Highly Comparative Time-Series Analysis

Benjamin D. Fulcher
Balliol College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

Hilary 2012

Acknowledgements

I would first like to thank my family for the support that they have always provided, regardless of what I'm doing or where I am in the world. With time, I am increasingly aware of the central role of their unconditional support on my life. During my time in Oxford I met some of the most beautiful and talented people, and I am extremely grateful for their friendship. That they came close to keeping me sane is commendable. My Australian friends also provided great support from a great distance, a grounding that was extremely valuable to me.

The development of ideas in this thesis required a lot of discussion, through regular meetings with my two supervisors, Nick Jones and Max Little. Nick Jones provided incredible support, assistance, and advice throughout my DPhil, showing genuine interest in both my personal wellbeing and my development as a researcher, and I am thoroughly grateful for it. My co-supervisor, Max Little, also provided lots of helpful feedback and advice on this thesis, which I value greatly. The Systems and Signals group in the Department of Physics was a pleasure to be a part of: fascinating informal discussions on an apparently endless range of topics were commonplace. The sharp curiosity of this group of scientists, and their willingness to learn, has left a lasting imprint on me. The support I have received from departmental colleagues has also been amazing. I would like to thank Sumeet Agarwal for valuable feedback and discussions on my work throughout the DPhil process. David Smith provided code and related assistance with network visualization that was much appreciated.

Lastly, that I was able to live and learn so much in Oxford is due to financial support from the *Commonwealth Scholarship Commission* and the *Oxford Australia Scholarship Fund*. Their assistance has provided me many unique opportunities so early in my life that I will always be thankful for.

Abstract

In this thesis, a highly comparative framework for time-series analysis is developed. The approach draws on large, interdisciplinary collections of over 9 000 time-series analysis methods or *operations*, and over 30 000 time series, which we have assembled. Statistical learning methods were used to analyze structure in the set of operations applied to the time series, allowing us to relate different types of scientific methods to one another, and to investigate redundancy across them. An analogous process applied to the data allowed different types of time series to be linked based on their properties, and in particular to connect time series generated by theoretical models with those measured from relevant real-world systems. In the remainder of the thesis, methods for addressing specific problems in time-series analysis are presented that use our diverse collection of operations to represent time series in terms of their measured properties. The broad utility of this highly comparative approach is demonstrated using various case studies, including the discrimination of pathological heart beat series, classification of Parkinsonian phonemes, estimation of the scaling exponent of self-affine time series, prediction of cord pH from fetal heart rates recorded during labor, and the assignment of emotional content to speech recordings. Our methods are also applied to labeled datasets of short time-series patterns studied in temporal data mining, where our feature-based approach exhibits benefits over conventional time-domain classifiers. Lastly, a feature-based dimensionality reduction framework is developed that links dependencies measured between operations to the number of free parameters in a time-series model that could be used to generate a time-series dataset.

Statement of Contribution

Listed below are a summary of the contributions made by me to the work contained in this thesis, including papers that have been submitted, or will in the future be submitted to journals. All text in this thesis was written by me, with feedback from my supervisors Nick Jones and Max Little, except in the case of the overview chapter, Chapter 2, where additional feedback was provided by Andrew Phillips, Stephen Blundell, and Anna Lewis.

Chapters 2-5 **Highly comparative time-series analysis: the structure of time series and their methods**

B. D. Fulcher, M. A. Little, N. S. Jones

Submitted to *Nature* (as Chapter 2, with Chapters 3–5 forming the basis for supplementary information to this paper).

I was primarily responsible for this work, including assembling all empirical time series from publicly-available data resources and synthesizing time series from a variety of time-series models, collecting existing toolboxes and pieces of time-series analysis code and implementing them appropriately in the form of operations, devising new ways of analyzing data using existing methods, and introducing novel operations with no precedent in the literature. The computational framework and the interface between MATLAB and mySQL was implemented by myself with initial assistance from M. A. Little. All analysis performed was performed by me using methods detailed in Chapter 3. Being a novel project, the content of these chapters was developed through frequent consultation with and guidance from N. S. Jones and M. A. Little.

Sec. 5.3 **Highly comparative fetal heart rate analysis**

B. D. Fulcher, A. E. Georgieva, C. W. G. Redman, and N. S. Jones

A manuscript based closely on the work in Sec. 5.3 of this thesis was submitted to the *34th Annual International IEEE EMBS Conference*.

This project was initiated by A. E. Georgieva, a Post-doctoral Research Fellow at Nuffield Department of Obstetrics & Gynaecology at the John Radcliffe Hospital, who provided the dataset and domain knowledge to guide my work in processing, calculating, and analyzing the data. The text was written by me, with feedback from A. E. Georgieva and N. S. Jones. The project was supervised by C. W. G. Redman and N. S. Jones.

Chapter 6 **Highly comparative time-series analysis for temporal data mining**

B. D. Fulcher, N. S. Jones

To be submitted to a data mining journal, such as *Data Mining and Knowledge Discovery*.

I was primarily responsible for this work: I discovered the dataset and performed all analysis on it. N. S. Jones supervised the project.

Chapter 7 **Highly comparative feature-based dimensionality reduction**

In future, a paper based on this work will be submitted to a machine learning or physics journal.

The initial ideas for linking the structure in datasets with redundancy in feature-based representations of them came from Sumeet Agarwal's work on networks. I was primarily responsible for this work, including the development of a theoretical framework for understanding the structure in feature-based representations of time series, the motivation for and generation of all synthetic datasets, and their analysis. The development of ideas, including the construction of the information theoretic framework for understanding dependencies between operations, was closely guided by N. S. Jones, who supervised this project.

Contents

Chapter 1

Introduction

In this section, we overview the time-series analysis literature in the context of the problems addressed in this thesis. First we explain what time series are and how they are analyzed in Sec. 1.1, and then describe two case studies of the development of time-series analysis methods in Sec. 1.2. We then explore three key scientific themes from the thesis: the importance of comparison in methodological literatures, in Sec. 1.3, the role of empirical approaches in addressing these problems, in Sec. 1.4, and the appropriateness of computational methods for automating science, in Sec. 1.5. The organization of the thesis is then outlined in Sec. 1.6.

1.1 A diverse literature

The passing of time is a fundamental component of the human experience, and as such, the *dynamics* of processes is a core focus of human curiosity. For example, on observing the motion of a leaf in the wind, it is natural to contemplate the burstiness of the wind speed, whether the wind direction now is related to what it was a second ago, or whether these observed characteristics of the wind might be the same when observed tomorrow. Similar questions arise naturally when observing fluctuations in the intensity of candle light, observing ripples in the ocean, or when feeling a human pulse. Dynamical processes such as these can be measured and recorded as a time series; a time series is defined as an ordered sequence of measurements of some process taken through time. As such, time series are fundamental data objects that are synthesized, recorded, and analyzed across the scientific disciplines. While common themes across the sciences can be difficult to identify, at least these data

structures are common: animal populations are measured over time in biology, the price of gold in economics, human heart rates in medicine, cell potentials in biochemistry, air temperatures in climatology, electroencephalogram (EEG) signals in psychology, radio signals in astrophysics, the output from chaotic systems in mathematics, and the heat emitted during a chemical reaction in chemistry. In contrast to a simple statistical collections of data, the *ordering* of measurements is crucial to understanding time series as records of the dynamics of processes at a given time scale. Time-series analysis therefore focuses on understanding patterns in the ordering of time series, including correlations between successive measurements, trends across recordings, long-range scaling behavior, local motifs, and so on. In this thesis, we undertake the challenge of unifying different scientific approaches to time-series analysis by assembling a large database of time-series analysis algorithms and applying them to different types of time series. The result allows us to deduce the types of statistics that are most useful for effectively describing different types of dynamics, and thereby provides a unified, quantitative framework for expressing an intuitive human interaction with the dynamical processes we observe in the world.

The modern time-series analysis literature is the result of the development of methods to address specific time-series challenges in particular disciplinary contexts. For example, generalized autoregressive conditional heteroskedasticity (GARCH) models were developed in the context of economic problems to accommodate for nonstationarity in the variance of economic variables like the inflation rate [1], the 'Approximate Entropy' regularity statistic was developed for heart monitoring [2], and detrended fluctuation analysis (DFA) was developed by physicists analyzing DNA sequences [3]. As methods are developed to meet the needs of specific scientific problems, the time-series analysis literature continues to expand, and papers introducing new methods are found both in high-impact interdisciplinary scientific journals (e.g., Refs. [4–7]), as well as a broad range of discipline-specific journals. Some time-series analysts are interested in a highly theoretical treatment of time series, including tests and proofs for different types of time-series properties (e.g., Ref. [8]), new time-series models such as Bayesian particle filters [9] and Gaussian Process models [10], new types of complexity and entropy statistics [11], and the nonlinear properties of time

series generated from dynamical systems models [12, 13]. However, there is a large divide between this theoretical literature developed by statisticians, mathematicians, physicists, and engineers, and the practical implementation of this theory to real time-series datasets of short, and inevitably noisy recordings.

The time-series analysis literature therefore encompasses a vast body of expert knowledge that is currently disjointed. New techniques for time-series analysis take a long time to diffuse through the literature because they are published in discipline-specific journals, or because it is not obvious how a technique for EEG analysis, for example, might be applicable to other types of data analysis problems. Furthermore, particular methods become 'standard' in some disciplines, preferred by some scientists but perhaps unpopular with others, and it is difficult to discern how the different alternatives relate to one another or which are indeed better for which tasks. By synthesizing the interdisciplinary literature on time-series analysis and drawing on the resource as a whole, we hope to contribute to developing an overarching and unified view of the field in this thesis.

1.2 Case studies in time-series analysis methods

Having established some background on the diversity of the time-series analysis literature, we now consider the process through which it has developed. This is particularly interesting because the time-series analysis community consists of scientists with very different backgrounds and research interests. Some new methods are published in a conference proceedings or a discipline-specific journal, but are never reported on again. Other methods take the interest of the community, diffuse into other disciplinary literatures, and become the subject of subsequent scrutiny, through a set of tests on synthetic time-series datasets for example. Such methods can eventually become part of the 'standard' time-series analysis methodology. In the remainder of this section, we describe two brief case studies of time-series analysis methods developed for practical applications, describing both their reception and long-term relevance to the time-series analysis community. Note that this practical perspective on time-series analysis as a methods-based literature for real problems is taken throughout this the-

sis; we are concerned much less with the time-series research conducted in parts of statistics, mathematics, and engineering that takes a highly theoretical approach to modeling and analyzing abstracted time series.

As a first example, we explore the recent plethora of time-series analysis methods that involve mapping time series to complex networks [6, 7, 14–23]. The constructed networks are subsequently analyzed using network analysis techniques and used to infer properties of the original time series. Because these techniques link methods developed in two different data analysis disciplines—network analysis and time-series analysis—they have attracted a lot of attention. The *visibility graph*, developed by Lacasa et al. [6], is one such method that involves representing time series as bars protruding from a baseline (defined as the minimum value across the time series) so that each point in the time series becomes a node with links to those other time-series points that it can 'see', with a straight line that does not intersect any of the intermediate bars [6]. The authors demonstrate a correspondence between dynamical properties of time series and the properties of the networks generated from them: for example, periodic time series yield regular graphs, and fractal series produce scale-free networks [6]. After its proposal, the visibility graph method was applied to a number of real-world and theoretical problems, including hurricane time series (even though these the time series are too short to give meaningful results) [24], exchange rate series [25], and as a means of estimating the Hurst exponent of fraction Brownian motion series [26, 27]. A related *horizontal visibility graph* method (where the straight lines are restricted to being horizontal) was also used to derive some exact theoretical results for time series [28]. The possibility that these two areas of data analysis: time-series analysis and network analysis, could be related is tantalizing. However, although these network-based methods take a qualitatively different approach, it is not obvious how they relate to existing time-series analysis methods, nor whether they offer any improvement over them. The unusual representation of time series as bars with straight-line visibility, is also difficult to motivate and slow to compute. Despite this, the novelty of the visibility graph method has seen it adopted in various forms by the time-series analysis literature.

As a second example, we discuss the complexity measure *Approximate Entropy*,

or *ApEn*, which was developed originally by Pincus et al. in 1991 [2, 29]. The method is based on the solid theoretical foundation of Kolmogorov-Sinai entropy estimates [30], from the nonlinear time-series analysis literature [13]. The problem of linking the theory of Kolmogorov-Sinai entropy to a reliable and accurate empirical method for estimating it has been a challenge, but ApEn yielded promising results, even for short and noisy time series [31]. The method has since become extremely popular, especially for problems in biomedical time-series analysis, including EEG analysis [32–39] and heart-rate analysis [40–42], among others [43]. The strengths and limitations of ApEn have also been explored through numerical simulations and theoretical analysis. Shortcomings of the method have been addressed through the proposal of alterations and new methods [44, 45], including multiscale entropy (MSE) [46], volumetric ApEn, or *vApEn* [47], fuzzy entropy, *FuzzyEn* [48, 49], and, most notably, *Sample Entropy*, or *SampEn* [50]. SampEn, which corrects a minor theoretical error in the derivation of ApEn, has since become very popular [42, 51–60], and the subject of various numerical tests itself [45, 61–63]. Despite lacking a formal definition [64], there are very many methods for estimating the 'complexity' of a time series [65]; these numerous algorithms that have developed out of the Approximate Entropy algorithm are hence part of a literature of complexity estimates that is already heavily populated. In this context, it is worth noting a comment made by C. R. Shalizi that "every few months seems to produce another paper proposing yet another measure of complexity, generally a quantity which can't be computed for anything you'd actually care to know about, if at all. These quantities are almost never related to any other variable, so they form no part of any theory telling us when or how things get complex, and are usually just quantification for quantification's own sweet sake."[1] As well as highlighting the vast size of the time-series analysis literature, and how rapidly it can grow, this case study also demonstrates how methods developed by physicists in a biomedical context have diffused through the biomedical sciences, and also across to other disciplinary literatures, from the analysis of earthquake activity [54], to network traffic [66]. We also recognize the relevance of this case study to the divide between a theoretically-motivated framework for understanding the dynamics

[1] From `http://cscs.umich.edu/~crshalizi/notebooks/complexity-measures.html`

of chaotic systems [12, 67] and the related methods that are applied in practice to 'finite, noisy and/or stochastically derived time series' that often produce 'confounding and non-replicable results' [31], a common challenge for the empirical time-series analysis community.

Through briefly describing these two examples of time-series analysis methods, we hope to have given an impression of the scale, complexity, and interdisciplinarity of the time-series analysis literature. As a result, it is difficult to determine which methods are appropriate for which tasks, or what the relationship of any new method is to existing, more familiar methods, and therefore whether it represents a genuine improvement or is simply a slight variation on some existing method. In particular, can the development of complicated new methods for time-series analysis be justified when simpler, existing alternatives may outperform them? In this work, we develop a highly comparative framework that facilitates methodological comparison on a large scale with the aim of addressing these shortcomings.

1.3 The importance of comparison

Many of the problems with the development of data analysis literatures stem from a *lack of comparison*: both the comparison of a proposed method to alternative methods, and of its performance on different datasets. Current papers in time-series analysis seldom compare their results to alternatives for a variety of reasons: (i) it is difficult or time consuming, (ii) it is not required of an author by journal editors/reviewers, (iii) it could undermine one's results. One aim of this work is to establish a framework in which methodological comparisons across wide varieties of data types can be performed transparently and straightforwardly. It is evidently in the interests of the community that new results hold up to comparative tests and that new methods are understood in the context of familiar existing methods.

Due to the large number of available methods and the myriad ways one can approach a given problem, data analysis has been referred to as an 'art' [68]. Time-series analysts develop an 'intuition' for problems by drawing on their prior experience, by inspecting the structure of a time-series dataset through exploratory data analysis

[69], and by linking domain knowledge of how the time series were generated with the scientific questions asked of the data. Different time-series analysts will therefore approach a given problem in quite different ways and using different types of techniques: from the preprocessing of the data (e.g., to reduce measurement noise, remove non-stationarities, or transform the distribution of the data to approximate a Gaussian), the choice of time-series models to fit, the analysis methods to subsequently apply, and the interpretation of the results. Although time-series analysis can be unsystematic, the choice of methods can be justified and interpreted within the context of a given scientific problem. However, it does raise the question of whether time-series analysis could be complemented by a more systematic approach that explicitly draws on the broad, interdisciplinary wealth of available knowledge from across the sciences.

Problems encountered due to a lack of comparison have been noted in the medical sciences [70] and in data mining [71, 72]. In mathematics and physics, there is often a strong guiding theoretical framework that new methods and theories can be evaluated and scrutinized against: through carefully-devised simulations, or by performing very specific and precise experiments. In contrast, progress in the more practical disciplines, from nutrition to empirical data analysis, is much harder to evaluate. Often target systems are poorly understood, containing many interacting components, and producing noisy outputs that are difficult to measure. Due to these challenges, the development of useful methods is a difficult task in itself, that is made more challenging by the massive literature of existing methods and competing claims about the usefulness of particular methods over others. Without a guiding theoretical background or a culture of comparison, one can reasonably be skeptical about whether progress is being made through such an unstructured process. For example, new data analysis methods developed in the machine learning literature: for dimensionality reduction, feature selection, classification, clustering, etc., are typically reported by presenting those datasets for which they perform favorably, but not others: an effect known as *data bias* [71]. This practice of highlighting strengths but hiding weaknesses is embedded in the literature and leads to a large number of available methods but no way to evaluate which are better than others, and for which types of problems. Furthermore, when comparisons to alternative methods are performed, the other methods are often

not optimized or implemented in a way that allows a fair comparison to a method that the author is obviously much more familiar with and committed to: an effect termed *implementation bias* [71]. Similar phenomena occur in the time-series analysis literature, for example, one study on nonlinearity measures found that 'results differ immensely from application to application', leading to 'partially contradicting claims in the literature' [73]. It is clear that a lack of comparison in methodological literatures hinders progress in these fields.

Another identified problem with methodological literatures stems from a scientific model that rewards a certain type of positive result, e.g., a useful clustering algorithm in data mining, an accurate entropy estimate in time-series analysis, or statistically-significant result in the medical literature [70]. In a literature focused on reporting such positive results, the most important parts of scientific investigation are often hidden, including the exploratory data analysis process that led to the method, the cases in which it fails, and how its performance compares to that of alternative methods [69]. For time series, exploratory data analysis may involve testing a variety of methods, fitting different types of time-series models, and trying different types of pre-processings for the data, in an attempt to arrive at some useful or meaningful result in the context of a given problem. Throughout this process, it is difficult to determine how many different methods were tested, how much of the data was used for exploratory data analysis, and how much was used for confirmatory data analysis; all of these factors influence the significance of the final result [69]. Therefore, although a large volume of literature is concerned with precisely defining and understanding the behavior of particular analysis techniques, an appropriate strategy for selecting amongst them and then subsequently implementing and applying them to new problems is typically left to the intuition and experience of the data analyst, even though this is the more subtle and perhaps important component of the scientific investigation [74, 75]. In this thesis, we address these concerns by making what can be an opaque data analysis process more transparent.

Given that modern data analysis literatures contain a large and growing volume of methods (due to the computer, which makes it 'easy to invent and program a new algorithm' [72]), it has been suggested that perhaps 'a natural selection process will

thin out those algorithms that add little useful information' [72]. On one hand, it is reasonable to expect that the limitations of popular methods will be discovered with time and perhaps fixed, but it is optimistic to expect that different methods from across science will receive a fair comparison through many isolated and small-scale scientific efforts. This is especially the case given the bias towards the publishing of positive results, the tremendous rate at which new methods are proposed, and the time and effort required to demonstrate the limitations of any particular method. But even if this scientific process involving a large number of small-scale tests of the merits of different methods does indeed filter out poorly-performing methods and leave those that are the most useful, it is certainly an opaque and inefficient way of performing the required comparison.

1.4 Collecting observations: an empirical approach

With the aim of synthesizing different approaches to time-series analysis in this thesis, we take the approach of collecting many 'observations' of both time series and analysis methods, which are then catalogued and organized. This process represents a fundamental theme throughout the history of science: by first collecting and then structuring observations, humans have found patterns and developed models for understanding the world. Such is the case for the periodic table with collections of observations about the behavior of different chemical elements, for example, and the development of cladistics from collections of observations of the properties of living organisms. In these cases, patterns found in the relationships between apparently different chemical elements or organisms motivated an organizational structure that ultimately aids our understanding of these objects. In this thesis, we set about applying a similar approach to time-series analysis, a field that encompasses a large variety of methods developed in diverse scientific disciplines. In our framework, a hypothesis test developed in Economics for the stationarity of a time series is put alongside an entropy measure developed in the heart rate analysis literature and the parameters of a Gaussian Process model fit to the data. In this way, we form a structured library of methods developed by scientists. The result allows us to address the question: how

many different types of time-series analysis methods are really required to capture the dominant structural differences between the types of time series that we measure from the world? And a related question: do methods for summarizing structure in time series developed in different disciplines really exhibit qualitatively different behavior from one another, or can they be condensed into a unified set that simultaneously draws on the diversity in the literature? Investigating structure in a sufficiently diverse collection of time-series analysis methods can therefore inform us about patterns in the scientific practice of developing techniques for understanding the dynamics of processes. By taking a survey of our data and methods in this way, we can also understand the relationship between analysis methods developed for time series and the actual structure in time-series data that we observe in the world. Undertaking such a survey therefore allows us to assess the expectation that the time-series analysis methods developed by scientists are indeed suited to the types of structure present in real world time series.

1.5 Automation for science

In recent years, there has been a rapid increase in both the volume of data recorded in numerous scientific disciplines and the availability of greater computational power. The result has led to the development of statistical techniques that allow data to be analyzed automatically on an unprecedented scale. A well-known example is the genetic microarray, which allows biologists to identify specific genes from the thousands of gene expression levels measured in each experiment [76]. This high-throughput approach is now an important complement to traditional, smaller-scale research efforts in biology. In this thesis, we produce an analogous tool for time-series analysis by analyzing two large databases: one containing thousands of time-series analysis methods, and the other containing thousands of time series. It is unfeasible to analyze data on this scale manually; instead we make use of statistical machine learning methods that allow us to answer meaningful scientific questions asked of these large collections. For example, we developed methods that automate the selection of relevant methods and models for specific problems in time-series analysis. In so doing,

we hit on a recent theme in the scientific literature: to what extent can the scientific process be automated?

In the context of the data mining literature, Glymour et al. [68] have argued that the "'art" [of data mining] is gained through experience (at present at least) rather than taught. The implication for data mining is that human judgement is essential for many non-trivial inference problems.' The authors continue, that 'automation can at best only partially guide the data analysis process. Properly defining the goals of an analysis remains a human-centered, and often difficult, process' [68]. By automating the selection of time-series analysis methods in this work, we are made to confront this fundamental question: to what extent is it appropriate to automate what is currently a subtle and subjective data analysis process? As statistical and experimental automation techniques become increasingly sophisticated, the boundary between how much of the scientific process can be automated is continually shifting: from the generation of hypotheses, the measurement and collection of data, its processing and analysis, and its interpretation. For example, the concept of the 'robot scientist' [77], that forms hypotheses, tests them by performing experiments, and uses the results to perform subsequent experiments, has recently been realized experimentally [78]. A robot scientist named "Adam" was shown to generate relevant hypotheses and subsequently carry out appropriate experiments—in total, over 6.6 million biomass measurements—to identify genes encoding orphan enzymes in the yeast *Saccharomyces cerevisiae* [78]. Adam's behavior demonstrates some ability to experiment and generate hypotheses from the outcomes of experiments, but the premise of the problem and all the knowledge required to generate hypotheses and perform the analysis is pre-defined, and ultimately the conclusions need to be understood and interpreted by a human scientist. Despite these limitations, this result represents an important step in the direction of scientific automation and raises some deep issues in computer science about the ability of machines to perform inference on systems and learn beyond the mechanisms that they have been explicitly programmed to perform [79]. In another example of science automation, a recent paper describes the automatic deduction of appropriate physical laws underlying observations of simple mechanical systems [80]. In that work, experimental data was used

to guide the process through which appropriate analytical expressions describing the system were generated. Thus, even this celebrated hallmark of human creativity—the deduction of physical laws from human observations of the natural world—is now being attempted with some preliminary success by computers. While it is interesting to speculate about the future of this so-called 'science automation' [78], the questions asked of a scientific endeavor and the evaluation of the significance of its conclusions, are currently at the direction of a human scientist. However, computers can enhance the way science is performed by collecting greater quantities of data more accurately, and analyzing it in new ways. As noted by Dumouchel and Demaine [81], 'while knowledge discovery techniques can uncover hidden relationships in the data, only the user's expertise can give those relationships meaning'.

It is tempting to view automation methods with heavy skepticism, as encroaching on the territory of traditional scientific exploration. While there is potential for automation methods to be misused at the expense of rigorous scientific investigation, in the right hands, they can instead form a powerful tool that improves scientific practice. In this thesis, we cannot hope to replicate the subtle 'art' involved in the data analysis process, but we will automate key components of it using statistical machine learning methods. As with all scientific procedures, there is potential for their incorrect use, and this is no different: the data analyst must first carefully set-up the problem by processing the data appropriately, and also interpret the results in the context of the problem. The data analysis techniques developed in this thesis represent a step in the direction of automated data analysis to complement traditional methodological practice, and require the usual care in implementation and interpretation by a scientist with an understanding of the data and the methods applied to it.

Our automatic approach can also complement conventional approaches to modeling time-varying systems. Which approach is best suited to a given problem depends on the questions asked of the data, and how much is known about the system that generated it. For example, a large literature of domain knowledge is available for some systems, and can be used to build informative, mechanistic models of them. This process builds an understanding of how the modeled mechanisms give rise to

the observed time series produced by the system. While our methods could perhaps be used to help with model selection for these systems—e.g., to choose simple models that best reproduce the properties of the observed signals—our automatic approach does not engage directly with this mechanistic modeling process. However, as we demonstrate in Chapter 7, in some cases we can infer the number of free parameters in a model used to generate a given time-series dataset and also provide estimates of these parameters. This knowledge could be used to guide the modeling of time-varying systems, as it highlights the types of properties that must be captured by free parameters in such a model (e.g., it might tell the time-series analyst that their model ought to capture distinct variation in stationarity, entropy, and auto-correlation). However, as we show in Chapter 5, our automatic methods are best suited to particular types of tasks, including automatic classification of different types of time series, automatic regression onto time-series characteristics, and automatic clustering to time series into meaningful groups. This is likely to be most useful when: (i) there is little domain knowledge about the system such that a mechanistic approach to modeling it is unfeasible, or (ii) statistical approaches to modeling it cannot capture the complex dynamics in the time series. However, even when a time-varying system can be modeled successfully using a mechanistic or statistical approach, we show in Chapter 5 that our automatic approach can still provide uniquely useful results. For example: we select new and unexpected types of informative analysis techniques for a range of time-series analysis tasks. These techniques can then be investigated further by the time-series analyst and may provide some additional intuition about the system, may stimulate subsequent theoretical investigation about why the algorithm is behaving in this way on the data, or may motivate a certain approach to modeling the system. In summary, the automatic methods introduced in this thesis do not attempt to replace conventional approaches to modeling them, but perform a different role that can be applied even when no domain knowledge about a system is available, or when statistical modeling is inappropriate.

1.6 Organization of the thesis

The thesis is organized as follows. In a somewhat unusual structure, we first present an overview of some of the main results of the thesis written for a general scientific audience in Chapter 2. This chapter includes some key results of the thesis, and explains their significance. In subsequent chapters, the content of this chapter and the ideas it contains are expanded upon in more detail. In Chapter 3, we describe our collections of over 9 000 time-series analysis methods and over 30 000 time series that we have assembled. We also describe and justify our highly comparative framework for time-series analysis that is used throughout the thesis, including methods for feature selection, classification, and clustering. The empirical structure in our collections of time-series analysis methods and time-series data is investigated in Chapter 4. In this chapter, we describe how interesting connections between different analysis methods, and between various pieces of data, can be discovered. In Chapter 5, we present applications of our highly comparative methods to specific classification and regression problems, including distinguishing Parkinsonian phoneme speech samples from those of healthy controls, separating congestive heart failure heart beat interval series from normal sinus rhythms, classifying the emotional content in long speech recordings, predicting the scaling exponent of self-affine time series, learning estimators of the Lyapunov exponent of time series generated from the Logistic Map, and distinguishing babies with an increased risk of compromised health from fetal heart rates recorded during labor. In Chapter 6, our highly comparative feature-based representation of time series is applied successfully to standard temporal data mining datasets, which typically treat short time-series patterns and use distances between the objects in the time domain to classify them. A feature-based dimensionality reduction framework for time series is developed in Chapter 7, that allows us to estimate the number of free parameters in a model responsible for generating a time-series dataset and to identify what types of time-series properties they control. The main findings of the thesis are summarized in Chapter 8, where some future directions for this highly comparative approach to data analysis are discussed.

Chapter 2

Highly comparative time-series analysis

This chapter represents a self-contained summary of the main methods and findings of this thesis, summarized in the format of a high-impact scientific journal and presented for a general scientific audience. The purpose of this chapter is to provide a concise overview of the main findings of this thesis, but is not to give a thorough justification of the methods used, nor a detailed analysis of the results obtained. The material in this chapter instead represents abbreviated versions of the more detailed analysis that will be performed in subsequent chapters (Chapters 3–5). In the context of this thesis, therefore, this chapter should be considered as an overview, with details provided subsequently. For completeness, relevant sections of the thesis in which more details can be found are referenced throughout this chapter. Note that these references do not need to be followed during a reading of this chapter, as the remainder of the thesis is self-contained; rather, the these forward references simply indicate where methods and results will later be explained in more detail in the thesis.

Abstract The process of collecting and organizing sets of observations and methods represents a common theme throughout the history of science. But despite the ubiquity of scientists measuring, recording, and analyzing the dynamics of different processes, an extensive organization of scientific time-series data and analysis methods has never been performed. Addressing this problem, we collected and organized over 35 000 empirical and synthetic time series and over 9 000 different algorithms for time-series analysis. Analyzing the interdisciplinary resource as a whole allowed us

to construct reduced representations of time series that efficiently encapsulate their dynamical properties. Furthermore, we introduce a set of comparative tools that allow many elements of time-series analysis tasks to be automated, including the selection and organization of useful models and metrics for a given dataset. The broad scientific utility of these tools is demonstrated through a variety of case studies using electroencephalograms, self-affine time series, heart-beat intervals, and speech signals.

Introduction Time series, measurements of a quantity taken over time, are fundamental data objects studied across the scientific disciplines: from measurements of stock prices in finance and ion fluxes in astrophysics, to atmospheric air temperature in climate science, and human heart beats in medicine. In order to understand the underlying mechanisms that generate these signals, scientists have developed many different techniques: from various summary statistics, to time-series models. For example, methods based on fluctuation analysis are frequently used in Physics [3], generalized autoregressive conditional heteroskedasticity (GARCH) models are a standard technique in Economics [1], and entropy measures like 'Sample Entropy' are popular in medical time-series analysis [50]. Are these methods, that have been developed within specific disciplinary contexts, summarizing empirical time series in unique and useful ways, or is there something to be learned by synthesizing and comparing them? We address this question by assembling extensive annotated libraries of both time-series data, and methods for time-series analysis. By coupling these two libraries together, we used each to organize the other: methods are characterized by their behavior across a wide variety of different time series, and time series are characterized by the outputs of a diverse library of methods applied to them. In this way, by treating our data and methods for time-series analysis as objects that require organization, we are able to give structure to the inherently interdisciplinary field of time-series analysis.

Synthesizing diverse sets of time-series data and approaches to time-series analysis yields many interesting comparative results. For example, connections are formed between different types of time series, and between apparently disparate analysis methods. Furthermore, despite variability in both the measurement process and in the range of dynamics present in natural systems, we find that empirical time-series data

exhibit structures that can be effectively summarized by reduced representations that simultaneously draw on multiple interdisciplinary techniques for time-series analysis.

As well as providing insights into the structure of comprehensive libraries of scientific time series and analysis methods, we also demonstrate how these libraries can be used to automate many time-series analysis practices. Through numerous case studies, we use the structure and known properties of the time-series data to select useful types of analysis methods—by comparing their performance across our library. This general approach allows us to tackle a variety of diverse problems, from the classification of speech signals, to the estimation of scaling exponents in self-affine time series, where we present significant contributions to the existing literature on each problem. We contrast our methodology to conventional studies that focus on small sets of manually-selected techniques with minimal comparison to alternatives.

Framework (Chapter 3) We have assembled an annotated library of 38 190 univariate time series measured from diverse real-world systems and generated from a variety of synthetic systems, and another containing 9 613 time-series analysis algorithms developed in a variety of scientific literatures (see Appendices E and F for full lists). Methods for time-series analysis come in various different forms, including summary statistics and model fits. In our library, we implement each such method as an algorithm—an *operation* that summarizes an input time series with a single real number.

A major challenge of this work involves constructing a unified representation of very different time series and diverse time-series analysis methods that facilitates a direct and meaningful comparison between them. To this end, we form a data matrix, as in Fig. 2.1C, that contains the outputs of a set of operations applied to a set of time series. In this way, time series are represented by feature vectors that contain the results of measuring an extensive interdisciplinary range of their properties (Fig. 2.1A), and operations are represented by feature vectors that contain their outputs across the time-series dataset (Fig. 2.1B). Cases for which operations do not yield a real number, or are inappropriate (e.g., fitting a positive-only distribution to non-positive data), are referred to as 'special values' throughout this chapter, and are

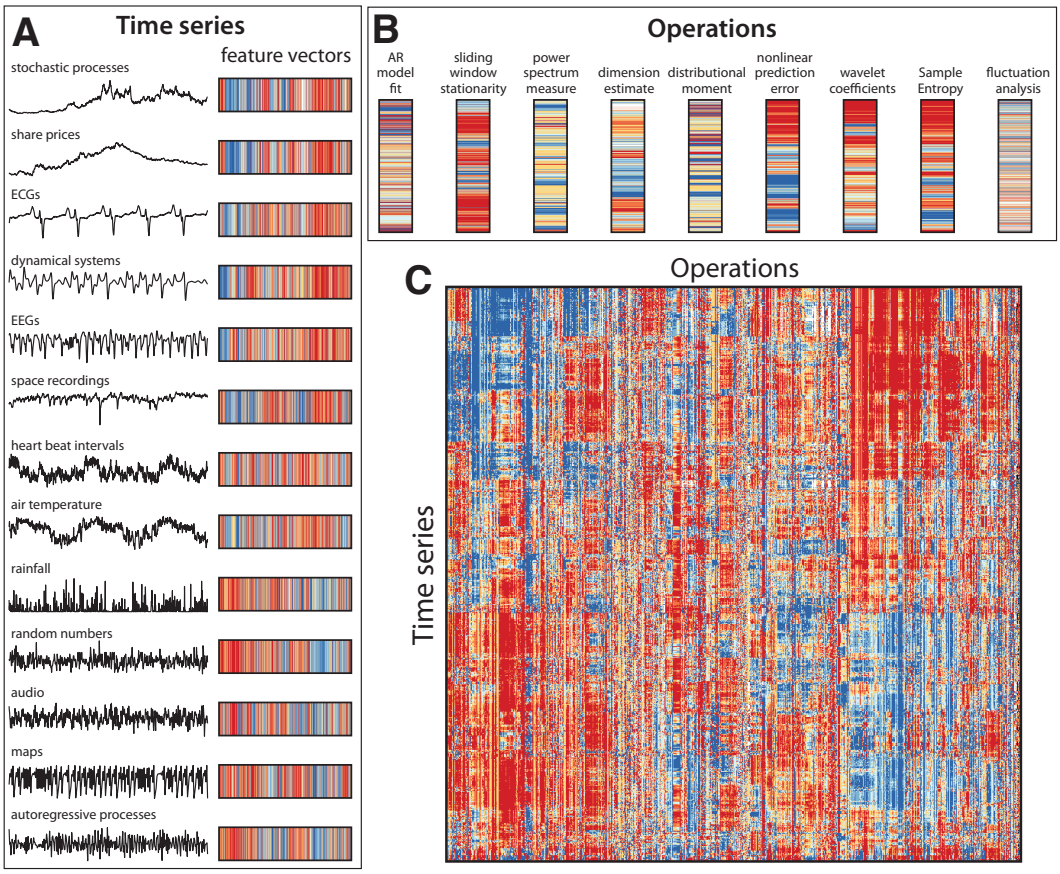

Figure 2.1: **A unified representation of time series and their analysis methods.** Time series are represented as feature vectors containing a large number of their measured properties, and operations are represented as feature vectors containing their outputs across a time-series dataset. We use an interdisciplinary set of 875 time series and 8 651 operations to illustrate this. The feature-vector representations for a range of time series and operations are shown in **A** and **B**, respectively. **C** The full dataset is visualized as a data matrix with columns that have been normalized (see Sec. 3.3.2), and where columns and rows have been reordered to put similar time series and operations close to one another (see Sec. 3.4.2). Each element of the data matrix represents the output of an operation applied to a time series, and is visualized here using color: from blue (low output) to red (high output). This representation, which allows us to draw on a diverse literature on methods for time-series analysis, forms the basis for the analysis performed in this thesis.

treated as missing values in the data matrix that can be filtered out (see Sec. 3.2.4).

Empirical structure of time-series analysis methods (Sec. 4.1) In this section, we analyze the structure in our library of time-series analysis operations, and show that significant redundancy within it can be exploited to form useful reduced representations of empirical time series. To study the behavior of our library operations, we assembled a representative interdisciplinary set of 875 real-world and

synthetically-generated time series (as illustrated in Fig. 2.1A, and described in Sec. 4.1.1). We then filtered the library to consider only the 8 651 operations that returned less than 20% special-valued outputs on these time series. Operations with highly-correlated outputs across this diverse set of time series are performing broadly similar functions for analyzing empirical time-series data, representing a redundancy in the set of scientific methods for time-series analysis. But how many different operations are required to summarize the structure we observe in these scientific signals most effectively? We address this question by clustering our library of operations [82]. For example, the results of a k-medoids clustering into four groups is depicted in Fig. 2.2A, reflecting an intuitive summary of the categories of operations developed by scientists to study time-varying signals.

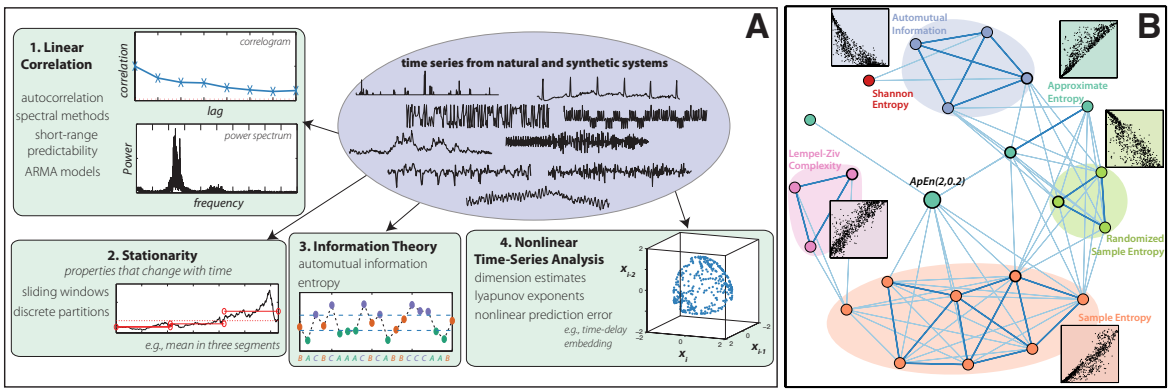

Figure 2.2: **Structure in a library of 8 651 interdisciplinary time-series analysis operations. A** The four main classes of operations in our library, as determined by a simple k-medoids clustering. **B** A network of operations in our library that are similar to the *Approximate Entropy* algorithm, ApEn$(2, 0.2)$ [2]. Each node in the network represents an operation and links encode the normalized mutual information between their outputs. Scatter plots showing the outputs of ApEn$(2, 0.2)$ (horizontal axis) against a representative member of each shaded community (indicated by a heavily outlined node, vertical axis) are annotated.

Next, we use clustering to isolate a reduced set of 200 operations that effectively summarize the behavior of these 8 651 operations. To quantify the quality of reduced sets of operations, we use the *residual variance* measure, $1 - R^2$, where R is the linear correlation coefficient of the distances between time series calculated in a reduced space compared to those calculated in the full space [83]. Using k-medoids clustering for a range of clusters, k, we found that a reduced set of 200 operations is a very good approximation to the behaviors contained in all 8 651 operations, with a resid-

ual variance of just 0.05 (Sec. 4.1.2). This set of operations is a concise summary of the types of behavior present in our full library of operations when applied to diverse types of empirical time series, and draws on techniques developed in a range of scientific disciplines: autocorrelation, automutual information, stationarity, entropy, long-range scaling, correlation dimension, wavelet transforms, linear and nonlinear model fits, and measures from the power spectrum. Later, we will use this reduced set of operations to structure our library of time series in a meaningful way.

As well as analyzing the global structure of our library of operations, it is also informative to explore the local structure surrounding a given target operation. In this way, the behavior of the target operation is contextualized with respect to alternative approaches developed in other fields. The result can be visualized as a network, with nodes representing scientific operations and links indicating their similarities, as illustrated in Fig. 2.2B for the *Approximate Entropy* algorithm, $\mathrm{ApEn}(2, 0.2)$. This operation is a measure of 'regularity' that has been applied to a wide variety of time-series analysis problems [2]. In Fig. 2.2B, different communities of operations are revealed, including methods based on Sample Entropy, Lempel-Ziv complexity, automutual information, Shannon entropy, and other Approximate Entropies (evaluated using different parameters). Inset scatter plots explicitly show the dependence between the outputs of each of these types of operations with $\mathrm{ApEn}(2, 0.2)$. We thus discover that, even when mixed with 8 650 different operations, related families of entropy measures are retrieved automatically and organized meaningfully, according to their behavior on the types of empirical signals that they are applied to in practice. We found similar results for case studies involving fluctuation analysis and singular spectrum analysis, where connections are made between interdisciplinary communities of methods that nevertheless perform similar functions on empirical time series (Sec. 4.1.3). This highly comparative outlook encourages interdisciplinary collaboration on methods for time-series analysis.

Empirical structure of time series (Sec. 4.2) In this section, we use the representative set of 200 operations formed above to organize a set of 24 577 empirical and synthetically-generated time series (chosen as a sensible subset of the full time-series

library, see Sec. 4.2.2). By drawing on such a wide variety of operations simultaneously, time series of different types, lengths, and sampling rates are organized in a meaningful way. First, we probe the structure of our time-series library by forming 2 000 time-series clusters (using complete linkage clustering, see Sec. 4.2.2). Despite the size and diversity of the time series that populate our library, most clusters contain time series measured from the same system, some examples of which are shown in Fig. 2.3A (see Sec. 4.2.2 for more examples). We also found that the same representation clusters known classes of different time series correctly (Sec. 4.2.1). This interdisciplinary feature-vector representation is clearly a powerful one for distinguishing sets of distinct dynamical behavior from a library containing thousands of different types of time series.

Some clusters contain time series generated by different systems that nevertheless exhibit similar dynamics. One such cluster, illustrated in Fig. 2.3B, contains outputs generated from three different iterative maps: the *Cubic Map*, the *Sine Map*, and the *Asymmetric Logistic Map* [12]. Although this cluster of synthetic signals contains time series of different lengths and generated from maps specifying different recurrence equations, it groups outputs from each map with parameters that specify a common generative process, as shown in Fig. 2.3B. This cluster therefore distinguishes a distinct class of dynamical behavior from a large library of time series, demonstrating our ability to exploit an interdisciplinary literature on time-series analysis to uncover commonalities in the underlying dynamical processes that generate time series. We see this behavior repeated in other clusters (Sec. 4.2.2.2).

Apart from clustering, we can also target specific real-world time series and retrieve similar synthetically-generated matches. In this way, suitable families of models for understanding the dynamics of real-world measurements can be suggested. An example of this procedure is depicted in Fig. 2.3C, where real-world and synthetic matches to a series of opening share prices for *Oxford Instruments* (OXIG) are plotted. Real-world matches are other opening share price series, and synthetic matches are outputs from stochastic differential equation (SDE) models. Indeed, the form of SDE models suggested by this set of closest matches, e.g., $dX_t = aX_t dt + bX_t dW_t$, is the same as that used in financial modeling: e.g., *geometric Brownian motion* [84]:

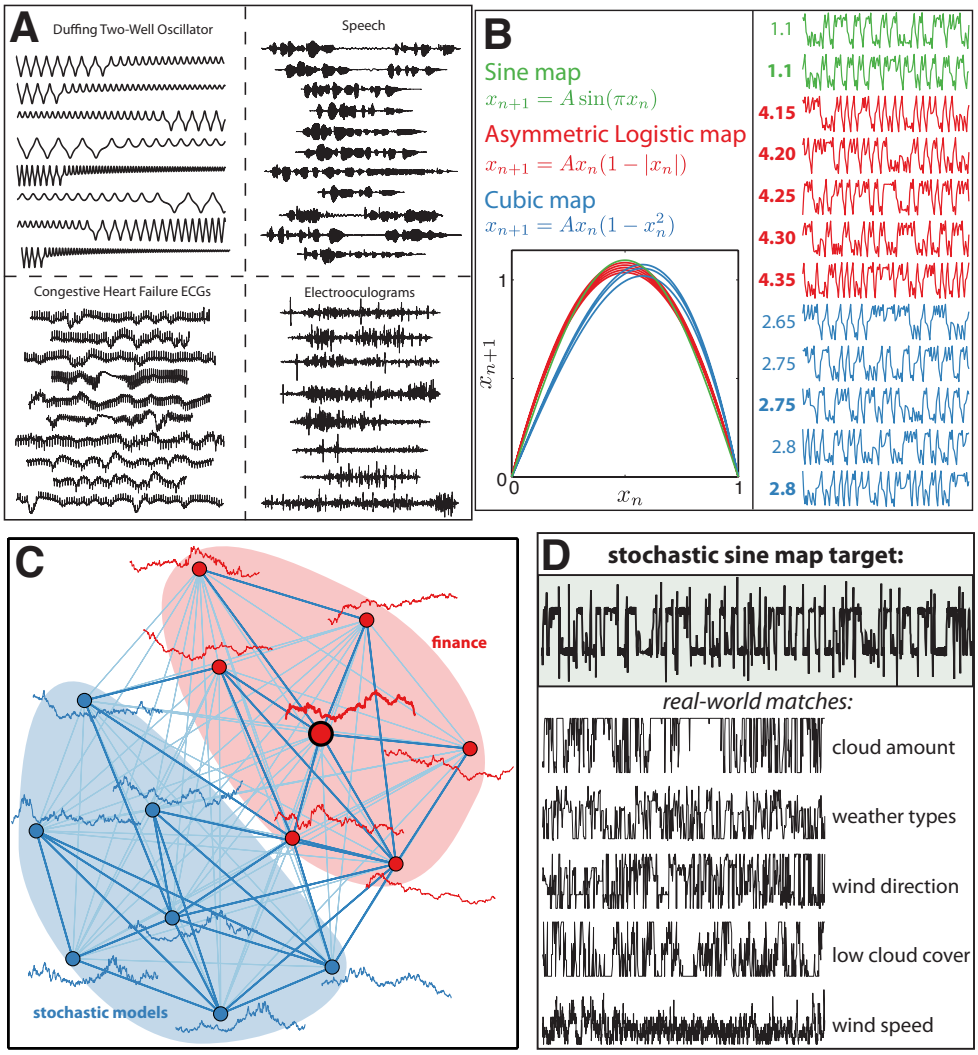

Figure 2.3: **Structure in a diverse set of 24 577 time series.** **A** Of the 2 000 time-series clusters formed from the time series, most are homogenous groups of outputs from a given synthetic or real-world system; four such examples are shown. **B** Some time-series clusters mix outputs from different systems that nevertheless exhibit similar generative mechanisms. This cluster groups outputs from three different maps with parameters that specify a common recurrence relationship. Time-series subsegments of 150 samples are plotted and labeled with the parameter A of the corresponding map that generated them (bold labels indicate 5 000-sample time series; others are 1 000 samples long), and the recurrence relationship $x_{n+1}(x_n)$ is plotted for all members. **C** Suitable families of models for real time series can be retrieved as near neighbors in our time-series library. In this example, we target a series of opening share prices for *Oxford Instruments* (OXIG): the larger, outlined node. The closest real-world time series are other opening share price series (red), and synthetically-generated neighbors are outputs from stochastic differential equations (blue). The result is represented as a network, where nodes represent time series and links encode distances between them. **D** The most similar real-world time series to a stochastic sine map target are noisy meteorological processes that display qualitatively similar 'noisy switching' dynamics.

$dX_t = \mu X_t dt + \sigma X_t dW_t$ (where X_t represents the time series, W_t denotes a Wiener process, and other variables are parameters). Matches to SDE models with slightly different forms, e.g., $dX_t = a(b - X_t)dt + \sigma\sqrt{X_t}dW_t$, suggest that other models with particular parameter values can also reproduce many dynamical properties of the target share price series that are not unique to the choice of geometric Brownian motion. The same phenomenon was observed for a range of other target time series, including rainfall patterns, astrophysical recordings, and human speech signals, for which relevant sets of real-world time series and appropriate families of models were retrieved automatically from our library (Sec. 4.2.3).

Outputs from synthetic systems can also be targeted to retrieve instances of real-world processes with similar dynamics. In Fig. 2.3D, we target an output of a stochastic sine map, which has a fixed probability of additive uniformly-distributed noise at each time step that can switch the system between two stable limit cycles [85]. The closest synthetic matches are stochastic sine maps with the same parameters as the target (Sec. 4.2.3.2), and real-world matches are meteorological processes that exhibit the same qualitative 'noisy switching' dynamics (Fig. 2.3D). In this way, the dynamics specified by the stochastic sine map model is explicitly linked to that of real-world meteorological processes. Similar results were also obtained for noise-corrupted sine waves and self-affine time series, where the types of real-world systems that can produce the dynamics specified by these time-series models is revealed, thereby bringing time-series models and time-series data closer together (Sec. 4.2.3.2).

Applications

Beyond organizing interdisciplinary collections of operations and time series, our libraries can also be used to select the most useful types of operations for analyzing specific time-series datasets and, because each operation is interpretable, new insights into their structure are thereby gained. In the remainder of this report, a series of case studies are presented to demonstrate the broad utility of our comparative approach to real problems in time-series analysis. Each case study is explained and analyzed in additional detail in Secs. 5.1 and 5.2 of this thesis.

EEG Classification (Sec. 5.2.4) In the first case study, we show how the collective behavior of our library of operations can be used to organize different types of electroencephalogram (EEG) time series, and we evaluate whether progress is being made in a literature concerned with the classification of epileptic seizure EEGs. The dataset contains 100 EEG signals from each of five classes: set A (healthy, eyes open), set B (healthy, eyes closed), set C (epileptic, not seizure, recorded from opposite hemisphere of the brain to epileptogenic zone), set D (epileptic, not seizure, recorded within the epileptogenic zone), and set E (seizure) [86]. The two-dimensional Principal Components projection of the data matrix, shown in Fig. 2.4A, summarizes the collective behavior of our library of operations, which automatically arranges the time series in a way that is consistent with their known properties: seizure time series (set E) are particularly distinguished, and the two healthy (sets A and B) and the two epileptic (sets C and D) sets are close in this space.

This dataset has previously been used to build classifiers that distinguish healthy from seizure signals using sets A and E. For example, one study reports a support vector machine classifier using a set of operations derived from the discrete wavelet transform with a classification accuracy of at least 98.75% using re-sampled subsegments of this dataset [87]. However, we can see that without knowledge of their class labels, these two types of signals are automatically separated (with 99% classification accuracy) in the two-dimensional Principal Components space shown in Fig. 2.4A, indicating that this task is relatively straightforward when exploiting the combined behavior of a large library of time-series analysis operations. Indeed, we found 172 different operations in our library that can each *individually* discriminate between these two classes with a 10-fold cross-validation linear classification accuracy greater than 95%, and eight operations with a classification rate exceeding 98.75% (cf. Sec. 5.2.4.3). These most successful operations are derived from diverse areas of the time-series analysis literature, and reveal that seizure EEGs have greater spread, lower entropy, lower correlation dimension estimates, and lower long-range scaling exponents. In contrast to existing research proposing a variety of complex classification methods for this problem, the comparative analysis performed here automatically selects simple and interpretable methods for the task.

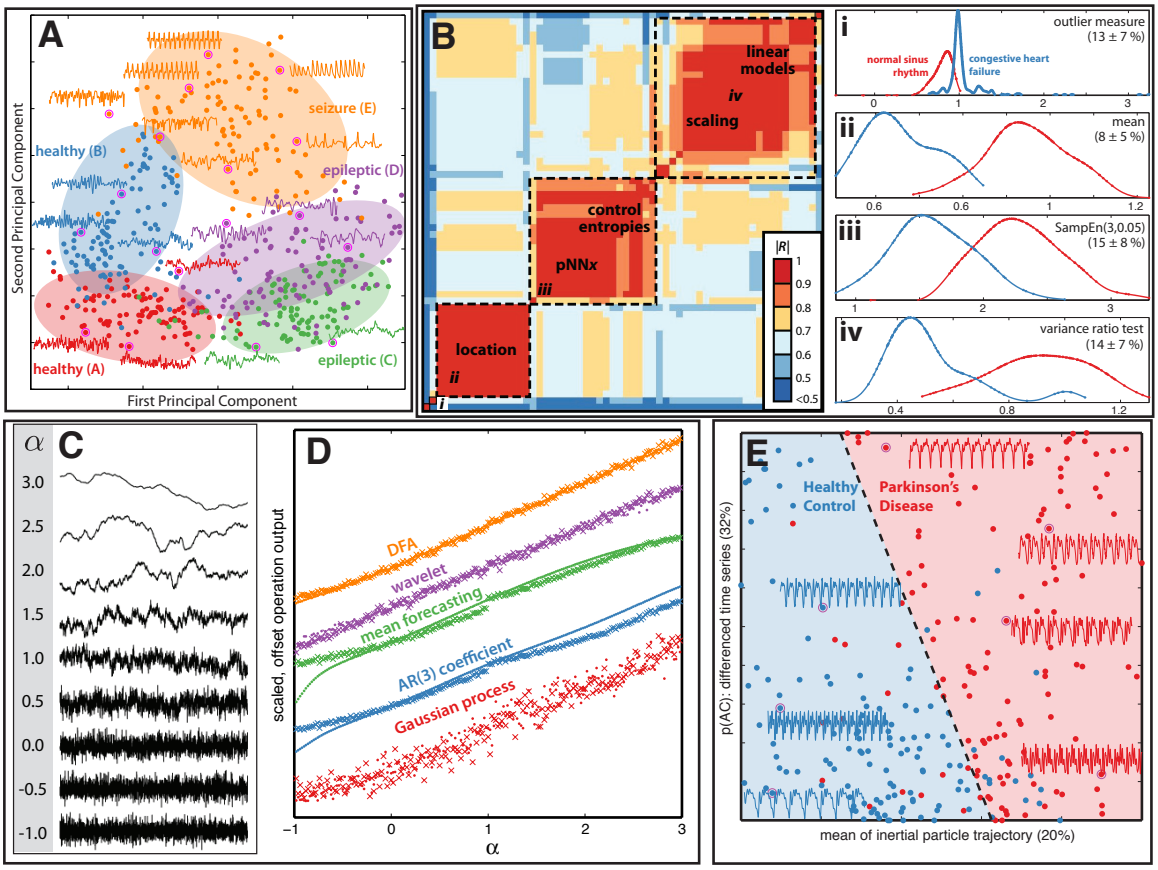

Figure 2.4: **Our extensive interdisciplinary library of time-series analysis operations allows time-series datasets to be analyzed in new ways. A** Five groups of EEG signals are meaningfully structured by the leading two Principal Components of the data matrix for this data. **B** A clustered matrix of linear correlations between the outputs of successful operations for distinguishing congestive heart failure from normal sinus rhythm RR interval series groups operations into three main types of behavior. Distributions of algorithms labeled **i**, **ii**, **iii** and **iv** in **B** are plotted for the two classes of time series, including the mean ± standard deviation of their classification accuracy using 10-fold cross-validation linear discriminant analysis. **C** Segments of time series generated with scaling exponents in the range $-1 \leq \alpha \leq 3$. **D** Five selected operations with outputs that vary approximately linearly with α. **E** An example of a linear classifier that combines the outputs from two different operations to classify speech recordings from healthy controls (blue) and patients with Parkinson's Disease (red) with a mean 10-fold cross-validation accuracy of 86%; the mean accuracy of each individual operation is given in parentheses.

This case study emphasizes the difficulty in interpreting the significance of high classification accuracies for empirical time-series classification tasks without comparison to other methods. This practice is common in the time-series analysis literature, from the classification of audio signals to that of human hand tremors, because the size and diversity of the field makes it unfeasible to compare the performance of any given method to many other possible choices. Our comparative approach provides an interdisciplinary context that can be used to assess whether scientific progress is being made through the proposal of new time-series analysis methods.

Heart rate variability (Sec. 5.2.6) In this case study, our highly comparative analysis is applied to an existing problem of classifying heart-beat interval (or RR interval) series into two classes: 'normal sinus rhythm' and 'congestive heart failure'. Many existing studies have analyzed RR interval data, each reporting on the usefulness of a particular set of techniques [88]. In contrast, we demonstrate how useful methods for this task are retrieved from our library and organized in a way that meaningfully synthesizes a large and disparate literature of existing approaches on the subject, and identifies promising new approaches. The dataset contains 105 recordings from each class, with lengths ranging from 800 to 19 900 samples.

The ability of each operation in our library to individually classify the two classes was calculated using 10-fold cross-validation linear discriminant analysis. The most successful operations spanned diverse areas of time-series analysis, from measures of location, outlier properties, entropy estimates, correlation structure, and long-range scaling exponents. In Fig. 2.4B, the 60 operations that each have a mean classification accuracy exceeding 85% are plotted as a pairwise similarity matrix, where color represents the linear correlation coefficient between the outputs of pairs of these operations across the dataset. As shown in Fig. 2.4B, most operations cluster into three main groups of behavior: (i) measures of location, e.g., mean, median, (ii) *PNNx* and entropy estimates, and (iii) linear correlation-based methods, e.g., autoregressive model fits, power spectrum scaling, and variance ratio tests. Two miscellaneous other methods in the bottom-left corner of Fig. 2.4B are relatively independent of the others: a discrete wavelet transform decomposition operation, and

an outlier-adjusted autocorrelation measure. Examples of distributions from each of these families of successful operations are plotted in panels of Fig. 2.4B: **i** an outlier measure returns the ratio of lag-3 autocorrelations of the time series before and after removing 10% of outliers, **ii** the mean, **iii** the Sample Entropy of the time series using an embedding dimension $m = 3$ and threshold $r = 0.05$ [50], and **iv** the output of a variance ratio test from the Economics literature [89]. Each of these operations represents an interpretable measure of structural difference between time series: e.g., the well-known results that normal sinus rhythm series tend to have longer inter-beat intervals (higher mean) and greater entropy (higher Sample Entropy) than congestive heart failure series.

This case study demonstrates the ability of our comparative approach to select and organize useful methods for a given time-series classification problem that provides a set of interpretable sources of difference between the known classes of time series. In this case, the operations retrieved from our library mostly correspond to existing families of methods with known utility for this dataset. This synthesis, which reveals commonalities across different approaches and distinguishes the novelty of others, can be used to assess new contributions to the time-series analysis literature. For example, a new study could focus on the novelty of the variance ratio test, a method originally developed in the Economics literature, and shown in panel **iv** of Fig. 2.4B. However, as shown this figure, the operation behaves in almost the same way as a range of simple linear model-based methods on this dataset—it is not actually measuring a new property of these time series. On the other hand, a new operation introduced in this work, based on the impact of outliers on time-series autocorrelation properties, is plotted in panel **i** of Fig. 2.4B and is distinguished as both unique and useful for this dataset. Referencing our comparative framework in this way can ensure that new analysis algorithms developed for any given dataset do not simply reproduce existing behavior.

Self-affine time series (Sec. 5.1.3) Algorithms based on fluctuation analysis have attracted substantial attention in the statistical physics literature, as providing evidence for scale-invariance in real-world processes. But are these conventional meth-

ods outperforming alternative approaches, or are there simpler methods that display comparable or superior performance? For this case study, we address this question by generating a synthetic dataset of self-affine time series and then searching our library for operations that vary with their known scaling exponents. Self-affine time series can be characterized by a single scaling exponent, α, according to $S(f) \propto f^{-\alpha}$, where S is the power spectral density of the time series as a function of its frequency, f [90]. Time series of 5 000 samples were generated with scaling exponents in the range $-1 \leq \alpha \leq 3$ by two different processes: (i) the *Fourier filtering method*, and (ii) by the *random midpoint displacement method* [90], as plotted in Fig. 2.4C.

We calculated the linear correlation coefficient between the output of each operation in our library to the known scaling exponent, α, that characterizes the process that generated each time series. A selection of five operations with the strongest linear correlations to α are plotted in Fig. 2.4D. Many fluctuation analysis-based algorithms perform well for this task, including the scaling exponent estimated using detrended fluctuation analysis (DFA) [3], and a wavelet-based alternative (labeled 'DFA', and 'wavelet' in Fig. 2.4D, respectively). Some other types of operations that are not based on fluctuation analysis also exhibit strong linear correlations with α: the autocorrelation of residuals from a local mean forecaster (labeled 'mean forecaster' in Fig. 2.4D), the first-order coefficient of a third-order autoregressive, AR(3), model fitted to the time series (labeled 'AR(3) coefficient'), and the mean log hyperparameter of the squared exponential length scale from a Gaussian Process regression on different local segments of the time series (labeled 'Gaussian process') (Sec. 5.1.3).

In this case study, we confirm the utility of methods based on fluctuation analysis to estimate the scaling exponent of self-affine time series, and also discover a surprising selection of other useful methods that require significantly less computation, and can be iteratively updated for real-time applications, unlike operations based on fluctuation analysis. A similar procedure was used to select suitable and interpretable estimators of the variance of contaminating noise added to periodic signals (in Sec. 5.1.1) and the Lyapunov exponent of time series generated from the Logistic Map (in Sec. 5.1.2). This highly comparative method for regression onto time-series characteristics could also be used to retrieve estimators of important diagnostic quantities

assigned to physiological recordings automatically, e.g., to estimate a patient's age from their gait time series, or the fetal cord pH from a fetal heart-rate recording obtained during labor (as studied in Sec. 5.3).

Parkinsonian speech (Sec. 5.2.5) Our final case study involves the particularly challenging task of distinguishing stationary phoneme speech recordings of patients with Parkinson's Disease from those of healthy controls [91]. We show how our general library of time-series analysis operations can be used to construct classifiers with high accuracy, despite having no knowledge of the generative process underlying the time series nor the types of algorithms that have previously been used for this task. Instead, the structure of the data is used to select the most useful types of analysis methods. The dataset contains 127 Parkinsonian speech signals, and 127 from healthy controls with recording lengths ranging from 800 to 20 000 samples, or from 50 ms to 1.25 s. We aimed to find measures that robustly distinguish the pathological dynamics despite this large range of recording lengths.

By comparing across our library, we found that the most useful operations for classifying Parkinsonian speech signals include measures of local waveform shape, summaries of the power spectrum, and stationarity near the time-scale of the fundamental period of the signal. Summaries from the power spectrum are standard for these data, and the stationarity measures we retrieve are similar to the traditional *jitter* and *shimmer* measures for speech analysis [91]. However, rather than restricting ourselves to individual operations, we also retrieved complementary combinations of operations that together achieve improved classification performance. An example classifier is shown in Fig. 2.4E that combines a new operation introduced in this work, that simulates the trajectory of an inertial particle that experiences an attractive force to the time series, with another operation that constructs a symbolic string from incremental differences of the time series using a three-letter alphabet: {'A', 'B', 'C'}, and returns the probability of the word 'AC'. The resulting classifier has a mean 10-fold cross-validation classification accuracy of 86%, a rate comparable to the state-of-the-art in this literature, that typically treats unbalanced datasets [91] (Sec. 5.2.5). In this case, two very different types of algorithms are combined: the first analyzes

asymmetries in the local shapes of the speech waveform, and the second quantifies the probability of large increases in successive increments of the signal. This result is in keeping with a more general trend that we observe for multi-feature classifiers constructed for other datasets: the most robust classifiers combine measurements of multiple different time-series properties simultaneously. The classifiers are built automatically, require no knowledge of the mechanisms underlying the time series, and can easily be extended to classifiers that incorporate more than two operations (as we describe in Sec. 5.2.5).

In other case studies in this thesis, we used a similar highly comparative analysis to classify an unknown seismic recording as an explosion (in Sec. 5.2.2) and produce competitive multi-class classifiers for identifying the emotional content in speech signals (in Sec. 5.2.3).

Conclusions In summary, we have given structure to extensive annotated libraries of time-series data and their analysis methods—representing analysis methods by their behavior on empirical data, and representing time series by their properties as measured by the analysis methods. The result is a highly comparative approach to time-series analysis that provides new insights into the properties of time series studied in science and the techniques used to study them. We also demonstrated how our library of methods can be applied successfully to a wide variety of data types and time-series analysis problems, including the analysis of self-affine time series, Logistic Map sequences, noisy sine waves, EEGs, RR interval sequences, seismic measurements, and speech signals.

Like the high-throughput analysis of the genetic microarray in biology, which is now a standard complement to traditional, more focused research efforts, we envisage our comparative tool being used to help direct progress in time-series analysis, as it is performed across the sciences. Traditionally, time-series analysis tasks have focused on the use of methods and models motivated by domain knowledge and an exploration of the empirical properties of a dataset. We argue that the comparative analysis introduced in this work contribute a powerful complement to such studies. As vast quantities of time-series data continue to be measured and new methods

for their analysis continue to be proposed, this comparative approach will help us to make sense of a large and complex resource and direct progress in an inherently interdisciplinary field. The library of operations used in this work is freely-available such that future contributions can be added and contextualized to provide a growing resource of time-series analysis methods to serve the scientific time-series analysis community.

Chapter 3

Methods

In this chapter, we provide details of, and justifications for, the methods used throughout this thesis. First, we explain the representation of time series and operations as a data matrix in Sec. 3.1. Issues related to the computation of operations on time series and the structure of the database used to store the results are detailed in Sec. 3.2. We then discuss methods for processing different types of time series data and time-series analysis operations in a way that allows a meaningful comparison between them in Sec. 3.3. Lastly, we justify our choice of methods for analyzing data matrices, including the choice of a distance metric, clustering, classification, dimensionality reduction, and feature selection in Sec. 3.4. We also explain, with examples, the implications of multiple hypothesis testing for our work.

3.1 Framework

In this thesis, we study univariate time series, $\mathbf{x}$, each of which is an ordered sequence of N real-valued measurements sampled discretely at equal time intervals: $\mathbf{x} = (x_1, x_2, ..., x_N)$. Over 20 000 real-world time series have been acquired, primarily from publicly-available databases, which have been selected to encompass a representative sample of the types of signals measured in scientific disciplines: from meteorology (e.g., temperature, pressure, rainfall, river flow), medicine (e.g., heartbeat intervals, electrocardiograms, electroencephalograms, gait, tremor data), audio (e.g., speech, music, sound effects, animal sounds), astrophysics (e.g., solar radio flux, magnetic field strength), finance (e.g., exchange rates, stock prices), and others. In addition, we have generated over 10 000 synthetic time series, including outputs from

a range of nonlinear and chaotic maps, dynamical systems/flows, and stochastic processes. A full list of time series used in this work, including where they were acquired from, or how they were generated, is in Appendix E.

We also assembled a database of algorithms for time-series analysis, or *operations*. Each operation is an algorithm that summarizes a time series with a single real number. Algorithms that return multiple outputs are incorporated into our general framework as multiple operations, each of which returns a single output. For this thesis, we formed diverse collection of over 9 000 operations that quantify various time-series properties, including the basic statistics of the distribution (e.g., location, spread, measures of Gaussianity, properties of outliers), linear correlations (e.g., autocorrelations, Fourier power spectrum measures), information theoretic/entropy measures (e.g., auto-mutual information, Approximate Entropy, Lempel-Ziv complexity), methods from nonlinear time-series analysis (e.g., correlation dimension, Lyapunov exponent estimates), and model fits (e.g., goodness of fit and parameter values from autoregressive moving average (ARMA) and state space models). A full list of operations developed for this work, including references and descriptions, is in Appendix F.

The output of a set of operations on a set of time series is represented as a data matrix, as we described in Chapter 2 above (cf. Fig. 2.1), where each row represents a time series and each column represents an operation. Elements of the matrix, D_{ij}, contain the result of applying the operation F_j to the time series $\mathbf{x}_i$, so that $D_{ij} = F_j(\mathbf{x}_i)$. Since all operations produce outputs over different ranges, we normalize them using a scaled sigmoidal transformation so that the range of all columns is between 0 to 1, as explained in Sec. 3.3.2. The data matrix allows very different time series and diverse time-series analysis methods to be represented in a unified way that allows a direct and meaningful comparison between them. Each time series is represented as a feature vector (its corresponding row of the data matrix) that contains the outputs of many different operations on it. Similarly, each operation is represented as a feature vector (its corresponding column of the data matrix) that contains its outputs across a set of time series. Distances between time series, and distances between operations, are calculated as distances between their feature vector

representations. The distance metrics used in this work are justified and discussed in Sec. 3.4.1.

3.2 Database structure and computation

In this section we describe computational aspects of the thesis, including the process through which time series and operations were collected, the database framework used to store our libraries of time series and operations, the results of applying each operations to each time series, and the computational structure used to evaluate it.

3.2.1 Collecting data and operations

Central to the analysis performed in this thesis are comprehensive collections of time series and operations, which have been prepared for the first time in this work. Our aims in this task were to try to be as comprehensive and even-handed as possible: collecting as many different types of data and operations as possible given the time constraints.

As described in Appendix E, time series collected in this work include publicly-available recordings from real-world systems, which have mostly been downloaded from publicly-available databases and processed appropriately (including denoising, detrending, interpolating, and breaking time series up into shorter subsegments) using code developed in this work. In addition to these real-world time series, we also generated time series from a range of synthetic systems, including sets of differential equations, discrete recurrence relations, stochastic differential equations, and outputs from various models.

Operations in our library rely to a varying degrees on existing code for time-series analysis; for example, functions in various MATLAB Toolboxes, or publicly-available code obtained from online resources. A detailed description of how this was achieved for operations implemented in this work is in Appendix F. Much creative freedom was allowed in this process, because there is little cost involved in adding novel types of analysis to the library alongside more traditional, and well-motivated approaches. We distinguish between three classes types of operations in terms of our contribution to the time-series analysis literature at the level of the operation library: (i) existing

code that is already in the form of an operation that takes in a time series and outputs a real number, that simply requires a straightforward implementation (e.g., moments of the distribution, or parameters from a model fit), (ii) an existing method that does not obviously fit into an automated framework: we have developed techniques to automate such methods and to extract appropriate summaries of their outputs (e.g., selecting an appropriate preprocessing and model orders for GARCH models, automating scaling results and related dimension estimates from time-delay embeddings), (iii) qualitatively new types of operations were thought up and implemented, both as completely new types of analysis (e.g., stick angle distributions and simulated particle motion trajectory-based operations) and new ways of applying existing analysis techniques (which are numerous, e.g., looking for variability in wavelet coefficients across different model orders, or fitting Gaussian Processes to local subsegments of time series to estimate stationarity).

3.2.2 Database structure

Our database is organized using *mySQL* and contains three main tables: `TimeSeries`, `Operations`, and `Results`. The `TimeSeries` table contains a unique time series identifier (an integer) for each time series, along with metadata, including the filename containing the time-series data, a list of keywords identifying the source and other descriptive characteristics of the time series, and its length (i.e., the number of samples). The `Operations` table contains a unique operation identifier (an integer), for each operation, along with the code to call to run the algorithm, and a list of keywords describing the basic class of the operation (e.g., 'correlation', 'stationarity', 'entropy', 'dimension'). Some operations return multiple outputs – these are incorporated into our framework by first calling these functions, storing the results in a MATLAB 'structure', and then assigning them to the appropriate individual metrics. The `Results` table stores the raw outputs obtained by applying every operation to every time series, and also stores the computation time and a quality code for each entry. The quality code allows us to encapsulate special values: Inf, $-Inf$, NaN outputs, and errors (described in Sec. 3.2.4), and to distinguish uncalculated ($NULL$) entries. Note that for operations that return multiple outputs, the computation time

for each single output is set to that of the entire routine.

3.2.3 Annotating metadata

For some applications, it is important to ensure that the time-series analysis methods used do not depend on trivial properties like the length or location (e.g., mean) of recordings, which could bias the results. This is especially true when comparing very different types of time series, as is done in this thesis, where we wish to focus on the dynamical characteristics of time series and not their length, location, or spread. To address this issue, we labeled operations that are length-dependent, location-dependent, and spread-dependent as keyword metadata in the database, so that an appropriate set of operations can be filtered quickly, given a specific time-series analysis task. These keyword labels were determined automatically by submitting each operation to a dataset of test time series. Location-dependent operations were flagged by applying each operation to a dataset of time series with different offsets in mean; (non-random) operations whose outputs varied strongly across this set were labeled. The spread-dependent test was performed in a similar way using a dataset of time series with different standard deviations. The length-dependent test was based on measuring the mutual information between operation outputs and time-series length across an interdisciplinary dataset of 875 time series (described in Sec. 4.1.2). Operations with these various dependencies were examined and the appropriate keyword: 'locdep', 'spreaddep', or 'lengthdep' was added to the keyword metadata for that operation (in the mySQL database described in Sec. 3.2.2).

3.2.4 Special values

Time-series analysis operations often do not return a real number: when its application is inappropriate (e.g., fitting a positive-only distribution to data that is not positive-only), when dividing by zero (returning $\pm Inf$), or when an error is encountered (e.g., the operation may exhaust the computer system's RAM when applied to a very long time series). Each of these outcomes is given a distinct label in the mySQL database, but all such cases are treated as missing values in data matrices analyzed throughout this thesis, which can then be filtered out from the data matrices prior to

analysis, as required. Although less than 6% of operation outputs stored in the full database are special values, it is important that they are treated appropriately.

For some analyses performed in this thesis, missing values in the data matrix cannot be tolerated. This includes dimensionality reduction techniques such as Principal Components Analysis and Isomap (cf. Sec. 3.4.4), and the training of classifiers. In this case, operations that output any special values on the dataset are removed prior to the analysis. The number of operations remaining after this filtration therefore depends on the dataset and will be noted for all applications throughout the thesis. Other analysis methods can tolerate some missing values in the data matrix, including the calculation of pairwise distances between time series or operations (prior to clustering), for example. Unless otherwise specified, operations that produced more than 20% special-valued outputs were filtered from the data matrix. Missing values are not included in any calculations used for the normalization (cf. Sec. 3.3.2), and are ignored in pairwise distance calculations (cf. Sec. 3.4.1).

We also mention a suite of tools we developed for detecting algorithmic errors, which are inevitable in a database of thousands of operations. Operations that are not deterministic are annotated in the database with the keyword 'stochastic' (e.g., algorithms that take random subsegments from a time series can give different outputs each time they are evaluated). Algorithms that return unexpected errors (e.g., for unusual time series that contain very few unique values, or are very short), have been identified and either routine tests have been added to the algorithm so that a *NaN* is returned for cases in which the algorithm cannot be applied meaningfully, or the code has been manually altered to correct the problematic behavior.

3.2.5 Computational details

The computation of operations on time series in our database was distributed over three quad-core 64-bit Windows machines running Windows XP and *MATLAB 2009b*, a 32-bit Windows machine running Windows XP and *MATLAB 2009b*, one 64-bit 16-core Windows Server machine running *MATLAB 2008b*, and a quad-core 64-bit Windows 7 machine running *MATLAB 2010a*[1]. On each machine, operations were

[1]MATLAB is a product of The MathWorks, Natick, MA.

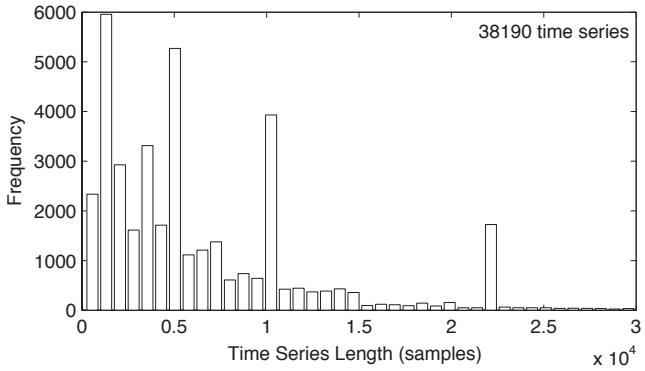

Figure 3.1: **Length distribution of the 38 190 time series in our database.** Peaks are due to many synthetically-generated time series that are 1 000, 5 000, or 10 000 samples long. The peak at $\approx 22\,000$ samples is from a large set of meteorological time-series of this specific length. The majority of time series are shorter than 20 000 samples.

computed in parallel over the available cores. Results were stored on a central mySQL database in which all machines stored their results, as described above in Sec. 3.2.2.

Lengths of time series in the database are mostly in the range 1 000 – 20 000 samples, as shown in Fig. 3.1. The time taken to evaluate all 9 496 operations in the database on time series of different lengths is shown in Fig. 3.2. To create this figure, we used a set of sinusoidal time series corrupted with Gaussian noise (the time series were generated from the model defined by Eq. (5.1) with $\eta = 0.2$) of length 2 000, 4 000, 8 000, 16 000, and 32 000 samples. All the MATLAB-based algorithms were calculated in parallel using the `parfor` function in MATLAB's *Parallel Computing Toolbox*, after which the algorithms derived from the TISEAN package of command-line executables [92]. The total time taken to calculate these 9 496 operations on a single machine is plotted in Fig. 3.2. For time series with less than 10 000 samples, the total calculation time for a single time series is less than 10 mins. Since each calculation involves running a given algorithm on a given time series, all calculations are independent and hence this task is massively parallelizable. As such, calculation times could be decreased further using parallel and distributed computing.

Also, for any given application, not all operations need to be calculated – instead an appropriate reduced subset could be selected given constraints on the total computation time. The calculation time of each algorithm, which is stored in the mySQL database, could be used to select such subsets of operations. For example, for long

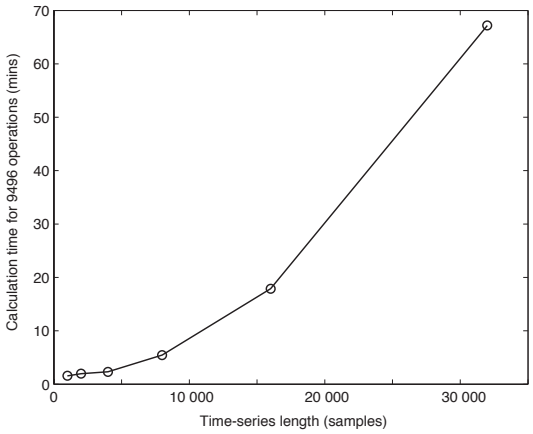

Figure 3.2: **Scaling of calculation time with time-series length**. Here, we use a test set of noisy sinusoidal time series of length 2 000, 4 000, 8 000, 16 000, and 32 000 samples. Most operations are calculated in parallel in MATLAB, after which operations from the TISEAN package are calculated serially in the command line. Calculation times plotted here are the total time taken to evaluate 9 496 algorithms using MATLAB 2010a on a quad-core Windows XP machine.

time series, the slowest algorithms can be ignored to make computation times more manageable.

3.3 Processing

In this section we discuss appropriate transformations that allow very different types of time series and operations to be compared meaningfully: for time series data in 3.3.1, and for operations in Sec. 3.3.2.

3.3.1 Data pre-processing

In this work, we have collected time series produced by many different systems and measured in many different ways, including bursty recordings from astrophysics, non-stationary medical data, noise-free oscillations from differential equations, and positively-skewed, positive-only rainfall time series. A particular challenge of this work involves representing these different types of time series in a way that facilitates a meaningful comparison between them. Applying an appropriate pre-processing is an important part of any application of time-series analysis, and is usually motivated by knowledge of how the data were measured and the questions one wishes to ask of it. For example, an increasing trend in climate data is important to climatolo-

gists interested in the trend, but uninteresting to scientists analyzing patterns in the fluctuations: in this case the low-frequency trend would first be removed. Similarly, some time series may be dominated by, for example, high frequency noise that is not relevant to the process of interest and should be filtered out to reduce this artifact of the measurement process. Pre-processings of this type are therefore motivated by domain-specific knowledge and should be performed prior to the inclusion of a time series in our database.

However, given the dominating effect of, say, a strong periodicity or linear trend in a given time series on the outputs of many time-series analysis algorithms, there is a case for automatically detecting and 'correcting' for such strong overriding trends in a time series. For example, consider a pre-processing procedure that transformed *all* time series to be approximately stationary using an aggressive detrending routine, or one that transformed highly non-Gaussian data (e.g., by applying log or Box-Cox transforms to skewed, positive-only data [93]). Applying automatic transformations of this type would convert all time series to a more comparable form, but doing so without empirical motivation would systematically impose structure on the data that may not suitably reflect the dynamics of the system of interest. Pre-processing transformations applied automatically would therefore compromise the strength of conclusions we can make from the result.

Given both the need to apply some transformation to time series to allow them to be compared meaningfully and also the need for care in applying such transformations without motivation, we reach a compromise in this work. We developed algorithms that automatically check for the presence of strong periodicities or linear trends in time series, which were subsequently analyzed visually and in some cases linearly-detrended or deseasonalized (i.e., with the strongest periodicity filtered out). For all 2 000 time series for which this detrending was performed, both the original time series *and* the transformed time series were included in the database to ensure that the original structure in the time series is still represented in the database, along with an appropriately detrended version.

Given the undesirable dependence of some operations on location and spread, we

use a z-scoring, which is treated as a standard pre-processing for time series:

$$\tilde{x} = \frac{x - \mu_x}{\sigma_x}, \tag{3.1}$$

where the z-scored version, $\tilde{x}$, of the original time series, x, has mean zero and unit variance because the mean of the original time series, μ_x, and its standard deviation, σ_x, have been corrected for. Only 1 335 of the 9 613 operations in the database operate on the raw time series; the remaining 8 279 operate on the z-scored time series.

3.3.2 Operation normalization

One challenge of including very many different time-series operations with different distributions of outputs involves choosing a transformation that allows them to be compared in a meaningful way. For many applications, the distributions of outputs from different operations must be transformed to allow a fair comparison. In particular, when calculating distances between time series feature vectors, the range of outputs of all operations should be the same so that all dimensions are weighted equally for the calculation of distances in the space. There are a number of possible transformations that can be applied to different sets of operation outputs: e.g., (i) z-scoring, (ii) linear rescaling to the [0,1] interval, (iii) nonlinear rescaling to the [0,1] interval, (iv) a transformation individually tailored to the nature of each operation (e.g., p-values already lie in the [0,1] interval and may not need to be rescaled). We find that, due to the diversity of time series included in our database, distributions of operation outputs are often multi-modal, and frequently contain significant outliers. It is therefore inappropriate to employ transformations like the z-score that are highly sensitive to presence of outliers.

We overcome this problem by devising a new form of an outlier-robust sigmoidal transform:

$$\hat{\mathbf{f}} = \left\{1 + \exp\left[-\frac{\mathbf{f} - \text{median}(\mathbf{f})}{1.35 \times \text{iqr}(\mathbf{f})}\right]\right\}^{-1}, \tag{3.2}$$

where $\hat{\mathbf{f}}$ represents the normalized outputs of a given operation across all time series, $\mathbf{f}$, are the raw outputs, median($\mathbf{f}$) is the sample median of $\mathbf{f}$, and iqr($\mathbf{f}$) is its interquartile range. The constant 1.35 has been chosen so that for a Gaussian-distributed $\mathbf{f}$, it is

equivalent to the standard *logistic sigmoid function* [9]:

$$\hat{\mathbf{f}} = \left\{ 1 + \exp\left[-\frac{\mathbf{f} - \mu_{\mathbf{f}}}{\sigma_{\mathbf{f}}} \right] \right\}^{-1}, \tag{3.3}$$

which instead uses the mean, $\mu_{\mathbf{f}}$, in place of median, and the standard deviation, $\sigma_{\mathbf{f}}$, in place of interquartile range (and the factor 1.35). After applying the nonlinear transformation, Eq. (3.2), the result is then linearly rescaled to the unit interval so that every operation has the same range of normalized outputs (from 0 to 1). And, because the median and interquartile range are used instead of the mean and standard deviation, the transformation is much less sensitive to outliers or other peculiarities of the distribution of $\mathbf{f}$. Note that other transformations to the unit interval are also possible, as have been used in other work [94].

A problem with this transformation, Eq. (3.2), occurs for operations that return many outputs with the same value and therefore have an interquartile range of zero, yielding a singularity in the calculation. One approach is to treat such operations separately, and apply an ordinary logistic transform, Eq. (3.3), since in this case the 'outliers' are important. However, for simplicity, we ignore operations with an interquartile range of zero throughout this thesis, by removing them from normalized data matrices. Thus, every normalized data matrix analyzed throughout this thesis contains columns of operations that have each undergone the same transform: Eq. (3.2). The main results presented in this thesis should be robust to different sensible normalizing transformations and, given the range of possibilities, we have picked the most appropriate one for the distributions of operation outputs we observe.

This normalization for operations are used prior to performing Principal Components analysis or calculating distances between time series throughout this thesis. For this reason, operations with a zero interquartile range will be removed from such analyses, for reasons described above. Many other techniques used to analyze data matrices, however, do not require any normalization. These include algorithms for estimating mutual information (cf. Sec. 3.4.1), and for building linear classifiers using feature selection (cf. Sec. 3.4.5).

3.4 Analysis methods

In this section we justify our choice of a range of different methods for analyzing data matrices, including the selection of distance metrics to quantify the similarity between two time series or two operations (Sec. 3.4.1), clustering methods (Sec. 3.4.2), classification methods (Sec. 3.4.3), dimensionality reduction methods (Sec. 3.4.4), and feature selection methods (Sec. 3.4.5). We also describe a statistical multiple hypothesis framework for evaluating the significance of results involving the comparison of a large number of individual methods in Sec. 3.4.6, and finally outline our method for producing network visualizations in Sec. 3.4.7.

3.4.1 Distance metric

In this section, we explain our choice of metrics used to measure the distance between feature vectors for time series (rows of the data matrix) and for operations (columns of the data matrix). Time series should be judged as similar when many operations produce similar outputs for them; our distance metric should reflect this basic property. We choose the Euclidean distance metric,

$$d_{xy} = \sqrt{\sum_i (x_i - y_i)^2}, \tag{3.4}$$

to quantify the dissimilarity, d_{xy}, between time-series feature vectors $\mathbf{x}$ and $\mathbf{y}$. Other metrics that quantify cumulative differences in value across features, including the 'cityblock' (or 'Manhattan') distance, are other possible choices that should produce broadly similar results as the Euclidean distance metric.

Unlike time series, operations should be judged as similar not when their outputs are similar in value, but when they are highly correlated. For example, two operations that output values that are simply offset by some constant value are really behaving in the same way and should be judged as similar. Also, an operation that increases across a dataset and another that decreases across the same dataset are also showing similar behavior. It is clear that translational or scaling transformations should not affect the distance calculated between two operations. We consider measures of linear and nonlinear correlations in this thesis that have the desired properties described

above. First, to quantify linear correlations between pairs of operations, we use an absolute linear correlation distance metric that judges operations to be similar that have outputs that are either highly correlated or highly anti-correlated across a time-series dataset. The measure is defined as

$$d_{xy} = 1 - \left| \frac{E[(\mathbf{x} - \mu_{\mathbf{x}})(\mathbf{y} - \mu_{\mathbf{y}})]}{\sigma_{\mathbf{x}}\sigma_{\mathbf{y}}} \right|, \tag{3.5}$$

where μ and σ represent the sample mean and standard deviation, respectively, of operation feature vectors $\mathbf{x}$ and $\mathbf{y}$, and the expectation operator, $E[]$, represents taking the mean. The absolute value in Eq. (3.5) ensures that anti-correlated operations are judged as similar (in the same way as positively correlated operations).

For calculating potentially nonlinear correlations between two operations, we use a measure based on *mutual information*. The mutual information, $I(X;Y)$, between random variables X and Y is the reduction in the uncertainty of X due to the knowledge of Y [95], and can therefore be used to estimate the strength of any dependent relationship between X and Y. The mutual information is used here as the basis for a distance metric and is defined as:

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X), \tag{3.6}$$

where $H(\cdot)$ represents the entropy, $H(X) = -\sum_{x \in \mathcal{A}_X} p(x) \log p(x)$, for an alphabet $\mathcal{A}_X$ for X, and $H(X|Y)$ is the conditional entropy of X given Y [95]. Mutual information has been used previously as the basis for a distance metric [96], and has been successfully applied to gene-expression clustering problems, for example [97]. In this work, pairwise mutual informations are estimated using a histogram-based method with ten equiprobable bins (i.e., with histogram partitions chosen by appropriate quantiles of the marginal distributions). Given the size of our operation library, more complex methods of estimating mutual information, such as k-nearest neighbors and kernel-based methods [98], have not been investigated, as they require an even greater computational expense.

Converting mutual informations to distances requires a choice of normalization, for which there are a number of alternatives [95, 99, 100]. Throughout this thesis we

use

$$d_{\text{norm}}(X,Y) = 1 - \frac{I(X;Y)}{H(X,Y)}, \tag{3.7}$$

which satisfies the necessary axioms of a distance metric, and $d_{\text{norm}}(X,Y) \leq 1$ [99]. This metric allows us to detect possible nonlinear pairwise relationships between pairs of operations.

Note that in cases where missing values occur in the data matrix, distances between pairs of feature vectors are calculated by ignoring those elements that contain missing values. No correction is necessary for correlation-based distances that do not scale with the number of samples, but for Euclidean distances that scale linearly with the number of points, an appropriate multiplicative correction was included to compensate for the missing values. For example, if 4 of the 100 elements in a feature vector are special-valued for either of the two objects, the distance is calculated using the remaining 96 elements, and the result is multiplied by 100/96.

3.4.2 Clustering methods

Clustering aims to group a collection of objects into subsets, or 'clusters', such that objects within each cluster are more similar to each other than objects assigned to different clusters [82]. In this thesis, we use clustering methods to find structure in time-series datasets and in collections of operations. Many different clustering algorithms have been devised in previous work [82, 101, 102]. Throughout this thesis, we take the philosophy of choosing simple and transparent methods so as to focus on the highly comparative nature of our analysis, rather than the complexity of our underlying methodologies. For clustering, we implement two basic methods: hierarchical or agglomerative linkage clustering, and k-medoids clustering. These two methods are convenient to compare because both take as input a pairwise distance matrix, containing distances between all pairs of objects—either time series or operations.

Hierarchical linkage clustering is implemented using the `linkage` function from MATLAB's *Statistics Toolbox*. In hierarchical clustering, clusters are formed in an agglomerative scheme that iteratively groups objects, starting with the pair that is most similar, until all objects are included in a single group [102]. This procedure requires a choice of a linkage method: three common choices are 'single' (which

defines the distance between two clusters as the distance between the two closest elements in the two clusters), 'average' (which defines the distance as the average distance between elements in the clusters), and 'complete' (which defines the distance as the maximum distance between elements in the clusters) [101, 102]. In this work, hierarchical clustering is done using the 'average' linkage method unless otherwise specified. In cases where fewer than 1 000 objects are clustered, the computationally-expensive `optimalleaforder` code from MATLAB's *Bioinformatics Toolbox* is used to order objects in a way that minimizes the sum of distances between adjacent objects.

A k-medoids clustering algorithm was also implemented as described in Hastie et al. [82]. We chose k-medoids clustering instead of the more commonly-known k-means clustering because, like linkage clustering, it takes only a pairwise distance matrix as input. This is convenient because the bulk of the computational expense is involved in computing these pairwise distances, after which both linkage and k-medoids clustering can be performed straightforwardly. The algorithm initially assigns random data points as cluster centers, and iteratively assigns data points to their nearest cluster centre. New cluster centers are then recomputed and the procedure is repeated until convergence is achieved [82]. Since there are very many local optimums in this combinatorial problem, we repeat the procedure 2 000 times and return the clustering that minimizes the sum of distances of points to their the nearest cluster centre. Note that an analogous method is the partitioning about medoids (PAM) algorithm [103].

3.4.3 Classification methods

Classification problems are treated throughout this thesis, in which classification rules (or classifiers) are learned that allow different labeled classes of time series to be distinguished. Many different types of classifiers have been developed in the field of machine learning [9, 82]. Throughout this thesis, we use a simple linear discriminant for classification, as implemented using the `classify` function from MATLAB's *Statistics Toolbox*, which provides a highly intuitive and interpretable result: a linear partition of a feature space. For the case of one-dimensional spaces, the linear clas-

sification boundary is simply a threshold in this space; for pairs of features, linear classification boundaries are straight lines; for three features, classification boundaries are planes; and in higher dimensions they are hyperplanes. In the case of multi-class classification problems with more than two classes, linear classification boundaries are learned between all pairs of classes. Then to classify a new time series, classification rules between all pairs of classes are evaluated, and the time series is then assigned to the class with the most votes from this procedure. Note that we also investigated k-nearest neighbors (k-NN) [9] and support vector machine (SVM) [82] classifiers. These alternative methods were explored for comparison, but results were usually similar to those obtained using linear classifiers, which are quicker to evaluate and simpler to interpret. Linear classifiers are used throughout the thesis, except in Chapter 6, where k-NN classifiers are also implemented to compare to the standard k-NN classifiers used in temporal data mining.

When evaluating the performance of classifiers throughout this thesis, we always specify how this is done: either in-sample, out-of-sample, or using k-fold cross-validation [82]. Stratified k-fold cross-validation involves separating a dataset into training and testing portions by dividing it into k equal partitions, where each partition retains the approximate class proportions of the full dataset. The cross-validation estimate for the out-of-sample classification rate is achieved by training the classifier on $k-1$ of the data partitions and testing on the remaining partition, and repeating until all k partitions of the data have been tested on. The procedure can be repeated using different partitions to reduce the variance on the resulting error estimate [82]. Unless specified otherwise, 10-fold cross-validation is performed throughout this thesis, with errors in the form mean $\pm$ standard deviation.

3.4.4 Dimensionality reduction

The problem of dimensionality reduction involves finding meaningful low-dimensional representations of high-dimensional datasets [83]. Principal Components Analysis (PCA) is used throughout this thesis, which returns an orthogonal set of features that are linear combinations of the original feature set [9, 82] and is implemented using `princomp` in the MATLAB *Statistics Toolbox*. By ordering the Principal Com-

ponents (PCs) by their eigenvalues, datasets are projected into the space of the leading PCs to provide a lower-dimensional representation of them. The result allows us to investigate the dimensionality of our library of operations in Chapter 4, and to find appropriate low-dimensional representations of particular time-series datasets in Chapter 5. Throughout the thesis, data matrices are normalized (cf. Sec. 3.3.2) and operations with any special-valued outputs (cf. Sec. 3.2.4) are removed prior to performing PCA.

While reduced dimensions obtained from PCA are restricted to being linear combinations of features; in Chapter 7 we investigate the nonlinear dimensionality reduction method, *Isomap* [83]. Isomap is a method for extracting the intrinsic nonlinear degrees of freedom from a high-dimensional dataset [83]. Three main steps are involved in the method: 1. A weighted network is constructed with data points represented as nodes with links connecting nearest neighbors in the input space, 2. Geodesic distances between all pairs of data points are approximated by shortest path distances in the network, and 3. Multidimensional scaling (MDS) is applied to the set of shortest path distances by embedding the result in Euclidean spaces of different dimensions. The result is a set of reduced Isomap dimensions, and a residual variance measure allows us to quantify how much of the structure in the dataset is captured by these dimensions. The method has been used to identify the underlying degrees of freedom in datasets of various images, including a dataset of photos of tilted heads, where Isomap correctly identified the three free variables: left-right pose angle, up-down pose angle, and lighting direction [83]. MATLAB code for applying Isomap is publicly available[2].

3.4.5 Feature selection

Throughout this thesis, we often aim to achieve dimensionality reduction in a sparse way, such that the reduced dimensions correspond to a small subset of the original variables, while retaining much of the information present in the full set. This procedure is often called 'feature selection', and is distinguished from 'feature extraction', for which features can be transformations and/or combinations of the original vari-

[2]The Isomap algorithm is available at `http://isomap.stanford.edu/`

ables [104, 105]. Feature selection is typically performed in a supervised framework for which labels have been assigned to all data objects. Feature selection is used in this thesis to learn time-series classifiers that combine multiple operations.

Many different feature selection methods have been developed [104–106], including the *lasso* [107], *elastic net* [108], and *recursive feature elimination* [109]. In this thesis, we focus on a simple, transparent, and easily-interpretable feature selection method called *greedy forward feature selection* [82]. This method involves iteratively selecting the feature that maximizes some quality function: in this work, we maximize the classification accuracy of a linear classifier. The algorithm is as follows: (i) The classification rates of all individual operations are calculated and the operation with the highest classification rate is selected as the first member of the reduced set. (ii) The classification rates of all operations in combination with this first selected operation are then calculated, and the operation that yields the highest classification rate is added to the reduced set. (iii) The procedure is repeated, selecting the operation that most improves the classification rate at each iteration. For iterations at which multiple features produce equally good classification rates, one of them is selected at random. A heuristic for terminating the process is discussed in Chapter 6. A major drawback of this method is that it always selects the single best feature at each iteration such that cases where the combination of two individually poor performers can in combination achieve high performance cannot be realized, cf. Guyon et al. [106]. Note that recursive feature elimination [109] (using a linear SVM through the SPIDER Machine Learning Toolbox for MATLAB[3]), and the popular *elastic net* [108] (which was implemented in MATLAB using publicly-available code[4]) were both experimented with for comparison, but both methods generally yielded similar classification accuracies to the greedy forward selection method, which is used throughout this thesis.

Having selected a set of operations and then learned a classifier using their outputs, it is important to evaluate the performance of the resulting classifier using an

[3]The toolbox is available under GNU General Public License at `http://www.kyb.tuebingen.mpg.de/bs/people/spider/`

[4]Code available from `http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=3897`

unseen test dataset. This is especially important in this work, where we often have more operations than time series: the $p \gg n$ problem [110, 111]. Given the strong potential for bias in the in-sample classification accuracy [112, 113], we are careful to properly separate data for training (selecting features and learning a classifier) and testing (evaluating the classifier). We follow the fundamental methodology outlined in Smialowski et al. [113], a schematic of which is illustrated in Fig. 3.3. The general procedure is as follows: (i) the data is split into training and testing portions, (ii) a feature selection algorithm is run on the training set, (iii) a classifier is trained on the training set in the reduced feature space, (iv) this classifier is tested on the test data in the same reduced space of features chosen from step (ii). By repeating this method using different splits of the data, a distribution of test set classification errors is accumulated. A parameter of this procedure is the proportion of data to use for training: we use training sets containing 80% of the data, and test sets containing the remaining 20% of data. We repeat the process described above 200 times with random data partitions in these proportions. Methods that use all the data for feature selection (e.g., by cross-validation), and then reuse the data for model building (e.g., again by cross-validation) suffer an in-sample bias due to features being chosen that over-fit the dataset [112]. A useful check to ensure that over-fitting is being avoided involves verifying that the classification error distribution obtained by randomly permuting class labels is consistent with chance (i.e., classification accuracy reflects class membership proportions).

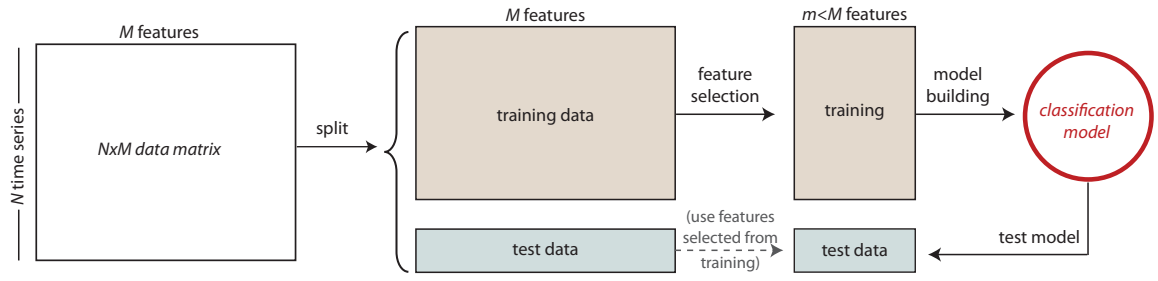

Figure 3.3: **Partitioning data into training and test sets for feature selection.** Within this general framework, three choices must be made: (i) how to separate data into training/test sets, (ii) the feature selection method, (iii) the classification model. Here we use a hold-out procedure with 20% testing, greedy forward feature selection, and a linear classifier, respectively.

Applying the procedure described above, it is important to note that *different sets of features* are selected at each iteration, such that the classification error distribution is made up of errors obtained using different classifiers learnt on different splits of the data. In this report, quoted error distributions are always obtained from 200 repeats of the procedure described above and illustrated in Fig. 3.3. Particular classifiers can be obtained from a single run of this process and hence represent a single sample from the full error distribution obtained through multiple iterations using different data partitions.

Performing feature selection using the full library of operations, which contains many groups of similar operations within it, can be problematic for feature selection. The main problems with this approach are as follows: (i) interpreting the result of feature selection algorithms is made more challenging because features selected at different iterations of the procedure are often simply slight variants of one another that nonetheless exhibit similar behavior on the data, and (ii) running feature selection algorithms using a large number of features is computationally intensive. Attempts to overcome this problem by first clustering the operations on the data as in Sec. 4.1.2 to form a less-correlated reduced set risk over-fitting this selection to the properties of the full dataset, even though the data labels are not used. For simplicity, we use the full set of operations for feature selection throughout this thesis. Correlations between operations can then be used to group them after feature selection to aid interpretation of the results, as is done for Parkinsonian speech and heart beat interval datasets in Secs. 5.2.5 and 5.2.6, respectively.

3.4.6 Multiple hypothesis testing

Given the large number of operations in our database, it is important to address statistical concerns involving multiple hypothesis testing [82] in many of the highly comparative analyses performed throughout this thesis. For example, consider the task of searching for operations in our database that effectively separate two classes of time series—a task that is demonstrated for a variety of case studies in Sec. 5.2. In this case, the performance of many operations are independently evaluated and only successes are reported; but what is the probability of finding such a successful oper-

ation, given that we are searching across a database of approximately 9 000 different operations? We form a null hypothesis to address this question: that operations have no ability to distinguish the class labels. We can then compare this null distribution to that obtained using the true labels of the data objects, whence the statistical significance of the results can be evaluated.

Null distributions for the test statistic are formed by the following procedure. First, the labels (values in the case of regression, Sec. 5.1, or class assignments in the case of classification, Sec. 5.2) are shuffled and assigned to the time series at random. Then, using these randomly permuted labels, an appropriate test statistic (a correlation coefficient in the case of regression, or a classification rate in the case of classification) is calculated for all n_{op} operations. The process is repeated K times (i.e., for K different permutations of the labels), and we accumulate a set of Kn_{op} values of the test statistic. A histogram is constructed using these values of the test statistic, where the counts in each bin is divided by K to produce a histogram with a total of n_{op} counts that can be compared to the distribution obtained using the 'true' assignments of the time series.

In order to quantify the significance of individual operations, we follow the treatment in Sec. 18.7 of Hastie et al. [82]. The test statistics obtained from all operations using permuted time-series labels are pooled to calculate p-values according to:

$$p_j = \frac{1}{MK} \sum_{j'=1}^{M} \sum_{k=1}^{K} I(|t_{j'}^k| > |t_j|), \tag{3.8}$$

where the test statistic for the known labeling is t_j, and across K different permutations of the data labels, $t_{j'}^k$ [82]. The indicator function, I, takes a value of 1 when $|t_{j'}^k| > |t_j|$, and is zero otherwise. The sum is over M operations with no special-valued outputs across the dataset ($M \approx 7\,000$–$8\,000$ in our case, depending on the dataset).

We consider two different methods for quantifying the significance of test statistics. First, the *Bonferroni* method makes each individual test more stringent by defining a null hypothesis, H_{0j}, that operation j shows no ability to distinguish the known classes of time series. This null hypothesis is rejected if $p_j < \alpha/M$, ensuring that the *family-wise error rate* (FWER) is at most α, where α is the type-I error (i.e., the

probability of falsely rejecting H_{0j}) [82]. The FWER is defined as the probability of at least one false rejection of H_{0j}. The Bonferroni method thus allows us to set a limit on this FWER, but in so doing it yields a conservative estimate of the significance of operations [82]. For this reason we also consider the *false discovery rate* (FDR), which is the expected proportion of operations that are incorrectly called significant among those operations that are called significant. For a given FDR, α, we place the p-values in ascending order, and then determine a threshold on them according to

$$L = \max\left\{j : p_{(j)} < \alpha\frac{j}{M}\right\}. \tag{3.9}$$

All null hypotheses H_{0j} for which $p_j \leq p_{(L)}$ are rejected. This is called the *Benjamini-Hochberg* method [82] and $p_{(L)}$ is the Benjamini-Hochberg rejection threshold.

These ideas are demonstrated in Fig. 3.4 using two analysis problems studied in this thesis. First, we focus on the task of selecting operations that distinguish sequences of random numbers from periodic signals, as studied in detail in Sec. 5.2.1. In this classification task, illustrated in Figs. 3.4(a) and (c), we wish to determine which of the $M = 6\,863$ operations with no special-valued outputs for the random/noise dataset classify the 105 time series in each group at a rate greater than could be expected by chance. Our test statistic in this case is the in-sample linear classification accuracy, and our null hypothesis, H_{0j}, is that the outputs of operation j exhibit no ability to classify the two classes of time series using a simple linear classification threshold (i.e., with a classification accuracy consistent with chance, 50%). Histograms of classification rates for the 6 863 features and the permuted versions are plotted in Fig. 3.4(a), in red and blue, respectively. The permuted (or shuffled) distribution represents an average over $K = 200$ random permutations of each of the operations. The mean in-sample classification error of permuted operations is slightly lower than chance: approximately 48%, due to in-sample over-fitting. Visually it is clear that most of the M operations are picking out the basic differences in dynamical properties between uncorrelated random number sequences and highly-autocorrelated periodic signals. However, the multiple hypothesis testing framework described above allows us to quantify this using several thresholds shown in Fig. 3.4(a). The Bonferroni threshold with $\alpha = 0.05$, plotted green in Fig. 3.4(a), is the strictest, for

which 5 468 operations are judged to be statistically significant in this example. False discovery rate thresholds for $\alpha = 0.001$, 0.01, and 0.05 are less restrictive, and are also plotted in the figure. The Benjamini-Hochberg method, Eq. (3.9), for multiple hypothesis testing is illustrated in Fig. 3.4(c) for $\alpha = 0.01$.

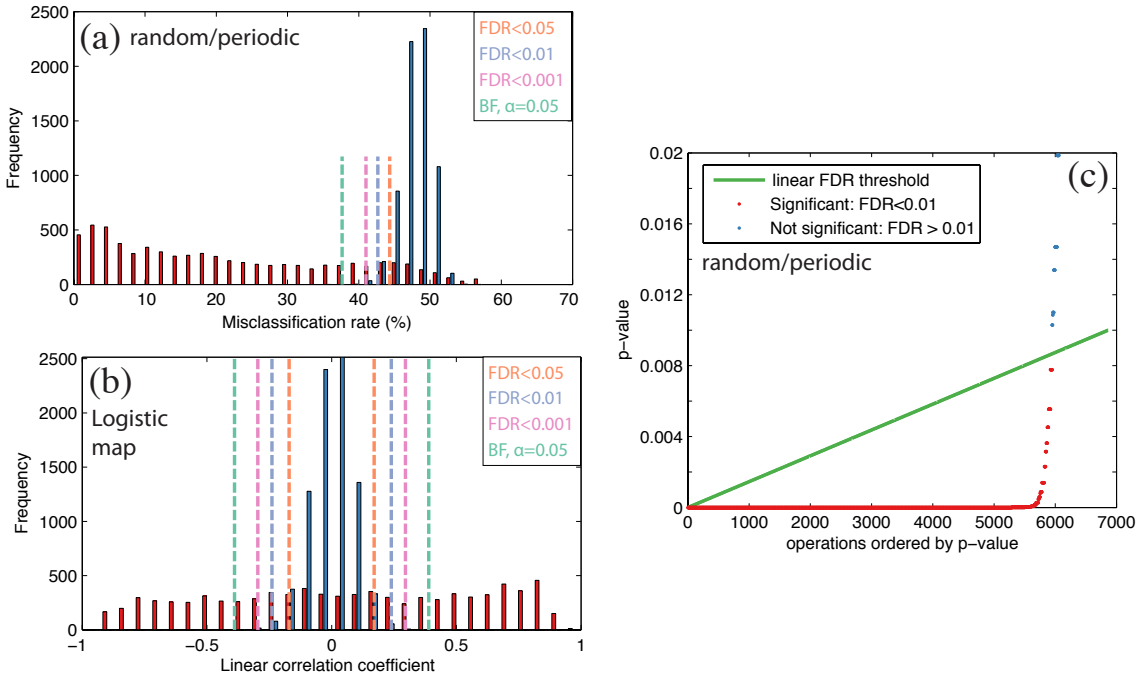

Figure 3.4: **Evaluating the statistical significance of results using multiple hypothesis testing**. This is necessary due to the large number of operations that we often test across in this thesis. We demonstrate the method on two datasets: one containing 105 examples of uncorrelated sequences of random numbers (studied in Sec. 5.2.1), and another containing 159 Logistic Map time series with a known Lyapunov exponent, λ (studied in Sec. 5.1.2). In plots (a) and (b), histograms of the test statistic for the set of operations is plotted in red, and that for permuted versions of the dataset in blue. Thresholds are based on the Bonferroni method with $\alpha = 0.05$ (green) and for different false discovery rates (FDR): 0.05 (orange), 0.01 (purple), and 0.001 (pink). In (a), operations with an in-sample linear misclassification rate less than these thresholds are judged significant. In (b), operations with a linear correlation coefficient at least as extreme as these thresholds are judged significant. The Benjamini-Hochberg method for determining significant p-values is illustrated in (c) for the dataset of random sequences and periodic signals. The linear significance threshold, defined by Eq. (3.9), is plotted in green, such that significant operations lie below this line, and are colored red.

As a second example, we consider the task of selecting operations whose outputs correlate with the Lyapunov exponent of 159 Logistic Map series, a problem analyzed in detail in Sec. 5.1.2. For this regression task, shown in Fig. 3.4(b), our test statistic is the linear correlation coefficient between the outputs of each of the $M = 8\,417$

operations (with no special-valued outputs on the dataset) and the known Lyapunov exponents, λ, assigned to each of the time series. In this case, the null hypothesis, H_{0j}, is that operation j exhibits no linear correlation with λ (i.e., a linear correlation coefficient $R = 0$). The thresholds for judging significance using the Bonferroni and the Benjamini-Hochberg methods are plotted in Fig. 3.4(b). In this case, the linear correlation coefficient can be positive or negative; thresholds are assumed to be equal on both sides of the distribution and we plot both in the figure. Even using the conservative Bonferroni method (with $\alpha = 0.05$), we find 4 593 operations with significant linear correlation coefficients for this dataset.

The methods demonstrated in this section show that, even though we are independently testing large numbers of operations, we are able to assess the significance of our results by comparing to permuted versions of operation outputs. Note that, because many operations produce highly-correlated outputs to one another, we could reduce the number of operations that we test according to methods demonstrated in Sec. 4.1.2. In this way, the behavior of operations in the database is represented using fewer operations, thereby reducing the number of operations that are independently tested. In this thesis, however, we always work with the full sets of operations for simplicity.

One may argue that the use of such a large number of operations is a drawback of our approach. However, as discussed in the Introduction, Chapter 1, reports in time-series analysis often involve hidden multiple hypothesis testing in the form of exploratory pre-processing, model-fitting, and analysis methods, of which many different options are trialled with only successes reported. Because this exploratory process is so opaque, it is difficult to estimate and account for the resulting bias. By contrast, our framework is transparent in explicitly testing a large number of methods for time-series analysis, so that multiple hypothesis testing can be accounted for.

We also mention connections between these concerns and the statistician John Tukey's notion of 'uncomfortable science', in which exploratory data analysis is performed on a finite sample of data such that an independent test, or confirmatory data analysis, is not possible [69]. Given finite time-series datasets, typically only successful approaches for the data at hand are reported in the scientific literature, despite

the fact that the same data has been used for exploratory data analysis. Empirical biases in this process occur in the time-series analysis literature, where there is no constraint on the set of methods that can be applied. In this thesis, by assembling and comparing many different methods systematically, we are able to make this process transparent, and thereby address the statistical issues appropriately.

3.4.7 Network visualization

Some plots in this work use a network representation to visualize a pairwise similarity matrix. Unless specified otherwise, networks are visualized using the Kamada-Kawai layout algorithm [114] on the inverse of the matrix of distances (between either operations or time series objects), which is solved using steepest gradient descent. Note that in some cases, an alternative spring-based network visualization method is used, as developed by Smith [115].

Chapter 4

The empirical structure of time-series data and their analysis methods

In this chapter we investigate the empirical structure of interdisciplinary collections of time-series data, and methods for time-series analysis. Time series are represented by the outputs of various analysis methods, and methods are represented by their behavior across the data. The chapter is divided into two parts: first, an analysis of time-series analysis methods is performed, in Sec. 4.1; a reduced set of operations formed in this section is then used to structure a large and diverse library of time-series data, in Sec. 4.2. We find that meaningful groups of operations and time series can be formed by clustering, that near neighbors of a given method or a piece of time-series data can be retrieved from the database to provide a scientific context for it, and that dimensionality reduction techniques can be used to reduce this large collection down to smaller, representative sets that efficiently summarize the behaviors present in the full library. The results offer insights into an interdisciplinary field.

4.1 The empirical structure of time-series analysis methods

In this section we investigate structure in a diverse set of approximately 9 000 operations for time-series analysis. Throughout the section, we represent operations using feature vectors that contain their outputs on a representative interdisciplinary set of 875 time series, which is described in Sec. 4.1.1. In Sec. 4.1.2, we show how reduced

sets of operations that approximate the types of behavior represented in the full database can be constructed using dimensionality reduction techniques. Searching for near neighbors of a target operation can be used to connect operations developed in different areas of science automatically, as described in Sec. 4.1.3. In Sec. 4.1.4, we present an example of how measuring correlations between sets of operations can give insights into the properties of a time-series dataset. Finally, in Sec. 4.1.5, we show how an intuition for the behavior of any operation can be gained by plotting an annotated distribution of its outputs. Broad implications of the results are discussed in Sec. 4.1.6.

4.1.1 An interdisciplinary set of time series

In this section, we evaluate the behavior of time-series analysis operations by analyzing their outputs when applied to a range of different time series. For this task, we constructed a representative set of 875 time series containing recordings measured from a variety of real-world systems, and time series generated from time-series models. By creating this controlled set of time series rather than using the full time-series database, we avoid over-representing particular classes of time series that happen to be more numerous in the database and also reduce the computation cost involved in the analysis. For example, although thousands of different rainfall time series are contained in the database, we do not want our analysis to be strongly biased towards their bursty, positive-only properties; rather, our analysis should reflect the diversity of time series studied in science as fairly as possible. Although there is no optimal way of selecting such a set of interdisciplinary time series, we have tried to encompass as many of the main types of time-series data as possible from the following classes: **synthetic time series**: random number sequences, correlated noise, periodic processes, stochastic processes, flows (i.e., sets of differential equations), and maps (i.e., iterative equations); **real-world time series**: *medical*: electrocardiograms (ECGs), electroencephalograms (EEGs), gait, heart-beat intervals, tremor, event-related potentials (ERPs), electrooculograms (EOGs), *meteorological*: rainfall, humidity, air temperature, air pressure, cloud height, soil temperature, visibility, wind speed, river discharge, river flow, dew point, *financial*: log returns, opening prices, high-low, vol-

ume, exchange rates, *astronomical*: geomagnetic, cosmic rays, ionosphere, magnetic field, *sound*: speech, music, sound effects, animal sounds, *others*: seismology, and (the length of consecutive sentences in) texts of literature. Selecting time series from these categories at random from the database allowed us to form a data matrix with 875 time series and 8 651 operations (those with less than 20% special-valued outputs on this dataset), as shown in Fig. 2.1C.

These 875 time series are produced by different generative processes, sampled at different rates, measured in different ways, and recorded over different time spans. Correlations between operation outputs that persist across such a diverse time-series dataset can therefore be expected to estimate similar time-series properties in similar ways.

4.1.2 Selecting a reduced set of operations

A time series, $\mathbf{x} = (x_1, x_2, ..., x_N)$, could be any sequence of N real numbers. But do time series from the systems studied in science represent more constrained realizations of this sequential object, $\mathbf{x}$? This question is often addressed by analyzing patterns in the measurements, $\mathbf{x}$, but in this section we instead consider the structure of *properties* extracted from them. We analyze the space spanned by operations and populated by time series. Low-dimensional structure in this space implies the existence of a reduced set of operations that provide a powerful summary of these time series, approximating the information contained in the full set of operations. Indeed, we do find redundancy in the space of operations applied to the diverse time-series dataset described in Sec. 4.1.1 above. Exploiting this redundancy allows us to construct reduced sets of operations that encapsulate the dominant behavior of methods for scientific time-series analysis. In so doing, we are able to structure and organize many of the myriad different methods that have been proposed throughout the highly diverse and interdisciplinary development of the time-series analysis literature.

In order to distill a reduced set of time-series analysis operations, we cluster the full set of operations (cf. Sec. 3.4.4) illustrated in the data matrix plotted in Fig. 2.1. We consider both hierarchical and k-medoids clustering (cf. Sec. 3.4.2). Both of these methods require only a pairwise distance matrix as input, which we compute

between all 8 651 operations using the absolute linear correlation distance metric (cf. Sec. 3.4.1). Dimensionality reduction is achieved by first clustering the operations to a given number of clusters, and then a reduced set of operations is selected as those that are closest to each cluster center.

Given any reduced set of operations, we can calculate a measure of the *residual variance*, V_{resid}, in the data matrix, which we define as in Tenenbaum et al. [83]:

$$V_{\text{resid}} = 1 - R^2(S, S_{\text{red}}), \tag{4.1}$$

where S is the vector of all pairwise distances (between time series) in the full feature space (of operations), S_{red} is that in the reduced feature space, and R represents the linear correlation coefficient. The residual variance is 0 when distances (between time series) in the reduced space are perfectly linearly correlated with those in the full space, and 1 when there is no correlation. Successful dimensionality reduction will mostly preserve distances from the full feature space in the reduced space and so return a residual variance close to zero. Note that the aim of our task is not necessarily to minimize the residual variance, rather, it is to reduce redundancy in the data matrix. For example, consider a set of ten slight variants of lag-1 autocorrelation-based operations and one stationarity operation: the dominant behavior of this set would be well approximated by just one of the autocorrelation-based operations (low residual variance) simply because lag-1 autocorrelation-based operations were over-represented in the initial set. Nevertheless, the residual variance measure provides an indicator of how well the set of behaviors of operations contained in our interdisciplinary database are encapsulated by reduced sets.

In Fig. 4.1(a), we present results for this cluster-based dimensionality reduction of 8 651 operations, using both hierarchical and k-medoids clustering on absolute linear correlation distances between operations. For a given number of features, clustering using the linkage method gives a much higher residual variance, and thus represents a less accurate representation of pairwise distances between time series in the original space. This can be understood in terms of the mechanism by which hierarchical clustering operates on a set that includes many highly-correlated groups of operations. Hierarchical linkage clustering, by progressively grouping similar operations, creates a

set of clusters that each displays a very different behavior. In so doing, linkage clustering emphasizes atypical operations—unusual ways of summarizing signals. Therefore, reduced sets of operations formed by hierarchical clustering will reduce redundancy by emphasizing unusual operations and hence require more clusters to reproduce the dominant behavior of the full set. In contrast, methods like k-medoids clustering (and the closely-related k-means clustering), place center clusters at densely-populated regions of the space, so as to minimize the sum of distances of all points to their nearest cluster centre. Since each point in the space (populated by operations) contributes to the sum of distances to be minimized by k-medoids, cluster centers occur near dense regions in this space—corresponding to sets of operations that perform broadly similar functions on empirical time series. In this way, k-medoids clustering can more accurately represent the behavior of the full set of operations using fewer operations. Note that similar residual variance results to that shown in Fig. 4.1(a) were obtained using a mutual information distance between operations instead of the absolute linear correlation distance (cf. Sec. 3.4.1).

In Fig. 4.1(b), we show the results using Principal Components Analysis (PCA), which reduces the dimensionality of the set of operations using feature extraction rather than feature selection, forming an orthogonal projection of the data onto a lower dimensional linear space such that the variance of the projected data is maximized [9, 116] (cf. Sec. 3.4.4). Since PCA involves performing an eigendecomposition (on the covariance matrix), it cannot tolerate any missing values in the data matrix such that residual variances plotted in Fig. 4.1(b) were calculated with reference to the 6 132 operations with no special-valued outputs on this dataset. As such the results are only approximately comparable to those in Fig. 4.1(a), where a larger set of 8 651 operations is considered. We find that the residual variance using the leading five Principal Components (PCs) is less than 0.1, implying that differences between these diverse scientific time series, as quantified using a set of 6 132 operations, can be approximately reproduced using just five dimensions, each of which is a linear combination of the outputs from all 6 132 original operations. Note that this residual variance measure, Eq. (4.1), returns values lower than would be expected from the proportion of variance explained by the PCs obtained by summing eigenvalues, which

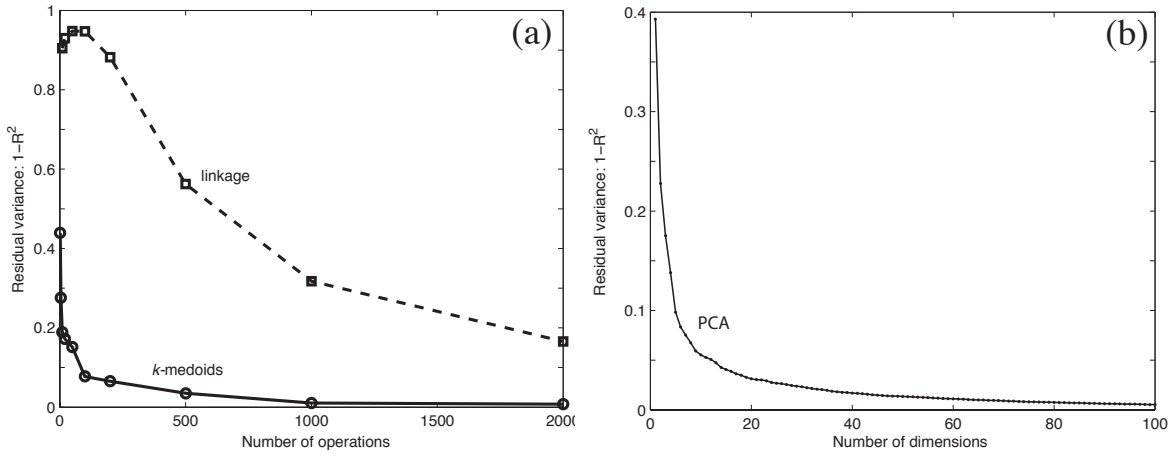

Figure 4.1: **Residual variance in reduced sets of operations**. (a) The residual variance, Eq. (4.1), in a data matrix containing a representative set of 875 time series from different real-world and synthetic systems as a function of the number of operations retained (from an original set of the 8 651 operations with less than 20% special-valued outputs). Reduced sets of operations are formed by applying hierarchical linkage clustering and k-medoids clustering to the library of operations and retaining a single operation from each cluster: the operation closest to the center of each cluster. (b) A comparison to Principal Components Analysis (cf. Sec. 3.4.4), where each Principal Component is a linear combination of all of the original dimensions (in this case, the 6 132 with no special-valued outputs). Principal Components are ordered by their eigenvalues, i.e., by the proportion of variance in the dataset that they explain. Dimensions chosen in this way have lower residual variance compared to those using k-medoids with the same number of dimensions.

is the usual measure for this technique [9]. Using this measure, 207 dimensions are required to achieve a proportion of variance explained over 90%, compared to just five operations using the residual variance measure, defined by Eq. (4.1). Regardless of whether we use the residual variance measure, which is directly interpretable as a correlation of distances between data points in the original space to those in the reduced space, or the more traditional sum of eigenvalues of a PCs decomposition, it is clear that there is significant redundancy in the set of operations included. Hence, producing reduced sets of operations for time-series analysis is a fruitful task that allows us to draw on an interdisciplinary literature to represent real-world signals in a succinct and informative way.

A core aim of this thesis is to find a representation of diverse types of time series that allows them to be compared meaningfully. The above results motivate the selection of such a set, which we choose to be 200 operations chosen by k-medoids

clustering. This set has a residual variance of 0.05, and hence distances between this set of interdisciplinary time series are correlated with those in the original space with a linear correlation coefficient $R^2 = 0.95$. A list of these 200 operations that effectively summarize the behaviors present in the full library is in Appendix G. These 200 operations will be used to structure time-series datasets in Sec. 4.2.

For some time-series analysis tasks, it is not appropriate to use operations that are sensitive to differences in the length, mean, or spread of a time series. To accommodate this, we created reduced sets of operations without these dependencies using the keywords annotated in the database (keywords were assigned on the basis of tests described in Sec. 3.2.3). To this end, the data matrix was first filtered to remove length-dependent, location-dependent, or spread-dependent operations (or any combination of these dependencies), after which the usual dimensionality reduction method was applied. Reduced sets of operations have been formed using k-medoids and linkage clustering with various sizes in the following sets: (i) no location dependence, (ii) no length dependence, (iii) no location or length dependence, (iv) no location or spread dependence, (v) no location, length, or spread dependence. In this way, a variety of operation subsets are pre-calculated and available for immediate use on a given analysis task.

4.1.3 Similarity search

In this section, as above, we use the behavior of operations on the representative set of 875 empirical and synthetic time series described in Sec. 4.1.1 to understand sets of operations with similar behavior across this dataset. This is like 'zooming in' on a small local neighborhood in the vast network of time-series analysis operations. We do this by selecting a target operation of interest, and then ranking all other operations in the database by their mutual information distance, Eq. (3.7), to the target operation. In this way, an ordered list of operations that exhibit (possibly nonlinear) correlations with the target is formed. Here we demonstrate the procedure using four different targets, as shown in Fig. 4.2. The nearest neighbors of the target are plotted either as a network, as in Figs. 4.2(a) and 4.2(b) (using the Kamada-Kawai network visualization algorithm described in Sec. 3.4.7), or as a similarity

matrix, as in Figs. 4.2(c) and 4.2(d). Both representations reveal connections between families of similar time-series analysis methods. The results of the similarity search are immediately interpretable as a set of relationships between different methods for time-series analysis.

Detrended Fluctuation Analysis (DFA) Figure 4.2(a) shows a network visualization of 19 operations with high mutual information with a target operation that estimates the scaling exponent of time series using Detrended Fluctuation Analysis (DFA). Detrended fluctuation analysis is a measure of the long-range scaling behavior in a time series, and was originally developed in the Physics literature in the context DNA sequence analysis [3]. The algorithm has since become a popular method for testing such 'fractal' behavior in many diverse meteorological [117, 118], economic [119], and physiological [120] systems, among others. Here we implement an algorithm for DFA that is in common use[1], which returns the exponent β for a linear fit in a log-log timescale-fluctuation plot.

As expected, this method shows strong, and approximately linear correlations with other fluctuation analysis-based algorithms, which implement similar ideas in different ways (e.g., the related rescaled range analysis [121, 122]). However, we also find unexpected connections to stationarity measures, including a family of *StatAv* measures [123] (with parameters $L = 50, 100, 150, 200$, and 250), the *KPSS* stationarity test from economics [124], and a measure that randomly retrieves 100-sample segments from the time series and returns the standard deviation of means obtained in each segment.

Given this unusual connection highlighted by this method, we proceed to explore the relationship between the DFA scaling exponent and the algorithm StatAv($L = 100$) in more detail. The StatAv algorithm is based on a basic stationarity algorithm proposed by Pincus et al. [123], that splits a z-scored time series into non-overlapping subsegments of length 100 samples, calculates the mean in each of these segments and returns the standard deviation of this set of means. An annotated plot of the dependence of StatAv($L = 100$) on the DFA scaling exponent is shown in Fig. 4.3,

[1]The *FastDFA* code was written by Max A. Little and is available for download at `http://www.physics.ox.ac.uk/users/littlem/software/index.html`

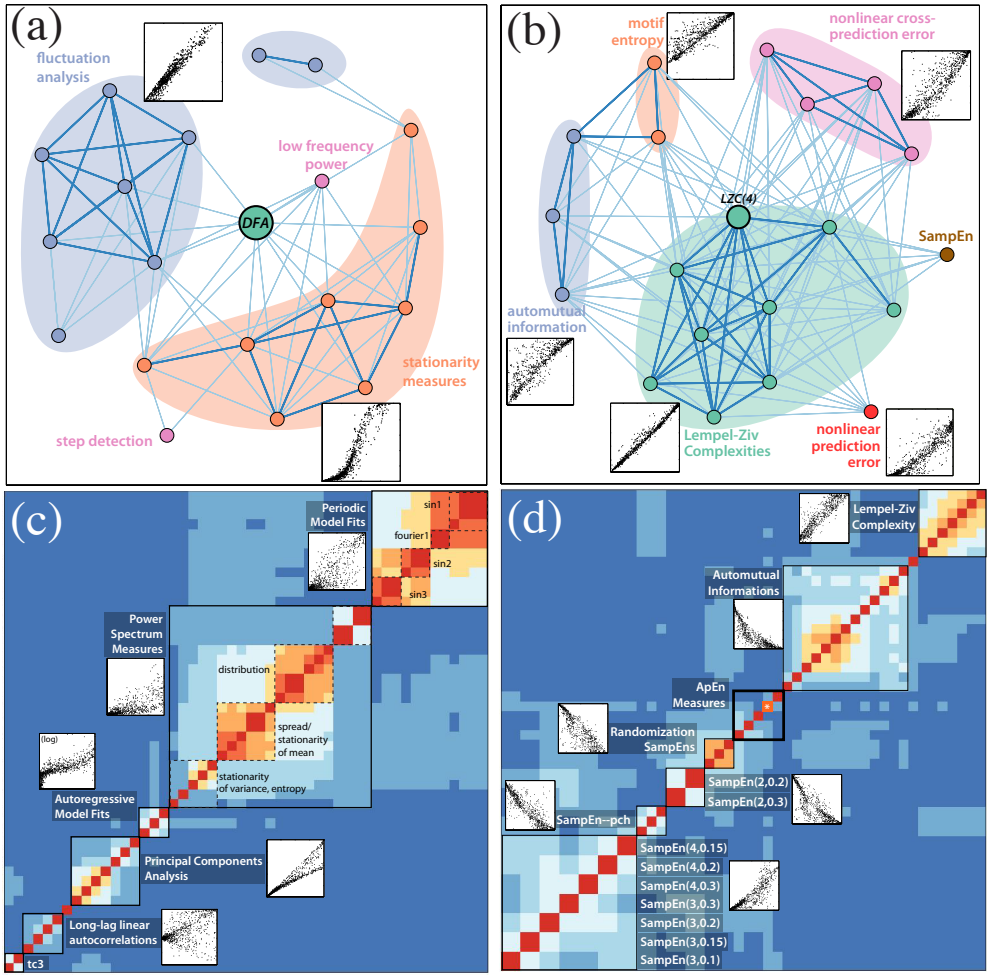

Figure 4.2: **Groups of similar operations**. In each plot we target a single operation, and then illustrate relationships between a group of operations with the highest mutual information to the target across an interdisciplinary set of time series: (a) the scaling exponent estimated using Detrended Fluctuation Analysis (DFA), (b) the Lempel-Ziv complexity of strings of length four, (c) the proportion of variance explained by two largest eigenvalues of a Principal Components projection of a 10-dimensional time-delay signal embedding, and (d) the Approximate Entropy, ApEn(2,0.2). In each case, the dissimilarity between two methods is quantified by the normalized mutual information distance, Eq. (3.7), between them, and is visualized as a network in (a) and (b), and as a clustered distance matrix in (c) and (d). The networks are fully-weighted and represent operations as nodes. Links are plotted that represent distances less than 0.6 (thick) and less than 0.75 (thin). Color in plots (c) and (d) encodes normalized mutual information distances between operations, from 0 (red) to 1 (dark blue). Plots showing the relationship between outputs of the target operation and a representative member of each community are annotated using inset scatter plots, where outputs from the target operation are always plotted on the horizontal axis. In each plot, groupings [with shading in (a) and (b), and black squares in (c) and (d)] have been added manually to aid interpretation.

representing a zoomed-in version of the inset scatter plot in Fig. 4.2(a). That these two measures are so highly correlated across this interdisciplinary range of time series is somewhat surprising: the simple stationarity measure, StatAv($L = 100$), considers variability in the mean at a single timescale (100 samples), while the DFA scaling exponent quantifies how fluctuations about a local linear trend vary across a range of different timescales. Although higher scaling exponents do imply greater non-stationarity of time series, that the measured DFA scaling exponent varies approximately monotonically across this broad and diverse set of signals with this simple, single-scale stationarity measure is a surprising result. This may be due to the fact that applying the DFA algorithm 'blindly', returns a scaling exponent for any time series, regardless of the linearity of the relationship between time-scales and fluctuations, and hence its appropriateness. This relationship should be checked in real applications to ensure that the algorithm is behaving appropriately. Given the prominence of DFA in the literature as providing evidence for long-range scaling in various real-world processes [125–128], the above result has implications for the interpretation of scaling exponents obtained using DFA and justifies further investigation.

A step-detection method, which calculates the proportion of step-changes in a time series also shows correlations with DFA, although the links in the network in Fig. 4.2(a) show that this metric has a higher normalized mutual information with the stationarity measures. This makes sense, as a time series with many step changes is likely to be judged as non-stationary by these measures and, as demonstrated above, this set of stationarity measures are correlated with the scaling exponent calculated from DFA. Another similar measure, labeled 'low frequency power' in Fig. 4.2(a), estimates a periodogram of the time series (using the `periodogram` function from MATLAB's *Signal Processing Toolbox* with a Hanning window), and returns the total power in the lowest 1.3% of frequencies. That this measure of very low frequency power exhibits correlations with the stationarity measures reflects existing knowledge: that strong low-frequency content is a basic indicator of non-stationarity [13].

Lempel-Ziv Complexity In the next example, an operation that estimates the Lempel-Ziv (LZ) complexity of a time series is targeted. This algorithm calculates

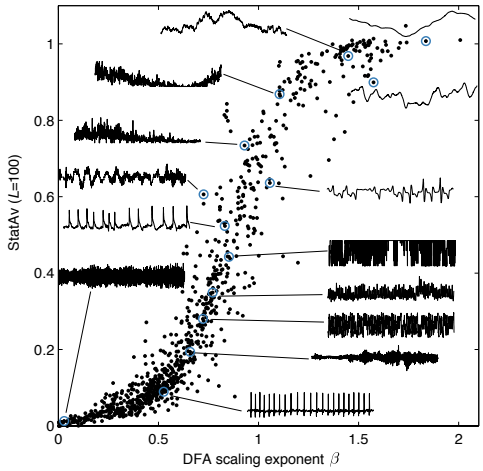

Figure 4.3: **The scaling exponent, β, obtained from detrended fluctuation analysis (DFA) is strongly correlated with a simple stationarity measure over a broad range of different time series**. We plot the outputs of the scaling exponent β obtained from DFA, measured according to a popular, publicly-available algorithm, against outputs from a simple stationarity measure StatAv($L = 100$), that measures the standard deviation of means calculated in non-overlapping 100-sample segments of a time series. Each point in this plot represents a time series from the representative interdisciplinary set of 875 time series described in Sec. 4.1.1. The first 3 000 samples from a selection of time series have been annotated to the plot. These two apparently different measures are actually approximately monotonically related across these diverse time series.

the number of distinct sequences in a symbolized time series, normalized by the expected number of distinct sequences for a random, uncorrelated sequence [129]. We target the LZ complexity of time series using strings of length 4, as implemented using freely-available code[2] [130]. As shown in Fig. 4.2(b), we find similarities to other entropy-related algorithms, including Lempel-Ziv complexities using different sized alphabets, automutual information-related measures, a 4-letter motif entropy measure, Sample Entropy [50], a nonlinear prediction error measure[3] [130], and some nonlinear nonstationarity measures[4] [92]. This result demonstrates our ability to retrieve different operations from a relevant literature of entropy and complexity measures. The operations play broadly similar roles for analyzing empirical signals

[2]**complexity.m** code, written by Michael Small, is publicly-available at `http://small.eie.polyu.edu.hk/matlab/`

[3]**nlpe.m** code, written by Michael Small, is available at `http://small.eie.polyu.edu.hk/matlab/`

[4]The function `nstatz` is part of the popular TISEAN nonlinear time-series analysis package, which is publicly-available at `http://www.mpipks-dresden.mpg.de/~tisean/Tisean_3.0.1/index.html`

as the LZ complexity, and were retrieved automatically from a database containing 8 651 different types of time-series analysis operations.

Leading eigenvalues from Singular Spectrum Analysis In this example, we target a technique known as *singular spectrum analysis* [131] and represent its nearest neighbors using a clustered pairwise similarity matrix as shown in Fig. 4.2(c). The target operation returns the proportion of variance explained by the leading two dimensions in a Principal Components decomposition of a ten-dimensional time-delay embedding of the z-scored time series, where the time delay, τ, is chosen in a standard way: as the first minimum of the auto mutual information [13, 132]. High values of this measure are indicative of low-dimensional linear structure in the embedding space, and hence strong linear autocorrelations in the target time series. As expected, similar methods retrieved from the database are mostly based on linear time-series analysis, including various linear autocorrelations and a variety of measures obtained from the Fourier power spectrum, implying that time series exhibiting strong linear correlations tend to also exhibit low-dimensional structure in a ten-dimensional time-delay embedding space. Since periodic signals are outputs from low-dimensional linear systems, it is unsurprising that operations measuring the quality of fits to periodic time-series models also show strong correlations with this measure.

Approximate Entropy (ApEn) Our final example in this section targets a popular entropy measure: the Approximate Entropy (ApEn) algorithm, that has been widely applied to different types of time series, and in particular in the biomedical time-series analysis literature [29]. The quantity $\mathrm{ApEn}(m, r)$ is approximately equal to the negative average natural logarithm of the conditional probability that two sequences that are similar for m consecutive time points remain similar, within a tolerance r, at the next point[5] [50]. The target for this example is $\mathrm{ApEn}(2, 0.2)$; its nearest neighbors in the database are shown as a clustered pairwise mutual information matrix in Fig. 4.2(d). Operations with outputs that correlate with $\mathrm{ApEn}(2, 0.2)$ include automutual information-based measures, measures derived from the related

[5]ApEn is only *approximately* this quantity because it counts self-matches, i.e., each template matches itself. The Sample Entropy, which was developed subsequently, corrects for this [50].

Sample Entropy (SampEn, which was proposed to reduce bias present in the ApEn algorithm [50]), and Lempel-Ziv complexities. These results suggest that Lempel-Ziv complexities, ApEn, SampEn, and simple automutual information measures play similar roles when applied to a wide range of empirical and synthetic time series.

In summary, we have applied a very simple and general method for retrieving similar objects from a large database to the interdisciplinary time-series analysis literature. The method reveals connections between operations in an immediate fashion using only their empirical outputs on a diverse set of scientific time series, but the results have much stronger theoretical implications. Operations developed in different disciplines and using apparently different underlying theories can be linked empirically, e.g., like the stationarity and DFA operations [Fig. 4.2(a)], or the ApEn and automutual information measures [Fig. 4.2(d)]. By linking apparently different methods and showing explicitly their relationship on real, empirical signals, the results of this process can provide time-series analysts a more unified outlook on the analysis methods at their disposal. It can also stimulate theoretical work to develop a framework that explains the observed links.

4.1.4 Relationships between sets of operations

Rather than just retrieving near neighbors of an operation to understand a local neighborhood of operation behavior, we can also retrieve a pre-defined set of operations to explore dependencies between them. For example, we can explore correlations between given sets of popular methods in a given discipline of time-series analysis, or investigate how a given algorithm with different sets of parameters behaves on the empirical signals they are applied to in practice.

In Fig. 4.4 below, we give an example of a set of auto mutual information (AMI) measures with time lags $\tau = 1, 2, ..., 10$. The AMI(τ) of a time series at a given lag, τ, is an estimate of $I(X_t; X_{t-\tau})$, the reduction in the uncertainty of X_t due to the knowledge of $X_{t-\tau}$ [95]. In Fig. 4.4, the outputs from each of AMI(1), AMI(2), ..., AMI(10) are plotted against each other, where color represents the mutual information-based distance measure, Eq. (3.7), calculated between all pairs of operations. Off-diagonal entries in this matrix reveal how the outputs of AMI(i) are related to the outputs

of AMI(j) across empirical signals. Although algorithms for estimating the AMI of time series at different lags, τ, are in principle distinct and could be independent when applied to some types of time-series datasets, we find that they exhibit strong correlations to one another when applied to empirical signals. We see that AMI(1) is quite a distinct operation—knowing the value of AMI(1) does not provide a lot of information about the value of AMI(2), AMI(3), ..., or AMI(10). However, as the lag τ increases, knowledge of AMI(τ) yields an increasing amount of information about the value of AMI($\tau + 1$). We notice, for example, that the outputs of AMI(9) and AMI(10) are very strongly linearly correlated: knowing the value of one allows the value of the other to be predicted with high accuracy.

This example demonstrates that we can use the empirical behavior of operations, and in particular their correlations to one another, to understand the properties of the time series that they are applied to. When two operations are highly correlated, not because they are structurally similar algorithms, but because of the properties of the time series that they are applied to, this hints at an underlying constraint on the time-series dataset. This key concept will be explored in Chapter 7, where we show that the measurement of correlations between operations can give insights into the number of parameters controlling variation in the time-series datasets that they are applied to.

4.1.5 Operation diagnostics

In this section we show how the reduced set of 875 interdisciplinary time series (introduced in Sec. 4.1.1) can be used to provide an intuitive illustration of the behavior of an operation—by annotating time series to an empirical distribution of its outputs. Four examples are plotted in Fig. 4.5.

In Fig. 4.5(a), a mean-stationarity measure, StatAv($n = 5$) [123], which we denote by **StatAv5** is plotted. This algorithm divides the time series into 5 equal segments and computes the mean in each segment. It then returns the ratio of the variance of this set of means to the variance of the full time series. In Fig. 4.5(a), most of the 875 time series are relatively stationary as measured by this metric, returning a value for **StatAv5** that is close to zero. Time series with segments containing moderate

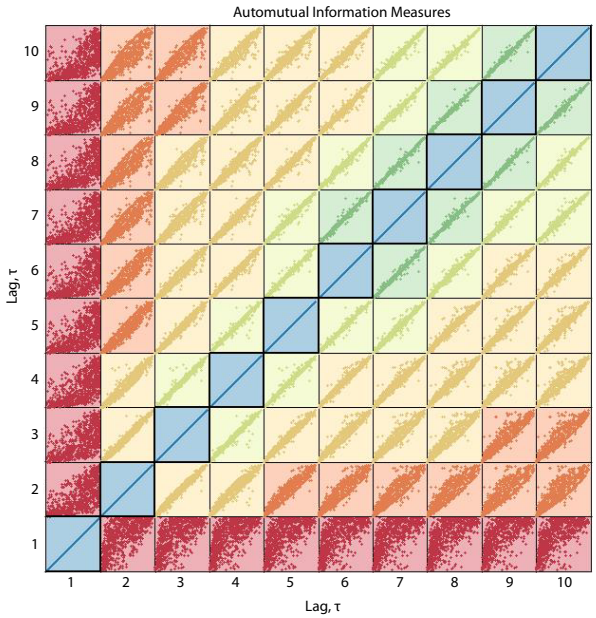

Figure 4.4: **Dependencies between ten operations that measure the auto mutual information of time series at different time lags,** $\tau = 1, 2, ..., 10$. In each subplot, normalized outputs across a representative set of 875 time series (described in Sec. 4.1.1) are plotted for all pairs of operations. Color indicates the normalized mutual information distance, from low (blue) through high (red). The automutual information of a time series measured at lag 1 is quite distinct from the others, but as the lag increases, the measures become more similar to their neighbors.

outliers or small drifts take intermediate values of `StatAv5`, and time series with low-frequency trends or dramatic step changes are assigned high values: the operation is behaving as expected.

In Fig. 4.5(b), the distribution of outputs for an algorithm that returns the number of times a time series crosses its mean as a proportion of its length, N, is plotted[6]. The time-series annotations in Fig. 4.5(b) make sense: slowly-varying time series cross their mean a small number of times compared to their length, while others cross their mean much more frequently.

The entropy measure, Approximate Entropy, ApEn(2,0.1) [29] is targeted in Fig. 4.5(c). Time series are ordered as expected, with highly autocorrelated and periodic processes receiving values close to zero, and complex and noisy processes receiving high values.

The fourth example, a measure of the linear autocorrelation at lag 1, AC(1), is plotted in Fig. 4.5(d). High frequency oscillations have AC(1) near -1, processes

[6]Because the maximum number of possible crossings is $N - 1$, we actually normalize by $N - 1$.

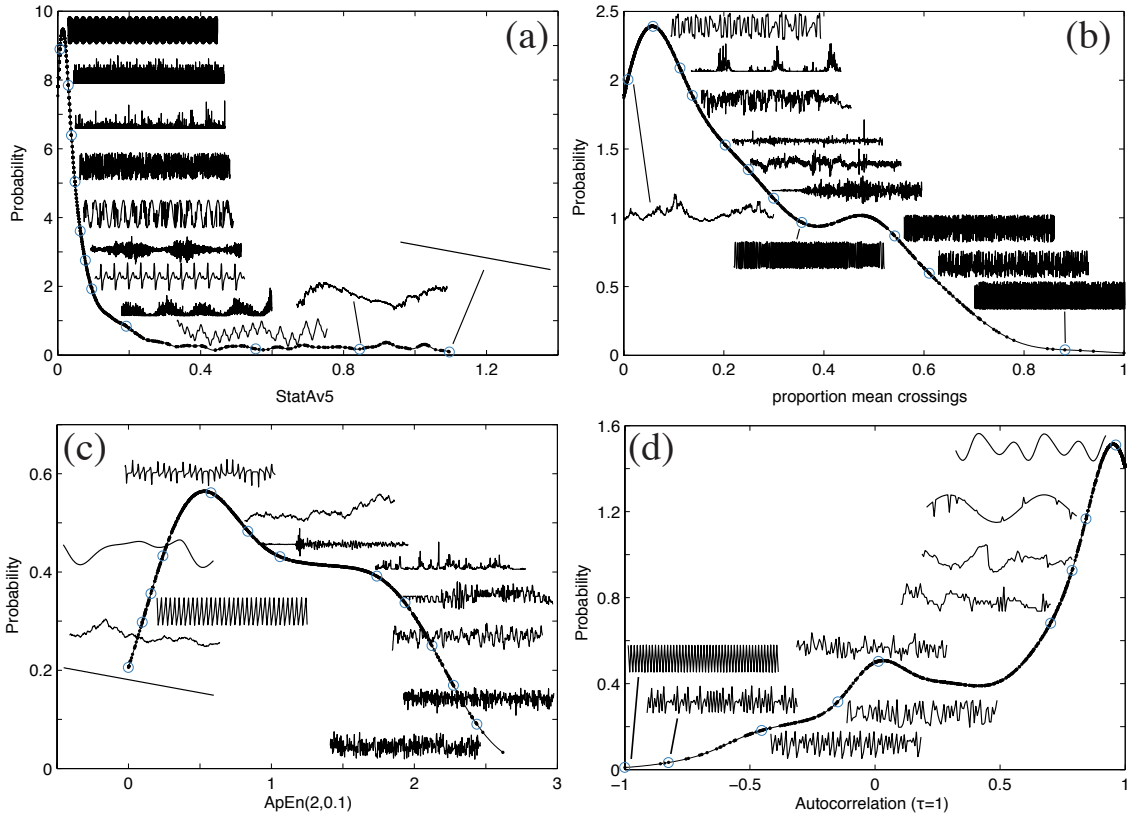

Figure 4.5: **Operations diagnostics illustrate the behavior of an operation across an interdisciplinary range of time series.** Annotated distributions of outputs from four operations across a representative interdisciplinary set of 875 time series (see Sec. 4.1.1): (a) a stationarity measure, **StatAv5**, (b) the number of times the time series crosses its mean as a proportion of its length, (c) an entropy measure, **ApEn(2,0.1)**, and (d) autocorrelation at lag 1, **AC(1)**. Distributions are estimated using MATLAB's **ksdensity** function with default parameters, and are annotated with all the actual data points. In each plot, some example time series are annotated: (a) full time series are annotated, (b) 1 000-sample segments are annotated, (c) 400-sample segments are annotated, and (d) 100-sample segments are annotated.

with no linear correlation at lag 1 receive values values near zero, and frequently-sampled periodic time series and diffusion processes have high outputs near +1. The distribution of outputs for this operation shows that many more time series in this set have AC(1) ≈ 1 than AC(1) ≈ -1.

Using a diverse database of time series to acquire an empirical intuition for one's methods in this way can hence be a useful tool for time-series analysts. Although here we have applied the method to an interdisciplinary set of time series, the procedure could be repeated on any set of time series of interest; e.g., to a dataset of electrocardiograms (ECGs) to provide some visual intuition into the output of a new

ECG analysis algorithm. Such plots also allow cases where algorithms are performing unexpectedly to be detected, potentially motivating subsequent modifications to that operation. We will make use of these plots at various points throughout the thesis to give an insight into the behavior of operations.

4.1.6 Discussion

To summarize, in this section we collected a representative set of interdisciplinary real-world and synthetically-generated time series (in Sec. 4.1.1) and analyzed the behavior of a large database of operations on it. Treating the database of operations as a unified scientific resource, we were able to use the redundancy contained in it to construct smaller sets of operations that approximate the information contained in the full database and are thus somewhat representative of the behaviors of an interdisciplinary scientific literature on time-series analysis (Sec. 4.1.2). Local families of similar operations were obtained by retrieving near neighbors of selected target operations, as in Sec. 4.1.3. Finally, we showed how relationships between sets of operations can be used to give insights into the properties of the signals they are applied to, in Sec. 4.1.4, and how the behavior of particular operations can be visualized by annotating a distribution of their outputs in Sec. 4.1.5.

As discussed in the Introduction to this thesis, Chapter 1, new methods and metrics are constantly being introduced into the time-series analysis literature. Without a means of comparing new methods to a large number of alternatives, it can be difficult to discern whether they offer any improvement over methods already available. The framework developed in this chapter addresses these concerns by representing all operations in a unified framework. In this way, we can aim to reduce redundancy in the set of algorithms we use to analyze time series collected from the world and direct progress by guiding scientific efforts towards new, unique ways of quantifying informative structures in time series. As we demonstrated, connections are revealed between methods developed in different areas of time-series analysis, linking groups of operations and hence different groups of theoretical ideas simply using their empirical behavior. Our highly comparative methodology therefore encourages collaboration between apparently distinct scientific areas.

Much creativity is involved in conventional time-series analysis – the process involves intensely studying and exploring the properties of a dataset and subsequently developing appropriate methods for their analysis (or selecting from existing methods) that may incorporate domain knowledge about the dynamics of the target system and the way it was measured. Although our approach cannot reproduce this intuitive process, it can incorporate the creativity in the process of devising new analysis methods. It is difficult to construct new methods and metrics other than those that are explicitly motivated by a particular data analysis problem, as their potential utility on such a large diversity of time-series analysis tasks is difficult to anticipate. By contrast, our framework encourages this freedom to innovate and construct new methods for time-series analysis because their utility does not need to be immediately demonstrated; instead new algorithms can be freely added to the database and later evaluated for different datasets. If a new operation shows utility for a given task, it will be highlighted in that context, and its behavior can also be compared to a comprehensive existing set of operations. Because of this freedom to innovate new methods, approximately 10% of our database correspond to new types of operations developed by us and introduced in this thesis, many of which turn out to be both unique and useful in different applications (cf. Sec. 5.2).

There is evidently much more work that could be done following what has been shown in this section. We have briefly represented the main types of problems that can be tackled by our approach, which has immediate potential for future work along the same lines. While we have shown only a handful of examples of clusters and similarity searches, many more interesting results could be obtained by analyzing this structure further, especially in the context of particular disciplinary problems for which results have important implications for the practice of time-series analysis in that discipline. For example, dependencies between standard sets of methods in a given field (e.g., the large number of methods of analysis for RR interval time series, cf. Sec. 5.2.6) can be visualized straightforwardly, and their connections to other methods used in other fields can also be made.

4.2 The empirical structure of time series

In Sec. 4.1.2 above, we distilled a set of 200 operations using k-medoids clustering that approximated the behavior of a large interdisciplinary database of 8 651 operations applied to diverse empirical signals. Throughout this section, we use this representative set of operations (excluding location-dependent, spread-dependent, and length-dependent operations) to investigate various types of structure in our database of time series. Using a range of examples, we show that representing time series by feature vectors that simultaneously estimate a range of time-series properties, forms a powerful means by which to compare empirical time series. In Sec. 4.2.1, we show how this set of operations can be used to successfully cluster known classes of interdisciplinary time series. In Sec. 4.2.2, 2 000 time-series clusters are formed by hierarchical linkage clustering from an initial set of 24 577 time series; most of the clusters are homogenous groups of time series measured from particular real-world or model systems. Finally, in Sec. 4.2.3, we show how connections can be made between time series from real-world and synthetic model systems that share similar dynamical characteristics using a large interdisciplinary time-series database. Implications of the results of this section are then discussed in Sec. 4.2.4.

4.2.1 Clustering ten classes of signals

In this section, different classes of time series are organized according to their broad class labels by representing them as feature vectors containing the outputs of 200 representative operations. We first constructed a time-series dataset containing 40 examples from each of the following ten categories: (1) sequences of random numbers, (2) output from stochastic differential equations, (3) electroencephalograms (EEGs), (4) normal sinus rhythm electrocardiograms (ECGs), (5) heart-beat intervals, (6) rainfall, (7) air temperature, (8) log returns of financial data, (9) human speech, and (10) birdsong recordings. In this case, 6 of the 200 operations have either more than 20% special-valued outputs (cf. Sec. 3.2.4), or zero interquartile range (cf. Sec. 3.3.2), and were removed so that feature vectors are of length 194. The data matrix, reordered by average linkage clustering, and colored by class labels, is shown in Fig.

4.6. Visual inspection of the data matrix reveals patterns of operation outputs that are peculiar to each class, and those that capture variation within a given class. The time-series classes are mostly reproduced by the clustering, and the relationships between the groups, as revealed by the dendrogram in Fig. 4.6 connect the different classes in a reasonable way. For example, the three classes of time series with low autocorrelation: log returns of financial data (orange in Fig. 4.6), random number sequences (blue), and rainfall (brown) are grouped. The colored data matrix also shows that uncorrelated noise and financial data actually have very many of the same properties, and hence they cluster together in the dendrogram. The two classes of audio recordings, the birdsongs, and human speech recordings also form a distinct group at the top of the data matrix in Fig. 4.6. Cases where members of a known class do not cluster together are also reasonable – for example, the non-stationary outputs from stochastic differential equations (green in the figure) cluster together near the heart beat interval time series, while the more stationary outputs (e.g., the output from the *M10a* model[7]) cluster instead between the EEG and air temperature time series.

In Fig. 4.7, we cluster these time series using leading Principal Components (PCs) of the normalized data matrix above (containing 194 operations). In this plot, time series are ordered by hierarchical linkage clustering and the `optimalleaforder` MATLAB code, using feature vectors containing the leading PC loadings. In this case, we see that as the number of PCs increases, the time series are ordered in a way that more closely mirrors their known class labels, with as few as three PCs already providing a reasonable clustering of most classes. This example demonstrates that the dominant behavior of a large interdisciplinary set of time-series analysis operations, as represented by their PCs for a given dataset, can provide a useful reduced representation of them—in this case clustering time series into distinct classes of dynamics. This preliminary finding, that Principal Components can provide a useful representation of time-series datasets, will be explored further later in the thesis, for identifying structure in time-series datasets in Chapters 5 and 6.

[7]Umberto Picchini. SDE Toolbox: Simulation and Estimation of Stochastic Differential Equations with MATLAB, `http://sdetoolbox.sourceforge.net`

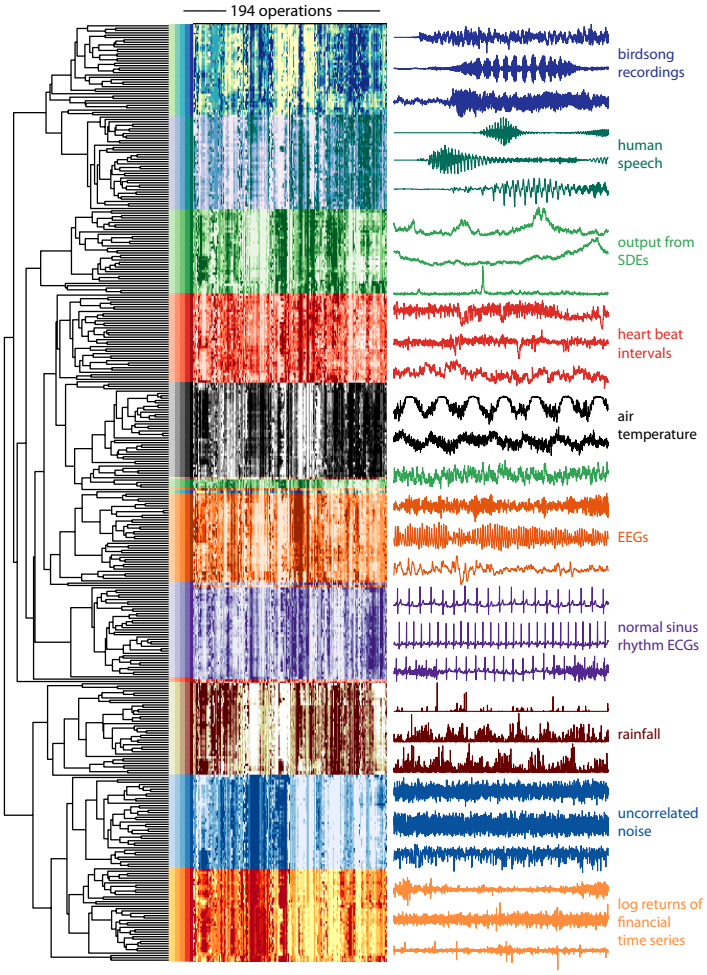

Figure 4.6: **Clustered data matrix containing 40 time series from each of ten categories**. Reordering the (normalized) data matrix using hierarchical clustering organizes it into meaningful categories of time series. The dendrogram structure reflects the often tight clustering of individual classes, and the relationships between them. The data matrix is colored using a different color map for each time-series class, as indicated to the right of the dendrogram: and runs from low values (light) to high values (dark). Some time series are annotated to the right of the row of the data matrix that represents them.

4.2.2 Fine time series clustering

Having shown that a set of 200 operations is able to summarize different types of time-series properties efficiently, in this section, we use them to cluster a large time-series database. The database contains 24 577 time series: those with length at least 1 000 samples (note that to make this set we have also filtered some members of over-represented time series classes to aid computation time[8]). We use complete linkage

[8]The groups that were trimmed are listed here. We kept: 1 000 of the ∼3 000 rainfall time series, 1 000 of the ∼3 000 normal sinus rhythm ECGs, 1 000 of the ∼3 000 congestive heart failure ECGs,

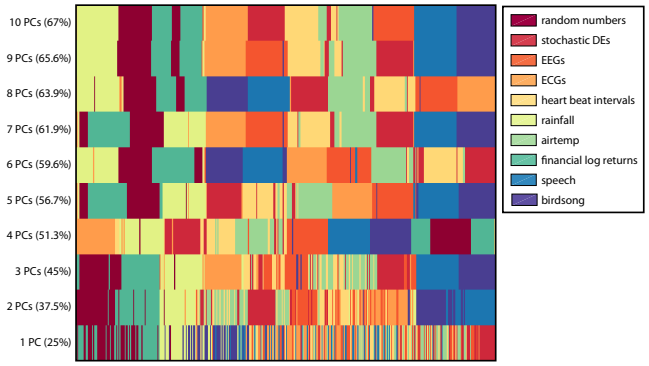

Figure 4.7: **Groupings of time series more closely correspond to their class labels when more Principal Components (PCs) are used to represent them**. Hierarchical clustering is used to order ten classes of time series containing 40 examples each, which are labeled according to the legend. Each row of the matrix corresponds to a different ordering the time series, using their Euclidean distances in the space of PCs of the data matrix to measure dissimilarity. As PCs are added, the groupings become more consistent with the known class labels. The percentage of variance explained by each set of PCs is indicated in parentheses.

clustering to form 2 000 groups of time series. Note that since location-dependent, length-dependent and spread-dependent operations were removed from the set of 200 operations (as described in Sec. 4.1.2), clusters are formed according to meaningful dynamical properties rather than similarities in mean, spread, or length.

Given such a large and diverse initial database of time series of different lengths, sampling rates, etc., one might expect the resulting clusters to be unusual mixes of time series that are correspondingly difficult to interpret. In fact, we found that most clusters are homogenous groups of time series of the same class or produced by the same process, and are discussed in Sec. 4.2.2.1. Clusters containing mixes of time series from different real-world or synthetic model systems are interesting because they imply commonalities in dynamical properties between different types of systems. When mixed clusters contain outputs from different model systems, we interpret the results in terms of our knowledge of the systems that produced the time series in the cluster, in Sec. 4.2.2.2. That we are able to compare such diverse time series meaningfully in this way is a strength of the reduced representation of our database of time-series analysis operations.

and 1 500 of the ∼3 500 outputs from stochastic differential equations. In each case the time series to be kept were chosen at random from the database.

4.2.2.1 Homogenous clusters

Of the 2 000 clusters formed from a highly diverse initial set of 24 577 time series, most are homogenous groups, of time series generated from the same system; some examples are plotted in Fig. 4.8. The first three clusters shown in the figure contain outputs from synthetic model systems: (a) a 'flow' (or set of differential equations), (b) a recurrence relation or 'map', and (c) a stochastic differential equation.

In Fig. 4.8(a), a cluster containing 5 000-sample outputs from a *Duffing two-well oscillator* [12] is plotted, governed by the following differential equation:

$$\frac{d^2x}{dt^2} + b\frac{dx}{dt} + x^3 - x - A\sin(\Omega t) = 0, \tag{4.2}$$

where in this case $A = 0$ such that the driving term is removed and the dynamics is governed by the single parameter b. The integration time step Δt of the time series in this cluster varies between 100, 200, and 400, although all time series are 5 000 samples long. The time series in this cluster have $b \in \{-0.01, -0.005, 0.005, 0.01\}$. Despite the differences in integration lengths and a small variation in parameters between these time series, the similarity in the dynamical behavior exhibited by them is inherited from the underlying differential equation, Eq. (4.2). This similarity is detected by the representative ensemble of time-series analysis operations and is sufficient to place them in their own cluster, amongst 1 999 other clusters.

The cluster plotted in Fig. 4.8(b), contains time series generated from the dissipative *Burger's map* [12], which is defined by the following pair of recurrence relations:

$$x_i = ax_{i-1} - y_{i-1}^2, \tag{4.3}$$

$$y_i = by_{i-1} + x_{i-1}y_{i-1}. \tag{4.4}$$

Nominal parameters are used: $a = 0.75$ and $b = 1.75$ [12]. Time series obtained by iterating these recurrence relations using initial conditions $x_1 \in \{-0.2, 0.1\}$, $y_1 \in \{0.1, 0.2\}$ and for lengths of either 5 000 or 10 000 samples are contained in this cluster. In this case, changes in length do not affect the clustering of these time series due to the marked similarities in their underlying dynamics, as measured by this representative set of operations for time-series analysis.

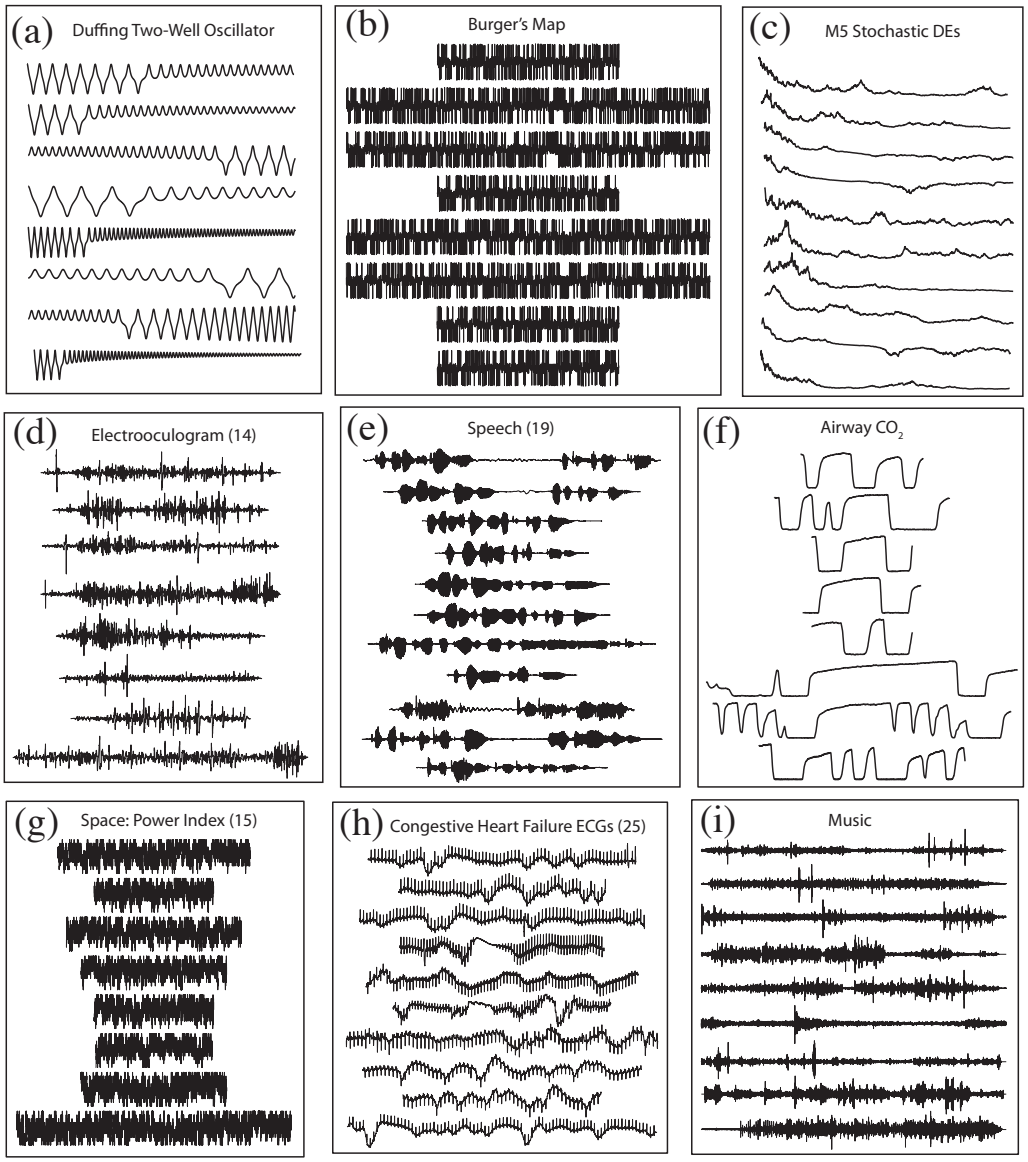

Figure 4.8: **Homogenous clusters of time series**. Most of the 2 000 clusters of time series, formed automatically by hierarchical linkage clustering of 24 577 diverse time series, are homogenous groups of time series produced from the same underlying synthetic or real-world system. Here we plot nine examples of such clusters where, in each case, time series are plotted in order of distance from the cluster centre. In cases where the clusters are large, only the members closest to the centre of the cluster are plotted, and the size of the cluster is indicated in parentheses; otherwise, all cluster members are plotted. (a) Outputs from a Duffing two-well oscillator. (b) Outputs from Burger's map. (c) Outputs from a particular class of stochastic differential equations. (d) Electrooculograms. (e) Speech signals. (f) Airway CO_2 time series. (g) 'Power index' space recordings. (h) Electrocardiograms from congestive heart failure patients (with baseline drifts). (i) Downsampled music recordings.

Figure 4.8(c) shows a homogenous cluster containing outputs from the following stochastic differential equation:

$$dX_t = (aX_t + c)dt + (bX_t + d)dW_t, \tag{4.5}$$

which is the four-parameter model, *M5a*, in the SDE model library from the MATLAB *SDE Toolbox*[9]. The time-series cluster contains a range of 1 000-sample outputs from this model, with parameters $a \in \{-0.01, 0, 0.01\}$, $b \in \{-0.6, -0.2, 0.6\}$, $c \in \{-0.1, 0\}$, and $d \in \{-0.1, 0, 0.1\}$. This homogenous cluster is formed because these time series from this model with these parameters exhibit similar properties.

The remainder of the example clusters in Fig. 4.8(d)–(i) are homogenous groups of real-world time series. These clusters, which are variable in size, contain time series of different lengths but similar properties. This can be appreciated visually for the electrooculograms[10] [133] in Fig. 4.8(d), the emotional speech recordings [134] in Fig. 4.8(e), the airway CO_2 time series[11] in Fig. 4.8(f), the power index measurements from hemispheric power index (HPI) recordings[12] in Fig. 4.8(g), the electrocardiograms (ECGs) from patients with congestive heart failure[13] in Fig. 4.8(h), and the music recordings[14] in Fig. 4.8(i).

4.2.2.2 Heterogeneous clusters

Although most of the 2 000 clusters are homogenous, some clusters contain mixes of different types of processes. When a cluster contains outputs from different model systems, we can evaluate the results in terms of the known properties of the system that generated them. Three examples of such time-series clusters are shown in Fig. 4.9.

[9] U. Picchini. SDE TOOLBOX: Simulation and Estimation of Stochastic Equations with MATLAB, `http://sdetoolbox.sourceforge.net`

[10] Data obtained from a large collection of time series collated by Eamonn Keogh. This particular dataset is studied in Kohlmorgen et al. [133]. A link to the data, available as part of a 'general time series tutorial', requires a request for a password, and is here: `http://www.cs.ucr.edu/~eamonn/tutorials.html`

[11] Data obtained from the Massachusetts General Hospital database on `physionet.org`, available at `http://physionet.org/pn3/mghdb/`

[12] From the National Oceanic and Atmospheric Administration (NOAA), information here: `http://spidr.ngdc.noaa.gov/spidrvo/viewdata.do?docname=Hpinoaa`

[13] Obtained from Physionet: `http://physionet.org/physiobank/database/chfdb/`

[14] Obtained from Ben Fulcher's personal music collection.

Maps The first example cluster, shown in Fig. 4.9(a), contains a mix of three different one-dimensional maps: the *Cubic map*, defined by

$$x_{n+1} = Ax_n(1 - x_n^2), \tag{4.6}$$

the *Sine map*, defined by

$$x_{n+1} = A\sin(\pi x_n), \tag{4.7}$$

and the *Asymmetric Logistic Map*, defined by

$$x_{n+1} = Ax_n(1 - |x_n|), \tag{4.8}$$

where each recurrence relation is governed by the single parameter A [12]. The cluster shown in Fig. 4.9(a) contains outputs from each of these maps with particular parameters: Cubic map series with $A = 2.65, 2.75$, and 2.8, Sine map series with $A = 1.1$, and outputs from the Asymmetric Logistic Map with $A = 4.15, 4.20, 4.25, 4.30$, and 4.35. The parameters of the model that generated each time series are annotated in the plot. In the inset plot in Fig. 4.9(a), the recurrence relationship for each of these maps is shown for $0 \leq x_n \leq 1$ (only this domain has been plotted; all maps are odd functions of x_n). Despite having different functional forms, these maps with these parameters are defining approximately the same recurrence relationship. This example demonstrates the ability of the reduced set of 200 operations, which simultaneously measure different types of structural properties of time series, to discern the strong underlying similarity between the systems that produced the time series in this cluster, despite differences in length (some time series are 1 000 samples, and others are 5 000 samples long).

Random number sequences Another example of a time-series cluster that combines outputs from different model systems is shown in Fig. 4.9(b), which contains random number sequences of length 1 000 from discrete *Poisson* and *Binomial* probability distributions. The cluster is visualized as a network (cf. Sec. 3.4.7) in Fig. 4.9(b). The Poisson distribution is given by

$$f(k, \lambda) = \frac{\lambda^k e^{-k}}{k!}, \tag{4.9}$$

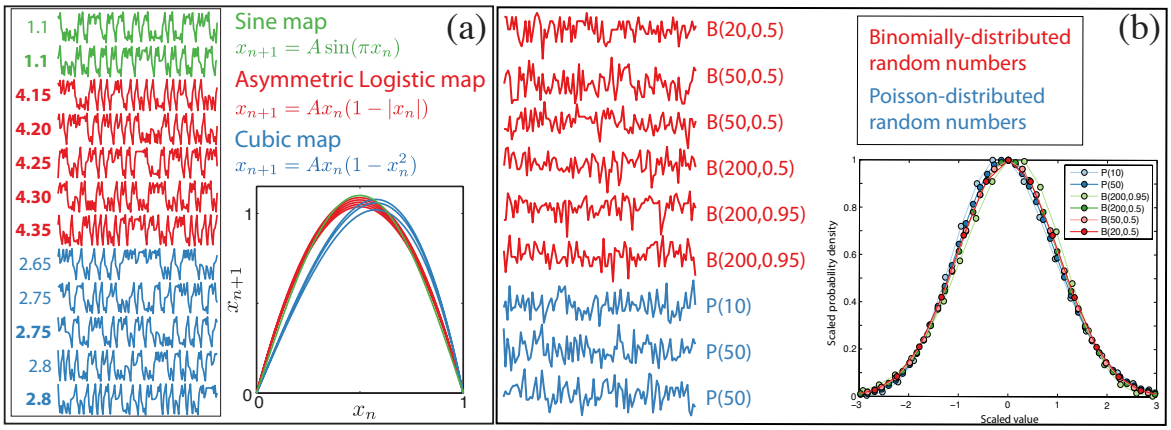

Figure 4.9: **Clusters containing mixes of different model-generated time series**. Two examples are shown here where members of each cluster have a common underlying generative process. These time-series clusters were formed by hierarchical linkage clustering of 24 577 interdisciplinary time series, cf. Sec. 4.2.2. (a) A time-series cluster containing outputs from the *Cubic map*, Eq. (4.6) (blue), the *Sine map*, Eq. (4.7) (green), and the *Asymmetric Logistic Map*, Eq. (4.8) (red), which for these parameters all show a similar recurrence relation. Time series in this cluster are either 1 000 or 5 000 samples long, with the latter indicated by bold labels. Subsegments of length 150 samples are plotted. (b) A time-series cluster containing 1 000-sample sequences of discrete random numbers drawn from either a Binomial distribution (red) or a Poisson distribution (blue). The shapes of these probability distributions with these parameters are plotted, and have been rescaled to have zero mean, unit variance, and a peak value of 1. Segments of length 100 samples are plotted.

where k and λ are parameters. This distribution is used to describe the probability that k occurrences of some process will occur in a given duration when the expected number is λ [135]. The Binomial distribution is given by

$$f(k;n,p) = \frac{n!}{k!(n-k)!}p^k(1-p)^{n-k}, \tag{4.10}$$

where k, n, and p are parameters. This distribution describes the probability of obtaining k successes in n trials of some process that occurs with probability p [135]. The random number sequences were generated using the functions `poissrnd` and `binornd` from MATLAB's *Statistics Toolbox*. Since the vast majority operations used in this work take the z-scored time series as input (cf. Sec. 3.3.1), and because spread and location-dependent operations were removed for this analysis, we plot the theoretical z-scored distributions corresponding to the parameters in this cluster in the inset of Fig. 4.9. Note that these are z-scores of the theoretical distributions known to have generated the time series in this cluster, not the empirical distributions

of each time series. The Poisson distributions have been z-scored by subtracting the mean, λ, and dividing by the standard deviation, $\sqrt{\lambda}$; the Binomial distributions have been z-scored, by subtracting the mean np and dividing by the standard deviation, $\sqrt{np(1-p)}$ [135]. For the purposes of visualization, we have normalized the peaks of the distributions in the inset plot to 1. We can see that, despite slight differences in the allowed values of the discrete distributions, the z-scored random number sequences from these probability distributions in this cluster are derived from approximately the same underlying probability distribution. Again we find that this reduced set of operations, which measure correlation, stationarity, fit models, estimate the correlation dimension, etc. of time series, is able to group time series with remarkable similarities in the processes responsible for generating them. In this case, our automatic approach distinguishes these systems with the same underlying distribution from amongst a large database of time series that contains many other uncorrelated random number sequences and uncorrelated real-world signals. The approach works because sufficiently many operations are sensitive to these distributional properties to overcome variability in other features and place these systems in their own cluster (in the presence of over 800 other random number sequences that are distinguished from these time series only by their distribution).

4.2.3 Similarity search

Given a target time series, other time series with similar properties can be retrieved from the database, using a feature vector containing a reduced set of 200 operations, as above. The dissimilarity between two time series is quantified as the Euclidean distance between their feature vectors. Given a target time series, close 'matches' are retrieved from the database using a method analogous to that described for operations in Sec. 4.1.3: distances between the target feature vector and that of all other time series in the database are calculated and an ordered set of the nearest neighbors is returned.

The nearest neighbors of a given target time series are frequently outputs from the same system as the target, but sometimes interesting links to other types of systems are found. Since the database contains both real-world and synthetic time series,

of particular interest is how these types of systems relate to one another—i.e., how closely do time-series models, which are often formulated with the aim of imitating the structure in real systems, actually reproduce a range of measurable properties of real measurements. Further, given a set of real data, how can we motivate the selection of appropriate models for them? In this section, we demonstrate an empirical approach to addressing these questions—using our database to search for synthetic time series that suggest models for real-world time series (in Sec. 4.2.3.1) and to search for real-world time series to suggest data that might be suitable for a given model (in Sec. 4.2.3.2).

4.2.3.1 Model systems for real data

In this section we target real-world time series and then search for time series generated from synthetic models that exhibit similar measured properties. In this way, we can gain an intuition for the mechanisms underlying a measured time series by comparing it to model data whose generative mechanisms are known. Some examples are shown in Fig. 4.10 below. In each case, we return a given number of the nearest real-world and synthetic matches to a target which is used to provide intuition about the time series and the types of models that most closely reproduce a range of its empirical properties. From the resulting set of real-world and synthetic matches, we use pairwise distances between the feature vectors representing each time series to construct a fully-weighted network for visualization (cf. Sec. 3.4.7). Each of five examples shown in Fig. 4.10 are described in turn in the remainder of this section.

Rainfall The first example, plotted in Fig. 4.10(a), targets a UK daily rainfall time series containing 3 653 samples, recorded from 1980–1989 which was obtained from the MIDAS database[15]. The closest four real-world neighbors and the closest five synthetic model neighbors were retrieved, and the resulting set of ten time series are represented as a network in Fig. 4.10(a). The real-world neighbors are all daily UK rainfall time series, also from the MIDAS database, and the synthetic neighbors

[15]UK Meteorological Office. MIDAS Land Surface Stations data (1853-current). NCAS British Atmospheric Data Centre, 2006. Available from `http://badc.nerc.ac.uk/view/badc.nerc.ac.uk__ATOM__dataent_ukmo-midas`

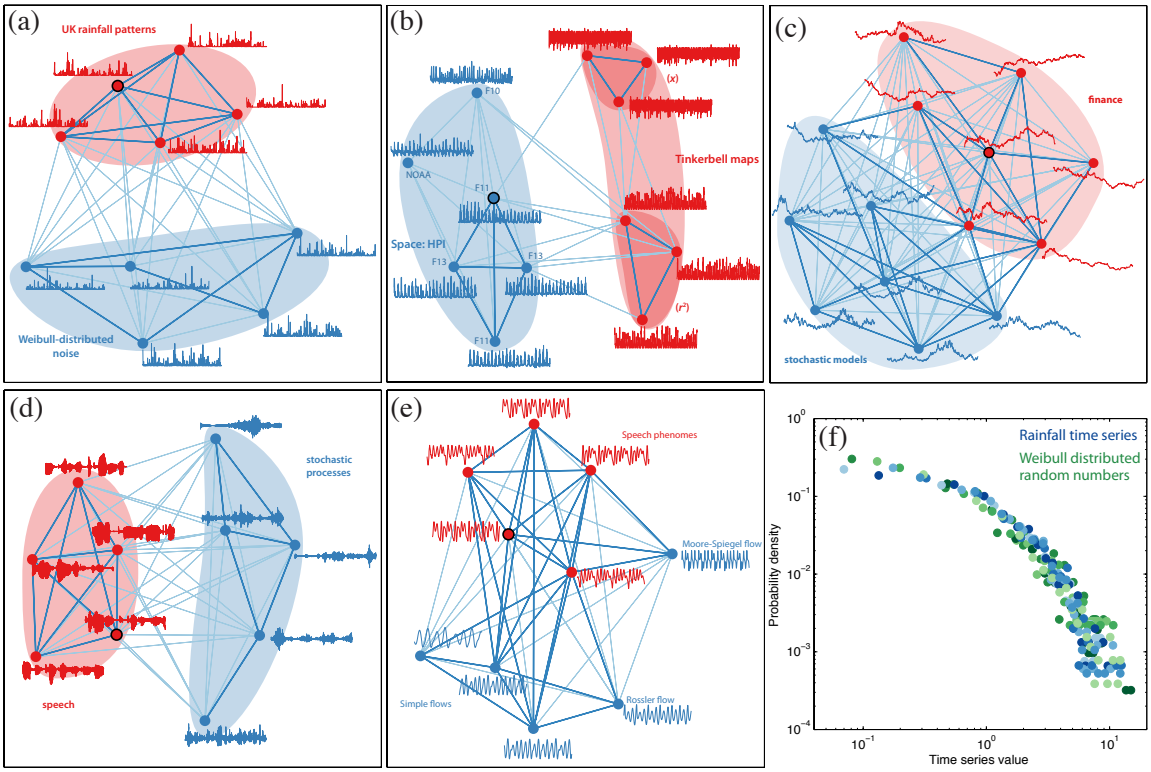

Figure 4.10: **Finding model candidates for real data using a highly comparative approach.** Target time series are: (a) a daily UK rainfall time series, (b) the hemispheric power index, (c) opening daily share prices of 'Oxford Instruments', (d) audio recording of a spoken phrase, and (e) a speech phoneme, and are distinguished using a bold outline in each case. In each case we retrieve real-world and synthetic time series. The network visualizations are fully-weighted and have been constructed as explained in Sec. 3.4.7. Only strong links between nodes (the Euclidean distance, d, between feature vectors is less than 3 units), and weak links between nodes ($3 < d < 4$ units) are plotted, using thick and thin lines, respectively. In (f), distributions of z-scored time-series values from rainfall and Weibull-distributed random number sequences shown in (a) are plotted on a log-log scale. The distribution of amplitudes for each rainfall time series is plotted as a different shade of blue, and that for each Weibull-distributed random number sequence is plotted as a different shade of green. The distributions overlap, contributing to the measured similarity between these two types of processes.

are random number sequences obtained from Weibull distributions (of either 1 000 or 5 000 samples long). The Weibull distribution is a positive-only distribution defined by the following probability density function:

$$f(x;\lambda,k) = \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k} \qquad x \geq 0, \tag{4.11}$$

$$= 0 \qquad x < 0, \tag{4.12}$$

where $k > 0$ is the shape parameter and $\lambda > 0$ is the scale parameter of the distribution [136]. In this case, four of the five Weibull-distributed random number sequences were obtained from a distribution with parameters $\lambda = 0.5$, and $k = 1$; the other is a 1 000-sample time series with $\lambda = 0.5$, $k = 10$. Weibull distributions are commonly used to model and predict extreme events in hydrological time-series analysis [137]. Here, this empirical class of rainfall time series and this theoretical distribution are meaningfully linked according to marked similarities in their measured properties: neither have strong autocorrelations and both have similar distributions of values. The z-scored empirical distributions of time-series values for all time series in this cluster are plotted in Fig. 4.10(f) for these rainfall (blue circles) and Weibull-distributed random number sequences (green circles) on a log-log plot using a histogram with 25 bins. Both types of time series indeed exhibit distributions of the same basic form.

Hemispheric Power Index (HPI) The second example, plotted as a network in Fig. 4.10(b), takes as a target the 'corr' measurement from the hemispheric power index (HPI), which measures high latitude precipitating energy flux carried by ions and electrons from the Defense Meteorological Satellite Program (DMSP) satellites[16], as obtained from the Space Physics Interactive Data Resource (SPIDR)[17]. This particular target time series was taken from station 'F11' from 1991–2000. The five closest real-world and six closest synthetic time series to this target are plotted as a network in Fig. 4.10(b). Five of the six real-world neighbors are also of this 'corr' HPI measurement from different stations: F10 and F13, and another space time series,

[16]Information here: `http://spidr.ngdc.noaa.gov/spidrvo/viewdata.do?docname=Hpidmsp`

[17]Data is publicly available as part of the National Geophysical Data Center (NGDC): `http://spidr.ngdc.noaa.gov/spidr/`

which is the 'normFactor' output from an HPI measurement obtained from a different source: the National Oceanic and Atmospheric Administration (NOAA) Assimilative Mapping of Ionospheric Electrodynamics (AMIE)[18], measured from station 7 over the period 1978–2003. Although obtained from a different source, this time series is also derived from HPI measurements, and our set of summary statistics identify its dynamics as similar to target HPI time series obtained from the DMSP. The HPI measurements connected with strong links in Fig. 4.10(b) are 'corr' measurements from DMSP stations F11 and F13. The noisier output from station F10 and the data obtained from NOAA are less similar.

All of the synthetic matches correspond to outputs from the two-dimensional *Tinkerbell Map* [12] defined by the following set of recurrence relations:

$$x_n = x_{n-1}^2 - y_{n-1}^2 + ax_{n-1} + by_{n-1}, \tag{4.13}$$

$$y_n = 2x_{n-1}y_{n-1} + cx_{n-1} + dy_{n-1}, \tag{4.14}$$

which contains the four parameters: a, b, c, and d, which are set to their nominal parameters: $a = 0.9$, $b = 0.6$, $c = 2$, $d = 0.5$. The matching Tinkerbell Map sequences in Fig. 4.10(b) are of two types: the first combines the x and y outputs of the Tinkerbell Map as $r^2 = x^2 + y^2$, and the other returns just the x output, Eq. (4.13). The links plotted in Fig. 4.10(b) show that the r^2 transformation of the Tinkerbell Map provides a better match to the space data in this case, given the positive-only, positively-skewed target time series. The weaker connection to the underlying x component, even though it differs visually, displays the characteristic Tinkerbell Map dynamics, which our set of 200 measures is sensitive to.

Share prices In the next example, plotted in Fig. 4.10(c), a financial time series of the opening share prices of 'Oxford Instruments' (OXIG) obtained from *Yahoo Finance*[19] was targeted, and the six nearest real-world neighbors and seven synthetic neighbors were retrieved. The real-world neighbors are opening prices of (in order of distance from the target): (i) sector index for automobiles (FTAM), (ii) shares in Oxford Biomedica (OXB), (iii) UK sector index for automobiles and parts (FTUB3300),

[18]Information here: `http://spidr.ngdc.noaa.gov/spidrvo/viewdata.do?docname=Amie`
[19]`http://uk.finance.yahoo.com/`

(iv) shares in Aegis group, a global marketing communications company (AGS), (v) the UK/FTSE foreign index for the health sector (FTMA), and (vi) UK Index for general retailers (NMX5370). Unlike conventional comparisons between financial time series where distances between time series are judged according to the strength of their temporal correlations to one another [138, 139], distances in this case are obtained from the feature vectors, and reflect (dis)similarities in the empirical dynamics of the process. Hence, time-series matches obtained in this way do not necessarily covary, but rather have many similar properties. Synthetic time-series matches in this case are mostly outputs from the *M5a* stochastic model, Eq. (4.5):

$$dX_t = (aX_t + c)dt + (bX_t + d)dW_t. \tag{4.15}$$

Here we have a range of different parameters for the model: $a \in \{-0.01, 0, 0.01\}$, $b \in \{-0.6, 0.2\}$, $c \in \{-0.1, 0, 0.1\}$, $d \in \{-0.1, 0, 0.1\}$. There is also a match to the *M10a* model from the SDE model library in the MATLAB *SDE Toolbox*[20]:

$$dX_t = a(b - X_t)dt + \sigma\sqrt{X_t}dW_t, \tag{4.16}$$

with $a = 1$, $b = 1$, and $\sigma = 1$. These biased random walks are able to reproduce the correlated behaviors of these share-price time series. There is a similarity in these processes to the classic Brownian motion models used in economics. The *Black Scholes* model [140], for example, considers that the price of an underlying asset follows

$$dX_t = \mu X_t dt + \sigma X_t dW_t, \tag{4.17}$$

which is the form of a *geometric Brownian motion* with parameters μ and σ [84, 141, 142]. The nearest neighbors of this Oxford Instrument share price time series therefore suggest a family of stochastic differential equation models that reproduce many of the characteristic properties of this data, and corresponds to an existing theory of modeling time series in this way.

Spoken phrases As shown in Fig. 4.10(d), we retrieved the most similar time series to an audio recording of a phrase spoken with a 'sad' emotion. This recording was

[20]U. Picchini. SDE TOOLBOX: Simulation and Estimation of Stochastic Equations with MATLAB, `http://sdetoolbox.sourceforge.net`

obtained from the *Berlin Emotional Speech Database* [134], which is analyzed in detail in Sec. 5.2.3. The nearest four real-world matches are other speech recordings from the database with either 'boredom' or 'neutral' emotions. The five nearest synthetic matches are seasonally-detrended outputs from stochastic models – four from the *M5a* model [Eq. (4.5)], and one match to a seasonally detrended *M10a* model [Eq. (4.16)]. The *M5a* model time series have $a \in \{-0.01, 0, 0.01\}$, $b \in \{-0.6, -0.2, 0.2, 0.6\}$, $c \in \{-0.1, 0.1\}$, and $d \in \{-0.1, 0, 0.1\}$, and the *M10a* match has $a = 1$, $b = 0.1$, and $\sigma = 1$. The seasonal detrending in this case removes the low frequency components of these time series, leaving the fluctuations, which appear to mimic the bursty, clustered oscillations that characterize speech. In this case, although the match does not capture many of the important subtleties of speech signals, we nevertheless obtain a sensible set of models that suggest possible mechanisms that could have generated many of the empirical properties of the time series.

Spoken phonemes The next example, shown in Fig. 4.10(e), targets a phoneme recording from a patient with Parkinson's disease, taken from a dataset of phonemes studied in Sec. 5.2.5. Four real-world matches and five synthetic time series were retrieved. The four real-world neighbors are phoneme speech recordings from this same dataset; synthetic neighbors are outputs from a selection of flows, or sets of differential equations. The first is the *Moore-Spiegel oscillator* [12]:

$$\dddot{x} + \ddot{x} + (T - R + Rx^2)\dot{x} + Tx = 0, \tag{4.18}$$

where dots over variables represent derivatives with respect to time: d/dt, and where normal parameter values are used: $T = 6$, $R = 20$. Another matching system is called the *simplest piecewise linear chaotic flow* [12], and is governed by

$$\dddot{x} + a\ddot{x} + \dot{x} - |x| + 1 = 0, \tag{4.19}$$

with $a = 0.6$. Three of the matches are the z components of this flow integrated over the range $1 \leq t \leq 1\,000$ and evaluated on a one-dimensional grid of 5 000 or 10 000 evenly-spaced points, or for $1 \leq t \leq 2\,000$ evaluated at 10 000 evenly-spaced points. The final time series that matches the target phoneme is an output from the *Rössler*

attractor:

$$\dot{x} = -y - z, \qquad \dot{y} = x + ay, \qquad \dot{z} = b + z(x - c), \tag{4.20}$$

where the parameters were set to their nominal values: $a = 0.2$, $b = 0.2$, and $c = 5.7$ [12]. As is evident from the time-series annotations in Fig. 4.10(e), these dynamical systems are able to reproduce the smooth, oscillatory local structure of these speech phoneme recordings. Indeed, biomechanical models of speech production use nonlinear differential equations to reproduce and understand observed patterns in speech phonemes, e.g., the two-mass model of vocal folds [143, 144].

4.2.3.2 Real data for model systems

Above, we showed how empirical data can be linked to appropriate models simply using measurements of their empirical properties. In this section we demonstrate this process in reverse: we target synthetically-generated outputs from particular model systems and retrieve real-world signals with similar properties. This procedure allows us to contextualize the dynamics of time series generated by a theoretical model with respect to recordings from real-world systems. Three examples of model systems are presented: stochastic switching, fractal (or self-affine) time series, and sine waves corrupted with additive noise.

Stochastic switching In Fig. 4.11, we target a 5 000-sample target produced from the *stochastic sine map* used by Freitas et al. [85]:

$$x_{n+1} = \mu \sin(x_n) + Y_n \eta_n, \tag{4.21}$$

where $\mu = 2.4$, Y_n is Bernoulli random variable that returns 1 with probability q and 0 with probability $1 - q$, and η_n represents a sequence of uniformly-distributed random numbers over the interval $[-b, b]$. We target such a process with noise amplitude $b = 3$ and noise propability $q = 0.2$; an example time series generated from Eq. (4.21) with these parameters is plotted in Fig. 4.11(b). The structure of this simple system, defined by Eq. (4.21), is illustrated in Fig. 4.11(a). As explained in the caption, the system contains two stable limit cycles (plotted green in the figure), with basins of attraction separated by $x = 0$. The addition of noise, with probability $q = 0.2$ (and

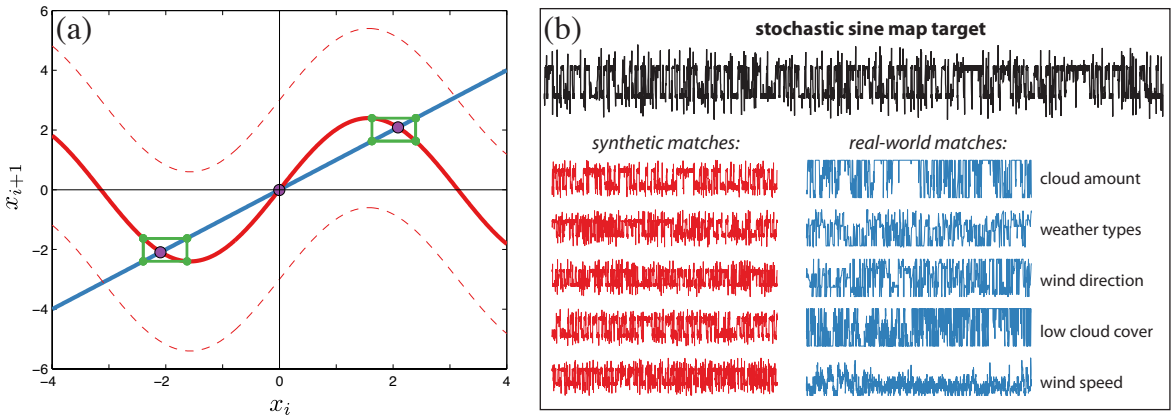

Figure 4.11: **The most similar real-world processes to an output from a stochastic switching model are meteorological time series with 'noisy switching' dynamics.** We target a stochastic sine map time series from Eq. (4.21) with parameters $b = 3$, $q = 0.2$, and return both synthetic and real-world neighbors. (a) An illustration of the structure of this stochastic map, including the noiseless case $x_{i+1} = \mu \sin(x_i)$, for $\mu = 2.4$, plotted in red. The dotted lines indicate the effect of noise, which is uniformly distributed in the range $[-3, 3]$. The line $x_i = x_{i+1}$ is plotted in blue, and equilibriums are plotted as purple circles. In the absence of noise, the dynamics of this map settle to one of the two limit cycles. The basins of attraction of these two limit cycles are $x_i > 0$ and $x_i < 0$. The presence of noise in the system (which occurs with probability $q = 0.2$) can cause it to switch from one stable limit cycle to the other. An example time series generated from this system is plotted as a black time trace in (b). Synthetic matches to this target time series are stochastic sine maps with the same parameters, while real-world matches are noisy switching processes from meteorological systems.

represented by dashed red lines in the figure), can perturb the system from one basin of attraction to another and hence the dynamics of the system involves stochastic switching between two stable limit cycles.

The closest synthetic and real-world matches to a time series generated by this process are shown in Fig. 4.11(b). The synthetic matches are other realizations of the stochastic sine map, Eq. (4.21), with the same parameters: $b = 3$, $q = 0.4$. Some matches are the same length as the target, i.e., 5 000 samples, and others are 10 000 samples long. Real-world matches are recordings from meteorological systems: time series of cloud amount, weather types, wind direction, low cloud cover, and wind speed all exhibit the same qualitative dynamics: noisy switching between transient states.

Self-affine time series In this example, shown in Fig. 4.12, self-affine time series generated by the random midpoint displacement method [90] (described in detail in Sec. 5.1.3) are used as targets to retrieve similar real-world time series. Self-affine time series can be characterized by a single scaling exponent, α, of the power spectral density, which scales as $S(f) \sim f^{-\alpha}$, where $S(f)$ is the square amplitude of the Fourier transform of the time series and f represents frequency. Fractal time series can be separated into two types: fractional Gaussian Noise (fGN) time series have $-1 < \alpha < 1$, and Hurst exponent $H = (\alpha + 1)/2$, and fractional Brownian Motion (fBM) time series have $1 < \alpha < 3$ and Hurst exponent $H = (\alpha - 1)/2$. In Fig. 4.12, we target two fGN time series and two fBM time series and seek matches to real-world time series. Note that in each case, we are not explicitly selecting time series with a similar scaling exponent as the target—operations that estimate α are determined in Sec. 5.1.3. Instead, assuming that the majority of the 200 operations are measuring informative characteristics of empirical time series, real-world time series are retrieved that share many similar *properties* with each target self-affine time series.

In Fig. 4.12(a), we target an fGN time series with $\alpha = -0.5$ ($H = 0.25$). Real-world matches are log returns of financial time series: the difference between maximum and minimum daily stock prices (high-low), and daily traded stocks volumes. In Fig. 4.12(b), we show neighbors of a fractal time series with $\alpha = 0.5$ ($H = 0.75$), which retrieves a diversity of time series from astronomy, current fluctuations on DNA pores, relative humidity recordings, and sequential tree ring data. In Fig. 4.12(c), we target an fBM time series with $\alpha = 1.5$ ($H = 0.25$). Again, we retrieve an interdisciplinary selection of real-world time series that display similar empirical behavior: cosmic ray recordings, air temperature, particular opening share price time series, and heart beat intervals. Neighbors of a persistent fBM with $\alpha = 2.5$ ($H = 0.75$), is plotted in Fig. 4.12(d), and include mainly financial time series, and an electrocardiogram (ECG) taken from a patient with congestive heart failure. This ECG has self-affine visual features, with the baseline drift reflecting that of each heart beat event, and visually similar fluctuations also on a finer scale.

Although not plotted, synthetic matches of each of the targets in Fig. 4.12 are mostly self-affine time series with similar scaling exponents, α, as the target in each

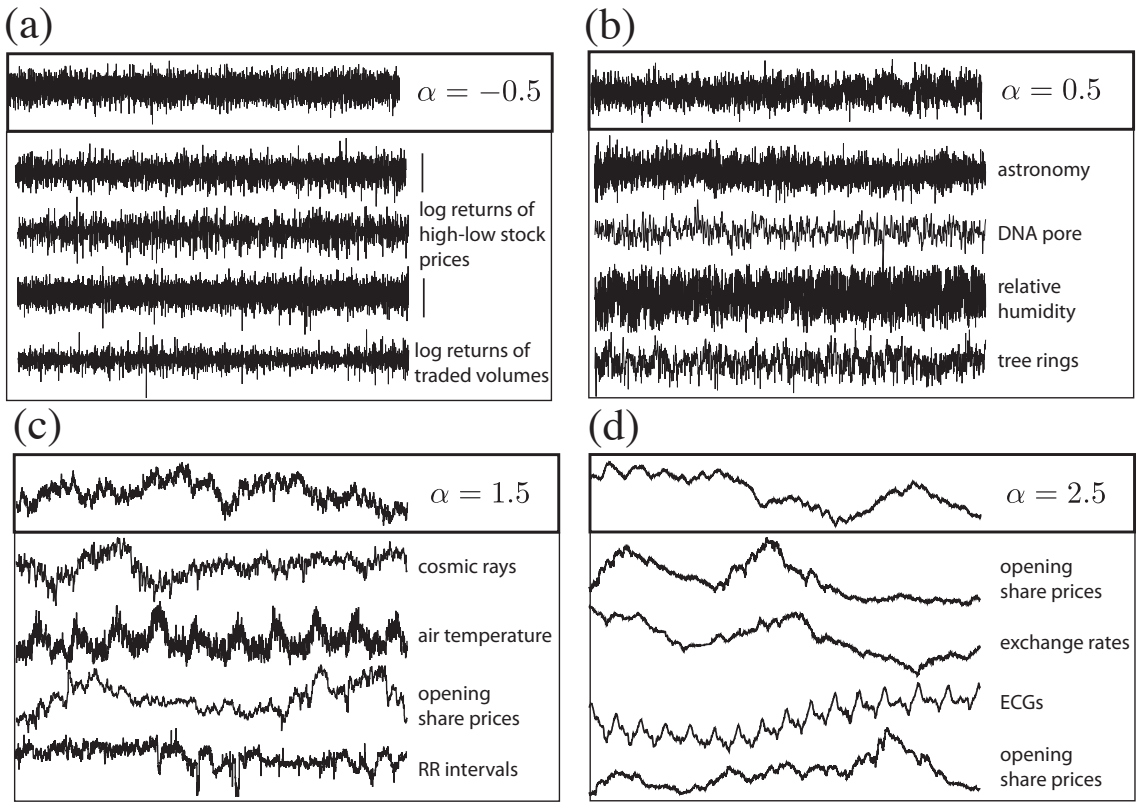

Figure 4.12: **Real time-series matches to target self-affine time series.** Self-affine time series can be characterized by their power spectral density scaling exponents, α. In each subplot, four real-world neighbors of the target time series are plotted: for (a) $\alpha = -0.5$ (i.e., fGN with $H = 0.25$), (b) $\alpha = 0.5$ (fGN with $H = 0.75$), (c) $\alpha = 1.5$ (fBM with $H = 0.25$), and (d) $\alpha = 2.5$ (fBM with $H = 0.75$).

case. Some exceptions include nonlinear transformations of autoregressive processes, which are similar to some fGN time series, and some random walk time series that matched with the $\alpha = 2.5$ time series (theoretically, random walks have $\alpha = 2$).

Although these similarity searches are performed in a way that does not involve any knowledge of the self-affine nature of the target time series, we are able to return synthetic time series with similar α to a target time series. This is because self-affine time series are characterized by α, and have feature vectors that are characteristic of the resulting behavior as measured not just by scaling exponent estimates (which make up only a small minority of the operations in the feature vector), but also by autocorrelation, stationarity, entropy, and other measures that each contributed to building a comprehensive signature of the empirical properties of time series generated by a given process. Since most operations measure informative characteristics of the

time series, this information is accumulated and used to retrieve meaningful time-series matches.

Noisy sine waves As a final example, we target periodic signals of unit amplitude and a period $T = 20$ samples corrupted with additive Gaussian-distributed noise of variance, η^2. The signal to noise ratio, defined as the variance ratio of the signal to the noise can be computed as $1/2\eta^2$. Outputs from this simple model, given by Eq. (5.1), are characterized in detail in Sec. 5.1.1. As shown in Fig. 4.13, four target time series are chosen, with (a) $\eta = 0$, (b) $\eta = 0.2$, (c) $\eta = 0.5$, and (d) $\eta = 2$. In each case, we show 5 examples of synthetic time series neighbors and 5 examples of real-world neighbors.

First, a pure sinusoid with $\eta = 0$ is targeted, as shown in Fig. 4.13(a). In this case, synthetic time-series matches are other sine waves (with different periods), and outputs from different oscillatory dynamical systems. Real-world time series are speech phonemes, a periodic recording of satellite position, and some very regular electroencephalograms (EEGs) recorded during seizure episodes. All matches display oscillatory dynamics, as defined by the $\eta = 0$ target.

In Fig. 4.13(b), a sinusoid corrupted with Gaussian-distributed noise with $\eta = 0.2$, or a signal-to-noise ratio of 12.5, is plotted. Synthetic neighbors are outputs from the same model, Eq. (5.1), with similar noise levels, and we also retrieve a matching noisy periodic process with a slightly longer period of $T = 25$ samples and the same noise level. Real-world matches all feature strong underlying periodic behavior with a visually similar signal-to-noise ratio. In Fig. 4.13(b), we plot monthly temperature recordings from the Hadley Centre[21], a noisy 'stationary phoneme' audio recording [144], an audio recording of a vibrating phone[22], a monthly air pressure recording from Nagasaki[23], and an astrophysical recording of the Ionosphere[24].

The next example, shown in Fig. 4.13(c), targets a noisy sine wave with $\eta = 0.5$, or a signal-to-noise ratio of 2. Synthetic neighbors are all periodic processes with similar signal-to-noise ratios; the closest neighbors have similar periods, and more distant

[21]Data available here: `http://www.metoffice.gov.uk/hadobs/hadcet/`
[22]Obtained from `http://www.soundjay.com/communication-sounds.html`
[23]Studied in [145]; data available from `http://www.cru.uea.ac.uk/cru/data/japan/`
[24]From the Space Physics Interactive Data Resource (SPIDR).

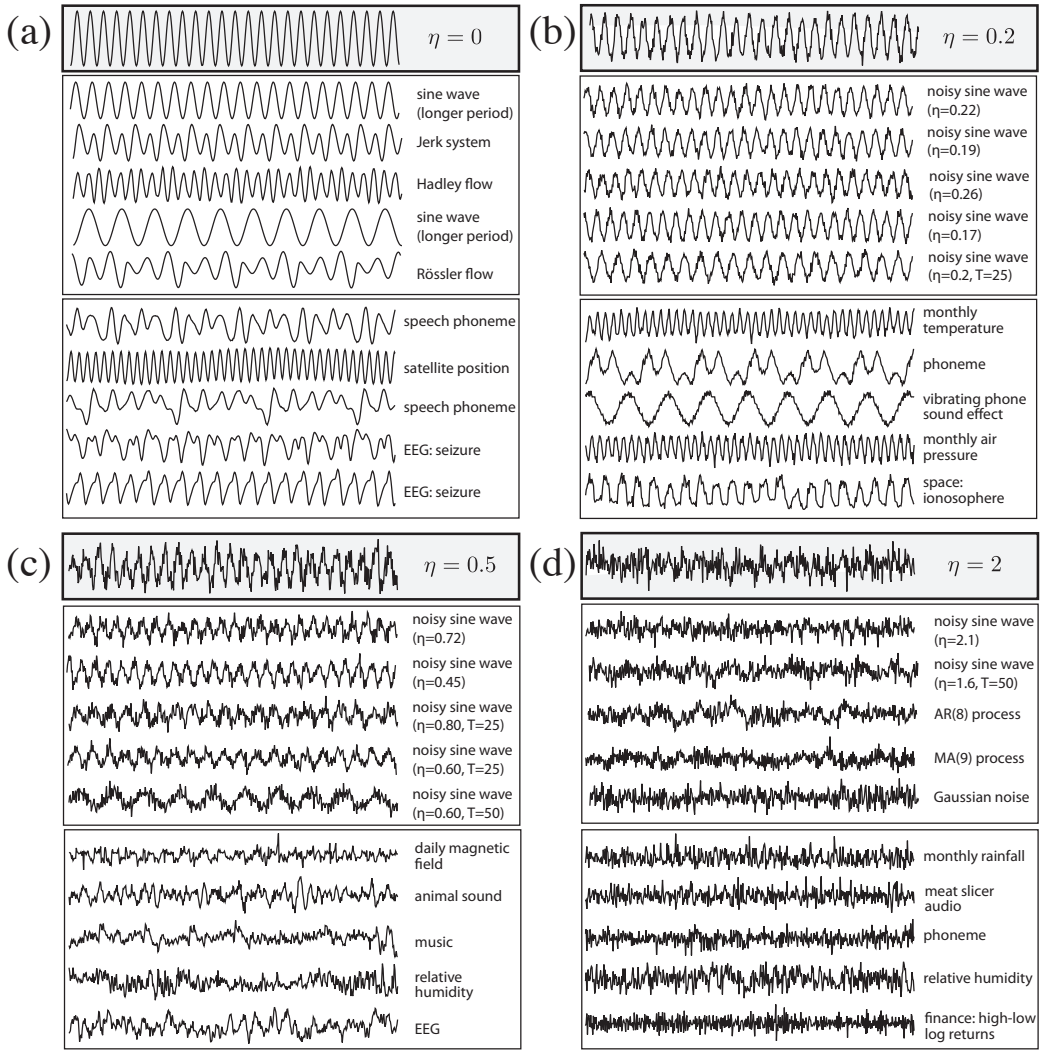

Figure 4.13: **Time series with similar properties to noisy periodic signals.** In each case, we retrieved synthetic and real-world neighbors of sine waves of period $T = 20$ samples, contaminated with Gaussian noise with standard deviation η. We consider (a) $\eta = 0$ (a pure sinusoid), (b) $\eta = 0.2$, (c) $\eta = 0.5$, and (d) $\eta = 2$. Target time series are plotted in thick boxes, followed by five of the closest synthetic matches and five of the closest real-world matches. In each case the first 500 samples of matches are plotted and are ordered from top to bottom by increasing Euclidean distance from the feature vector of the target, along with a brief label.

neighbors have longer periods. Real-world neighbors are processes with an underlying periodic behavior but with strong additional fluctuations, including daily magnetic field recordings[25], the sound made by a grouse[26], a snippet from a rock music recording[27], and a set of relative humidity measurements[28], and an electroencephalogram (EEG) recording[29].

Finally, as shown in Fig. 4.13(d), we consider a target time series with $\eta = 2$, or a signal-to-noise ratio of 0.125. In this case, the noise overwhelms the periodic signal, and again synthetic matches are meaningful: the closest matches are noisy sine waves with similar η, and, since the power contained in the signal is so low, we now find matches to some correlated noise processes: an autoregressive (AR) process of order 8 and a moving average (MA) process of order 9 [93], as plotted in Fig. 4.13(d). Because the periodic component in this signal is so weak, these correlated noise processes are able to closely reproduce many of the same structural properties. Although at a greater distance from the target, another match is a sequence of Gaussian-distributed random numbers, which is also able to closely reproduce many of the properties of this target, reflecting its very low signal-to-noise ratio. Like the synthetic target, real-world matches are visually noisy and have weak autocorrelations: a monthly rainfall time series from the Hadley Centre[30], an audio recording of a meat slicer[31], a noisy stationary phoneme recording[32], relative humidity measurements[33], and log returns of high-low prices[34].

[25]From the Space Physics Interactive Data Resource (SPIDR).

[26]From the Macaulay Library, a segment from audio file index number 137 313 which corresponds to the Rock Ptarmigan. Data available from `http://macaulaylibrary.org/index.do`

[27]The song, 'Take a Bow', by *MUSE*, from the album *Black Holes and Revelations*.

[28]The daily data from grid-points around the globe is available at `http://www.cru.uea.ac.uk/cru/data/ncep/qs_eurasia/daily/surface/air.sig995/`

[29]Data obtained from a large collection of time series by Eamonn Keogh. A link to the data, available as part of a 'general time series tutorial', requires a request for a password, and is here: `http://www.cs.ucr.edu/~eamonn/tutorials.html`

[30]`http://www.metoffice.gov.uk/hadobs/hadukp/`

[31]Obtained from `http://www.soundjay.com/household-sounds.html`

[32]Information here: `http://www.eng.ox.ac.uk/samp/members/max/publications/little_thesis.pdf`

[33]The daily data from grid-points around the globe is available at `http://www.cru.uea.ac.uk/cru/data/ncep/qs_eurasia/daily/surface/air.sig995/`

[34]Of NMX3780 (Tobacco) from the UK Sectors Indices, FTSE 350 Sectors. Data obtained from `http://uk.finance.yahoo.com/`

4.2.4 Discussion

In this section, we have demonstrated the usefulness of our feature vector representation of empirical time series for a variety of applications. The feature vectors are a powerful generic summary of time-series behavior and represent an informative synthesis of a previously disjoint literature on time-series analysis. This representation allows time series measured from different systems, for different lengths of time, and in different ways, to be meaningfully compared to one another and to various model systems. We showed how this representation can be used to structure a large and diverse interdisciplinary time-series database using clustering or similarity searches, yielding scientifically interesting and useful results.

While the results of this section are interesting in themselves, as examples of general methods they also have implications for the future practice of time-series analysis. The scientific process we have presented begins with an input of data, proceeds through an exploratory investigation of different real-world and synthetic time series matches, and then an interpretation of interesting matches which can be used to motivate further analysis. In contrast to conventional time-series analysis, we automatically obtain information about the characteristics of the target time series by contextualizing it within an extensive database of empirical and model time series studied in different scientific disciplines. By suggesting these connections, scientific investigation can be guided towards families of models that may be relevant to understanding the dynamics of the target process. Even if a model for real data suggested in this way is inappropriate for describing the actual mechanics governing the system of interest, it will share some of the basic behaviors of the system that generated the data and may still provide a useful reduced picture of some of its important dynamical characteristics. Although this process is no substitute for traditional model building motivated by informed questions asked of the data; the connections this process highlights can guide this process and hence represent a powerful complement to traditional time-series analysis by suggesting other scientific systems—both theoretical and empirical—that exhibit similar dynamics.

The operations we used to summarize time-series properties in this section were

selected from the behavior of a large set of operations on a dataset containing diverse empirical signals, as described in Sec. 4.1.2. It is not obvious that by combining so many different operations in this way that a meaningful clustering of time series will be obtained, as demonstrated in Secs. 4.2.1 and 4.2.2, or that neighbors of particular time series will correspond to meaningful real and synthetic time series matches, as in Sec. 4.2.3. The success of this procedure relies on this set of operations being diverse and informative enough to provide a characteristic signature of different types of dynamical behaviors. Although operations may not be designed to estimate some specific time series property, they are often sensitive to many different dynamical properties of time series. For example, dimension estimates are sensitive to the presence of outliers, thereby providing some information about the distribution, similarly some model fits are sensitive to the time series autocorrelation and stationarity properties, etc. Thus, combining a collection of informative time-series analysis operations together becomes a versatile, robust, and powerful way of representing time series.

Note that the similarity searches performed in Sec. 4.2.3 for individual target time series could easily be generalized to sets of time series. The most straightforward way to achieve this would be to simply represent the group by a central feature vector (e.g., the mean of all feature vectors in the group, or the feature vector with the minimum sum of distances to all others) and computing distances of all other time series to this representative feature vector. A more sophisticated treatment would involve finding neighbors to the group as a whole. These neighbors could be defined to minimize the average distance to all members of the group (as in 'average' linkage clustering), the minimum distance to group members (as in 'single' linkage clustering), or the maximum distance to group members (as in 'complete' linkage clustering). Such an approach contextualizes a time-series dataset amongst similar time series studied in different disciplines.

The analysis performed in this section is clearly dependent on the diverse set of 200 operations that make up the time series feature vectors; the analysis could also be performed using different sets of operations. For example, the characteristic properties of a particular group (i.e., those that most distinguish it from other signals, like the salient skewness and burstiness properties of rainfall time series) could be weighted

more heavily in the similarity search. Included operations could also be chosen by a clustering of operation behavior not on an interdisciplinary set of time series but a more specific time-series dataset (e.g., a dataset of ECGs) that best represent the main types of operation behaviors on that particular data type. We could also select a set of operations as those most useful for a given task, e.g., operations could be retrieved that most effectively measure scaling exponents in time-series (as in Sec. 5.1.3) and then time series with similar measured scaling exponents to a target could be retrieved using these successful operations.

We finally note that the analysis of time-series models performed in this section is relatively unusual. Time-series models generally have a strong underlying theoretical framework that provides an understanding of the mechanisms underlying the model that give rise to the properties of the time series it generates, how parameters affect these changes, and appropriate methods for fitting the models. In contrast, we showed how an understanding of a time-series model can instead be built up *empirically*, by analyzing the types of time series that it generates. Of course this should not be considered a substitute for conventional time-series modeling, but it can be quite a powerful complementary technique for understanding time-series models. For example, we showed how models and their parameters can be retrieved for new pieces of data, and how the types of data that a model might be able to fit can be retrieved from a large time-series database. While the examples presented are simple, they could be much more powerful on a larger scale, and could be applicable to very complex time-series models for which a theoretical analysis is intractable. This empirical analysis can also provide an intuition for understanding the connections between data and models that may otherwise be unattainable. We develop some of these ideas further and present a framework for analyzing empirical outputs of time-series models in Chapter 7.

Chapter 5

Applications to classification and regression tasks

In the previous chapter, we demonstrated a highly comparative empirical approach to time-series analysis by structuring diverse interdisciplinary collections of time series and operations. In this chapter, we present many specific applications of our highly comparative methodology to tasks in time-series analysis. The chapter is divided into three sections. In the first, Sec. 5.1, study time-series regression problems, and in the second, Sec. 5.2, we analyze time-series classification tasks. In Sec. 5.3 we present a more detailed case study in which both classification and regression techniques are applied to the problem of fetal heart rate rate monitoring.

5.1 Regression tasks

In this section we demonstrate a highly comparative method for selecting operations that exhibit some desired variation across a set of signals. The general method simply involves calculating linear correlations between the outputs of all operations and some quantity that has been assigned to each time series. The magnitude of the linear correlation coefficient is used to rank the operations, and we then investigate and interpret these linear predictors in the context of the given problem. Note that special values (cf. Sec. 3.2.4) are ignored in this process – for each operation, correlations are calculated across the range of operation outputs that are not special values. By comparing across thousands of operations, the highly comparative method allows automatic selection of relevant operations for general time-series datasets.

Rather than only considering linear correlations, mutual information-based mea-

sures could alternatively be used to select operations that show some possibly non-linear dependence on the target quantity. We could also extend the approach to search for combinations of operations that together vary strongly with the target variable. However, for simplicity, in this section we seek only linear correlations for single operations. This allows us to demonstrate the utility of our approach with maximal transparently, while leaving the door open for extensions of the approach in the future.

Because we are repeatedly testing the ability of individual operations to predict a target quantity, we are effectively performing multiple hypothesis testing, as explained in Sec. 3.4.6. In each of these case studies, we therefore compare the distribution of correlation coefficients obtained from operations acting on the time series to shuffled (null) distributions in order to assess the significance of our results. This allows us to verify that operations are quantifying real structure in the data rather than simply demonstrating an ability of a massive database of operations to statistically fit an arbitrary variation. Note that we do not analyze lower-dimensional projections of the data matrices studied in this section of the thesis, as this will be treated in detail for each of these datasets in Chapter 7.

We consider three regression problems separately in which we are able to successfully select operations that predict: (i) the signal-to-noise ratio of noisy periodic time series in Sec. 5.1.1, (ii) the Lyapunov exponent from Logistic Map time series in Sec. 5.1.2, and (iii) the scaling exponent of self-affine time series in Sec. 5.1.3. The same regression method is also applied to a real-world dataset of fetal heart rate time series in Sec. 5.3.

5.1.1 Noisy sine waves

In this section we seek operations whose outputs correlate with the variance of Gaussian-distributed noise added to a periodic signal. We begin with such a simple and easily interpretable problem to provide some intuition for our highly comparative method for time-series regression. The dataset is described in Sec. 5.1.1.1, and successful individual operations are described in Sec. 5.1.1.2.

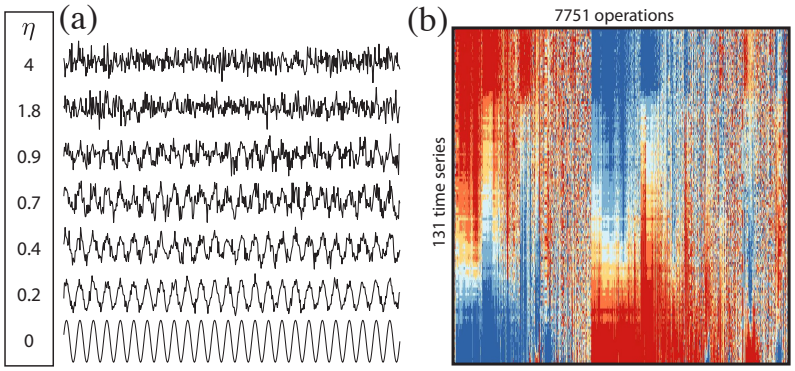

Figure 5.1: **The noisy sine wave dataset**. (a) 500-sample subsegments from selected time series are plotted, labeled by the standard deviation η of added Gaussian-distributed noise. The full dataset contains 131 time series generated using different values of η in the range $0 \leq \eta \leq 4$. (b) Patterns in the clustered data matrix for this dataset reflect changes in η across it. Most operations either show a steady increase from low values (blue) to high values (red) across the time series, or a steady decrease from high values (red) to low values (blue).

5.1.1.1 Data

We used a dataset containing 131 time series generated from the following simple model:

$$x_t = \sin(2\pi f t) + \eta_t, \tag{5.1}$$

where the frequency $f = 0.05\,\text{sample}^{-1}$ is fixed and $\eta \sim \mathcal{N}(0, \eta^2)$ represents independent draws from a normal distribution with zero mean and standard deviation η. Each time series contains 5 000 samples, i.e., for $t = 1, 2, ..., 5\,000$ in Eq. (5.1), and the single control parameter η is varied through the range $0 \leq \eta \leq 4$: for $\eta = 0, 0.01, 0.02, ..., 1$ and $\eta = 1.1, 1.2, ..., 4$. Examples of time series generated from Eq. (5.1) are shown in Fig. 5.1.

5.1.1.2 Individual operations

Linear correlation coefficients were calculated between the output of operations and the known noise standard deviation, η, that characterizes each time series. Their distribution across the 8 808 operations in our database with less than 20% special-valued outputs on this dataset are plotted as a red histogram in Fig. 5.2. As a comparison, the distribution of correlation coefficients obtained by randomizing the assignment of η to time series is plotted as a blue histogram in Fig. 5.2. The random

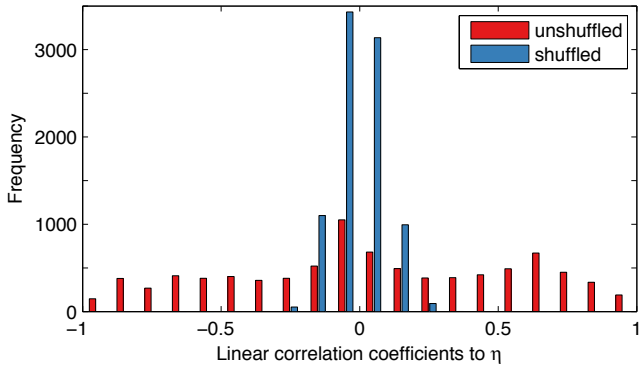

Figure 5.2: **Many operations in our database show a strong linear correlation to the standard deviation of added white noise, η.** The plot contains a histogram of the linear correlation coefficients calculated between the output of each operation and η across our dataset of 131 noisy sine waves, in red. A comparison to a shuffled version of the dataset, for which the values of η are assigned to time series in the dataset at random, is plotted in blue. The shuffling has been performed ten times, and we plot the average result. It is clear that many operations in our database exhibit linear correlations with the known noise standard deviation, η, beyond what would be expected by chance.

shuffling was done 10 times, and frequency counts are averaged across these 10 shuffles. Linear correlation coefficients obtained by shuffling are quite sharply distributed around $\eta = 0$, with a mean of 0.00 and a standard deviation of 0.09. It is clear that many of the operations exhibit significant linear correlations to η.

Given the dramatic change in the properties of the time series as η is varied, as depicted in Fig. 5.1(a), it not surprising that a large number of operations exhibit strong linear correlations with η. A benefit of this analysis is that these operations can be interpreted. A selection of the most successful operations for this task are plotted in Fig. 5.3. Many different types of time-series analysis operations exhibit correlations with η, including various measures of spread [Fig. 5.3(a)], summaries of the power spectrum [Fig. 5.3(b)], local correlation lengths [Fig. 5.3(c)], local predictability [measured by an exponential smoothing fit in Fig. 5.3(d)], a nonlinear autocorrelation [Fig. 5.3(e)], and entropy [Fig. 5.3(f)]. Each of these trends is interpretable in terms of the properties of the dataset: e.g., standard deviation increases with η [Fig. 5.3(a)], the peak of the power spectrum decreases with η [Fig. 5.3(b)], and measures of local correlation properties decrease with η [Figs. 5.3(c)–(e)]. Each of the operations shown in Fig. 5.3 are described in more detail and interpreted in the context of this problem

in Sec. B.1.1 of the appendix.

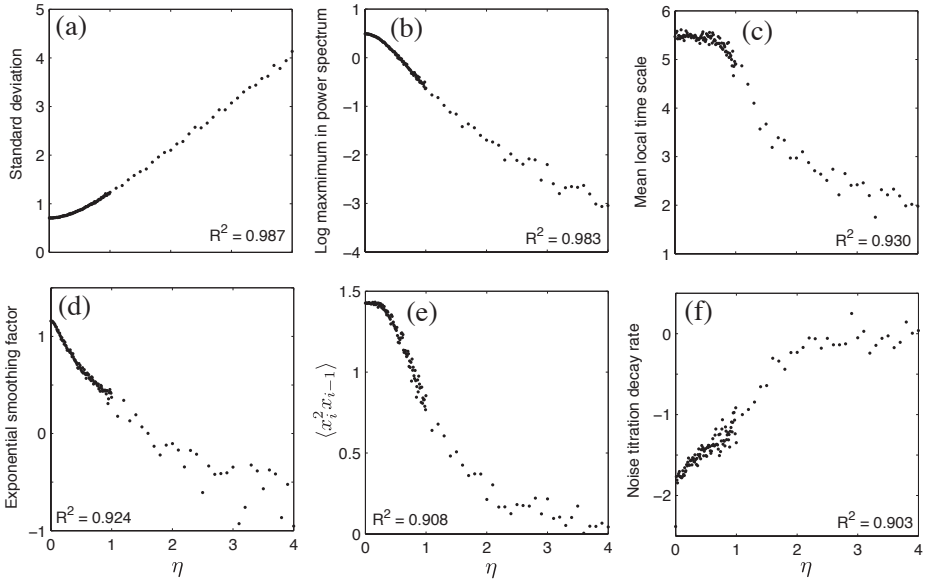

Figure 5.3: **Some operations predict the standard deviation of noise, η, added to sinusoidal time series**. Selected operations with high linear correlations to η are plotted. The coefficient of determination, R^2, measuring the strength of linear correlation between the outputs of each operation and η, is annotated to each plot. Operations span distributional summaries, measures of the power spectrum, correlation and model fit statistics, among others. Each operation is briefly described in each subplot, summarized in the main text, and described in detail in Sec. B.1.1 of the appendix.

5.1.2 Logistic Map time series

In this section, we search for operations with outputs that vary linearly with the Lyapunov exponent, λ, of Logistic Map time series. The Logistic Map is defined by the recursive relationship:

$$x_{n+1} = Ax_n(1 - x_n), \tag{5.2}$$

and is characterized by a single control parameter, A [12, 67]. The map has been studied extensively as the prototypical example of a simple nonlinear system that can produce a broad range of dynamical behaviors: from regular oscillations to unstable chaotic dynamics [12, 67]. We produced 159 Logistic Map time series for 53 different values of the parameter $A = 3, 3.2, 3.4$, and $A = 3.50, 3.51, 3.52, ..., 4.00$, each being generated using three different initial conditions x_1, chosen uniformly at random in the interval $0 \leq x_1 \leq 1$. Each time series is 10 000 samples long.

The Lyapunov exponent, λ, can be thought of informally as quantifying the 'amount of chaos' in a system: a bounded dynamical system with $\lambda > 0$ is chaotic, and λ encapsulates the average rate at which predictability is lost [12]. For a given A, the Lyapunov exponent was estimated numerically according to

$$\lambda \approx \log A \left\langle \log(A|1 - 2x|) \right\rangle_x , \tag{5.3}$$

where the average $\langle \cdot \rangle_x$ is performed over a 20 000-sample simulated Logistic Map trajectory, which was adequate to obtain a sufficiently converged estimate for our purposes [12]. Examples of Logistic Map time series, ordered by their Lyapunov exponent, λ, are plotted in Fig. 5.4.

Only those 8 417 operations that returned less than 30% special-valued outputs on this dataset were included in our analysis (cf. Sec. 3.2.4). This threshold was increased to 30%, from the nominal value of 20% used elsewhere in the thesis, to allow algorithms that give invalid outputs for $\lambda < 0$ to be included. As we will find below, many algorithms only yield a valid estimate of λ for $\lambda > 0$. Linear correlations are calculated between the operation outputs and the Lyapunov exponent, λ, calculated according to Eq. (5.3) for each time series (from the value of A used to produce it).

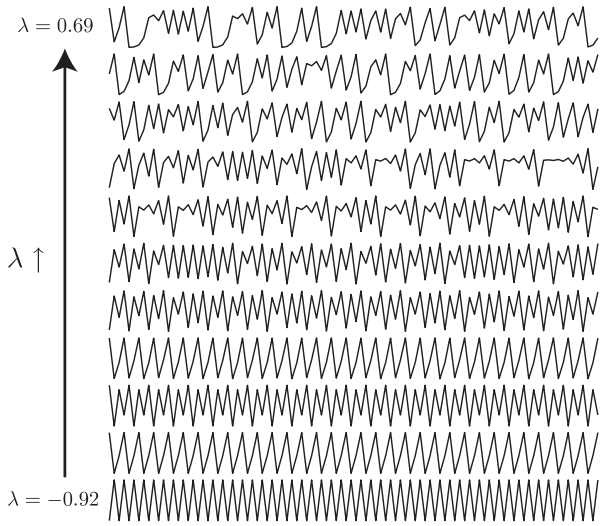

Figure 5.4: **Logistic Map time series ordered by their Lyapunov exponent λ.** For this task we generated a dataset containing 159 Logistic Map time series of length 10 000 samples using a range of values of the control parameter A. Operations with outputs that correlate linearly with λ are investigated in this case study. Each time series plotted here is a 100-sample segment of a time series in this dataset.

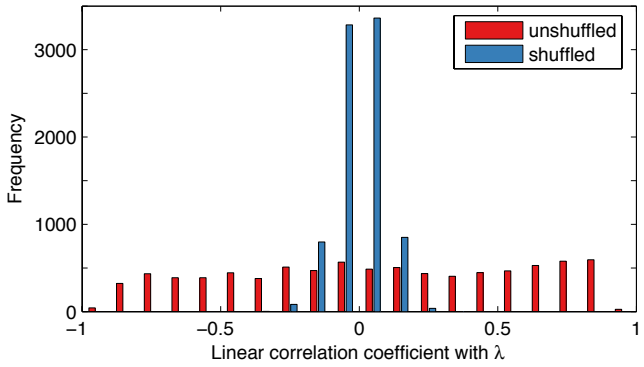

Figure 5.5: **Many operations exhibit a strong linear correlation with λ.** The plot contains a histogram of the linear correlation coefficients calculated between the output of each operation and the Lyapunov exponent, λ, of 159 Logistic Map time series, in red. A comparison to shuffled versions of the dataset, for which the values of λ are assigned to time series in the dataset at random, is plotted in blue.

5.1.2.1 Individual operations

In Fig. 5.5, a histogram of linear correlation coefficients calculated between the output of 8 417 operations and λ is plotted. Compared to the shuffled distribution, plotted blue, many operations in our database are sensitive to the change in dynamics associated with changes in λ, as illustrated in Fig. 5.4.

Given the nonlinear nature of this problem, we expect nonlinear time-series analysis operations to perform the best for this task [13]. In our database, some operations make use of algorithms from two existing packages for nonlinear time-series analysis: *TSTOOL*[1] and *TISEAN*[2] [92]. As described in Appendix F, we have implemented operations that process the time-series data, apply these algorithms, and then return a range of suitable results as outputs. Indeed, the usefulness of nonlinear time-series analysis methods is confirmed by our results: 91 of the 100 operations with the strongest linear correlations to λ are derived from nonlinear time-series analysis methods.

In Fig. 5.6, a sample of successful operations is displayed. In each subplot, operation outputs are plotted as a function of the theoretical Lyapunov exponent,

[1] A publicly-available set of MATLAB code for nonlinear time-series analysis available at `http://www.physik3.gwdg.de/tstool/index.html`

[2] A popular command-line package of algorithms for performing nonlinear time-series analysis. Code is available here: `http://www.mpipks-dresden.mpg.de/~tisean/Tisean_3.0.1/index.html`

λ, for the Logistic Map that generated the time series. The operations plotted in Fig. 5.6 include many operations from nonlinear time-series analysis: as expected, a largest Lyapunov exponent estimator [Fig. 5.6(a)], a simple local density estimate in a three-dimensional time-delay embedding space [Fig. 5.6(b)], various dimension estimates [Figs. 5.6(c)–(e)], an Approximate Entropy algorithm [Fig. 5.6(f)], and a noise titration-based operation [Fig. 5.6(g)]. The most successful linear time-series analysis method in the database is shown in Fig. 5.6(h), which returns a spectral flatness measure from a periodogram obtained from the time series. Like the operations above, its variation is sensible: chaotic time series with high λ produce disordered time series with many frequency components and hence a flatter spectrum than the more periodic time series with low λ. The behavior of each operation plotted in Fig. 5.6 is explained in detail in Sec. B.1.2.

We note that many of these operations implement classic nonlinear time-series analysis methods in an automatic way, which is particularly prone to error [13]. As mentioned previously, an understanding of the operations selected using our highly-comparative approach requires careful interpretation of the operations in the context of the problem at hand. We also note that many of these operations are detecting trends in λ that are peculiar to the Logistic Map; in future we could construct more diverse datasets of time series generated from a range of model systems with known λ, and repeating the regression task to find more general estimators of λ.

5.1.3 Self-affine time series

In this section, we construct a set of self-affine time series and seek operations that produce outputs that are linearly correlated with the scaling exponent that characterizes each time series. A summary of the theory behind self-affine time series and details of how we constructed the dataset is in Sec. 5.1.3.1, and individual operations with outputs that exhibit strong correlations with the scaling exponent are examined in Sec. 5.1.3.2.

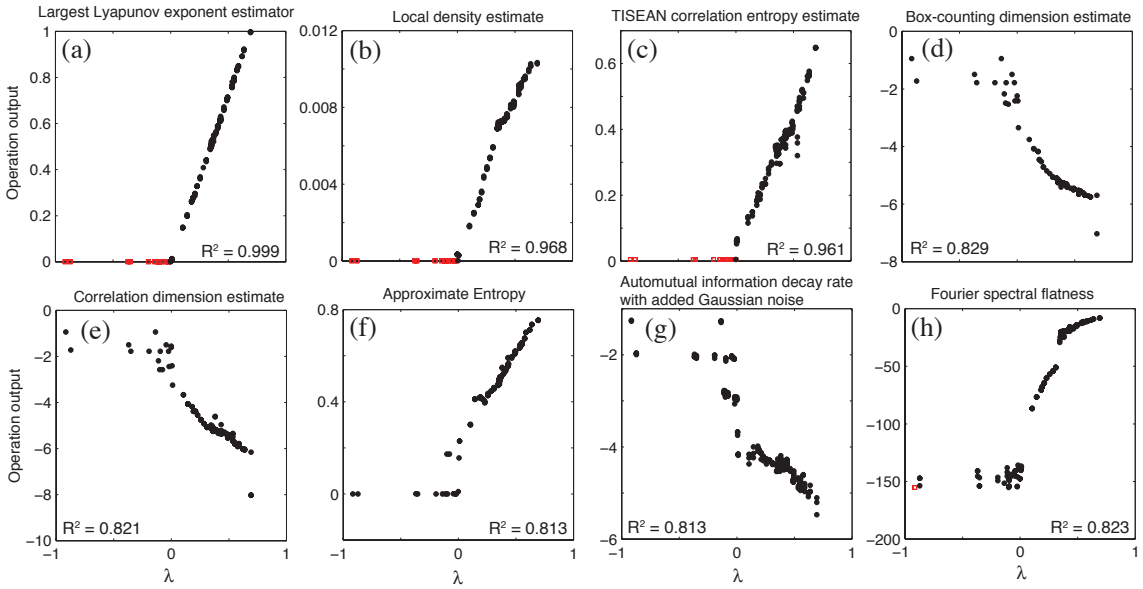

Figure 5.6: **Some operations correlate with λ across a dataset of Logistic Map time series.** A sample of operations that show strong linear correlations with the Lyapunov exponent, λ, of Logistic Map time series. Outputs of operations, briefly labeled in the title of each subplot, are plotted against the Lyapunov exponent, λ, of each Logistic Map time series in each case, with special values denoted by red squares and real-valued outputs denoted by black circles. Coefficients of determination, R^2, are annotated to each plot. These algorithms are summarized in the text and described in detail in Sec. B.1.2.

5.1.3.1 Data

Self-affine time series are self-similar after scaling by different factors in different dimensions [90]. As a consequence, their power spectral density scales as a power of their frequency [90]. For example, so-called *anomalous* diffusion processes obey

$$\langle \Delta x^2 \rangle \propto \Delta t^{2H}, \tag{5.4}$$

and are characterized by the *Hurst exponent*, H, where Δx correspond to increments in the time-series values, and Δt correspond to increments in time [13]. The power spectral density, $S(f)$, of self-affine time series goes as $S(f) \propto f^{-\alpha}$, where α is the *scaling exponent* of this power-law spectrum [146]. Self-affine time series can be divided into two categories: *fractional Gaussian noise* (fGN) time series, which have $-1 < \alpha < 1$, and *fractional Brownian motion* (fBM) time series, which have $1 < \alpha < 3$ [147]. Increments of a fBM series give a fGN time series, and fBM series can be produced by a cumulative summation of fGN time series. The Hurst exponent

relates to the power-spectral density scaling exponent by $H = (\alpha + 1)/2$ for fGN and $H = (\alpha - 1)/2$ for fBM. The Hurst exponent, H, can take values between 0 and 1; if successive increments for fBM (or successive values for fGN) are positively correlated, $H > 0.5$ (*persistence*), and if they are anti-correlated, $H < 0.5$ (*anti-persistence*) [13]. Many algorithms exist for estimating the Hurst exponent, and have been compared in previous work [147–150]. Some methods are valid only for fGN, and others only for fBM. Estimators of the scaling exponent of self-affine time series have attracted attention recently in, e.g., statistical and interdisciplinary physics, to find evidence for scale-invariance in real-world processes [151].

In this section, a dataset of fGN and fBM time series was generated with the aim of selecting operations that vary approximately linearly with the scaling exponent, α, across the full range of fGN and fBM: $-1 < \alpha < 3$. These time series are generated via two different methods. The first method, called the *Fourier filtering method* [90], constructs the desired scaling behavior in the frequency domain and then converts to time domain series via the inverse Fourier transform. The second method is the *random midpoint displacement method*, which is described in [90]. We implement publicly available code [152] to generate time series of length 5 000 samples. Examples of self-affine time series are plotted in Fig. 5.7. Based on existing work [147, 150], we expect methods based on fluctuation analysis, which have been used to infer scaling properties in real time series in the past, to perform well for this task.

5.1.3.2 Individual operations

In this section, we describe individual operations in our database that are good linear predictors of the power spectrum scaling exponent, α, across the dataset of self-affine time series described above. A histogram of correlation coefficients for each of the 8 672 operations with less than 20% special-valued outputs on this dataset is plotted in Fig. 5.8.

A selection of successful operations is shown in Fig. 5.9, each of which is described in detail in Sec. B.1.4. As expected, methods based on fluctuation analysis perform well, including various operations based on detrended fluctuation analysis, rescaled range analysis, and wavelet-based analysis fluctuation analysis [Figs. 5.9(a)–(e)], all

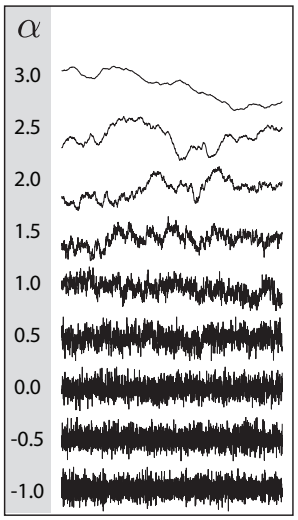

Figure 5.7: **A dataset of self-affine time series.** Self-affine time series can be characterized by a scaling exponent, α. We generated 451 self-affine time series by two different methods with scaling exponents in the range $-1 \leq \alpha \leq 3$.

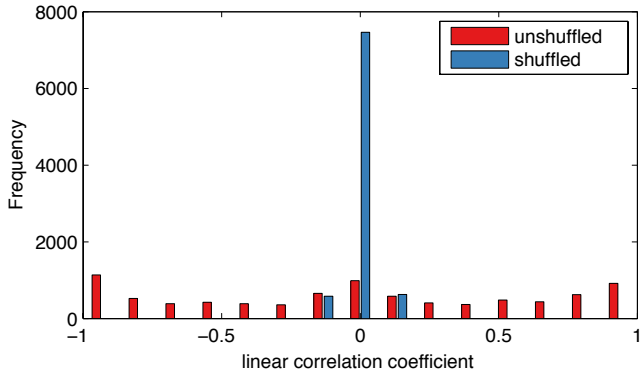

Figure 5.8: **Outputs from many operations in our library correlate strongly with the scaling exponent, α, of self-affine time series.** In this plot, the individual linear correlation coefficients between the outputs from 8 672 operations and the the known scaling exponent, α, of 451 self-affine time series are plotted as a red histogram. The histogram reveals that many of the operations exhibit correlations with λ that are more significant than would be expected by chance. The histogram obtained by shuffling the values of α at random is shown in blue as the average of ten independent shuffles.

of which vary approximately linearly with α. In addition to these fluctuation analysis-based operations, many model-fit based operations also exhibit strong correlations to α, including an autoregressive model parameter [Fig. 5.9(f)], a local mean predictor [Fig. 5.9(g)], a state-space model parameter [Fig. 5.9(h)], and a Gaussian Process model fit [Fig. 5.9(i)], which are picking up changes in the correlation properties of the time series with α (as shown in the time series plotted in Fig. 5.7). Some other operations are also selected, including the variability of local autocorrelation estimates [Fig. 5.9(j)], a summary of the trajectory of an imaginary particle that moves in response to the time series [Fig. 5.9(k)], and the frequency of a local motif 'CACB' in a symbolization of the incremental differences of the time series [Fig. 5.9(l)].

In this section we confirmed the utility of fluctuation analysis-based measures for estimating the scaling exponent in these finite-length, self-affine time series. We also highlighted a great diversity of different ways of summarizing time series that exhibit strong linear correlations to the scaling exponent, α, across a dataset of self-affine time series. In particular, we have shown that measures using autoregressive and state-space models, motifs, Gaussian processes, and quantities based on imaginary particles and residuals of a sliding mean predictor all show good performance. Many of these unusual estimators, while not as accurate as fluctuation analysis, are computationally much cheaper to perform, and some can be updated incrementally with the arrival of new data points, for applications in streaming data: e.g., Fig. 5.9(g) sliding mean residuals, Fig. 5.9(k) particle crossings, Fig. 5.9(l) 'CACB' motifs. The fundamental property of these self-affine time series is that they are self-similar, displaying analogous dynamics on all scales—thus this theme of operations that extract information at a range of scales makes sense for this task. Reflecting the multi-scale nature of these self-affine time series, many of the more successful operations combine local information with some global process, e.g., Fig. 5.9(f): residuals are calculated by local mean predictions, but global information is added through autocorrelation of the residual time series, in Fig. 5.9(g) the current position of the walker is conditioned on all the past values of the time series, in Fig. 5.9(j), we calculate many local autocorrelations and look at the spread of these across the time series, and in

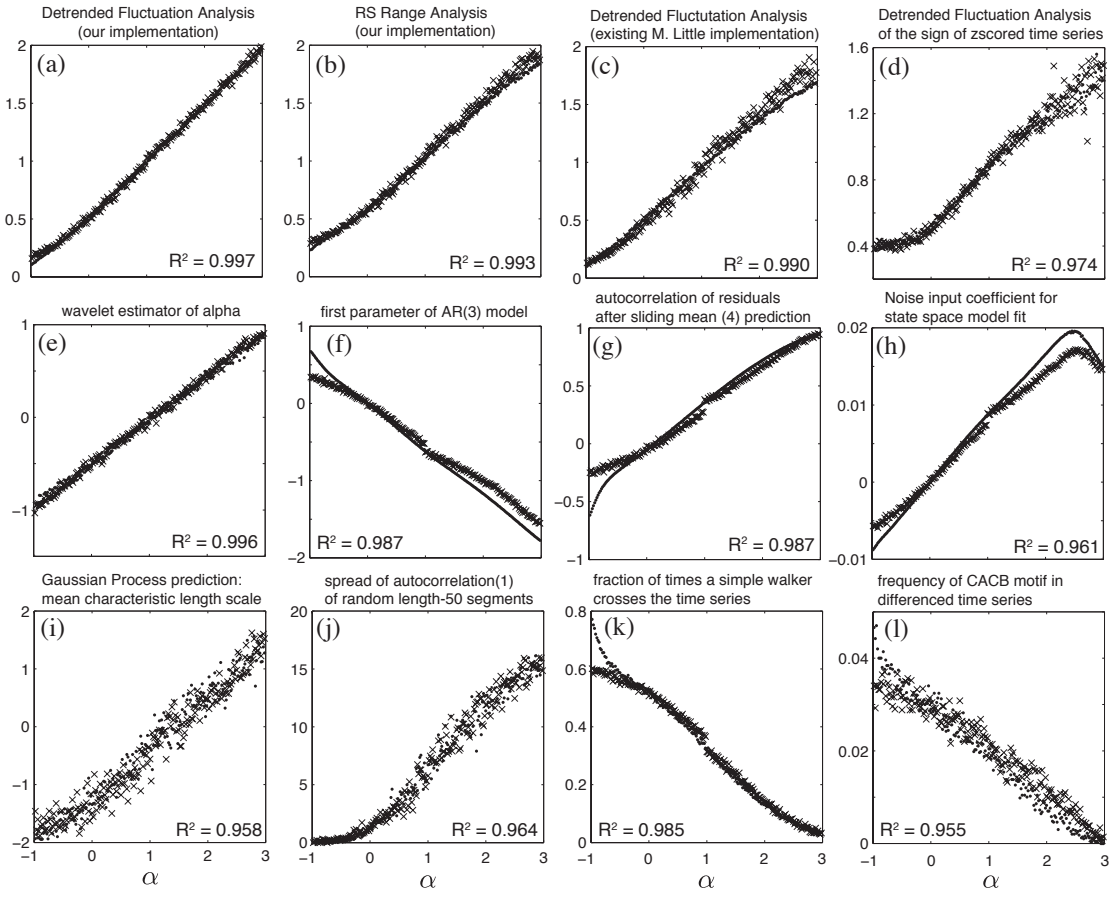

Figure 5.9: **Selected operations that correlate with** α, the scaling exponent of fGN and fBM fractal time series generated by the Fourier filtering method (dots) and the random midpoint displacement method (crosses) [90]. Brief descriptions of each operation are given in the title of each subplot, and their raw outputs are plotted against α across the range $-1 < \alpha < 3$. Coefficients of determination, R^2, are annotated to each plot. Fluctuation analysis methods, model fit metrics, and other types of time-series summary statistics exhibit strong linear correlations with α. Each operation in this plot is described in detail in Sec. B.1.4.

Fig. 5.9(l) we use the distribution of values across the whole time series to determine bins for labeling time series points as symbols 'A', 'B', or 'C' and then measure the probability of the motif 'CACB'. This is an example of how our methodology can highlight a range of unexpected and interesting scientific results because it makes no theoretical assumptions, and selects operations based only on their empirical behavior. The results may be able to stimulate the development of new theory in the future to understand how these unusual operations are able to inform the scaling exponents of these time series.

5.1.4 Discussion

Choosing appropriate techniques for a time-series analysis task by automatically comparing the performance of a large set of interdisciplinary operations on a specially-designed dataset of time series is a novel concept demonstrated in this section. Rather than relying solely on theoretical reasoning or a 'data analyst's intuition', here we take a purely empirical approach to selecting useful operations and subsequently use theoretical and domain knowledge to interpret the results. We have demonstrated this procedure with three simple synthetic examples here, but the method is general and could be extended to more subtle problems straightforwardly: by constructing appropriate empirical datasets. For example, white noise could be added to the time series studied in this section so that operations are retrieved that can also be effectively applied to noisy time series. Further, rather than developing theory to understand the impact of non-Gaussian noise distributions on operations for estimating the scaling exponents, a desired noise distribution can simply be added to the time series. In the same way, linear trends, or other types of non-stationarities can be accommodated by our general approach. By constructing well-motivated calibration sets of time series and comparing the performance of a large number of time-series analysis operations, we are therefore able to retrieve useful operations for a given time-series analysis task automatically.

The method can also be applied to sets of real-world time series, one example of which is shown in Sec. 5.3 for fetal heart rate time recordings. For example, we may wish to estimate a patient's age from an electrocardiogram, or the location on Earth where a rainfall time series was measured. By preparing a calibration set of real-world time series with a known target value assigned to each (e.g., the age of the patient that the ECG was recorded from or the latitude and longitude of the position on Earth that a rainfall time series was measured), we can repeat the same method proposed in this section to select empirical estimators of the target variable automatically for problems where there is no obvious theoretical motivation for performing the selection.

5.2 Classification tasks

In this section, datasets containing different classes of time series are analyzed using our highly comparative methodology. All of the real-world time-series datasets are publicly-available, and all synthetic data sets have been generated by us. Six datasets are analyzed in total: the discrimination of random number sequences from periodic signals in Sec. 5.2.1, earthquake and explosion signals in Sec. 5.2.2, human speech recordings spoken with different emotions in Sec. 5.2.3, five classes of electroencephalograms (EEGs) in Sec. 5.2.4, speech phonemes from healthy controls and patients with Parkinson's disease in Sec. 5.2.5, and heart-beat interval series of normal sinus rhythm and patients with congestive heart failure in Sec. 5.2.6. Our database of operations is used to explore structure in each dataset and to distinguish operations that measure differences between the labeled classes of time series most effectively. In each problem we demonstrate different characteristics of the highly comparative approach, including how to evaluate significance using multiple hypothesis testing for small datasets (seismic data in Sec. 5.2.2), how to treat multi-class classification problems (emotional speech data in Sec. 5.2.3), and how to build classifiers that combine the outputs of multiple operations using feature selection (Parkinsonian speech data in Sec. 5.2.5). In many cases, the collective behavior of the large interdisciplinary set of time-series analysis operations, judged by a reduced Principal Components (PC) projection, effectively distinguishes the classes and reproduces the labeled structure in the dataset, a result noted throughout this section, and explored further in Chapter 7.

5.2.1 Random number sequences and periodic signals

One of the most basic time-series classification tasks involves distinguishing uncorrelated sequences of random numbers from periodic time series that are regular and strongly autocorrelated. In this case study, we examine 175 random number sequences and 175 periodic time series, selected at random from the database. More information about this dataset is in Sec. B.2.1 of the appendix. Since we have a very clear knowledge of the differences between the two classes, we use this initial dataset to

develop an intuition for our methods: revealing structure in the dataset using PCA (in Sec. 5.2.1.1), and selecting interpretable features that best distinguish the two classes (in Sec. 5.2.1.2).

5.2.1.1 Unsupervised analysis

Given the dramatic difference between random number sequences and periodic signals, we expect many operations in the database to be sensitive to the differences between the two classes, which is confirmed in Fig. 5.10. The data matrix contains 6 863 operations (those with no special-valued outputs and a non-zero interquartile range for this dataset, and also excluding any location-dependent, spread-dependent, or length-dependent operations) is plotted in Fig. 5.10(a). The data matrix shows that the outputs of many operations are relatively uniform within each group but very different between the groups. The dendrogram, calculated using the Euclidean distances between time series in this space clusters time series into their known group identities: periodic signals and random number sequences. Segments of time series are annotated next to the data matrix in Fig. 5.10(a).

The two-dimensional PC projection of the data matrix is shown in Fig. 5.10(b). Periodic and random series are perfectly separated by the first PC, as shown in the upper distribution plot. This leading PC reflects the dominant covarying behavior of operations in the database, and represents the automatic inference of an algorithm that accurately distinguishes these two classes of time series. Operations that most contribute to the first PC include fitting AR and ARMA models, conducting seasonality tests, local mean prediction, estimates of the smoothing parameter, α, in Exponential Smoothing, measures from the power spectrum, fitting periodic models, etc. Therefore, the behavior of the first PC primarily reflects the behavior of correlation-based operations, which explain the most variance in this dataset. This result, which makes sense in the context of this dataset, was obtained simply by performing dimensionality reduction on the data matrix.

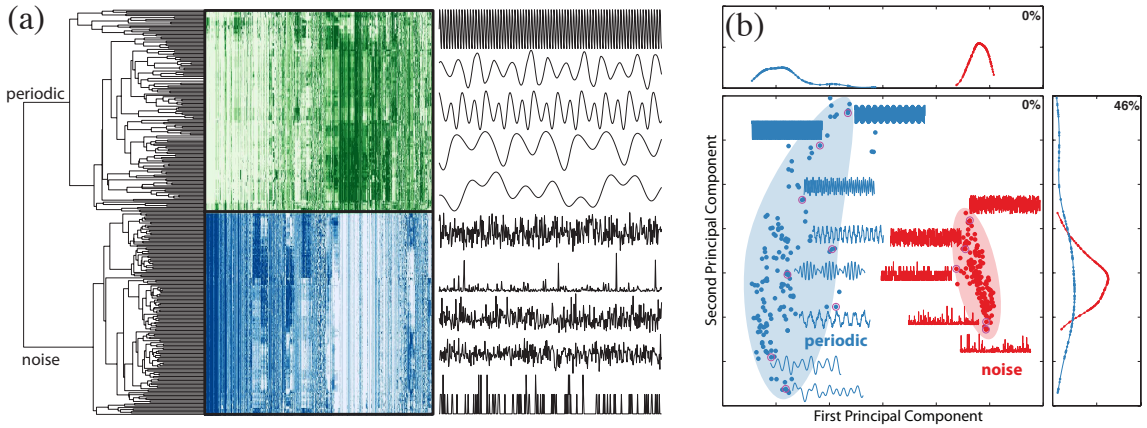

Figure 5.10: **Random number sequences are automatically distinguished from periodic signals using our library of operations.** (a) The clustered data matrix and dendrogram with annotated time series. Periodic signals are colored green, and random number sequences are colored blue, from low outputs (white) to high outputs (dark). For selected time series, 400-sample segments are annotated adjacent to the rows of the data matrix that represent them. (b) A two-dimensional Principal Components (PC) projection of the uncorrelated noise (red) and periodic (blue) time series. The first PC separates the two classes perfectly. Background shading has been added manually to aid the eye, and 500-sample segments of selected time series have been annotated to the plot.

5.2.1.2 Individual operations

We calculated 10-fold cross validation linear classification rates for each of the 7 210 individual operations with no special-valued outputs on this dataset. The distribution of misclassification rates is plotted as a red histogram in Figure 5.11, and is clearly classifying the two types of time series with an accuracy that greatly exceeds what could be expected by chance.

Some examples of individual operations that perfectly separate random number sequences from periodic signals are plotted in Fig. 5.12. The operations include entropy measures that give higher outputs for the uncorrelated time series [Figs. 5.12(a)–(d)], the goodness of fit of a periodic time-series model, which sensibly gives high values to the periodic time series [Fig. 5.12(e)], and a measure of the peak in the Fourier power spectrum, which is higher for periodic time series and lower for the flat, white noise spectra of the random number sequences [Fig. 5.12(f)]. Note that each operation shown in Fig. 5.12 is described in detail in Sec. B.2.1.

All of the above results are sensible and interpretable given the stationary autocor-

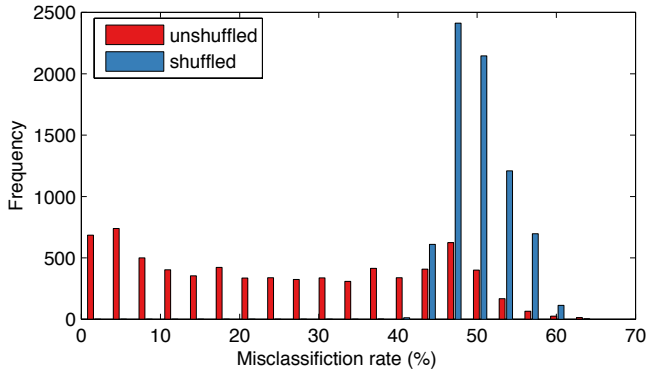

Figure 5.11: **Histogram of misclassification rates for the noise/periodic dataset across our library of operations.** Across the 7 210 operations with no special-valued outputs on this dataset (red), we find that many can distinguish these two types of time series better than chance (blue). Classification errors are determined as the mean of a 10-fold linear discriminant analysis, repeated with 10 different splits of the data to reduce the variance. The shuffled version shuffles the labels on the data at random and then calculates each classification error; the process is repeated 10 times, cf. Sec. 3.4.6 for details. The operations that most strongly contribute to the first Principal Component are mostly related to autocorrelation properties.

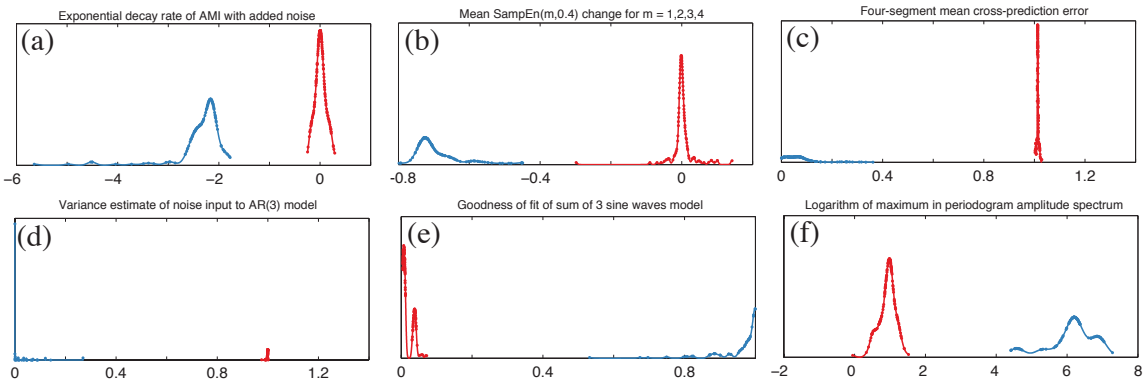

Figure 5.12: **Selected operations that distinguish periodic processes from sequences of random numbers with a cross-validation linear misclassification rate of zero.** In each case the distribution of outputs from the target operation is plotted for the random number sequences (red) and periodic signals (blue). Autocorrelation measures and entropy-based estimates dominate the list of most successful operations. Detailed explanations of each operation are in Sec. B.2.1.

relation structure present in the periodic processes compared to the random number sequences that exhibit minimal autocorrelation (which arises only by chance). The results of this section demonstrate, using two simple groups of synthetic time series, how our highly comparative approach can reproduce meaningful structure in datasets (by clustering the data matrix and analyzing the leading PCs) and retrieve estimators of interpretable structural differences between labeled groups of signals automatically. In subsequent case studies in this section, interesting insights into more challenging datasets will be gained through analogous analyses.

5.2.2 Seismic activity

In this section we aim to distinguish explosion and earthquake time series, and to classify an unknown event. The problem is used as a proxy for monitoring a nuclear test ban treaty, for which it is crucial to be able to discriminate mere earthquakes from nuclear explosions [153, 154]. Because this dataset contains only eight examples of each class, and we are independently testing the ability of thousands of operations to distinguish them, we might expect our highly comparative approach to be inappropriate for this task. We thus use this case study to demonstrate the multiple hypothesis testing framework described in Sec. 3.4.6, which allows us to evaluate the significance of our results. The time-series data are described in Sec. 5.2.2.1, the ability of individual features to classify the two sets is explored in Sec. 5.2.2.2, and two-dimensional PC projections of the data are analyzed in Sec. 5.2.2.3.

5.2.2.1 Data

This dataset consists of eight known explosion time series, eight known earthquake time series, and an unknown event, which took place near the Russian nuclear test facility in Novaya Zemlya [154]. All time series are 2 048 samples long. The dataset, which contains regional (i.e., 100 – 2 000 km) recordings of typical Scandinavian earthquakes and mining explosions, was constructed originally by Blandford [155], and is publicly-available[3]. The sixteen events were listed and combined with the unknown event by Kakizawa et al. [156]. The problem was discussed in detail in

[3]The dataset can be downloaded here: `http://lib.stat.cmu.edu/general/stoffer/tsa2/`

Shumway and Stoffer [153] and has been studied since [154]. Previous analyses have classified the unknown event as an explosion [153, 154, 156].

5.2.2.2 Individual operations

We calculated the linear classification rates of all 8 586 operations that produced no special-valued outputs across this dataset. Classification rates were computed as the mean of ten repeats of 2-fold cross-validation and summarize the ability of each individual operation to separate the two classes of time series. Note that 2-fold cross validation was chosen instead of the more usual 10-fold cross-validation in this case because of the small size of this dataset. Histograms of these misclassification rates obtained using the correctly labeled dataset, and using randomly-permuted class labels are shown in Fig. 5.13. The shuffled distribution was obtained by generating 500 permutations of outputs from the 8 586 operations (i.e., obtaining 4 293 000 'randomized' linear classification rates) using the method outlined in Sec. 3.4.6. The two distributions overlap significantly, and the tail of the shuffled distribution extends to low classification rates, indicating that even when the information contained in the labels is shuffled, some operations can still distinguish the two classes by chance. We use our multiple hypothesis testing framework to evaluate the significance of operations acting on the correctly-labeled data, using the shuffled data as a comparison. The false discovery rate for features with a linear classification error of 0% was calculated to be 0.012, of which there are 15 operations; i.e., the expected proportion of these 15 operations that are incorrectly called significant is 0.012, or approximately 0.2 operations. Misclassifying a single time series corresponds to an error of 6.25%, and has a false discovery rate of 0.032, of which there are 98 operations; i.e., approximately three of these 98 operations are expected to be judged significant by chance. Given that thousands of operations are tested independently on a single, small time-series dataset, these statistical calculations allow us to assess the usefulness of particular operations. It is clear that we have some statistical power to be confident in the significance of especially the set of operations with 0% classification error, as the expected number of these fifteen operations that are separating the time series by chance is less than one. We note that the false discovery rates calculated

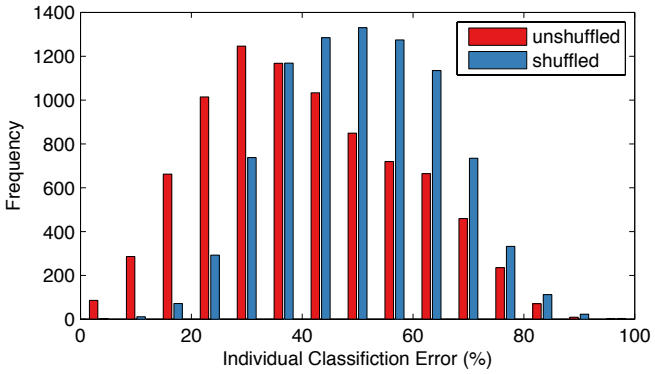

Figure 5.13: **Histogram of classification rates for classifying explosions and earthquakes across operations in our library**. The 8 586 operations with no special-valued outputs and a nonzero interquartile range are each individually tested using two-fold cross validation linear discriminant analysis. Two-fold cross validation was chosen instead of ten-fold cross validation because there are only eight examples of each class of time series. The distribution for these operations using the correctly-labeled data is shown in red, and for ten random permutations of class labels in blue. Although the distribution for the data as labeled is different to that for the shuffled data, because there are so few time series in this dataset—only eight from each class, we use multiple hypothesis testing to quantify the significance of operations for this classification dataset.

here are much lower than what is tolerated in genetic microarray studies, for example [82], indicating that our highly comparative approach can be used to select relevant operations for classification tasks, even for very small datasets. In future we could use the selected operations to understand the types of time-series properties that differ between earthquakes and explosions; this process is not followed here, but will be covered in detail in the remaining case studies in this chapter.

5.2.2.3 Unsupervised structure

In this section, we focus on an unsupervised analysis of the dataset, which does not involve multiple hypothesis testing, nor do we risk over-fitting, as the data labels are not used. A two-dimensional PC projection of the dataset is plotted in Fig. 5.14. In this representation, explosions are placed towards the left of the plot, and earthquakes towards the right—i.e., the first PC is contributing to separating the two types of time series. The unknown event (plotted green) is suggested to be an explosion in this representation.

Apart from the PC projections, we can also calculate pairwise distances between

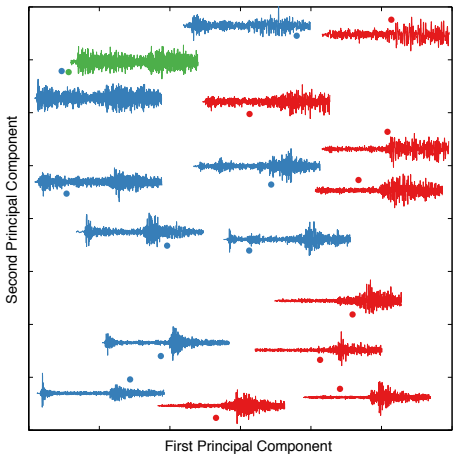

Figure 5.14: **The two leading Principal Components (PCs) organize the seismic dataset in a way that suggests that the unknown event is an explosion**. Eight explosions (red), eight earthquakes (blue), and an unknown event (green) are included. The first PC explains 30% of the variance, and the second PC explains 18% of the variance in the set of 8 586 operations applied to this dataset. Full 2048-sample time series are annotated to each point.

all time series in the dataset and use the resulting structure to organize the dataset Using a reduced set of 200 operations (described in Sec. 4.1.2), we show two such representations of the data: as a dendrogram in Fig. 5.15(a), and as a network in Fig. 5.15(b). In both cases the time-series data are organized in a sensible way, with the unknown event classified as an explosion. Note that these results are not strongly dependent on the number of operations used—we obtained qualitatively similar results using reduced sets of 10, 50, and using all 8 639 operations.

In this section we showed that a highly comparative approach can be applied to small datasets, using multiple hypothesis testing to evaluate the significance of the results. Regardless of the statistical power of any supervised analysis, we also showed that unsupervised representations can be used to structure the data meaningfully. The unknown time series is classified as an explosion in our analysis, consistent with the results of previous studies [153, 154, 156].

5.2.3 Emotional speech

In this section we analyze a dataset of German phrases spoken by actors using different emotions. The dataset contains seven different emotional classes of recordings, and we will consequently focus on the application of highly comparative analysis to multi-

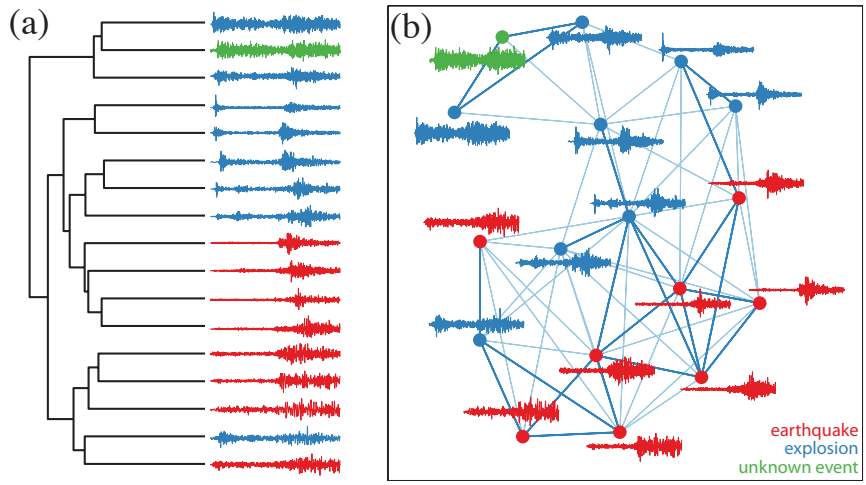

Figure 5.15: **Structure in a dataset of seismic recordings.** The data is represented as (a) a dendrogram, and (b) a network, where time series are labeled as either earthquakes (blue), explosions (red), or an unknown event (green). Euclidean distances are calculated between feature vectors containing outputs from a set of 197 operations with no special-valued outputs from the reduced set of 200 distilled in Sec. 4.1.2. Full 2048-sample time series are plotted in each case. The unknown event clusters with explosion time series, suggesting that it is itself a recording from an explosion, in agreement with previous analyses [153, 154, 156]. The network is fully connected but only the strongest links are plotted: those that represent Euclidean distances less than 5 units (thin lines) and 6 units (thick lines).

class classification problems in this case study. Our generic library of time-series analysis operations does not contain operations tailored to the specific speech analysis problem at hand, yet still allows us to classify the recordings successfully. The dataset is described in Sec. 5.2.3.1, differences between the various classes of speech signals are explored in Sec. 5.2.3.2, Principal Components projections of balanced subsets of data are analyzed in Sec. 5.2.3.3, and relevant individual operations and useful combinations of operations are used to classify the full dataset in Sec. 5.2.3.4.

5.2.3.1 Data

For this case study, we analyze a publicly-available dataset of speech utterances[4], developed by the Technical University of Berlin [134]. Ten different German phrases (five short and five long) were spoken by ten different actors (five males and five females) using seven different emotional states: 'neutral', 'happy', 'angry', 'sad', 'boredom', 'fear', and 'disgust'. The 535 utterances were recorded at a sampling

[4]Available at `http://www.expressive-speech.net/emodb`

frequency of 48 kHz and later down-sampled to 16 kHz. The length of these audio files makes them too computationally expensive to analyze in full and so we further down-sampled each audio recording by a factor of two to 8 kHz using the MATLAB function `resample(x,1,2)`, where x has been imported directly from the audio wave file. The length of time series that can be analyzed is limited by the computation time; we restrict ourselves to time series no longer than 30 000 samples (or 3.75 s) for this analysis, as depicted in Fig. 5.16. This leaves 67 (/71 total) 'happy', 77/79 'neutral', 115/127 'angry', 28/62 'sad', 68/69 'fear', 68/81 'boredom', and 33/46 'disgust' recordings to analyze.

The dataset has been used in a number of existing studies targeting the prediction of the emotional content of speech signals [157–160]. Note that most speech analysis operations use sliding windows across the data to accommodate the length of the signals [159]. By contrast, we investigate what insights can be gained into the structure of the data using a large sample of generic algorithms for time-series analysis applied to full speech recordings.

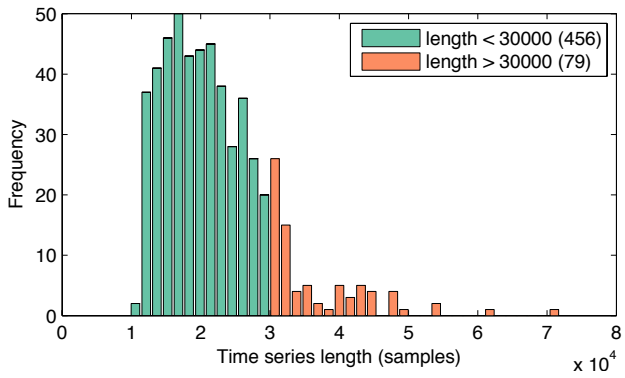

Figure 5.16: **Distribution of lengths of emotional speech time series.** The time series have been downsampled to 8 kHz and only the 456 time series of length less than 30 000 samples (or 3.75 s) are analyzed here, due to computational constraints (cf. Sec. 3.2.5). The other 79 recordings are not included in our analysis.

5.2.3.2 Class structure

To investigate class structure in this dataset, we chose 28 recordings at random from each of the seven emotional states, forming a dataset of 196 time series in total. A reduced set of 200 operations developed in Sec. 4.1.2, of which 194 produce less than 20% special-valued outputs on this dataset, are used to calculate Euclidean

distances between all pairs of time series, as shown in Fig. 5.17. Members of each class are relatively similar to other members of the same class, and three main groups are formed: (i) 'anger' and 'happiness', (ii) 'sadness', and (iii) 'boredom', 'neutral', 'disgust', and 'fear'. The dendrogram between class centers of each group, plotted above the pairwise distance matrix in Fig. 5.17, reflects this grouping. The 'sad' speech signals are particularly distinguished from the other emotions, and especially from the 'happy' and 'angry' recordings.

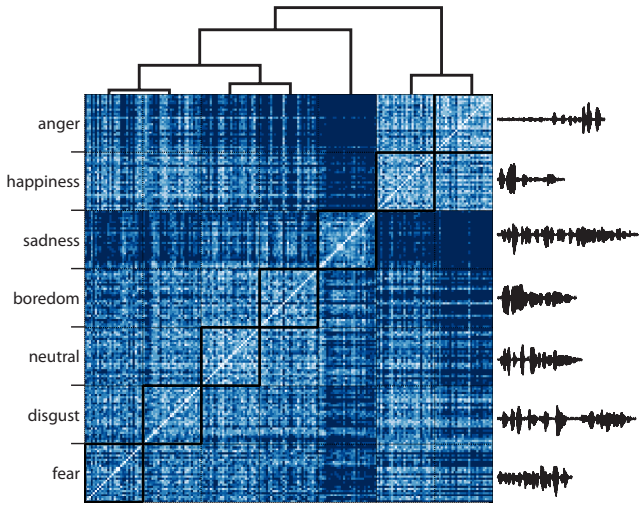

Figure 5.17: **Relationships between classes of emotional speech signals.** A pairwise Euclidean distance matrix calculated between feature vectors containing a reduced set of 194 operations is plotted for 28 randomly-selected examples from each of the seven classes: 'anger', 'happiness', 'sadness', 'boredom', 'neutral', 'disgust', and 'fear'. The distance matrix is visualized using color, which ranges from a Euclidean distance of $d \leq 3$ (white) to $d \geq 6$ (dark). All 28 examples from each labeled emotional class are grouped manually, but the groups have been reordered according to the dendrogram computed between class centers using average linkage clustering, which is annotated above the figure. Single examples of time series from each class are also annotated to the right of the plot.

5.2.3.3 Principal Components projections

In this section, we show that two-dimensional PC projections of balanced datasets can inform differences between classes of emotional speech recordings. We choose two cases: a two class balanced classification between 76 recordings each of 'happy' and 'boredom' types in Fig. 5.18(a), and a three class balanced classification between 28 examples each from 'happy', 'sad', and 'boredom' classes in Fig. 5.18(b). In each case,

the number of time series retrieved was limited by the class with the fewest examples, and time series were retrieved at random from the database. All operations with no special-valued outputs and non-zero interquartile range are retrieved from our library, and the data is plotted in a space of the first two PCs of the resulting data matrix. In Fig. 5.18(a), the first PC alone is sufficient to separate the two classes, whereas in the second example in Fig. 5.18(b), the second PC aids in the discrimination of the boredom and sadness classes. In both cases the PC projections provide a meaningful space in which to distinguish these classes of emotional speech.

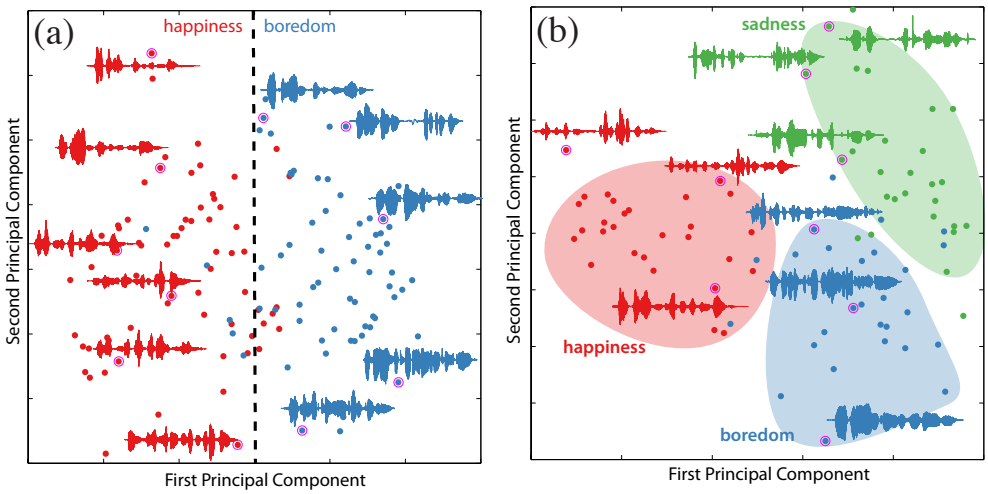

Figure 5.18: **Two-dimensional Principal Components projections organize speech signals by their emotional contents.** (a) 'Happy' and 'bored' speech signals, and (b) 'happy', 'bored', and 'sad' speech signals. Selected time series have been annotated to the plot. The optimal linear classification boundary is plotted as a dashed line in (a). Background shading in (b) has been added manually to aid visualization.

5.2.3.4 Training classifiers

Having shown how unsupervised representations of the data can reproduce basic aspects of the known class structure in the dataset, in this section we demonstrate how the class labels can be used to learn classifiers for this multi-class dataset. We treat the full dataset of 456 speech recordings which contains 67 'happiness', 77 'neutral', 115 'anger', 28 'sadness', 68 'fear', 68 'boredom', and 33 'disgust' speech signals[5]

[5]As described in Sec. 5.2.3.1, the original dataset contained 494 recordings, and only those that were sufficiently short (these 456 with less than 30 000 samples after downsampling) are analyzed here (cf. Fig. 5.16).

Linear classification rates were calculated for all 8 087 operations for this multi-class classification problem. As described in Sec. 3.4.3, linear classification thresholds are learned between all pairs of classes (in this case $7\times6/2 = 21$), and the new sample is evaluated according to all of these thresholds and classified as the class that received the most 'votes'. Although the classes are unbalanced in this problem, we evaluate each operation using the simple classification rate statistic, defined as the proportion of correct classifications. This statistic could be altered in future work to take the class imbalance into account, but in this section our aim is simply to demonstrate an ability to retrieve relevant operations for this task.

Three selected operations with the lowest individual misclassification rates is shown in Fig. 5.19. Note that the mean ($\pm$ standard deviation) misclassification rate of linear classifiers trained using the same 8 087 features but using 200 random permutations of the class labels was 84 (±1)%. By contrast, the misclassification rates achieved by the best operations are significantly lower—less than 60%. The plots in Fig. 5.19 immediately provide some intuition for the types of properties that differ between the classes of emotional speech recordings. For example, Fig. 5.19(a) shows that anger and happiness time series have a lower interquartile range (and hence are on average softer) than the other classes. From Fig. 5.19(b), we learn that sadness time series have less energy in a particular (low) frequency bin (in the spectrum of logarithmic deviations from the mean), and from Fig. 5.19(c), we see that the sadness time series also have a higher automutual information at lag 6 than the other time series.

As well as individual features, we can also search our library for pairs of features that accurately separate the classes. Two examples are shown in Fig. 5.20. The first, in Fig. 5.20(a), combines two operations: (i) the limiting autocorrelation at lag 4 calculated by randomly permuting samples in the the time series, and (ii) the probability of an increase in the time series. Although both of these operations achieve individual misclassification rates exceeding 60%, in combination, the mean misclassification rate is reduced to just 48%. The second classifier, shown in Fig. 5.20(b), combines two different operations: (i) the R^2 goodness of fit of a quadratic fit to the cumulative power spectrum (estimated using a periodogram with a Hamming

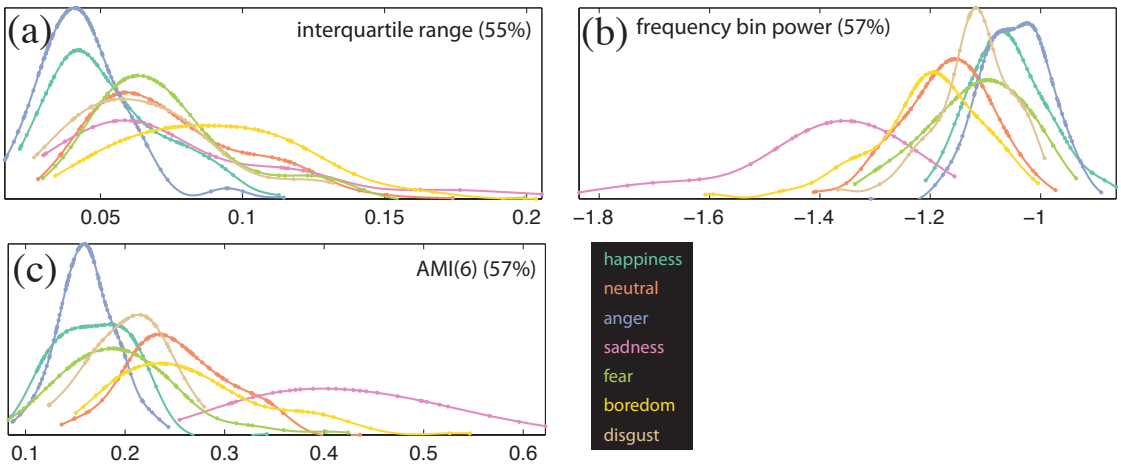

Figure 5.19: **Some operations show a strong ability to distinguish different classes of emotional speech recordings.** The distributions of outputs from three operations are plotted for each of the seven classes, as labeled. In each subplot the operation is labeled along with the in-sample linear misclassification rate, in parentheses.

window), and (ii) the ratio of the length of the longest consecutive string of increases to the longest consecutive string of decreases in the time series. Although this second operation in particular has a relatively high individual misclassification rate (71±5%), it works well in combination with the power spectrum measure, attaining a mean misclassification rate of 46%. All of these operations are individually interpretable, e.g., from the horizontal axis of Fig. 5.20(b), we learn that the cumulative power spectrum of anger recordings show the best fit to a quadratic function ($R^2 > 0.8$).

Finally, we show that classifiers combining multiple operations can be constructed using the greedy forward selection procedure described in Sec. 3.4.5. Mean out-of-sample misclassification rates are plotted as a function of the number of features in the classifier in Fig. 5.21. The mean classification error improves as operations are added up to the ten operations shown, where a mean misclassification rate of 40% is attained.

In future work, these classifiers could be investigated in more detail in the context of the existing speech classification literature. Humans are able to recognize the different emotions with an accuracy ranging from 79.6% for 'disgust', to 96.9% for 'anger' [134]. A variety of classification rates have been reported for this dataset: 50–65% [158], 65%–79% [157], approximately 75% [160], 59%–69% [159], 86.7% [161],

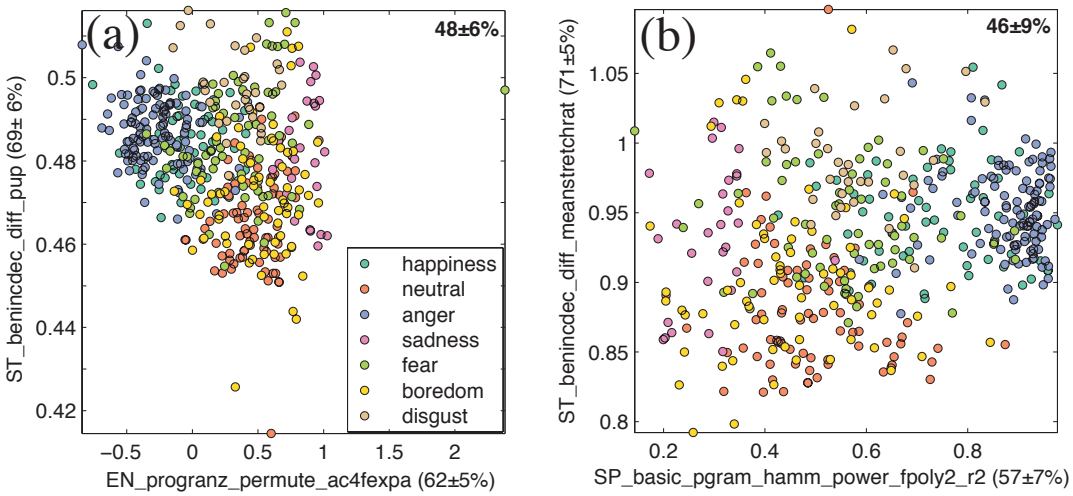

Figure 5.20: **Pairs of features that best classify the emotional classes.** Here we consider select the best linear classifiers by comparing across all possible pairs of operations. By combining complementary operations, 10-fold cross-validation misclassification rates are reduced (given in parentheses in each axis label, and for the combined classifier in bold in the top right of each plot).

and 55%–76% [162]. In these studies, a variety of complex classifiers have been implemented, from SVMs to artificial neural networks, and using a variety of techniques and large sets of features related to audio analysis. The range of classification rates reported in the literature is due to different sets of features, different classifiers, and different evaluation methods. Indeed it is difficult to compare classification studies even on the same dataset, because of differences in the evaluation scheme – some may be over-fitting features to the structure of the data, or optimizing parameters of the classifiers for the particular task. The studies above also use slightly different versions of the database, e.g., some contain 494 utterances, others use a smaller set of 488 [161]. Our dataset is comparable to the original dataset, but differs slightly, because it contains only 456 of the 535 utterances contained in the original set (those that were short enough for us to analyze using our full library of operations, cf. Fig. 5.16), and we have further down-sampled all audio recordings in the dataset from 16 kHz to 8 kHz. We expect our results to be pessimistic compared to what would be obtained using the full original dataset for three main reasons: (i) approximately one third of the filtered recordings (34/103) are from one class: 'sadness', which is the most distinct and hence the easiest class to distinguish, as has been found in

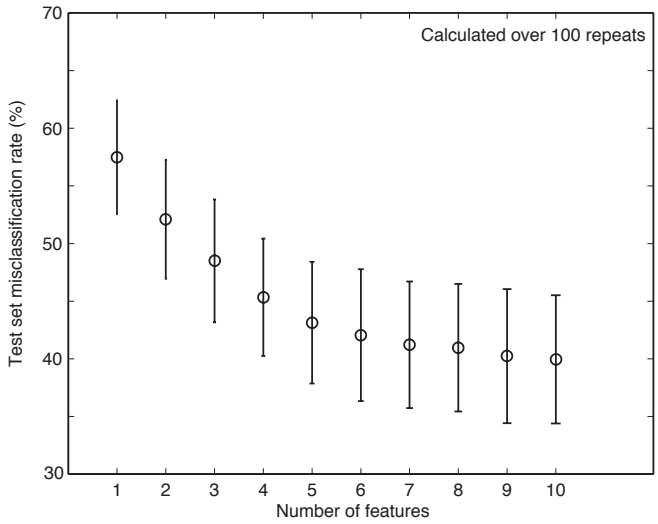

Figure 5.21: **Out-of-sample misclassification rates as a function of the number of features in linear classifiers.** Classifiers are trained on 100 different 80% portions of the data, using greedy forward feature selection; test set classification rates are then evaluated on the remaining 20% of the data (cf. Sec. 3.4.5). Mean misclassification rates are plotted with circles, and standard deviations are represented as error bars. We can achieve a competitive 60% mean out-of-sample classification accuracy using linear classifiers that combine the output of ten operations.

the literature, and as we found in Fig. 5.17, (ii) we use downsampled (8 kHz) audio recordings in our analysis and may therefore be losing information that could improve the performance of our classifiers, (iii) we restrict ourselves to just the simplest linear classifiers. However, despite difficulties of comparison, we have shown in this section how successful classifiers for this multi-class problem can be constructed automatically from our library of operations. Linear classifiers were learnt for single operations, pairs of operations, and larger sets of operations. As shown in Fig. 5.21, ten-feature classifiers trained on 80% portions of the dataset have an expected out-of-sample classification accuracy of approximately 60%, which is comparable to the rates achieved in the literature using complex classifiers on this problem. As well as being accurate, these classifiers are transparent and interpretable, in contrast to the high-dimensional feature spaces and complex classifiers that are often used to attain high classification accuracies in other reports on this data.

5.2.4 Electroencephalograms (EEGs)

So far we have shown how multiple hypothesis testing can be used to assess the significance of highly comparative results for small datasets (in Sec. 5.2.2), and how competitive multi-class classifiers can be constructed automatically using feature selection (in Sec. 5.2.3). In this section, we select operations that are effective at distinguishing different classes of electroencephalogram (EEG) recordings, and show how the results can be used to assess the significance of existing reports in the literature.

5.2.4.1 Data

We analyzed a publicly-available dataset of EEG time series[6] [86]. Each time series was sampled at 173.61 Hz and contains a 23.6 second segment (i.e., containing 4 097 samples) of a single-channel EEG recording. The segments were selected so as not to contain artifacts, and to fulfill a weak stationarity criterion [86]. As suggested in the original paper [86], we applied a low-pass filter at 40 Hz to the data. The recordings are divided into five groups containing 100 time series each, labeled *A*, *B*, *C*, *D*, and *E*. Each category label is described in Table 5.1. While the original paper was mainly concerned with the nonlinear and fractal properties of these time series [86], here we apply our database of time-series analysis methods to investigate structure in the data and sources of difference between the classes. The database has been used for classification previously, including studies attempting to separate sets *A* (healthy) and *E* (seizure) of the dataset [87, 163, 164], and others attempting to classify sets *A* (healthy), *D* (epileptic), and *E* (seizure) [165].

5.2.4.2 Unsupervised analysis

First we investigate whether the structure of the data in the two-dimensional PC space (obtained from all 8 285 operations with a non-zero interquartile range and no special-valued outputs on this dataset) corresponds to the labels assigned to the time series, as shown in Fig. 5.22(a). The classes are grouped in a way that reflects the known

[6]Data available at `http://epileptologie-bonn.de/cms/front_content.php?idcat=193&lang=3&changelang=3`

Table 5.1: Explanations of each of the 5 EEG data classes obtained from Andrzejak et al. [86].

Set	Description
A	surface recording of healthy volunteers: relaxed, awake, eyes open
B	surface recording of healthy volunteers: relaxed, awake, eyes closed
C	epileptic patients during seizure-free intervals: depth electrode recordings from hippocampal formation from opposite hemisphere of the brain to epileptogenic zone
D	epileptic patients during seizure-free intervals: recordings from within epileptogenic zone
E	epileptic patients during seizure episodes

labeled structure of the data: sets C and D, both recorded from epileptic patients during seizure-free intervals are grouped together; sets A and B, both recorded from healthy patients, are also grouped together; and set E is distinguished as a set of epileptic seizure recordings.

We used a reduced set of 200 operations, excluding location-dependent length-dependent, and spread-dependent operations (cf. Sec. 4.1.2) to measure distances between class centers of each group. The dendrogram between classes shown in Fig. 5.22(b) was obtained. Each split in the dendrogram is interpretable, and the dendrogram structures the time series meaningfully. The first split separates the healthy and epileptic groups, and within the epileptic group, the seizure recordings are distinguished. Sets A and B are judged to be very similar—healthy subjects with eyes either open or closed, as are sets C and D—epileptic patients during seizure-free intervals. Again, the unsupervised analysis provides a meaningful organization of the data. Note that the same qualitative structure can be reproduced using reduced sets of operations, as is described in Sec. B.2.2 of the appendix.

5.2.4.3 Distinguishing seizure recordings from healthy EEGs

In this section, we investigate a binary classification task between sets A (healthy, eyes open) and E (epileptic seizure). Existing studies on this classification between these healthy and seizure recordings [87, 163, 164] quote very high classification accuracies: 97.2% with a neural network classifier [163], and at least 98.75% using support vector machines (SVMs) with the radial basis function kernel and features chosen using dimensionality reduction techniques from a set of features derived from the discrete wavelet transform of the signals: 98.75% using PCA, 99.5% using in-

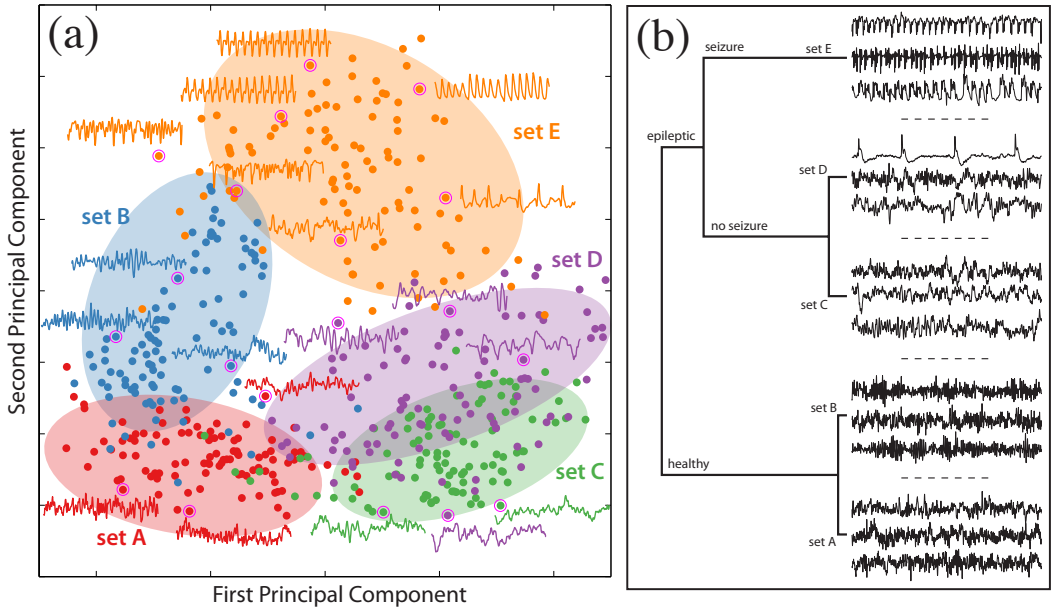

Figure 5.22: **The collective behavior of our database of time-series analysis operations structures five classes of EEG signals meaningfully.** (a) A two-dimensional Principal Components projection of the dataset. Background shading is used to guide the eye, and 400-sample segments of time series are annotated to selected points in the plot. (b) A dendrogram between class means in a reduced 197-dimensional space of operations; three 2 000-sample subsegments of time series from each class are annotated. Note that the same qualitative structure in this dendrogram is also observed using the full set of operations instead of this reduced set.

dependent components analysis (ICA), and 100% using linear discriminant analysis (LDA) [87]. In the latter paper by Subasi and Ismail Gursoy [87], it is argued that the high accuracy of their method demonstrates the importance of the chosen discrete wavelet transform-based features extracted from the signals, and they promote a clinical implementation of their system. How do we interpret the significance of these reported classification accuracies? Are classification rates high simply because the classification task is straightforward, or does the proposed technique represent a uniquely successful approach for a difficult problem? A comprehensive comparison to other possible methods is required to support the latter conclusion, and this can be achieved by simultaneously comparing the classification performance across our assembled library of operations.

In Fig. 5.23(a), we show that the collective behavior of the ensemble of 8 062 operations with no special-valued outputs and a non-zero interquartile range from our library, as summarized by the first two PCs, accurately distinguishes these signals.

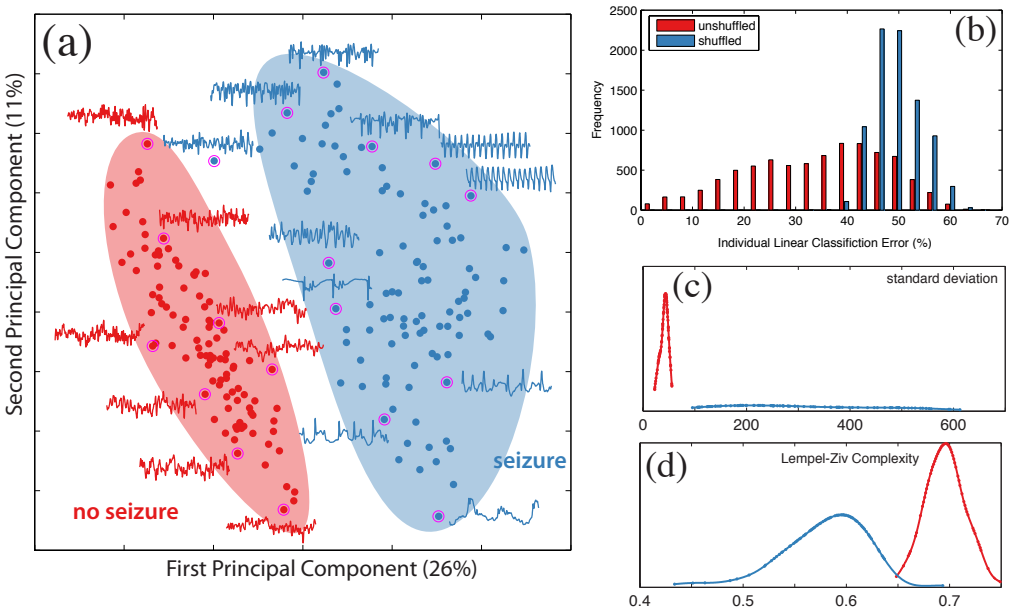

Figure 5.23: **Distinguishing 'healthy' from 'seizure' EEGs is straightforward using our library of operations.** We use sets A (healthy, eyes open) and E (epileptic seizure) from the EEG dataset, as has been done previously [87, 163, 164]. (a) In a two-dimensional PC projection of the data matrix, the two classes are clearly separated. The proportion of variance explained by each PC is indicated in parentheses. (b) A histogram of individual mean 10-fold cross-validation linear misclassification rates (red). The cross validation was repeated ten times to reduce the variance of the estimators, and a shuffled version (cf. Sec. 3.4.6) is plotted in blue. Many individual operations are clearly separating the two classes more than might be expected by chance. (c) Distributions of the standard deviation of both classes of time series. (d) Distributions of the output of a Lempel-Ziv complexity operation, which uses words of length 8 for a binary symbolization of the time series of successive increments (i.e., '1' for an increase and '0' for a decrease).

As shown in Fig. 5.23(a), the first PC alone accurately classifies healthy and epileptic signals, with a $97 \pm 3\%$ 10-fold cross-validation linear classification rate. Together, the first two PCs classify these two groups with $99 \pm 2\%$ error. Only the unlabeled structure of the dataset is used to select these two PCs with which to represent the data, yet excellent classification performance is achieved.

Ten-fold cross-validation linear misclassification rates for individual operations are plotted as a histogram in Fig. 5.23(b). Fifty-four individual operations have a cross-validation linear classification accuracy greater than that of an existing neural network classifier (97.2%) [163], which uses a multistage nonlinear pre-processing filter in combination with a LAMSTAR Artificial Neural Network (extracting relative spike amplitude and spike occurrence frequency features). There are 181 individual

operations that each separate the classes with a classification rate exceeding 95%. These operations include measures of spread (e.g., standard deviation), entropy (e.g., histogram entropy, quantized motif frequencies, permutation entropy, average local 1-point correlation entropy, Lempel-Ziv complexity, SampEn), dimension estimates (e.g., Taken's estimator), and scaling exponents (e.g., from detrended fluctuation analysis or rescaled range analysis). Two examples of such operations are plotted in Figs. 5.23(c) and (d). In Fig. 5.23(c) we see that measuring just the standard deviation is sufficient to separate the classes—seizure EEGs have a much greater standard deviation than healthy EEGs. This perhaps explains the motivation for the relative spike amplitude feature in Nigam and Graupe [163], that is itself a spread-dependent measure. The increase in the standard deviation at the onset of a seizure has also been noted in the seizure prediction literature [166]. In Fig. 5.23(d), we see that the Lempel-Ziv complexity of an 8-bit encoding of incremental differences of the time series (a similar idea as 'Control Entropy' [63]) also separates the signals with high accuracy, showing that incremental differences of seizure EEGs have lower complexity than healthy EEGs.

We do not present a full analysis of the problem of separating the three sets, *A*, *D*, and *E* here, but simply confirm that it our highly comparative approach can be used to construct successful classifiers in this case also. One study by Güler et al. [165] used recurrent neural networks with Lyapunov exponent estimates as features for this three-class problem, reporting a 96.79% classification accuracy using some split of the data into training and testing portions. Principal Components were calculated from the 7 988 operations with no special-valued outputs and a non-zero interquartile range on this data and, as shown in Fig. 5.24, the two-dimensional PC projection of the data separates the three classes of time series. The 10-fold cross-validation linear classification accuracy (with 10 repeats to reduce the variance) is $94 \pm 3\%$ in this space. Although this is a slightly lower accuracy than that reported in Güler et al. [165], it was obtained in a two-dimensional PC space constructed using the outputs of a diverse range of time-series analysis methods using the simplest possible linear classifier. Training classifiers using the labeled structure of the data, as above, could be used to construct interpretable classifiers with improved classification rates.

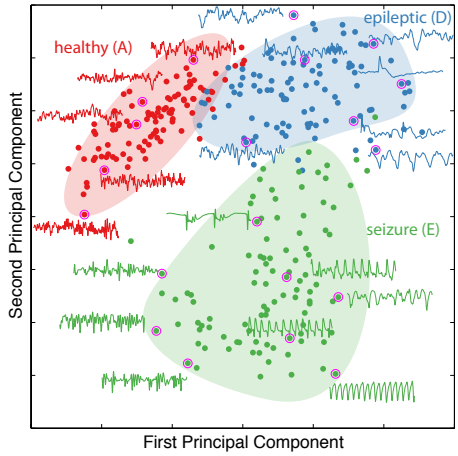

Figure 5.24: **A two-dimensional Principal Component projection of datasets A, D, and E separates the three classes of emotional speech signals.** All 100 time series from each of the sets A, D, and E (described in Tab. 5.1), are plotted. Selected 400-sample time-series segments were annotated to points on the plot, and background shading was added manually to guide the eye.

From our results, it is clear that many different types of operations can individually outperform existing classifiers, even using a simple threshold on the output of a single operation: including measures of spread, entropy estimates, methods derived from nonlinear time-series analysis, and scaling exponents. Our results therefore highlight the difficulty in evaluating the significance of a reported classifier without comparing the performance of alternative methods. Hence, rather than implementing increasingly complex methods and reporting subsequent minor improvements in classification accuracy for existing tasks, our results suggest that a broad, comparative approach that draws on an existing time-series analysis literature can be used to focus scientific effort towards developing techniques that probe difficult problems in new and useful ways.

5.2.5 Parkinsonian speech

In this section, a dataset containing phoneme audio recordings from healthy controls and from patients with Parkinson's disease is analyzed. Our aim with this case study is to demonstrate how a highly comparative classification approach can be applied to this difficult problem, with results that can be interpreted in the context of an existing literature on the topic. Studies of this type of data have in the past aimed to separate healthy controls from patients with Parkinson's disease using a range of classical

speech analysis algorithms and a variety of novel new methods [91, 152, 167–169]. Other studies have used outputs from various operations to predict patients' scores on the Unified Parkinson's Disease Rating Scale (UPDRS) [170, 171]. Traditional measures known to be useful for these types of tasks include *jitter*, which measure modulations in frequency between vocal cycles [172], and *shimmer*, that measures variation in amplitude between vocal cycles [91]. Estimates of the fundamental frequency, F_0, and the energy in different frequency bands are also common measures for this task [167, 168, 171]. Newer studies have focused on various nonlinear algorithms, including *pitch period entropy* (PPE) [91], a correlation dimension estimate D_2, [169], detrended fluctuation analysis (DFA) [152, 171], and nonlinear recurrence and fractal scaling [152, 173].

In Sec. 5.2.5.1, we describe the time series and outline the pre-processing applied to them. We then describe some individual operations that exhibit high classification accuracies in Sec. 5.2.5.2. In Sec. 5.2.5.3, combinations of operations are used to build classifiers for this task. The implications of the results obtained in this section are discussed in Sec. 5.2.5.4.

5.2.5.1 Data

The data analyzed in this section was obtained from an original dataset containing 195 voice recordings from 31 subjects, 8 of which are age-matched healthy controls; a similar dataset to that studied by Little et al. [91]. On average, there are six recordings per subject of sustained phonations of the vowel 'ahhh'. Originally recorded at 44.1 kHz, they have been downsampled to 16 kHz [91], and a 'vocally stable' section of the middle of the recording has been retained, starting at 1.0 s. Each recording is 2.0 s in duration, and contains 32 000 samples.

For the present analysis, a set of shorter, non-overlapping segments of time series with length chosen randomly in the range 800–20 000 samples (50 ms–1.25 s) has been obtained from each time series. This provides additional examples of each class and decreases computational time (which scales nonlinearly with the length of time series, cf. Sec. 3.2.5), and ensures that operations are returned that distinguish the time series despite their variable length. For this classification task, 127 examples of each of

the two classes: 'Parkinson's disease' and 'healthy control', were retrieved at random from the database to create a balanced dataset for classification. Some examples of time series from each of the two classes are shown in Fig. 5.25. Note the variation in length of the full signals, shown in the upper panels, and the visual similarity of the two classes in 1 000 sample (62.5 ms) subsegments plotted in the lower panels of the figure. Both classes have similar length distributions.

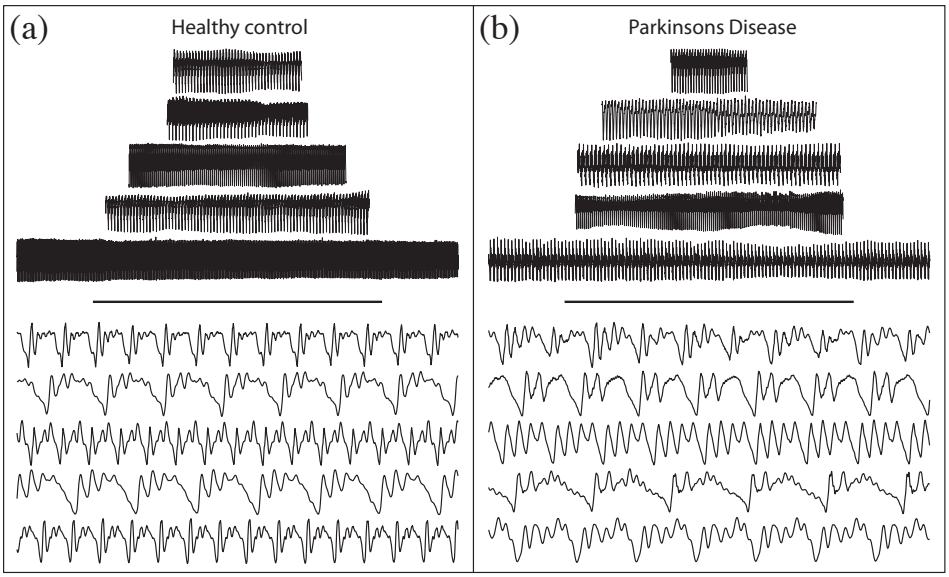

Figure 5.25: **Phoneme audio recordings from healthy controls and patients with Parkinson's disease.** In the top section of each subplot, full time series are plotted. Lengths of recordings in this plot range from 2 500 samples (156 ms) through to 17 300 samples (1.08 s). In the lower sections, random time series segments of length 1 000 samples (62.5 ms) are plotted.

5.2.5.2 Individual operations

Unlike many of the other case studies described in this chapter, unsupervised representations of this dataset are not informative of its known class structure. For example, in a two-dimensional PC projection of the dataset, the classification rate for a linear classifier is consistent with chance: $50 \pm 6\%$ (see Sec. B.2.3.1 of the appendix for details). It is clear that a more targeted approach is required to detect the subtle differences between these two classes of speech signals. We calculated 10-fold cross validation linear classification errors for our library of 8 399 operations (those with no special-valued outputs on this dataset), using ten repeats to reduce the variance of

the estimates. A histogram of 10-fold cross-validation misclassification rates (using ten repeats to reduce the variance of the estimates) across our library of 8 399 operations (those with no special-valued outputs on this dataset) is shown in Fig. 5.26. The result highlights the difficulty of this classification task: no operations classify the Parkinsonian speech signals at a rate exceeding 80%. However, it is clear that many operations classify Parkinsonian time series at a rate better than chance (cf. Sec. 3.4.6), which is represented as a blue histogram in Fig. 5.26.

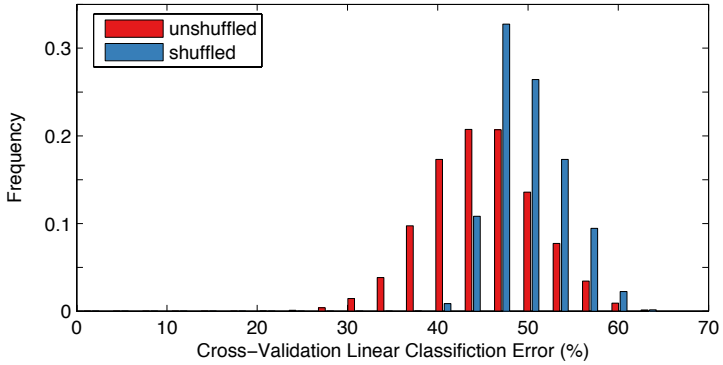

Figure 5.26: **Operations in our library that classify the Parkinsonian speech signals most accurately.** We plot histograms of individual 10-fold linear classification errors for the 8 399 operations with no special-valued outputs on this dataset (red), and for the same operations across 10 random permutations of the data labels (blue), as explained in Sec. 3.4.6. It is clear that some of the operations classify the two classes with an accuracy that greatly exceeds what would be expected by chance.

A selection of the best performing individual operations for this classification task are shown in Fig. 5.27. These operations are described in detail in Sec. B.2.3, and are summarized below. Interestingly, we find that a number of operations based on particle trajectories perform well for this task. Indeed, the operation with the highest classification accuracy for this task ($80 \pm 7\%$) is plotted in Fig. 5.27(a). These particle trajectory-based operations are a new type of time-series analysis operation developed by us in this work, and involve simulating the motion of an imaginary Newtonian particle that moves in response to the time series. As explained in Sec. B.2.3, these operations are able to detect some of the subtle characteristics of the waveforms of the Parkinsonian speech as compared to that of healthy controls. The other successful operations can be grouped into the following categories: (1) those that measure the asymmetry of the time-series above and below the mean are often useful, including

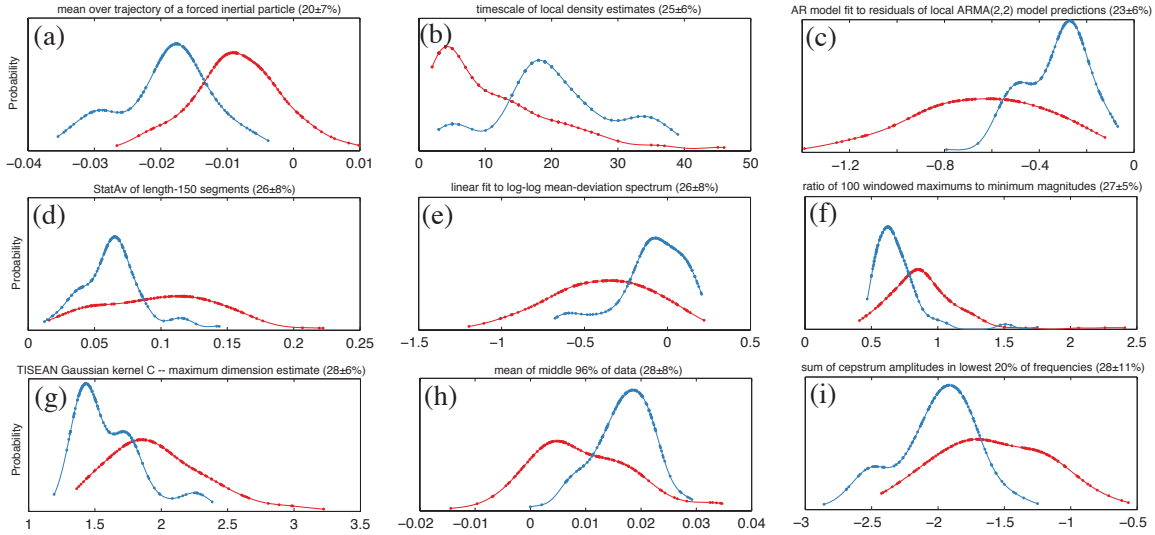

Figure 5.27: **A selection of the most successful operations for Parkinsonian speech discrimination.** Operations are identified in the title of each subplot, with 10-fold cross-validation linear classification errors (using ten repeats to reduce the variance) are also quoted in parentheses in the form: mean ± standard deviation. Kernel-smoothed distributions of the output of each operation are plotted for both Parkinson's disease (red) and healthy control (blue) time series. Detailed explanations of each operation are in the main text.

a measure of local extreme values [Fig. 5.27(f)] and a basic outlier-trimmed mean [Fig. 5.27(h)], (2) operations from nonlinear time-series analysis, including a method based on local density estimates in an time-delay embedding space [Fig. 5.27(b)] and a Gaussian kernel-based correlation dimension estimate [Fig. 5.27(g)], (3) linear model fits such as the autoregressive moving average (ARMA) model [Fig. 5.27(c)], (4) a stationarity measure, StatAv [Fig. 5.27(d)], and (5) summaries of the Fourier power spectrum [Figs. 5.27(e) and (i)].

While the particle trajectory-based operations are new, many of the other selected operations are closely related to existing measures for this task. Firstly, a related dimension estimate, D_2, to that shown in Fig. 5.27(g), has been shown already to be higher for speech recorded from patients with Parkinson's disease [169]. The traditional speech analysis methods 'jitter' and 'shimmer' measure variation in fundamental frequency and amplitude between vocal cycles, respectively [91]; jitter has been shown to be useful in a number of studies into the problem [91, 169, 171]. The stationarity measure shown in Fig. 5.27 calculates stationarity at approximately their period of oscillation and hence is conceptually similar to 'jitter' measure. Fi-

nally, the operation that estimates low-frequency power in logarithmic magnitudes of deviations from the mean, shown in Fig. 5.27(i) is related to traditional statistics involving power in low frequency bands of the power spectrum [167, 168, 171], or the related cepstrum coefficients [171].

Comparison to existing measures Two existing operations that have been implemented and used to analyze Parkinsonian speech recordings in the past are shown for comparison by Fig. 5.28, as adapted from an original figure in Little et al. [91]. Note, however, that the dataset used by Little et al. [91] is unbalanced, containing 48 recordings from healthy subjects and 147 recordings from patients with Parkinson's disease. It is therefore difficult to make a direct comparison between our results for a balanced set of time series, where the expected accuracy of a 'null' classifier is 50%, to these results, where the expected accuracy of a 'null' classifier (that always classifies time series as Parkinsonian) is 75.4%. The first measure, shown in Fig. 5.28(a), is termed *jitter*, and measures the variation in frequency between vocal oscillation cycles [172]. Using a support vector machine classifier with a Gaussian radial basis kernel function, the classification accuracy using this feature is $80.6 \pm 9.9\%$ (95% confidence interval), where 75.4% accuracy is the null rate [91]. Our best measure, plotted in Fig. 5.27(a), achieves a similar classification accuracy of $80 \pm 7\%$ using a linear (threshold) classifier on a balanced set, where a random classifier achieves an expected classification accuracy of 50%. The classification accuracy of the second measure, pitch period entropy (PPE) in Fig. 5.28(b), is greater, at $85.6 \pm 5.4\%$. Our results, which were obtained by automatically comparing the performance of our library of time-series analysis operations to the dataset achieves similar classification rates but using a balanced dataset.

5.2.5.3 Building Classifiers

Using greedy forward feature selection with a linear classifier on the full set of 8 399 operations, as described in Sec. 3.4.5, misclassification rates are plotted as a function of the number of operations contained in the classifier in Fig. 5.29(a). Each point in Fig. 5.29(a) summarizes a distribution of test set error rates across 500 different

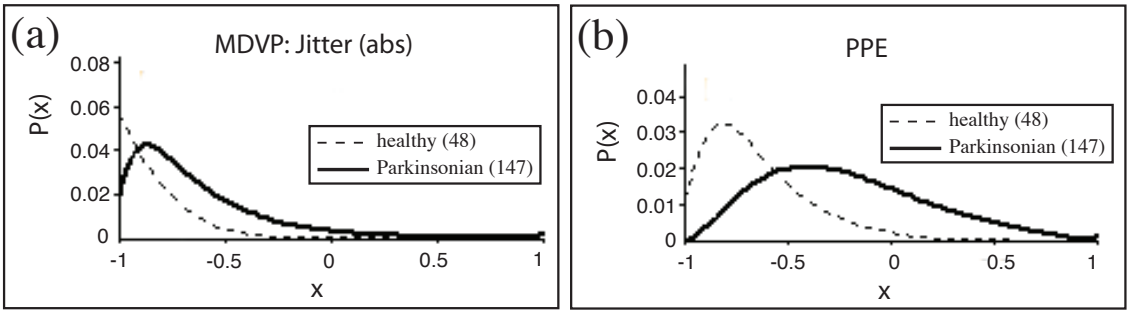

Figure 5.28: **Performance of two existing measures for Parkinsonian speech classification.** Figure adapted from Little et al. [91]. The two measures are (a) a standard speech analysis statistic: *jitter* [172], and (b) a novel measure: *pitch period entropy* (PPE) introduced by Little et al. [91]. In each plot a kernel-smoothed distribution of normalized outputs, x, from each operation is plotted kernel for the 48 healthy speech recordings and 147 Parkinsonian speech recordings. The jitter measure has a classification rate of $80.6 \pm 9.9\%$, and PPE has $85.6 \pm 5.4\%$, where the null rate, reflecting the class proportions, is 75.4%.

balanced partitions of the data into training sets (80% of data) and test sets (the remaining 20% of data). Examples of these distributions are plotted as red histograms in Figs. 5.29(b) and (c). Test set misclassification rates decrease up to approximately four features, after which the improvement plateaus. This trend is consistent with the results presented in [91], where the optimum number of features was also found to be four using a support vector machine classifier with a Gaussian radial basis kernel function. Although, classifiers other than linear classifiers, and feature selection methods other than greedy forward selection may show different trends.

Because the feature selection and classifier building process is done on the training set and errors are computed only on unseen data (cf. Sec. 3.4.5), repeating the process using randomly-permuted class labels produces test set errors centered at approximately 50%, as shown in the blue histograms in Figs. 5.29(b) and (c). As the number of features increases from one (in Fig. 5.29(b)) to four (in Fig. 5.29(c)), the blue distribution remains approximately unchanged, while the distribution using the real data labels shifts towards lower test set errors.

Given the difficult nature of this problem, in which the differences between the two sets are subtle and the range of recording lengths varies by more than an order of magnitude, our ability to construct four-feature classifiers automatically with an expected classification rate of approximately 85% is encouraging. Previous reports

have proposed classifiers with accuracies as high as 91.4% [91] or $91.8 \pm 2\%$ [152], but on highly-unbalanced datasets (75.4% Parkinsonian examples in Little et al. [91] and 81.8% Parkinsonian recordings in Little et al. [152]), whose quantitative results are difficult to compare to our results using a balanced dataset. Furthermore, our library of operations does not contain any operations specific to speech analysis, and we do not use knowledge of the sampling rate in any of our operations; whereas traditional approaches to this problem exploit particular domain-motivated frequency bands [167, 168, 171].

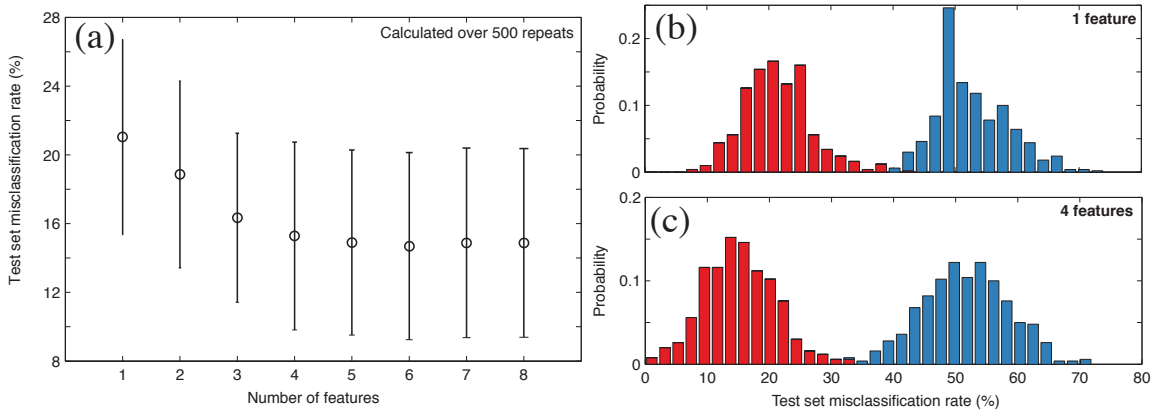

Figure 5.29: **Test set misclassification rates across 500 repeats of a greedy forward selection algorithm using simple linear discriminant classification for the Parkinsonian speech dataset.** (a) Misclassification rate as a function of the number of features. After four features, the classification rate shows little improvement. Distributions of test set misclassification rates are plotted as red histograms for (b) one feature, and (c) four features. In (b) and (c), test set errors for the time series with randomly permuted labels are also plotted in blue. Note that plots like (b) and (c) can be obtained for each point in (a) corresponding to a given number of features: all are qualitatively similar to these. The distributions obtained from fitting to shuffled labels are always consistent with chance (50% for this balanced dataset).

Each sample from distributions such as those plotted in red in Figs. 5.29(b) and (c), represents a linear classifier that combines the outputs of multiple operations in our library. We could therefore select classifiers from these distributions as those that were chosen most frequently, or those with the lowest cross-validation misclassification rate, for example. This approach will be explored using the RR interval data in Sec. 5.2.6. An example of how to construct two-feature classifiers for this Parkinsonian speech data—by considering an exhaustive search of all (35 267 401) pairs of

operations—is described in Sec. B.2.3.4 of the appendix.

5.2.5.4 Discussion

Having described a variety of useful time-series analysis operations in this section, a time-series analyst could use this information to understand the dynamics in these time series, and explore those operations that most accurately and reliably estimate these properties. Our results can also be considered a starting point to the optimization of the parameters of useful operations to suit the task at hand, and potentially to develop clinically-useful classifiers.

Of the operations that we retrieved from our library, many corresponded to operations with known utility for this task, including 'jitter'-like operations, scaling operations, dimension estimates from nonlinear time-series analysis, and various summaries of the Fourier power spectral density function, despite the fact that our library contains no operations that are specific to speech analysis. We also retrieved a number of novel operations including simple measures of distribution asymmetry, ARMA model fits, and summaries of simulated particle motions driven by the speech recordings. These operations have never before been applied to this task. We contrast our ability to retrieve and organize these operations automatically from a large and diverse library of operations with conventional reports on this topic, that use a small number of standard methods and sometimes propose one or two new methods. Analysis on this very small scale is typical: rarely are methods compared to one another, previous results are difficult to repeat due to limited data availability and private code, and the literature develops slowly as new methods are introduced individually. In the future, all the standard operations for Parkinsonian speech analysis could be added to our library, which would allow them to be simultaneously compared and put in the context of the rest of the time-series analysis literature (by visualizing them as correlation matrices like Fig. B.8, for example, or using similarity searches as in Sec. 4.1.3).

5.2.6 Heart rate variability

In this section we study heart-beat interval series from two classes: 'normal sinus rhythm' and 'congestive heart failure'. The RR interval is defined as the duration between the peak of one QRS complex in the electrocardiogram (ECG) to the next. Hence RR interval series are sequences of the durations between successive QRS complexes in the cardiac cycle. RR intervals have been studied across a range of disciplines, from biomedical engineering to physics, and there exists a large volume of research into time-series analysis techniques that discriminate between different pathologies [57, 174], quantify the effect of heart transplants [175], evolve with aging [176], predict ventricular tachycardia or fibrillation [177], assess the impact of different pharmaceutical treatments [178], anticipate neonatal sepsis [51], and even to classify different behavioral states in dogs [40]. All of these studies of heart rate variability (HRV) aim to discover structure in RR interval series and develop accurate ways of measuring it. Our library of operations is therefore in a good position to contribute to this literature on RR intervals by comparing the behavior of a very large number of operations on the data. We aim to distinguish and organize those that are the most useful for distinguishing normal sinus rhythm from congestive heart failure RR intervals.

A wide variety of techniques have been employed on these RR interval data in the past. Firstly, there is a large literature of standard measures relating to basic statistical time- and frequency-domain properties of the RR interval series, including the mean and spread of RR intervals, a statistic called PNNx that gives a proportion of successive beat length differences that are below a certain threshold x, and the proportion of energy in different bands in the power spectrum, including very low frequency (VLF), low frequency (LF), high frequency (HF), and a ratio LF/HF [57, 88, 177–179]. Physicists have contributed to fractal analyses of these signals, as a means of distinguishing pathological dynamics, or to understand the process of aging [127, 180–182]. Entropy measures have also been well-applied to RR interval series, with healthy patients having a characteristically higher entropy than various pathological dynamics; entropy has been measured using SampEn [51, 57, 61, 178], ApEn [40,

178], Shannon entropy of the distribution [177, 179], and measures from symbolic dynamics [40, 177, 179, 183, 184]. Operations derived from nonlinear dynamics have also been applied to RR-interval series, including estimates of correlation dimensions [175], Lyapunov exponents [177], Poincaré plots [185], and a multi-scale time-reversal asymmetry estimate [176]. All of these studies have uncovered interesting patterns in RR interval series and interpreted them in the context of an interesting biomedical problem. Rather than selecting those operations that have been used in the past and therefore being biased by a precedent in the literature, in this section, the empirical structure in the dataset is used to motivate the selection of appropriate methods from our library.

In Sec. 5.2.6.1, we describe the dataset and our processing of it, and in Sec. 5.2.6.2, a two-dimensional Principal Components projection of the data is analyzed. In Sec. 5.2.6.3, we discuss the behavior of individual features with good performance on the dataset, and use their outputs to organize them into meaningful groups in Sec. 5.2.6.4.

5.2.6.1 Data

A set of RR interval sequences was obtained from two *PhysioNet* databases: the *Congestive Heart Failure RR Interval Database*[7], and the *Normal Sinus Rhythm RR Interval Database*[8]. These data are strictly sequences of intervals and not time series, but because they are ordered sets of measurements they can be analyzed in the same way as time series. Each recording in the dataset belongs to one of two classes: 'normal sinus rhythm' or 'congestive heart failure'. The RR interval sequences are also annotated by the age of the subject recorded. From the recordings, we took non-overlapping subsegments of length chosen uniformly at random in the range 800–20 000 samples from an original set of 29 congestive heart failure recordings and 54 normal sinus rhythm recordings. We finally obtained a balanced dataset containing 105 RR interval series from each class, which is analyzed in this section. We will therefore assess whether, given a recording of some length between 800 and 20 000

[7]Data available at `http://www.physionet.org/physiobank/database/chf2db/`.
[8]Data available at `http://www.physionet.org/physiobank/database/nsr2db/`.

consecutive RR intervals, there exist time-series analysis operations in the database that can effectively measure differences in the underlying dynamics of the two classes of recordings. Note that distributions of both subject age and time-series length are approximately the same for both classes of RR interval series. Examples of RR interval sequences belonging to each class are plotted in Fig. 5.30.

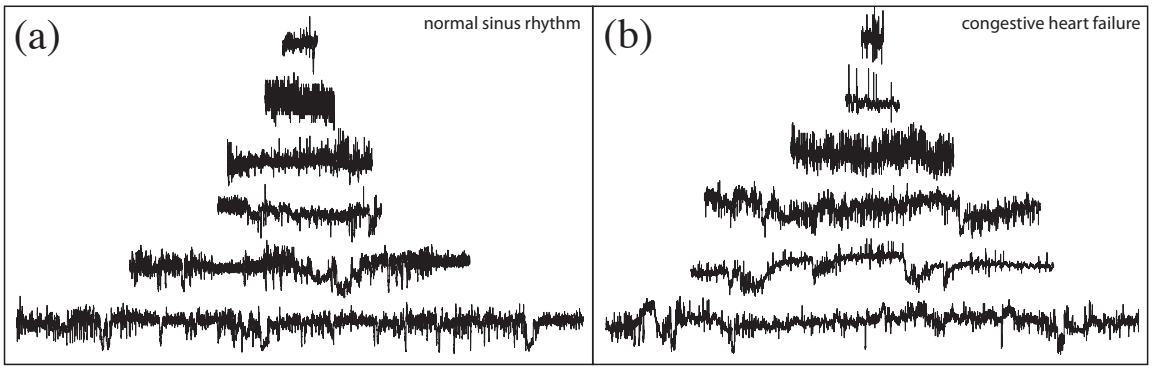

Figure 5.30: **Examples of RR interval series.** All recordings belong to one of two classes: (a) normal sinus rhythm, or (b) congestive heart failure. In this plot, the longest time series plotted is 19 000 samples long, and the shortest is 800 samples long. Despite the large range in lengths, our extensive library contains time-series analysis operations that can classify them with high accuracy.

5.2.6.2 Principal Component projection

A two-dimensional PC projection of this dataset is shown in Fig. 5.31. The PCs were calculated from the normalized data matrix containing the 7 478 operations with no special-valued outputs and a non-zero interquartile range on this dataset. In this representation, the two classes of time series are well-separated, even though their class labels have not been used to determine this representation. Instead, the combined behavior of our large and diverse database of time-series analysis operations provides a natural and informative representation that mirrors the known structure of the dataset. Interestingly it is the second PC that plays the dominant role in this classification, as the classification boundary is approximately horizontal in Fig. 5.31.

5.2.6.3 Individual operations

Ten-fold cross-validation linear classification rates were calculated for the 7 672 operations with no special-valued outputs on this dataset, repeating the process ten

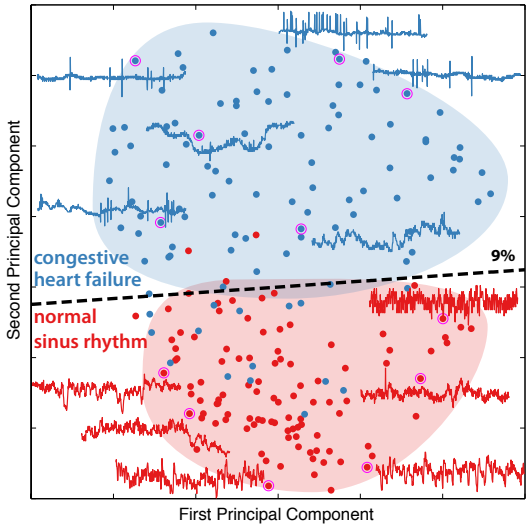

Figure 5.31: **A two-dimensional Principal Component (PC) projection of the RR interval dataset separates the two classes.** The optimal linear classification boundary in this space is plotted as a dashed line. The 10-fold cross-validation linear misclassification rate in this space is $9 \pm 3\%$. The first and second PCs explain 20.6% and 12.9% of the variance in the data matrix, respectively. Some 500-beat segments of RR sequences are annotated to selected points in the plot, and background shading has been added manually to guide the eye.

times to reduce the variance of the estimates. A histogram of misclassification rates obtained from this process is plotted in Fig. 5.32. Many operations classify the data with an accuracy that greatly exceeds what would be expected by chance. The most accurate individual classifiers encompass a wide variety of different types of operations, from measures of location, characteristics of outliers, model fit statistics, entropy estimates, long-range scaling and autocorrelation, as well as the traditional heart rate variability metric *PNNx*. Twelve examples of operations with low classification errors that encompass these areas of time-series analysis are plotted in Fig. 5.33. The behavior of these operations is summarized below and explained in more detail in Sec. B.2.4.1.

The most successful operations for this task are measures of location, such as the mean [Fig. 5.33(a)], which separates the two classes with 10-fold cross-validation linear misclassification rate of just $8 \pm 5\%$; other measures of location such as median and harmonic mean have similar performance. That normal sinus rhythm RR intervals are longer for healthy patients confirms established knowledge in the literature [57]. Aside from these location-based operations, measures of outliers also perform well,

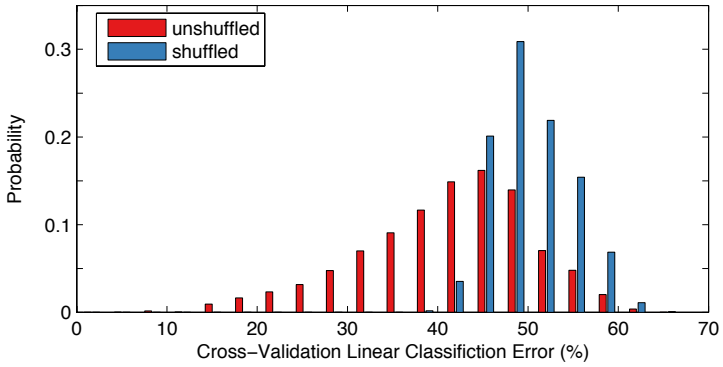

Figure 5.32: **Many individual operations accurately classify the two classes of RR interval sequences.** We plot the individual mean 10-fold linear classification errors for the 7 672 operations with no special-valued outputs on this dataset as a red histogram. Ten repeats (with different splits of the data) are used to reduce the variance of these estimates. The same errors obtained by permuting the labels on the data 10 times are plotted as a blue histogram, as described in Sec. 3.4.6.

including a new measure introduced by us in this work with a classification accuracy of $89 \pm 7\%$ that removes 10% of the time series values that are furthest from the mean and then calculates the autocorrelation at lag three of the resulting time series, as shown in Fig. 5.33(b). The success of this operation implies that the magnitude and temporal distribution of anomalous beats is a significant source of difference between the two classes of recorded RR interval sequences. Confirming the use of entropy-based measures for this task in the past, for which it is known that the RR intervals of healthy patients have higher entropy than those of congestive heart failure patients [57, 61, 178], we find a number of entropy-based operations appear as the most successful for this task. These include the Sample Entropy of the differenced series, ΔRR [Fig. 5.33(c)], and entropy measures using a binary symbolization of incremental time-series differences, ΔRR [Figs. 5.33(f) and (l)]. Next, we notice that some model fit statistics, including a nonlinear prediction error operation [Fig. 5.33(d)], and a parameter from an AR(2) model [Fig. 5.33(i)]. Differences in the long-range scaling behavior of RR interval series measured from different patients have been noted in the past [127], and indeed various fluctuation analysis-based operations are highlighted in our analysis, including a wavelet-based estimator of the scaling exponent [Fig. 5.33(g)] and a power spectrum-based iteratively re-weighted least squares estimate [Fig. 5.33(k)]. Autocorrelation properties are another important source of

variation between these two RR interval series, including the linear autocorrelation at lag 1 [Fig. 5.33(e)], and the generalized self-correlation function [186] with $\alpha = 1$, $\beta = 10$, and $\tau = 5$ [Fig. 5.33(j)]. Finally, we note that one of the classic statistics designed for HRV analysis, PNNx, is also amongst the most successful operations for this task. The operation calculates the proportion of successive RR intervals that differ by at least a duration x (typically measured in milliseconds) [174]. Variations on this statistic were returned from our analysis, including that shown in Fig. 5.33(h).

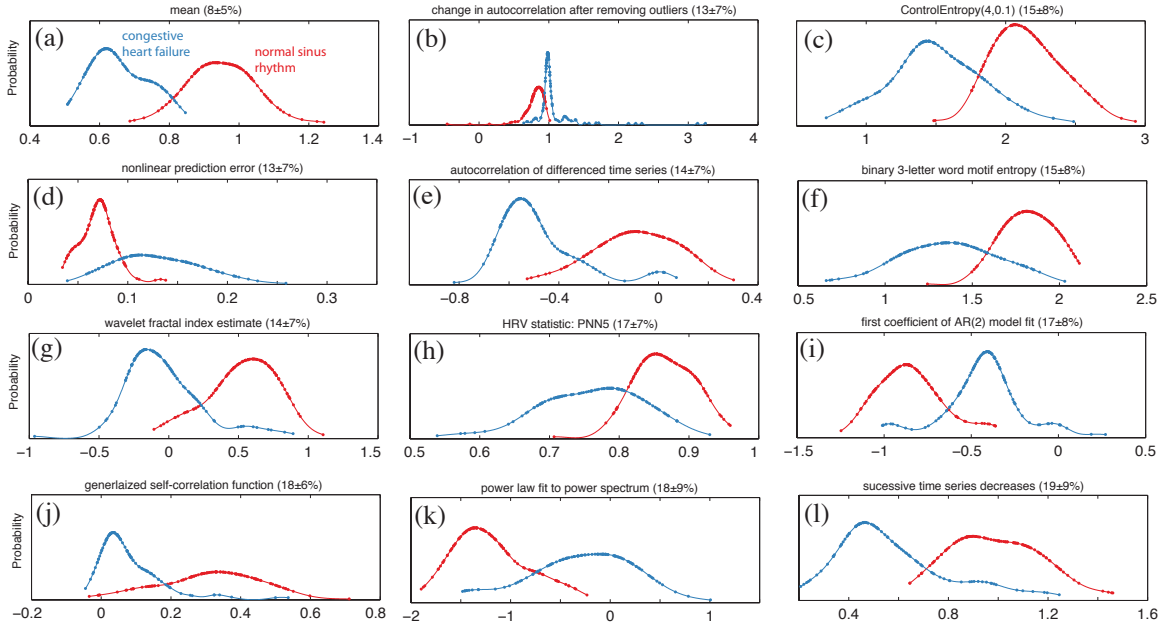

Figure 5.33: **A selection of operations with high classification accuracy rates for the RR interval dataset.** The distribution of outputs for normal sinus rhythm (red) and congestive heart failure (blue) RR intervals are plotted for each of twelve selected operations. A brief description of each operation is provided in the title along with the mean and standard deviation of 10-fold cross-validation linear classification errors with 10 repeats in parentheses. Descriptions of each operation can be found in Sec. B.2.4.1.

5.2.6.4 Organizing successful operations

From the results above, it is clear that a diverse range of properties can be measured to distinguish between these two classes of heart beat intervals. The most successful are the mean and other measures of location, but outlier properties, entropy estimates, correlation structure, and scaling exponents, can also classify the two groups with high accuracy. But are all of these operations behaving in distinct ways, or are many simply reproducing the behavior of others? In this section, we show how this

diversity of individual operations can be organized by their behavior on this dataset. In so doing, we group similar operations that are simply minor variants of one another, establish connections between apparently different types of successful operations, and distinguish those that are unique.

Pairwise linear correlation coefficients, R, were calculated between the 60 operations with a mean 10-fold cross-validation linear misclassification rate less than 16% on this dataset. The resulting (symmetric) matrix was then reordered such that covarying operations (with high $|R|$) are placed near one another using average linkage clustering (as described in Sec. 3.4.2). The result is plotted in Fig. 5.34A, and shows that most operations fall into one of three main groups. Operations in each of these groups perform similar functions on this dataset, with strong linear correlations to other operations in the group, but relatively low correlations to operations outside their group.

The first group, labeled 'location' in Fig. 5.34A, contains location-dependent operations: the arithmetic mean, median, truncated means, the midhinge (i.e., the mean of the first and third quartiles), harmonic mean, and the mean of a fitted Gaussian distribution. The behavior of the mean is plotted in Fig. 5.34C, and is representative of other operations within this 'location' cluster.

The second cluster, labeled with 'PNNx' and 'control entropies' in Fig. 5.34A, contains entropy-like operations: SampEn, ControlEn, motif frequencies, motif entropies, a nonlinear prediction error; and pNNx measures. It is interesting that these classic pNNx measures from the HRV literature are performing similarly to these entropy-like algorithms on this dataset. All of these operations produce distributions that resemble that plotted for the Sample Entropy in Fig. 5.34D.

The third cluster, labeled with 'scaling' and 'linear models' in Fig. 5.34A, contains operations measuring different correlation properties of the RR intervals: goodness of fit of linear models, autocorrelation of residuals from local linear forecasting, wavelet and power-spectrum-based measures of fractal scaling exponents, and the variance ratio hypothesis test from the Econometrics literature [89] (using the function `vratiotest` in MATLAB's *Econometrics Toolbox*, which is based on the null hypothesis of a random walk). Distributions from this variance ratio hypothesis test

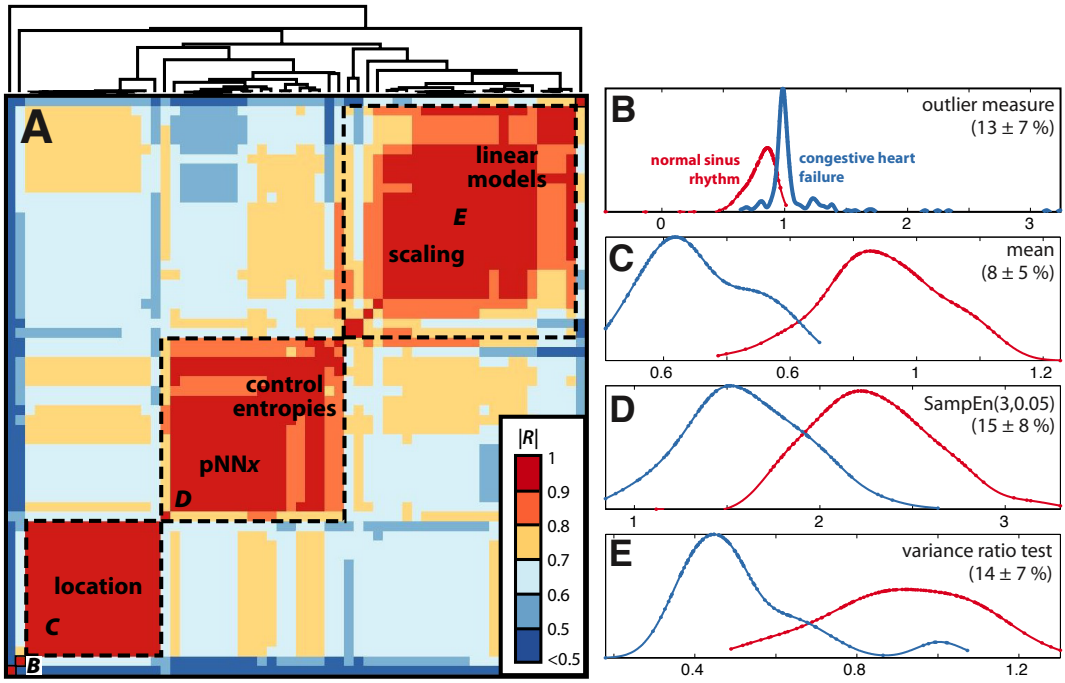

Figure 5.34: **Successful HRV operations are grouped into three main types of behavior.** The 60 operations with less than 16% mean 10-fold cross-validation linear classification error (over 10 repeats) are included in this plot. Members of each group exhibit high absolute linear correlation coefficients with other members of the group across these RR interval sequences, but relatively low correlations to operations in other groups. The clustered correlation structure is plotted in A, where color indicates the magnitude of linear correlation coefficients, $|R|$, between outputs of all pairs of operations, from blue ($|R| < 0.5$) to red ($|R| > 0.9$). Brief labels are annotated, summarizing the type of operations located in different regions of the plot. A dendrogram is annotated above the plot. The behavior of a representative member of each of the groups labeled in A is plotted in Figs. B–E. The 10-fold cross-validation misclassification rates for each of these representative operations are given in parentheses as mean ± standard deviation.

are plotted in Fig. 5.34E; other members of this group produce similar distributions.

There are two relatively unique operations in the bottom left corner of Fig. 5.34A. The first, the AC(3) ratio from removing outliers from the RR interval sequences (as described in Sec. 5.2.6.3 above) is plotted in Fig. 5.34B, has a low misclassification rate on this dataset ($13 \pm 7\%$: lower than all other operations that have no location dependence), and exhibits low correlations to all other measures in this set ($|R| < 0.6$). The operation has been introduced here for the first time, and is behaving in a unique way compared to other successful operations for this classification problem. The second unique operation in the bottom left corner of Fig. 5.34A performs a discrete wavelet transform on the z-scored RR interval series using a second-order

Symlet wavelet and returns the minimum detail coefficient at level 5. This operation has a $16 \pm 7\%$ misclassification rate on this dataset, and correlates somewhat with location-dependent measures ($0.5 < |R| < 0.8$), despite not using any location-based information, and also with the outputs of some linear models ($0.6 < |R| < 0.7$), as can be seen in Fig. 5.34A.

Another operation with minimal correlations to other operations is in the top right corner of Fig. 5.34A. This operation, which uses the mean of the past two values of a time series to predict the next value and returns the first zero crossing of the autocorrelation function of the resulting residuals, correlates most strongly ($0.7 < |R| < 0.8$) with outputs from linear models, as labeled in Fig. 5.34A.

This ability to organize a range of successful operations by their behavior is very useful because it allows us to visualize and understand linear dependencies between different types of operations immediately. In so doing, operations with distinct behavior and high classification rates are distinguished. New reports on time-series analysis often promote the utility of a new method for a particular classification task by demonstrating good performance on a dataset. For example, the variance ratio hypothesis test, developed in the Economics literature [89, 187, 188], is successful at this task, as shown in Fig. 5.34E, and is novel in the context of heart rate variability. However, from Fig. 5.34A, we can see that there already exists a large interdisciplinary literature of other methods exhibiting highly-correlated outputs and hence comparable performance for these HRV signals, e.g., the first coefficient from an AR(5) model fit to the series. By performing a thorough comparative analysis, and using a representation like that shown in Fig. 5.34A, new operations can be compared immediately to existing ones, and contributions to time-series analysis can therefore focus on the development of algorithms with unique and useful behavior.

Note that relationships between sets of approximately 10–15 standard operations for HRV analysis have been analyzed previously using dendrograms [57, 178]. However, such analyses are severely limited in scope due to the small sets of operations that are chosen based on popularity within the HRV literature. As such, they do not allow for the discovery of new operations developed in different areas of time-series analysis nor the synthesis of varied approaches achieved by clustering them

into highly-correlated groups.

The predominance of incremental differencing as a built-in pre-processing amongst the most successful operations for this classification task suggests that this pre-processing is a useful one for this data. This discovery could motivate a time-series analyst to investigate and compare a range of pre-processing choices for the dataset to account for its nonstationarity. Many of the most common and useful statistics that are derived from RR-interval series indeed involve successive increments of RR intervals, including pNNx and RMSSD [88, 178, 179], and also evidence for long-range scaling is typically measured from series of successive increments, ΔRR [182, 189]. Without a highly comparative approach, this ability for operations to hint at suitable types of pre-processing in this way would be unfeasible.

In summary, we have demonstrated how to select useful operations from our library automatically, given a well-defined classification task. We retrieved location-dependent, entropy-based, pNNx, and long-range scaling measures, all of which correspond to known differences between these types of RR interval series. We also found new and unique measures that have high classification rates for this task, including an operation developed by us that returns the ratio of autocorrelations at lag 3, before and after removing 10% of outliers from the time series. Having assembled a set of successful operations, we used their measured pairwise correlations to organize different ways of classifying the data. Many different entropy measures behave similarly to pNNx measures, as do a diverse set of correlation and scaling-based operations.

Note that we investigated how operations can be combined to construct useful classifiers for this dataset using feature selection in Secs. B.2.4.2 and B.2.4.3 of the appendix. We found that, unlike for the emotional speech and Parkinsonian phoneme datasets analyzed above, combining multiple operations yields minimal improvement in out-of-sample classification accuracy for these RR intervals (an approximately 2% improvement with two features, and minimal improvement thereafter).

5.2.6.5 Discussion

In our highly-comparative analysis of RR interval series, we retrieved operations that were effective at separating normal sinus rhythm and congestive heart failure RR

interval series, and organized them into meaningful groups based on their behavior across the dataset. Operations measuring such properties as location, PNNx, entropy, and long-range scaling correspond to standard measures that have been used to analyze this dataset previously, and we also highlighted some interesting new measures for RR interval analysis. We contrast this highly comparative approach with the extensive and diverse literature on heart rate variability, that consists of isolated studies utilizing either single operations or small sets of operations. Each of these studies could benefit from the context provided by a unified resource of operations for time-series analysis, as is provided here. Not only are standard measures in the literature structured, but new operations are also incorporated and systematically compared to all existing operations. Our highly comparative framework therefore represents a means of automatically synthesizing existing knowledge and directing future research in directions that encourages the development of new methods that do not simply reproduce the behavior of existing approaches.

In future work, our highly comparative approach could be applied to other problems in HRV analysis, including distinguishing different types of pathological dynamics from that of healthy controls, or the effect of interventions like heart transplants or pharmaceutical treatments, to predict the occurrence of pathological cardiac events (which can be framed as a classification task where time series preceding the target event are labeled), or to find operations whose outputs vary with aging or some other characteristic (as a regression task). Hence our framework provides a flexible way of comparing the behavior of operations that scientists have developed both for HRV and for other types signals that can be used to provide an understanding of the relevant differences in dynamics for classification or regression tasks, and to construct accurate classifiers that draw on different time-series analysis techniques simultaneously.

5.3 A highly comparative analysis of fetal heart rate time series

In this section, we present a highly comparative analysis of fetal heart rate (FHR) recordings measured during labour within 30 min before birth from 7 221 different

patients. The case study has both classification and regression components. First we report a set of five operations that are most successful at classifying a balanced training set containing 59 compromised and 59 healthy FHR recordings. We then select five operations with the strongest linear correlations to cord pH across the full dataset of 7 221 FHR time series. By comparing the performance of our extensive library of operations, we discover the types of properties that are important for distinguishing patients with an increased risk of compromise. As with previous studies on this problem, no operations derived from the fetal heart rate series are sufficiently accurate to be clinically viable. Hence, we suggest that future work on this problem might be better directed at combining patient information and features derived from other physiological recordings obtained during labour to better predict fetal health at birth rather than developing new types of operations for FHR analysis.

As noted by Georgieva et al. [190], this problem is one of great significance to the medical community. During birth, a baby's oxygen supply can be compromised and cause birth asphyxia (suffocation). Birth asphyxia can lead to seizures, permanent brain damage, and the death of the newborn. Intervention in the form of a Caesarean section, or the use of forceps or ventouse (vacuum), is required to prevent this chain of events, but can themselves cause complications and would preferably be avoided. Currently, the decision to intervene is made on the basis of an electronic recording of the baby's heart rate during labour, a cardiotocogram (CTG). The mechanisms underlying this recording are complex and its analysis by eye is highly unreliable, whereby different experts have made conflicting decisions on the basis of the same CTG trace [191]. This subjectivity in decision-making can also lead to litigation when an 'incorrect' decision results in a complication. These factors have led to a push for research into an objective, computerized system for analyzing CTG recordings to assist the decision-making process [191]. Previous reports on this topic have been plagued by very small datasets (typically containing less than 500 time series) [192–200]; it is very difficult to make reliable conclusions when so few compromised cases are available. The present work is distinguished both by the large size of the dataset: 7 221 FHR time series, and the scale of the analysis: over 7 000 time-series analysis operations are used.

Note that our aims in this section of the thesis are to contribute to an existing Oxford System for intrapartum CTG analysis *OxSys* [190] by providing a set of useful operations from fetal heart rate time series. Given that our method retrieves many different operations that may be related to one another to varying degrees, an important part of this challenge is ensuring that the set of operations we contribute to this system are a relatively independent set. As shown above for the heart rate variability data (Sec. 5.2.6), we can calculate correlations between the most successful operations on a given classification or regression task to investigate their dependencies. In this section we show that by clustering these operations using this correlation matrix representation, reduced sets of operations can be constructed and interpreted straightforwardly. Note also the term 'features' will be used interchangeably with 'operations' throughout this section, as 'features' is the more familiar term in the biomedical literature.

5.3.1 Data

The FHR time-series dataset analyzed in this work contains 7 568 heart rate time series sampled at 4 Hz. These time series are those that meet a set of quality-based criteria from an initial set of 107 614 deliveries in John Radcliffe hospital, Oxford, UK between 20 April 1993 and 28 February 2008 [190]. We received the data already pre-processed to remove various known artifacts from the recording process [201]. The data were further processed by linearly interpolating short durations of missing values, and trimming longer durations of missing values, and removing time series with a large proportion of missing values, as described and justified in Sec. B.3.1. In Sec. B.3.1 we also describe a partition of the data into training set and testing sets as constructed in a previous study [190]. The data are divided into two groups: *low pH* and *normal pH*, corresponding to fetal heart rate recordings from babies with low cord pH (below 7.1) and with normal pH (above 7.2), respectively. The training set contains 59 time series in each group, and the the test set contains 117 time series in each group. Some examples of time series in each group are shown in Fig. 5.35.

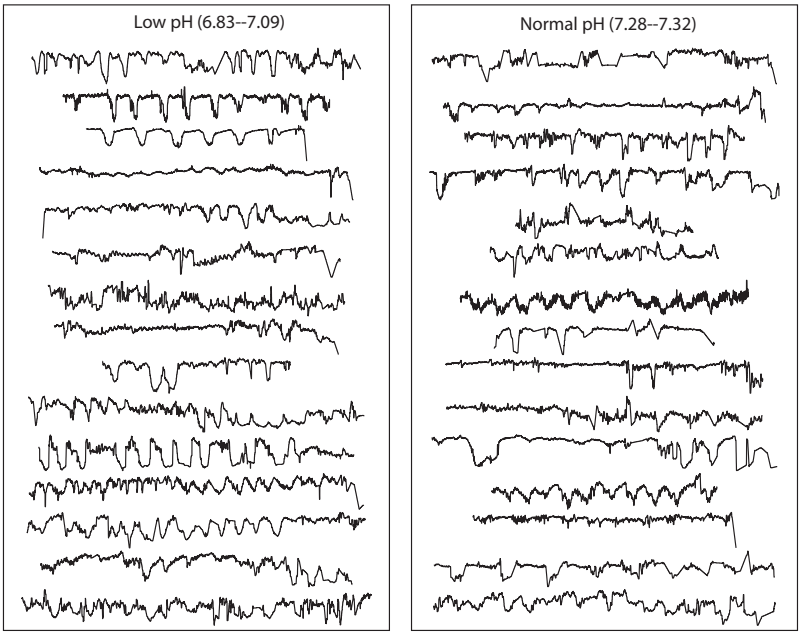

Figure 5.35: **Examples of fetal heart rate time series in each of two classes: low pH and normal pH**. The time series plotted span the full range of pH values in each group, which is given in parentheses.

5.3.2 Classification

In this section, we analyze the balanced training and test sets of time series described above with the aim of classifying those fetuses with low cord pH. We construct a representative set of the most successful individual operations applied to the training set and then evaluate their performance on the test set.

In Fig. 5.36A, a red histogram represents the in-sample training set error across the 7 586 individual operations (that have no special-valued outputs on the training set nor the test set) using a linear classifier. The corresponding out-of-sample test set errors of the same operations are shown in Fig. 5.36B for comparison. For the in-sample classification errors, the permuted distribution peaks at approximately 48%: slightly lower than a random classifier (50%) due to in-sample over-fitting. The out-of-sample shuffled distribution is symmetric and centered around what would be expected of random chance (50%), as it should be. There are clearly many operations that classify the test set with a accuracy much greater than would be expected by chance, indicating that some characteristics of the training set indeed do carry across to the test set.

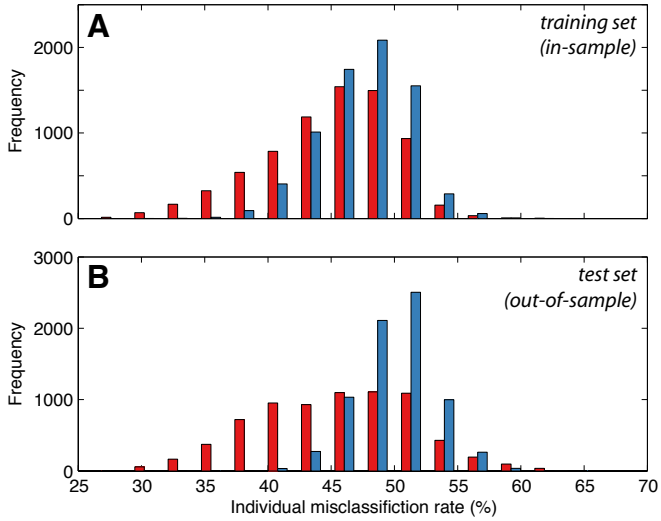

Figure 5.36: **Distributions of in-sample and out-of-sample misclassification rates across 7 586 operations. A** In-sample linear-classification errors (red) and for 200 random permutations of the data labels (blue), **B** Out-of-sample linear classification errors on the testing data using thresholds learned on the training set (red) and for 200 random permutations of the data labels (blue). Many operations are able to classify the fetal heart rate time series measured from babies with low cord pH with an accuracy that exceeds what could be expected by chance.

We used the training set to select the 19 operations with a false discovery rate less than 0.001 (cf. Sec. 3.4.6), which corresponds to operations with an (in-sample) linear misclassification rate under 30%. Since some groups within this set of 19 operations perform very similar functions on the FHR data, we proceed to construct a subset of five operations that minimizes this redundancy. We calculated linear correlation coefficients between all pairs of these 19 operations across the full dataset of 7 221 fetal heart rate time series, as shown in Fig. 5.37. Average linkage clustering was used to construct the dendrogram connecting operations shown in Fig. 5.37. Clustering these operations into five groups using the threshold on the dendrogram plotted with a dashed line, was then used to form the clusters grouped by black squares in the figure. Correlations between operations within each cluster are high and can be reasonably summarized by a single representative member. These representatives are chosen as the operations with the lowest misclassification rate in each group, and are labeled using stars in Fig. 5.37. In this way, the 19 operations with the highest classification rates on the training set were reduced to a more manageable set of five relatively independent measures that effectively summarize the most successful methods from

across many areas of time-series analysis for distinguishing babies with low cord pH from fetal heart rate time series recorded during labour.

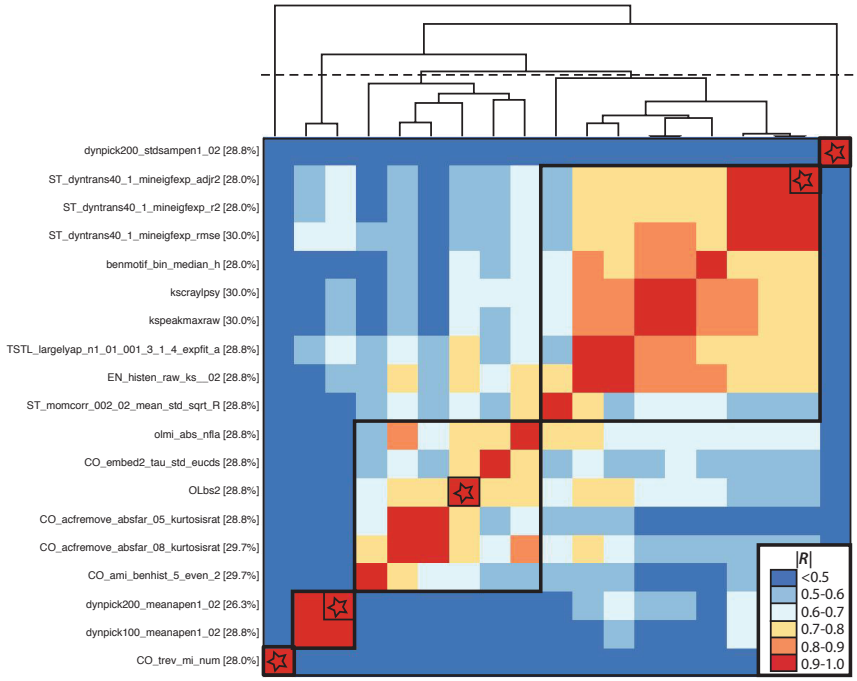

Figure 5.37: **Selecting five operations that best represent the 19 operations with a misclassification rate under 30%.** The magnitude of linear correlation coefficients, $|R|$, calculated between all pairs of the top 19 operations are plotted as a colored matrix. The name of each operation is labeled to the left of the plot and is annotated with its in-sample training set misclassification rate. A dendrogram constructed using average linkage clustering is plotted above the pairwise correlation matrix, and is cut at the point plotted with a dashed line to create five clusters of operations. These groups are represented using black squares in the pairwise correlation matrix. The operations with the lowest misclassification rates within each cluster are selected to represent that cluster, and are represented as stars in the correlation matrix. These five operations are described in the main text.

Given that we are using a fixed partition of training and test data, it is important to evaluate how our results might be dependent upon this choice of partition. This is investigated in Fig. 5.38 for the five operations selected above. We find that each of these operations exhibits an optimistic in-sample classification rate, and a conservative out-of-sample classification rate. This finding is consistent with a general trend that we observed on this dataset and others: when datasets are relatively small yet diverse, selecting the best operations in a single partition is likely to pick out those 'lucky' operations that exhibit higher than average performance on that partition. Ideally, the training set will be large enough to capture the variation in the dynamics

of the time series sufficiently so that operations that are successful on the training set are also successful on the test set. By plotting the distribution of errors across different partitions, we can investigate this effect.

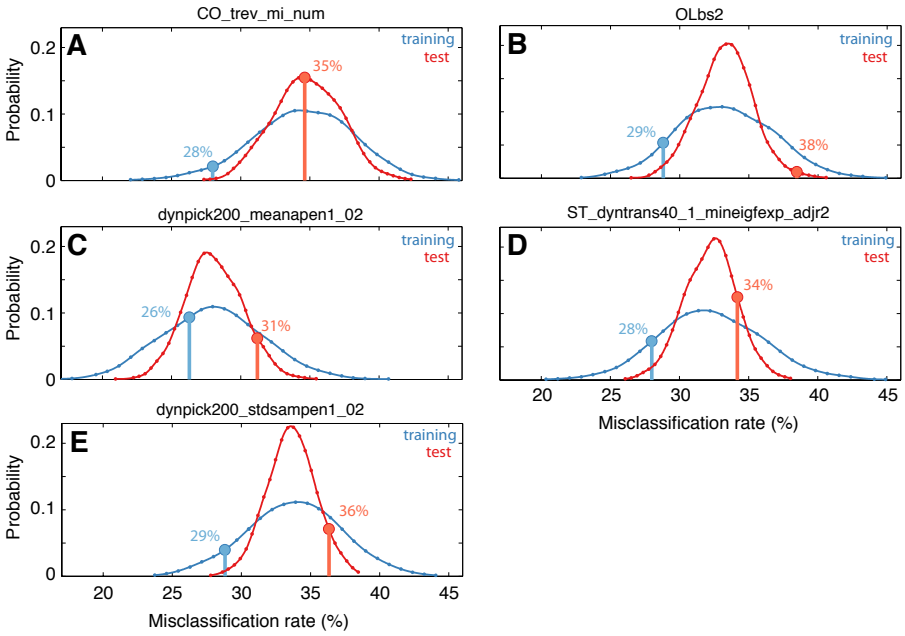

Figure 5.38: **Dependence of training/test classification results on training/test partitions for the top five operations.** In each plot, we show the distribution of training (blue) and test (red) errors over 1 000 different random balanced repartitions of training and test data. The training and test classification errors for the partition used in this work are shown with circles and lines, and labeled with text. We observed a general trend for different partitions of the data: lower than average training errors correspond to higher than average test errors, and vice-versa (not shown). This trend is seen here: operations with the lowest training errors were selected, and lead to test errors that are overestimated compared to what would be expected from the set of possible equivalent data partitions.

We now evaluate the performance of each of these five operations selected above on both the training and test sets. Distributions of their outputs are shown for both healthy and low pH FHR time series in Fig. 5.39. The five operations are summarized briefly here, and are described in more detail in Sec. B.3.3: (i) **CO_trev_mi_num** is a quantity related to the time-reversal asymmetry of a time series, (ii) **OLbs2** returns the ratio of standard deviations before and after removing 2% of the highest and lowest values of a time series, (iii) **dynpick200_meanapen1_02** averages local measures of ApEn(1,0.2), (iv) **ST_dyntrans40_1_mineigfexp_adjr2** calculates 1-step transition matrices for different alphabet sizes and fits a decaying exponential to the mini-

mum eigenvalues of these transition matrices, and (v) **dynpick200_stdsampen1_02** measures the variation in local SampEn(1,0.2) estimates from the time series. The behavior of each of these operations is plotted in Fig. 5.39. As shown in the figure, in-sample misclassification rates range from 26–29% and out-of-sample misclassification rates range from 31–38%.

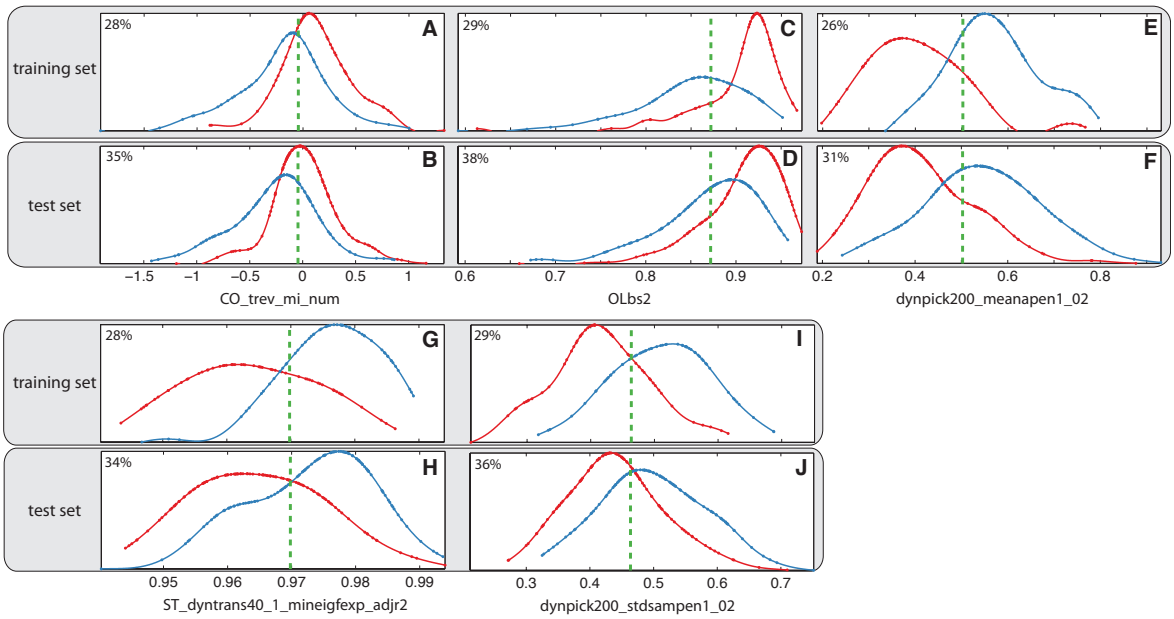

Figure 5.39: **Performance of the five representative operations with a classification error $< 30\%$, chosen from the training set, cf. Fig. 5.37.** For each of the five operations, the distribution of outputs in the 'compromised' group (red) and the 'healthy' group (blue) are plotted for both the training data (upper plots) and testing data (lower plots). The linear discrimination threshold (dotted green line) is fitted from the training data, and used to classify the testing data. Classification errors are annotated to the top left of all plots. Each operations is explained in detail in the main text.

Note that classifiers that combine multiple features for this dataset showed no improvement in out-of-sample performance over single-feature classifiers, as explained in Sec. B.3.1.1 of the appendix.

5.3.3 Regression onto arterial cord pH

Instead of the classification task studied above, in this section we investigate operations that correlate linearly with the known arterial cord pH. Linear correlation coefficients, R, between the outputs of 7 171 different operations and arterial cord pH were calculated, producing a distribution that is plotted in Fig. 5.40. Many

operations produce outputs that correlate with the known pH greater than would be expected by chance. Even using the conservative Bonferroni multiple hypothesis testing method with $\alpha = 0.05$ (cf. Sec. 3.4.6), 3 038 operations are significantly correlated with cord pH. Some structure is evidently being detected, demonstrating that feature-based summaries of fetal heart rate time series can indeed be used to contribute to predicting cord pH. However, this prediction cannot be done accurately using single FHR features alone: linear correlations are weak, $|R| < 0.3$. Stronger correlations could be achieved in future by combining more information into the regression procedure, including information about the mother and other physiological features recorded during labour.

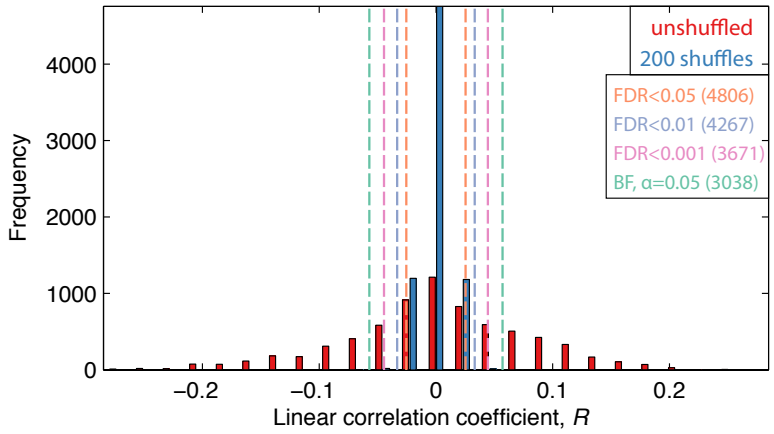

Figure 5.40: **Distribution of linear correlation coefficients, R, of the output of 7 171 operations on 7 221 fetal heart rate recordings and the corresponding arterial cord pH of the newborn babies**. The distribution of coefficients are plotted as the red histogram, and a shuffled version, for which the pH assignments to all time series are randomly permuted 200 times, is plotted in blue. Various statistical significance thresholds are also plotted, as explained in Sec. 3.4.6, whereby operations with greater $|R|$ than these thresholds are judged as significant. Although correlation coefficients are weak ($|R| < 0.3$), many of the operations are clearly measuring useful structure in the fetal heart rate time series.

We now use clustering to select a relatively independent set of operations that summarize those that perform best at this regression task onto arterial cord pH in the same way as for the classification task described in Sec. 5.3.2 above. Like for the classification task above, the fifty operations with the strongest linear correlation coefficients, $|R|$, to cord pH, can be effectively summarized by just five operations, as explained in Sec. B.3.2 of the appendix. Scatter plots of these five representative

operations and cord pH are plotted in Fig. 5.41. A brief description of each operation is as follows: (i) **cv2** returns the second order coefficient of variation: $(\sigma/\mu)^2$, where σ and μ are the standard deviation and mean of the time series, respectively, (ii) **mead** returns the median absolute deviation, $\langle|x - \text{median}(x)\rangle$, a measure of spread of the time series, x, (iii) **ST_dyntrans40_1_mineigfexp_adjr2** is the quantity derived from transition matrices described in the classification task above, (iv) **fd_exp1_rmse_h30** returns the goodness of an exponential fit to the time-series distribution, and (v) **CO_embed2_tau_arearat** returns the ratio of areas spanned by points in a two-dimensional time-delay embedding space for the time series. Each of these these operations are described in more detail in Sec. B.3.3. Linear correlation coefficients for these operations range between $|R| = 0.24$ to $|R| = 0.28$, as shown in Fig. 5.41. Interpreting the sign of the correlation also allows us to interpret the results, e.g., Fig. 5.41D shows that FHR recordings associated with high cord pH have distributions that are closer to exponential than those with low cord pH.

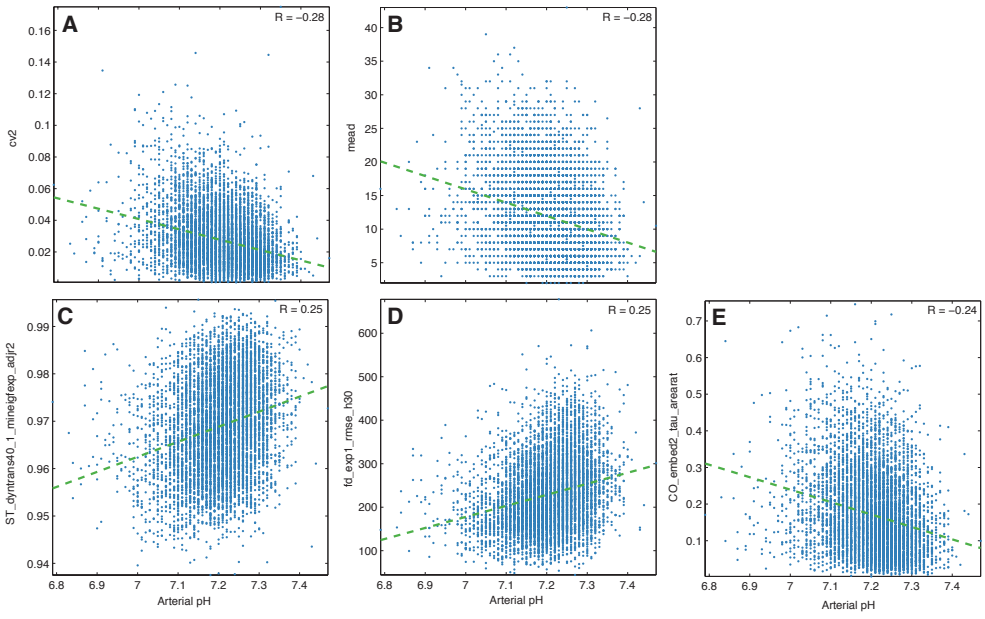

Figure 5.41: **Scatter plots of the output of five representative operations against arterial cord pH.** Least-squares linear fits to each scatter plot are plotted as dashed green lines, and the correlation coefficients, R, are annotated to the top-right of each plot. Each plot contains 7 221 points, each corresponding to a time series in the full database of fetal heart rate recordings.

5.3.4 EveREst plots

In the large database of fetal heart rate recordings studied in this work, the great majority correspond to patients with no compromise. For example, consider the following three groups: (i) *low pH*, defined as patients with an arterial cord pH ≤ 7.05, contains 302 patients, (ii) *compromised*, defined as patients with a reported severe, moderate, or mild reported compromise [190], contains 795 patients, and (iii) *low pH and compromised*, defined as patients that fulfill both of the above criteria, contains just 110 patients. These are the patients that are of the most interest to clinicians that must decide whether an intervention is appropriate during labour. Distinguishing such a small number of problematic scenarios from such a large total cohort of 7 221 patients is a difficult problem. Plotting distributions of outputs from an operation for each class, as in Fig. 5.38, now requires a different interpretation because the healthy distribution contains orders of magnitude more samples than those with low arterial pH or compromise. One way of proceeding, which we follow here, is to divide the total cohort into N_{group} equally-populated groups and compare the proportion of compromised cases in each group. A graphical representation of this concept has been termed an *Event Rate Estimate* (EveREst) plot [190]. By ordering all fetal heart rate time series by the output of a given operation on them, we constructed such plots using $N_{\text{group}} = 10$ here.

An example is shown in Fig. 5.42 for the *mead* measure of spread, $\langle |x - \text{median}(x)| \rangle$. The distributions in Fig. 5.42A are misleading for predicting an ability to classify the low pH condition because there are only 302 recordings in this group (plotted red), compared to the 6 919 other recordings with normal pH that make up the blue distribution. However, dividing the patients up into equal groups, as in Fig. 5.42B, yields a representation that shows the proportion of target patients in each equally-populated group, which can be used to determine thresholds by which the two groups could be separated. For this measure, there is a relatively sharp rise in target cases in the final bin; patients in this bin, with $\langle |x - \text{median}(x)| \rangle > 19.25$, therefore have an increased risk of delivering a compromised baby, or one with low arterial cord pH. The *mead* spread measure is extremely simple to compute, and could in this way be

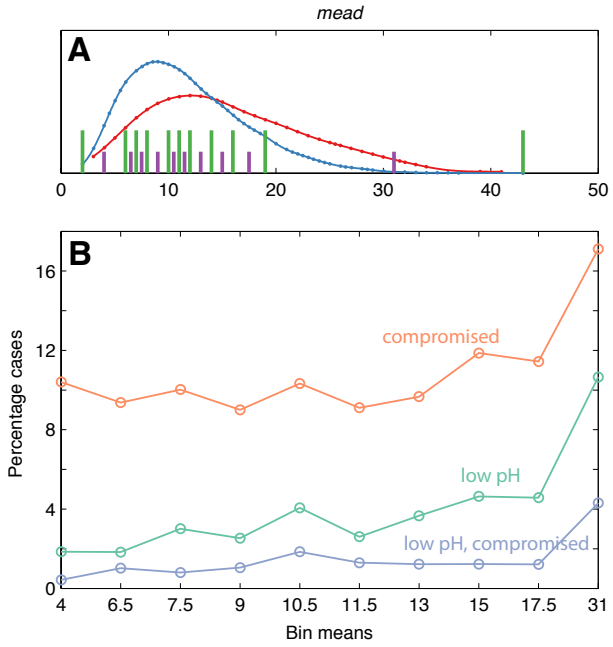

Figure 5.42: **Distribution and EverEst plot for a measure of spread: mead.** In the upper plot, we show distributions of the *low pH* (red) and *normal pH* (blue) groups defined using the pH threshold 7.05. There are 302 fetal heart rate recordings with a corresponding arterial cord pH $\leq$ 7.05, and 6 919 with pH $>$ 7.05. The EverEst plots in the lower panel was generated by dividing all 7 221 patients into 10 equally-populated groups, ordered by the outputs of the target operation. The data partitions that define these equally-populated groups are shown using green lines in the upper plot and in the lower EverEst plot, the ten groups are represented by their group means (purple lines in the upper plot). In these plots we represent three different types of compromised patients: (i) pH $\leq$ 7.05, plotted green (and represented as distributions in the upper plot), (ii) compromise, and (iii) pH $\leq$ 7.05 and compromised. This indicates that a useful indicator of a potential compromise would be simply to measure the spread of recordings during delivery, and a high value (greater than $\approx$ 25 for this *mead* statistic) could be used to alert an increased probability of compromise.

used as input to a decision-making algorithm. Note that the other features selected in the classification and regression tasks above have qualitatively similar EveREst plots to that shown in Fig. 5.42B for this mead operation.

The idealized situation has the two distributions (in Fig. 5.42A) completely separated so that the EveREst plot jumps abruptly from 0 to 100 (or 100 to 0). In this case, that operation is able to predict the target condition without error, and the decision to intervene during labour would be unambiguous. Of course, as with most medical applications, this is an extremely complex problem that depends on a large

number of variables, just one of which may be a summary of a fetal heart rate time series. In this case, we find that using just the mead statistic allows the probabilistic inference of a group of patients with a greater risk of compromise and low arterial cord pH. Although as much as an 8-fold increase in risk is seen here (for the mead statistic and the low pH and compromised group in the final bin), this is clearly too low to be clinically useful in isolation. However, as mentioned previously, measures from other physiological recordings during labour and prior information about the mother and the pregnancy may ultimately be combined with this information in future work to produce a decision rule that forms the basis of some objective framework for intervention during labour.

5.3.5 Conclusions

In summary, we have applied our highly-comparative method for selecting features that are useful for predicting babies with low cord pH from fetal heart rate time series recorded during labour, within 30 min from birth. Five features were selected as representative of those most successful on a training classification dataset, and five were selected to represent those with the strongest linear correlations to arterial cord pH across a large dataset of 7 221 time series. One of these features occurs in both sets, leaving a set of nine candidate features that will become part of the Oxford System for intrapartum CTG analysis: *OxSys* [190]. Although not clinically useful on their own, combined with other CTG features and clinical data, future work will focus on using these features to build a commercial diagnostic system—an intrapartum analogue of the established Dawes-Redman system [202].

This case study demonstrates the broad applicability of our highly-comparative methodology that allow us to address specific questions asked of different datasets. In this case study, we were interested in determining which properties of the fetal heart rate time series help to predict a low pH condition in newborn babies. After processing the data appropriately, we set-up a classification task and a regression task, and were able to automatically retrieve sets of useful operations, and cluster them based on their correlations to one another to further distill a relatively independent set. Although our set of thousands of operations from across time-series analysis is far

from exhaustive, our results suggest that future research that continues to focus on developing operations that might be able to distinguish compromised cases from fetal heart rate time series may be better targeted towards other aspects of the task. For example, future work might be better focused on combining features of fetal heart rate time series with knowledge of the patient and other recordings taken during labour with the aim of developing an objective system for guiding intervention during labour.

Chapter 6

Highly comparative time-series analysis for temporal data mining

In this chapter, we introduce a highly comparative, feature-based approach to time series clustering and classification and demonstrate its utility on a broad range of standard datasets in the field of temporal data mining. The chapter is somewhat self-contained and is written for a data mining audience. Our method relies on the extensive database of over 9 000 operations for extracting features from time series developed in this thesis. Rather than selecting a time-series similarity measure manually, our framework allows the structure of the data to motivate the selection of useful features. Because our approach uses features extracted from time series, it can straightforwardly accommodate long time series, datasets containing time series of different lengths and, because classification rules can be learned on small sets of extracted features, our approach is applicable to very large datasets.

6.1 Introduction

Throughout this thesis, we have demonstrated the usefulness of a feature-based representation for time series, using a large and diverse library of operations. Hitherto, this interdisciplinary wealth of scientific methods for understanding the properties of time-series has been largely absent from the temporal data mining literature, which mainly treats the problem of clustering and classifying large time-series datasets. In this chapter, we address this problem by exploiting our collection of over 9 000 operations for estimating different properties of time series to tackle a range of time-series clustering and classification problems.

The two main challenges of time series classification (or clustering) involve: (i) choosing an appropriate *representation* of the time series, and (ii) selecting a suitable measure of similarity or *distance* between time series [203]. The literature on representations and distance measures for time series clustering and classification is vast [71, 203, 204]. The simplest representation of time series is simply their time-domain form, such that distances between time series relate to distances between the temporal objects themselves. This basic representation of time series in the time domain dominates the temporal data mining literature. However, an alternative approach involves representing time series using a set of derived structural properties, or features. To take a very simple example, one may choose to represent a time series using just two numbers: its mean and variance, thereby transforming time-series objects of any length into short vectors that encapsulate two important properties. There are many examples of successful applications of this approach: e.g., Timmer et al. [205] characterized hand tremor time series using a variety of time- and frequency-domain measures, Deng et al. [206] represented time series using ARMA(p, q) model parameters, Nanopoulos et al. [207] used the mean, standard deviation, skewness, and kurtosis of the time series and its successive increments to represent time series, Siirtola et al. [208] compressed accelerometer time series to local maximums before extracting basic distributional features from them, and Mörchen [209] used features derived from wavelet and Fourier transforms of time series. More recently, a broader set of thirteen features has been assembled that contains measures of trend, seasonality, periodicity, serial correlation, skewness, kurtosis, chaos, nonlinearity, and self-similarity [210], and has since been extended to represent multivariate time series [94]. Another recent paper has attempted to combine distances between pre-defined time-series patterns (or 'templates') and a time series, with a small set of basic features extracted from the time series [211]. The process of selecting which features (and how many of them) to use is rarely discussed and never treated systematically in the data mining literature, although it is often a very difficult problem that lacks a theoretical framework to justify any particular choice. By comparing the performance of a large number of features and selecting those that are most effective for a given task, our highly comparative method is able to guide the selection of features for any

given dataset in a data-driven fashion.

In Chapter 4 of this thesis, a library of over 9 000 features for time-series analysis was used to organize an extensive interdisciplinary database of over 20 000 empirically-measured and synthetically-generated scientific time series. The information contained in this library was used to automate many aspects of time series regression and classification problems. For many time-series datasets, it is natural to use a feature-based representation of time series: from detecting the emotional content of speech signals to diagnosing epileptic electroencephalogram (EEG) recordings, as demonstrated in Chapter 5. For these applications, relevant summaries of the type of dynamics present in each time series are important, whereas a direct comparison of the actual values in the time domain is not. For other applications, however, where short time series encode meaningful patterns, for example, it therefore makes sense to work directly with the time-domain representation; this type of problem has traditionally been the focus of the time series data mining community [71, 203]. However, given the improved dimensionality reduction of a feature-based representation, perhaps it may also be a useful representation of such time series. We address this question in this work, and present a general feature-based classification method that can be used to analyze time-series datasets effectively. We find that feature-based representations of many time-series datasets can yield faster and superior classification performance compared to conventional time-domain representations.

Many papers in the field of data mining champion a particular method—be it a new representation, distance measure, or classifier—typically by demonstrating its utility on a small number of datasets (in one survey, such studies apply their methods to an average of 1.85 datasets [71]), and with minimal comparison to other methods (such studies compare their method to an average of 0.91 others [71]). This lack of comparison to alternative analysis methods and 'cherry-picking' of favorable datasets makes it difficult to discern whether progress is being made in the field of time-series classification [71]. Two key problems with the time series data mining literature have been identified as (i) *implementation bias*: the disparity in the quality of implementation of a proposed approach versus that of competing approaches, and (ii) *data bias*: the use of a particular set of testing data to confirm a desired finding [71]. We raise an

additional concern about the focus on developing 'better' methods in isolation from a guiding motivation from either a scientific question asked of the data, or its structural characteristics. There are principled reasons why we can expect classification methods to perform well on some datasets and fail on others [212], but it is important to understand what characteristics of different datasets make them more suited to particular methods of analysis. In this way, we move away from the search for 'one-size-fits-all' classifiers, that can lead to misleading generalizations—e.g., "keeping low-frequency information, as in DFT, is good for most applications" [213] (see also Keogh and Kasetty [71]), and towards a framework in which the structure of the dataset is used to motivate appropriate analysis methods. In the framework we develop in this chapter, the structure of the dataset is used to select features that summarize important properties of time series. By selecting features in this way, classifiers adapt to structure in the dataset and allow us to understand which properties of a time-series dataset are important for classifying it.

This chapter is structured as follows. In Sec. 6.2, we introduce our general framework for representing time-series datasets, describe the datasets analyzed in this work, and explain our method for retrieving relevant features for time-series classification tasks. In Sec. 6.3.1, we show how a general feature-based representation of time series can uncover meaningful structure in time-series datasets through clustering, and in Sec. 6.3.2, appropriate low-dimensional, feature-based representations of time series are constructed automatically using feature selection and used to train successful classifiers. The main results are discussed in Sec. 6.4, and the key findings are summarized in Sec. 6.5.

6.2 Data and Methods

Central to our approach to time series clustering and classification is the ability to represent time series using a large and diverse set of their measured properties. In this section, we describe our methods for constructing this representation, and explain how this representation is used to cluster and classify time-series datasets. First, in Sec. 6.2.1, the datasets analyzed in this chapter are described. The feature-vector

representation of time series is explained in Sec. 6.2.2, and we then discuss the methods used to perform feature selection and classification in Sec. 6.2.3.

6.2.1 Data

The twenty time-series datasets analyzed in this work were obtained from *The UCR Time Series Classification/Clustering Homepage* [214]. Note that this resource has since been updated (late in 2011) to include an additional twenty-five datasets [215], which are not considered here. The datasets are listed in Table C.1, and span a variety of time-series lengths, N, from $N = 60$ (for the 'Synthetic Control' dataset) to $N = 637$ samples (for 'Lightning (two)'); dataset sizes, from a number of training time series, $n_{\text{train}} = 28$, and test time series, $n_{\text{test}} = 28$ (for 'Coffee'), to $n_{\text{train}} = 1\,000$, $n_{\text{test}} = 6\,164$ (for 'Wafer'); and the number of classes, n_{classes}, from $n_{\text{classes}} = 2$ to $n_{\text{classes}} = 50$. The datasets also encompass a broad range of applications: including measurements of a vacuum-chamber sensor during the etch process of silicon wafer manufacture ('Wafer' [216]), spectrograms of different types of lightning strikes ('Lightning' [217]), and the shapes of Swedish leaves ('Swedish leaf' [218]), and yoga poses ('Yoga' [219]). Note that many of these datasets are not strictly time series, but all are sequential measurements (e.g., leaf shapes are converted to 'time series' by rotating them about a point and measuring the distance to the edge of the shape, and 'time series' of yoga poses are sequences of distances measured from a yoga mat to the body for a given pose); however, being discrete, sequential measurements, all such data objects are analyzed in the same way as time series. All the data is used exactly as obtained from *The UCR Time Series Classification/Clustering Homepage* [214] (i.e., without any preprocessing), including the specified partitions of each dataset into training and testing portions. The sensitivity of results to these given partitions for all datasets are investigated in Sec. C.4 of the appendix.

6.2.2 The feature vector representation

Representing time series as sets of features summarizing their properties is a very general task that could be achieved in a variety of ways. One approach involves manually analyzing the structure in a given dataset, and using that insight, combined

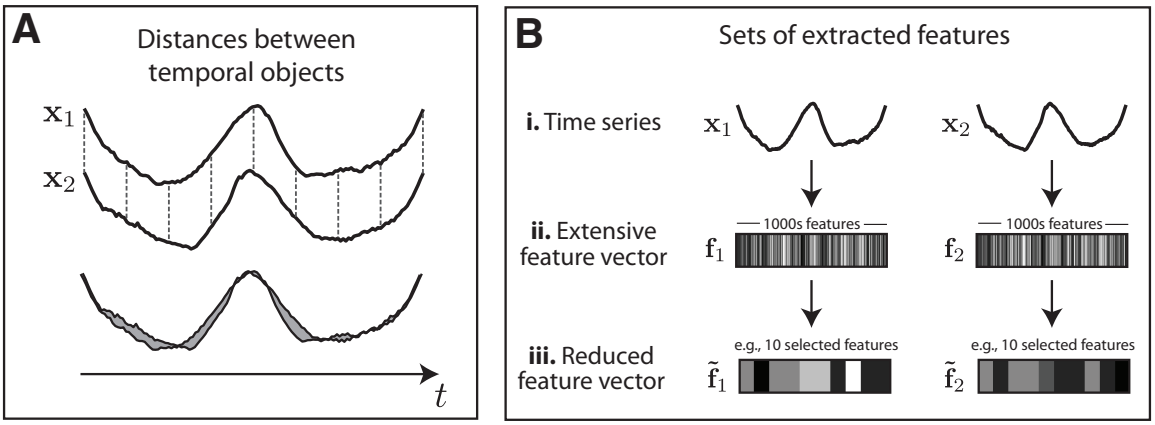

Figure 6.1: **An illustration of two approaches to representing time series.** **A** The first approach involves measuring the distance between the two objects in the time domain. In the upper portion of the plot, the two time series, $\mathbf{x}_1$ and $\mathbf{x}_2$, are offset vertically for clarity, but are overlapping in the lower plot, where shading has been used to illustrate the total distance between $\mathbf{x}_1$ and $\mathbf{x}_2$. The simplest distance measure, that simply compares the values of the time series at each time point, is visualized here, although more complex choices exist (see main text). **B** An alternative approach, that we focus on in this work, involves representing time series by a set of features that summarize their properties. Time series, $\mathbf{x}$, are used as inputs to a large number of time-series analysis operations, producing an extensive set of thousands of features, $\mathbf{f}$, that summarize properties of the distribution, autocorrelation, entropy, goodness of fits to various models, etc. These features can be filtered or transformed to produce a reduced representation of the time series, $\tilde{\mathbf{f}}$. Distances between $\tilde{\mathbf{f}}$ then define distances between the time series they represent, or can be used to learn a classification rule. In the figure, feature vectors are represented using a grayscale colormap, where black represents low values and white represents high values of a given feature.

with knowledge of the generative processes underlying the data, to motivate a relevant set of features to distinguish them. This approach is obviously best suited to applications in which knowledge of the structure in the data and the process by which it was measured (or synthesized) is available. However, for many applications this is not the case, and even when knowledge of a time-series dataset is plentiful, it is not obvious what combinations of extracted features will yield the best representation with which to distinguish the known data classes. By computing a very large number of features using a diverse library of time-series analysis operations and selecting those with the best performance, the feature selection process can be automated, requiring no knowledge of the time series or the processes through which they were generated.

The distinction between a more conventional approach to time series data mining,

which measures distances between the time series represented as objects in the time domain, and our method, which measures distances between a large set of features extracted from time series, is illustrated in Fig. 6.1. The so-called 'lock step' distance measure is visualized in Fig. 6.1A, which measures distances between time series at each time point, and is the simplest class of distance measure [203]. Other choices, like 'elastic', 'editing', or 'threshold' measures are more complex, and can accommodate unaligned patterns, for example [203]. Dimensionality reduction methods operating in the time domain can speed up these calculations, e.g., using discrete Fourier transforms, piecewise constants, piecewise linear interpolants, symbolizations such as SAX, and so on [220]. Our feature-based method is illustrated in Fig. 6.1B, and involves constructing an extensive feature vector which is used to represent, cluster, and classify time-series datasets.

6.2.3 Feature selection and classification

In this chapter, we use greedy forward feature selection with either linear classifiers or k-NN classifiers with $k = 1$ (i.e, 1-NN classifiers). For 1-NN classifiers, each point in the training data is classified as its nearest neighbor in the training set (using Euclidean distances in a given feature space) and the resulting classification rate is used as the statistic to select features. To keep the testing data completely unseen, the parameters of the normalizing transformation, Eq. (3.2), i.e., the median and interquartile range of a feature's outputs, are calculated using just the training portion of each dataset; the transformation is then applied to the full dataset (including the testing data). Linear classifiers can be constructed in either the raw or normalized feature spaces, whereas 1-NN classifiers require each feature to have the same range and are only applied to normalized feature spaces.

As explained in Sec. 3.4.5, the greedy forward selection procedure incrementally grows a set of important features in a way that reduces the in-sample misclassification rate on the training set. An important choice, therefore, is when to terminate this process—how many features should be used to form a classifier for the test data? In this work, we choose a simple cutoff as the point at which the improvement in the training set classification rate drops below 3%. Alternatively, if the training

set classification rate reaches 100%, no improvement is possible and the process is terminated. This choice to set a threshold at 3% is an arbitrary one that allows us to demonstrate the usefulness of the feature selection procedure for time-series classification in this work; a more principled method for determining this cut-off (e.g., using cross-validation) will be explored in future work.

Throughout this chapter, we distinguish classifiers operating on time series represented in the time domain with the label 't :', and those operating on a feature-based representation with the label 'f :'. We show results for three different temporal representations, which are labeled as: (i) 't : Euclidean 1-NN', a 1-NN classifier using Euclidean distance, (ii) 't : DTW 1-NN', a 1-NN classifier using a dynamic time warping distance, and (iii) 't : DTW 1-NN (best warping window, r)', a 1-NN classifier using a dynamic time warping distance with a warping window learned using the 'Ratanamahatana-Keogh Band' [221]. These results were obtained from *The UCR Time Series Classification/Clustering Homepage* [214]. The Euclidean results using a 1-NN classifier were verified by us: results were consistent with those reported in the *UCR* source [214]. Three feature-based classifiers are compared in this work, which are labeled as (i) 'f : linear', linear discriminant classification using unnormalized features, (ii) 'f : linear (norm)', linear discriminant classification using normalized features, and (iii) 'f : 1-NN', nearest neighbor classifier using normalized features.

6.3 Results

In this section we present our analysis of time-series datasets using the feature-based representation of time series described above. In Sec. 6.3.1, we show how PCA can be used to construct useful low-dimensional representations of some datasets. In Sec. 6.3.2, feature selection and classification results are presented and compared to conventional classifiers that use distances calculated in the time domain.

6.3.1 Low-dimensional representations

Each Principal Component (PC) is a linear combination of all features in the original (normalized) feature space, and the leading PCs approximate the combined behavior of the full set of operations acting on a dataset. A useful two-dimensional representa-

tion of a given dataset in terms of its measured properties can therefore be constructed using these two PCs. We have seen this in the time-series datasets studied in Chapter 5, and the results can also provide useful organizations for these datasets, as described in Sec. C.2 of the appendix.

Here we focus on a single example of how PC projections of feature spaces can give insights into the structure of time-series datasets, using the 'Swedish Leaf' dataset, which contains shapes of fifteen types of Swedish leaves (cf. Table C.1). The two-dimensional PC projection of this dataset (using the 6 557 features with a non-zero interquartile range and no special-valued outputs) is shown in Fig. 6.2A. Each leaf type occupies a distinctive region in this space, and similar leaf shapes are positioned near to one another. The known labels of the data were also used to construct a dendrogram between the fifteen classes, which is shown in Fig. 6.2B. Euclidean distances between classes were calculated in the two-dimensional PC space in Fig. 6.2A, and each class is represented by its class center, defined as the mean value of points in the class in this space. Even though the patterns themselves are not being clustered—rather a reduced two-dimensional representation of their properties—the dendrogram corresponds to an intuitive grouping of them. With just these two dimensions, which were constructed using only the training data and using no knowledge of the class labels, we are able to form a useful feature-based representation of this dataset of leaf patterns automatically. Given that this dataset is naturally suited to a conventional representation of the shapes as objects in the time domain, the success of this unsupervised feature-based approach is encouraging.

This result, along with those described in Sec. C.2, demonstrates that unsupervised, feature-based representations of short, pattern-based time series can indeed be useful. However, it is often more important for practical applications to learn classification rules in low-dimensional feature spaces. This supervised classification problem using feature selection is explored in the following section.

6.3.2 Classification

In this section, feature selection is used to select interpretable subsets of features to form classifiers for time-series datasets. First we demonstrate the method on specific

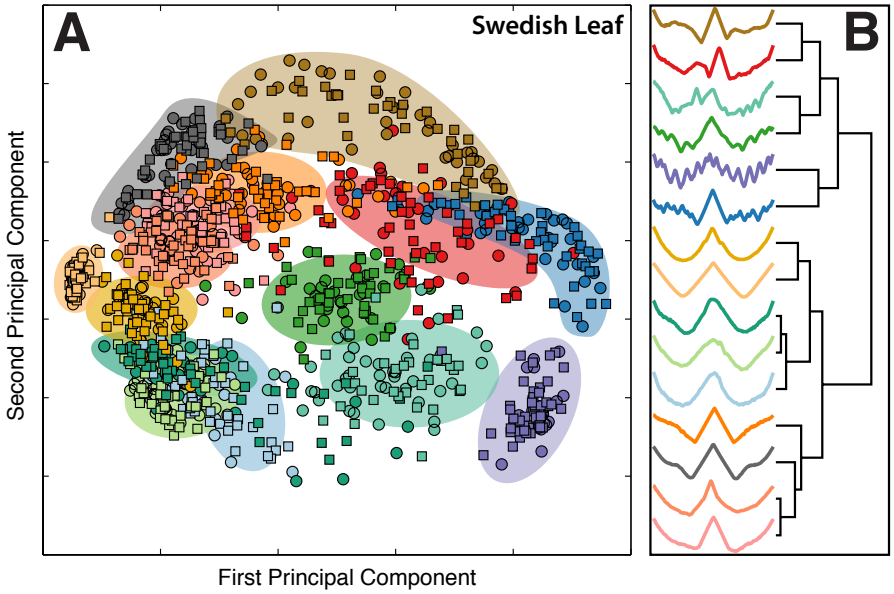

Figure 6.2: **The class structure in the 'Swedish Leaf' data set is reflected in the two-dimensional Principal Components space of our library of features**. **A** The space of the two leading PCs, formed using the normalized training data. When the training (circle) and test (squares) data are plotted in this space, known classes of leaf patterns occupy characteristic regions of the space, and similar leaf patterns are placed near one another. Background shading has been added manually to guide the eye. **B** A dendrogram between class centers in this space was constructed using average linkage clustering and reflects a visually intuitive structuring of the types of leaf patterns in this dataset.

datasets, and then compare the results to alternative classification methods.

6.3.2.1 Specific datasets

In this section, we demonstrate the feature selection method for constructing classifiers for time-series datasets using selected examples.

For some datasets, we found that the first feature selected using greedy forward selection accurately distinguishes the labeled classes. Examples are shown in Fig. 6.3. For the 'Trace' dataset, the first selected feature in our database (i.e., that with the lowest misclassification rate on the training data) is a time-reversal asymmetry statistic, t_{rev}. As shown in Figs. 6.3A and 6.3B for the training and test sets, respectively, this single feature accurately distinguishes all four patterns in this dataset. The quantity is defined by

$$t_{\text{rev}}(\tau) = \frac{\langle (x_{t+\tau} - x_t)^3 \rangle}{\langle (x_{t+\tau} - x_t)^2 \rangle^{3/2}}, \tag{6.1}$$

where x is the time series, τ is the time lag ($\tau = 3$ for this operation), and averages, $\langle\cdot\rangle$ are performed across the time series [222–224]. Using a basic linear classifier, simple thresholds in the feature space are learned from the training set, allowing the test set to be classified without error. This also circumvents the need to calculate any distances between time series, allowing new time series to be classified extremely quickly (cf. Sec. 6.3.2.3).

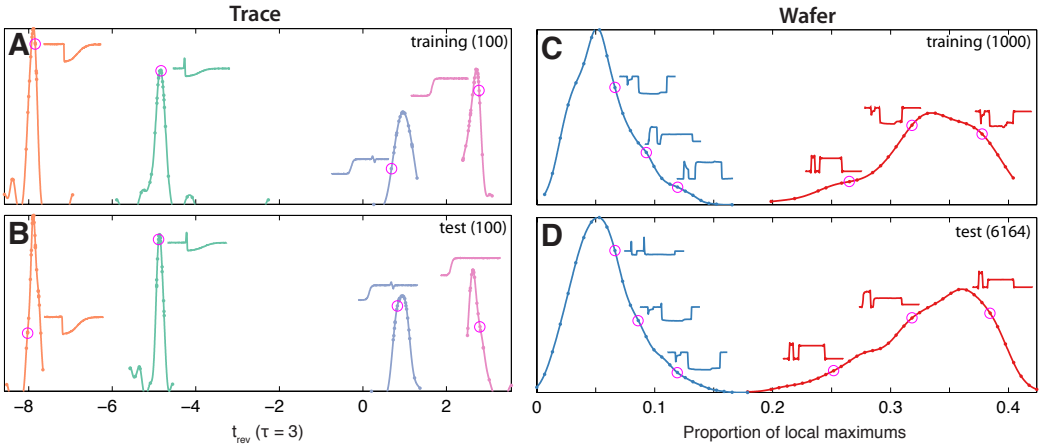

Figure 6.3: **For some datasets, single extracted features can separate the labeled classes.** We show two examples: **A** and **B** show the feature t_{rev} with $\tau = 3$ for the 'Trace' dataset, and **C** and **D** show the proportion of local maximums in time series from the 'Wafer' dataset. The distribution of the labeled test statistic is plotted for each of the labeled classes for the training (upper panels) and test (lower panels) data separately. The classes are distinguished by color, and selected time series (indicated with a purple circle) are annotated to each distribution.

Another example is shown in Figs. 6.3C and 6.3D for the 'Wafer' dataset. Both classes in this dataset are quite heterogenous, as can be seen from the time-series annotations in the figures. However, a very successful single feature for this dataset simply counts the number of local maximums in the time series (i.e., the number of times the time series increases in one time step and then decreases in the next time step), expressed as a proportion of the time-series length. Using the simple classification threshold learned on the training set, the test set is classified with an accuracy of 98.4%.

Instead of selecting just a single feature, classifiers can also be constructed using combinations of multiple features. One example is shown in Fig. 6.4 for the 'Synthetic Control' dataset. The 'Synthetic Control' dataset contains six classes of time series,

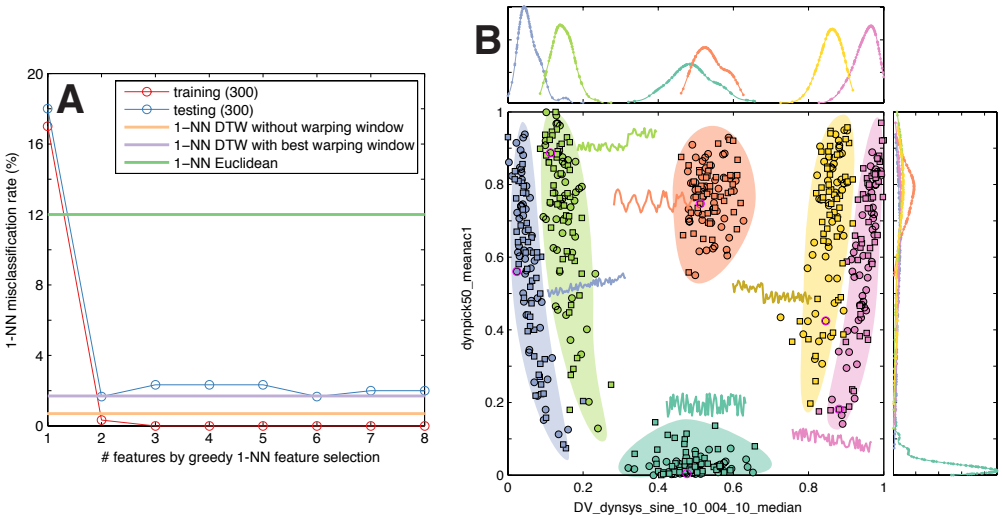

Figure 6.4: **Feature selection for the 'Synthetic Control' dataset using a 1-NN classifier with normalized features. A** The misclassification rate decreases as the number of features increases, and approximately two features are required for accurate classification (approximately 97% accuracy). Misclassification rates for classifiers that use distances between time-series objects are also shown. **B** The training (circles) and test (squares) data clearly separated by two features, which are described in the main text. Selected time series have been annotated to the plot, as indicated by pink circles, and background shading has been added manually to guide the eye.

each with different distinctive dynamical properties; it therefore makes sense to use a feature-based approach to classify the signals according to their properties rather than their values. Appropriate features could be selected by a time-series analyst studying the patterns in the time series and devising appropriate statistics that might separate them into their known classes. However, we show here that this feature selection can be performed automatically by comparing the classification performance of a large number of time-series analysis operations. In Fig. 6.4, we show the results of a 1-NN classifier in the feature space, but qualitatively similar results were obtained using a linear classifier.

The greedy 1-NN feature selection process (using normalized features) is illustrated in Fig. 6.4A. At each iteration, the feature is selected that minimizes the in-sample training set misclassification rate. The misclassification rate in both the training and test sets drops dramatically when a second feature is added to the classifier, but plateaus when subsequent features are added. These first two selected features are used to represent the time-series dataset in Fig. 6.4B. The first selected

feature, **DV_dynsys_sine_10_004_10_median**, behaves in a way that is analogous to performing a cumulative sum through time of the (z-scored) time series (the cumulative sum, S_t, is defined as $S_t = \sum_{i=1}^{t} x_i$), and then returning its median[1]. This operation gives high values to time series that decrease (the cumulative sum increases and then decreases back to zero), moderate values to time series that are approximately mean-stationary (the cumulative sum oscillates about zero), and low values to time series that increase (the cumulative sum initially decreases and then increases back to zero).

As shown in Fig. 6.4, this first cumulative sum-based feature distinguishes all the classes well, except for the uncorrelated random number series (green) and oscillatory time series (red), neither of which exhibit a strong underlying trend. To compensate for this, the second selected feature, **dynpick50_meanac1**, clearly distinguishes these two confused classes using their difference in autocorrelation properties. This operation behaves like the autocorrelation at lag 1 of the time series[2]. The oscillatory time series have a high autocorrelation, whereas the uncorrelated random number sequences have an autocorrelation close to zero.

For this dataset, the behavior of each selected feature is interpretable and facilitates an understanding of how the classifier acts to distinguish the six labeled time-series classes. Each time series is represented compactly as a two-component vector through an extremely simple and computationally inexpensive transformation learned automatically: the first feature quantifies the basic direction of the trend in the time series, and the second measures the linear autocorrelation at lag 1. Because both features are simple to interpret and quick to compute, this classifier could be applied to very long time series straightforwardly (unlike conventional approaches that measure distances between the time-domain objects, cf. Sec. 6.3.2.3). Confu-

[1] In fact, this operation treats the time series as a drive to a particle in a sinusoidal potential and outputs the median values of the particle across its trajectory. But the parameter values for this operation are such that the magnitude of the potential energy landscape is much lower than that of the input drive from the time series—thus the effect of the sinusoidal potential can be neglected and the result is approximately the same as taking a cumulative sum.

[2] In fact, this operation selects one hundred 50-sample time-series segments of the time series, measures the autocorrelation at lag 1 in each of them, and returns the mean value. Because for this dataset, time series are only 60 samples long, the average autocorrelation across 50-sample segments is approximately the same as the autocorrelation of the full time series.

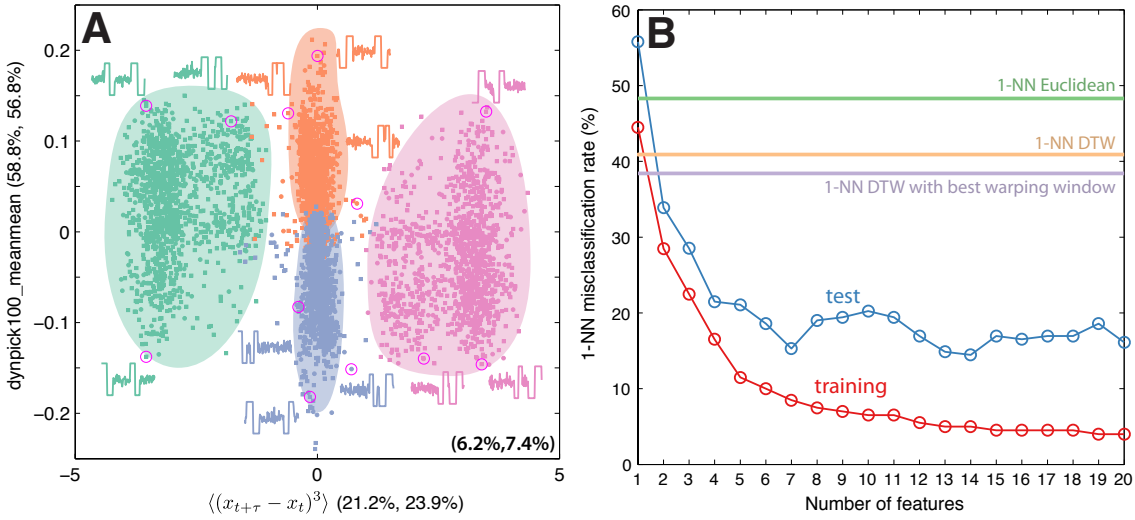

Figure 6.5: **Feature-based classification of the 'Two Patterns' and 'OSU Leaf' datasets. A** A two-feature representation of the 'Two Patterns' dataset. Linear misclassification rates on the training and test sets are given in parentheses for each individual figure, and for their combination, which is annotated in the bottom right corner of the plot. Selected time series are annotated to the plot, as indicated by pink circles, and background shading has been added manually to guide the eye. The 1 000 training data are represented as circles, and the 4 000 test data points as squares. **B** Training and test misclassification rates as a function of the number of features learned using a greedy forward feature selection procedure using a 1-NN classifier for the 'OSU Leaf' dataset. Misclassification rates for time-domain 1-NN classifiers are shown using horizontal lines for comparison.

sion matrices for one- and two-dimensional feature-based classifiers, and for a 1-NN Euclidean time-domain classifier are shown in Sec. C.3.2.

Further examples of our feature-based feature selection process can be found in Sec. C.3.3 and Sec. C.3.4 for the 'Two Patterns' and 'OSU Leaf' datasets, respectively. Two key figures from these analyses are shown in Fig. 6.5. For the 'Two Patterns' dataset, we found that the first selected feature calculates a time-averaged third order quantity that effectively distinguishes the pink and green classes, but not the blue and orange classes. The second selected operation compensates for this shortcoming by separating the blue and orange classes. Together, the classifier has a 7.4% misclassification rate on the test set. For the 'OSU Leaf' dataset, we found that the classification accuracy can be improved by constructing classifiers with more than just two features, as shown in Fig. 6.5B. For this example we used a 1-NN feature-based classifier for the greedy forward feature selection, but similar results were found

when using a linear classifier. The figure shows that the out-of-sample classification accuracy of time-domain 1-NN classifiers is exceeded by the feature-based classifier containing just two features. The performance increases further when more features are added, down to a misclassification rate of just 16% with seven features. Note that in general, we do not have access to the test data to select the number of features to use in a classifier; in this thesis we propose a simple heuristic to make this selection as the point at which the improvement in the in-sample misclassification rate from adding an additional feature drops below 3% (cf. Sec. 6.2.3). For this dataset, that corresponds to a classifier with five features for this dataset with an error of 21.1% (cf. Table 6.1).

6.3.2.2 All results

In this section, classification results for all twenty time-series datasets are presented in Table 6.1. As described in Sec. 6.2.3, two types of classifiers are compared: (i) the conventional approach of calculating distances between time-series objects, labeled with 't : ', and (ii) classifiers are learned using feature selection on sets of features extracted from time series, labeled with 'f : '. As with all classifiers, feature-based classifiers are more successfully applied to some datasets than others [212]. In this case, time-domain classifiers are expected to be applicable to datasets such as those studied in this work, where time series are patterns whose values through time can be meaningfully compared and used as the basis of computing a distance between them. Feature-based classifiers may be able to outperform time-domain-based methods when capturing relevant features from the time series makes the classifiers more robust to noise, for example, or when important features in the time domain are poorly aligned. Indeed, we find that for some datasets, feature-based methods are more accurate than time domain alternatives for some datasets: e.g., for the ECG dataset, all three feature-based methods are able to classify the test set with at least 99% accuracy using just a single extracted feature, whereas the best time-domain classifier has just 88% accuracy. The 'Coffee' dataset is another example where linear classifiers can achieve perfect test-set classification with just a single feature, whereas the best time-domain classifier has a misclassification rate of 82.1%. For other datasets, however, 1-NN

time domain classifiers outperform our feature-based classifiers: e.g., the '50 words' dataset (a 50-class classification problem), the 'Face (all)' dataset and the 'Lightning (seven)' dataset. Overall, however, both types of classification methods give relatively similar results, or modest deviations in favor of one or the other. Interestingly, it is often the case that when time-domain classifiers show poor performance that feature-based classifiers show good performance, and vice-versa. This indicates that feature-based classifiers may be a good complementary method for temporal data mining classification tasks.

Table 6.1: A summary of classification results for the time-series datasets analyzed in this work. For each dataset, we list the results of both temporal classification (labeled 't :'), in which distances are calculated between objects in the time domain, and feature-based classification (labeled 'f :'), in which time series are instead represented as a set of extracted features. All numbers in the table are misclassification rates expressed as percentages, and numbers in parentheses have a meaning as labeled in the corresponding column. Results using dynamic time warping (DTW) were obtained from Keogh et al. [214]. 'Warping window' has been abbreviated as 'WW' in the third column. For the feature-based classifiers, the number of features, n_{feat}, was chosen as the point at which improvement in the training set classification rate from adding another feature drops to less than 3%. The classifier with the lowest misclassification rate for each dataset is printed in boldface.

	t : Euclidean 1-NN	t : DTW 1-NN	t : DTW 1-NN (best WW, r)	f : linear (n_{feat})	f : linear (norm) (n_{feat})	f : 1-NN (n_{feat})
Synthetic control	12.0	**0.7**	1.7 (6)	3.7 (2)	2.0 (2)	2.3 (2)
Gun point	8.7	9.3	8.7 (0)	**7.3** (2)	8.7 (1)	13.3 (1)
CBF	14.8	**0.3**	0.4 (11)	28.9 (2)	4.1 (2)	0.9 (2)
Face (all)	28.6	**19.2**	**19.2** (3)	29.2 (5)	38.0 (6)	35.2 (7)
OSU leaf	48.3	40.9	38.4 (7)	**16.5** (5)	23.6 (5)	21.1 (5)
Swedish leaf	21.1	21.0	15.7 (2)	22.7 (5)	**12.5** (5)	15.0 (6)
50 words	36.9	31.0	**24.2** (6)	45.3 (7)	46.6 (7)	42.7 (5)
Trace	24.0	**0.0**	1.0 (3)	1.0 (1)	**0.0** (1)	**0.0** (1)
Two Patterns	9.3	**0.0**	0.2 (4)	7.4 (2)	5.3 (2)	1.8 (3)
Wafer	0.5	2.0	0.5 (1)	**0.0** (1)	4.7 (1)	0.03 (1)
Face (four)	21.6	17.0	**11.4** (2)	26.1 (3)	54.5 (3)	55.7 (3)
Lightning (two)	24.6	**13.1**	**13.1** (6)	19.7 (2)	19.7 (2)	32.8 (2)
Lightning (seven)	42.5	**27.4**	28.8 (5)	43.8 (4)	41.1 (6)	49.3 (3)
ECG	12.0	23.0	12.0 (0)	1.0 (1)	1.0 (1)	**0.0** (1)
Adiac	38.9	39.6	39.1 (3)	35.5 (5)	**30.2** (6)	**30.2** (7)
Yoga	17.0	16.4	**15.5** (2)	22.6 (3)	22.2 (3)	16.4 (5)
Fish	21.7	16.7	**16.0** (4)	17.1 (6)	18.9 (6)	29.7 (7)
Beef	46.7	50.0	46.7 (0)	43.3 (5)	**36.7** (3)	40.0 (4)
Coffee	25.0	17.9	17.9 (3)	**0.0** (1)	**0.0** (1)	7.1 (1)
Olive oil	13.3	13.3	16.7 (1)	**10.0** (2)	26.7 (2)	33.3 (2)

A summary of the classification performance of various classifiers across these twenty datasets is shown in Fig. 6.6 using box plots. In addition to the classifiers considered in Tab. 6.1 above, we also include feature-based classifiers that use the test set to select an optimal number of features for comparison. These classifiers

are obviously not applicable to real situations, nor are they directly comparable to the other methods that treat the test data as unseen (as they should). However, we include them here to show that, given future improvements in methods to select a suitable number of features from the training set (rather than the threshold of a 3% improvement used here, cf. Sec. 6.2.3), classification rates may be improved significantly. The box plots show that the t : Euclidean 1-NN classifier has the highest median misclassification rate across these twenty datasets. However, the other time-domain classification methods have lower median misclassification rates than the feature-based methods. The difference is not large, however (less than 5%), and the lower-quartile misclassification rates are lower for the feature-based classifiers. The feature-based classifiers that select an optimal number of features must perform better than the other feature-based classifiers, and here they also perform better than the time domain-based classifiers.

These results show that feature-based classifiers are generally competitive with classifiers that use distances between objects in the time domain, even for datasets for which this time-domain representation is well-motivated and sensible. Indeed, for many examples, simple linear feature-based classifiers outperform conventional approaches despite dramatic dimensionality reduction (as low as one). In the following section we show how feature-based classification can also reduce classification times for new time series.

6.3.2.3 Computational complexity

In this section, we compare the computational effort required to classify time series using extracted features to time-domain approaches. First we outline the classification process using a distance measure in the time domain. This basic representation of time series in the time domain encompasses a wide variety of classification approaches, including various dimensionality reduction methods such as piecewise polynomial representations, symbolic representations, and representations using some basis function set [220]. Calculating the pairwise Euclidean distance between two objects has a time complexity of $\mathcal{O}(N)$, where N is the length of each time series (which must be identical). The distance calculation for dynamic time warping (DTW) has a time

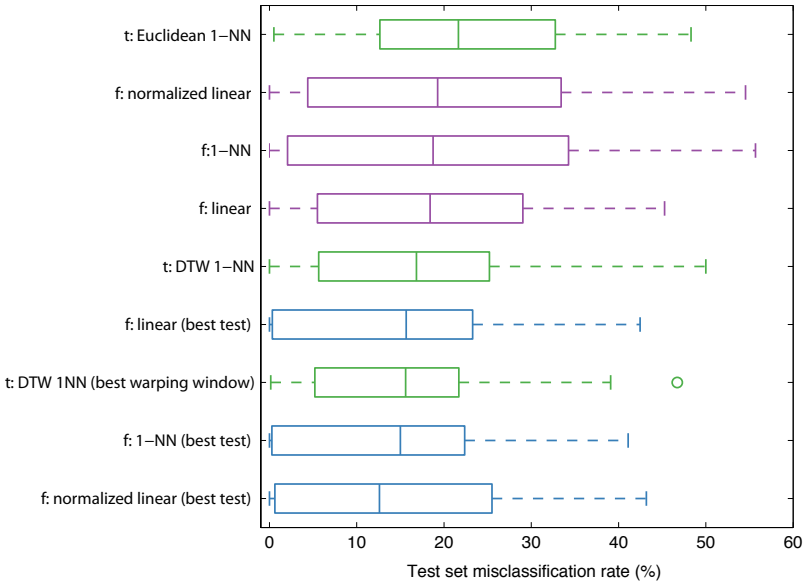

Figure 6.6: **Box plots of test set misclassification rates for different classifiers across the twenty datasets considered in this study**. The three types of methods are grouped using color: (i) 1-NN classifiers using distances between time-series objects (green), (ii) classifiers for an extracted feature space where the number of features is chosen as the point where the training set misclassification rate decreases by less than 3% (purple), and (iii) classifiers for an extracted feature space with a number of features chosen as that which minimizes the test set error (blue). This latter approach, labeled 'best test', is shown for illustration but cannot be directly compared to the other methods for which the test data is left completely unseen. Labels are as follows: 'f:' – a feature-based classifier, 't:' a time-domain classifier, 'normalized linear' a linear classifier trained on the data normalized using the training data by a sigmoidal transform Eq. (3.2), and 'linear' a linear classifier trained on the un-normalized data. The features are ordered by their median misclassification rate.

complexity of of $\mathcal{O}(N^2)$, or, with a warping window it is $\mathcal{O}(Nw)$, where w is the warping window size [203]. Classifying a new time series using a 1-NN classifier and sequential search (i.e., sequentially calculating distances to each time series in the training set) therefore has a time complexity of $\mathcal{O}(Nn_{\text{train}})$ for Euclidean distances and $\mathcal{O}(Nwn_{\text{train}})$ for DTW, where n_{train} is the number of time series in the training set [203]. Note that when using the Euclidean distance metric to retrieve nearest neighbors, the amortized time complexity[3] of the distance calculation can be improved: e.g., for a Euclidean distance calculation, not all N points need to be summed if

[3]Amortized analysis is a way of analyzing sequences of operations in a computation that takes into account the frequency of performing costly operations. For example, computing lower bounds contributes an additional computational cost to the procedure but can lower the amortized cost by reducing the total number of distance calculations that need to be performed.

the current distance is already greater than the nearest neighbor computed so far [71]. The improvement on $\mathcal{O}(Nn_{\text{train}})$ depends on properties of the dataset[4]. We also note that indexing schemes can greatly improve the time complexity of classification [225], e.g., the iSAX method [226, 227]. Although much research has focused on optimizing time-domain-based time-series classification using various dimensionality reduction schemes, lower bounds, and indexing methods, the need to calculate distances between time-series objects means that scaling with the time-series length, N, is inevitable and (at least for sequential search) the time complexity of classifying a new time series scales linearly with the size of the training set, n_{train}. Thus, for long time series or large datasets, time-domain classification can quickly become prohibitive.

In contrast to time-domain classifiers, feature-based classifiers can be evaluated by simply calculating the relevant summary statistics on the time series and then applying the classification rule. This approach was demonstrated in this chapter using a linear classifier, although many other types of classification rules can be learned (cf. Sec. 3.4.3). In this case, the bulk of the computational burden involves learning the classification rule using the training set: thousands of operations are computed on each time series, and then feature selection is used to train a classifier. However, once the classifier has been trained, new time series are classified quickly *and independently* with a computational complexity limited by that of the selected operations. For most cases in this work, successful time-series analysis operations are simple algorithms with a time complexity that scales linearly with the time series length as $\mathcal{O}(N)$. For linear classifiers, the classification of a new time series then involves computing the n_{feat} operations and then evaluating a simple linear classification rule. Hence, for operations with a time complexity that scales as $\mathcal{O}(N)$, the total time complexity of classifying a new time series also scales as $\mathcal{O}(Nn_{\text{feat}})$. This result is independent of the size of the training dataset and, importantly, the classification process does not

[4]For example, the amortized computational cost for techniques such as DTW can also be sped up dramatically on massive datasets by using a linear-time lower bound (such as **LB_Keogh** or **LCSS** [203]), whence the time complexity scales as $\mathcal{O}(pN + (1-p)Nw)$ per distance calculation, where p is the fraction of DTW calculations that can be skipped. The fraction p varies for different datasets, and with their size: e.g., for the 'Two Patterns' dataset with 50 objects, $p = 0.1$ and the speedup is minimal, but with 6400 objects, $p \approx 0.97$ so that now 97% of distances can be computed with the same time complexity as the Euclidean distance metric [203].

require any training data to be loaded into memory, unlike time-domain approaches. Large test sets of time series could thus very easily be classified using our feature-based approach. Since the dimensionality reduction of transforming time series to features can be extreme, we can also accommodate databases of very long time series of variable length, as was demonstrated using case studies for time series as long as 30 000 samples in Chapter 5 of this thesis. Furthermore, because each time series can be classified independently, classification can be performed using distributed or parallel computing straightforwardly.

Having outlined the computational process involved in feature-based classification, we now consider the actual time taken to perform classification using some examples from this chapter. The calculations are performed using MATLAB 2011a[5] on a basic desktop PC. First we show that even though the methods used in this chapter were applied to very short time series (of 60–637 samples), they are also applicable to time series that are orders of magnitude longer. For example we apply the operations shown in Fig. 6.3 to time series of different lengths, as shown in Fig. 6.7. We see that both of these operations have a time complexity that scales approximately linearly with the time-series length, as $\mathcal{O}(N)$. The feature extraction process is evidently applicable to time series that are many orders of magnitude longer than the short time series patterns that distance-based classifiers are restricted to.

Next we analyze the calculations required to classify the 'Wafer' dataset, which involves three stages: (i) calculation of features on the training data, (ii) learning a classifier, (iii) evaluating the classifier on time series in the test set. The calculation of 9 288 operations on a time series in the 'Wafer' dataset (of length 152 samples) took an average of approximately 31 s per time series. Calculating each of the training time series serially scales linearly with n_{train}, and because this is a very large training set with $n_{\text{train}} = 1\,000$, this amounts to a total calculation time of 8.6 hours (although note that the calculation could also be distributed across 1 000 machines for a total calculation time of 31 s, for example). This calculation only needs to be performed once, and the computational cost could be significantly reduced if desired, by using reduced sets of operations rather than the full set of over 9 000, as will be discussed

[5]MATLAB is a product of The MathWorks, Natick, MA.

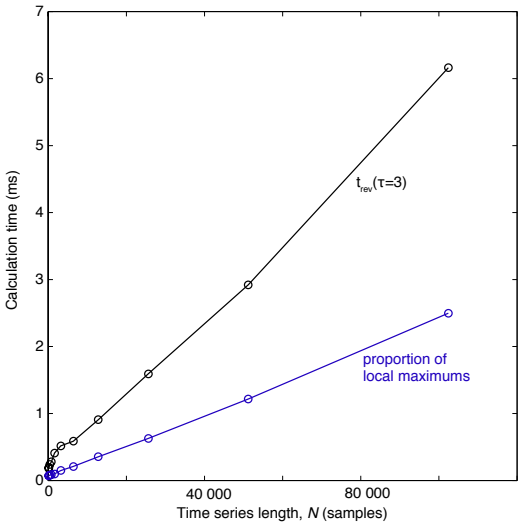

Figure 6.7: **Scaling of two operations with time-series length,** N. These operations are those shown in Fig. 6.3: $t_{\rm rev}(\tau = 3)$ (for the 'Trace' dataset), and the proportion of local maximums (for the 'Wafer' dataset). Both operations scale approximately linearly with the time-series length, N. Calculation times were evaluated on Gaussian-distributed white noise time series and each point represents the average of 200 repeats of the calculation. Unlike time-domain-based classification methods that require the calculation of distances between time series, this feature-based approach is applicable to much longer time series.

in Sec. 6.4. Operations in this set of 9 288 have computational time complexities that scale in different ways with the time-series length, N, but as a total set, the relationship is nonlinear (see Fig. 3.2). Having calculated all operations for the training data, the resulting data matrix is then trimmed of operations with any special-valued outputs, and then a linear classifier is learned using the training time series. The greedy forward feature selection in this case took only 6 seconds because the first selected feature has 0% classification error on the training set. For other datasets, more features are selected according to the normal feature selection process. With the linear classifier trained, we then evaluated it on all 6 164 test time series. It took a total of 32.5 s to load the time series data into memory, 0.1 s to calculate the proportion of local maximums feature for all time series, and 0.2 ms to evaluate the linear classification rule on all time series. The result classified 6 163 of the 6 164, or 99.98%, of the test time series correctly. Hence we find that the bulk of the computational burden of our highly comparative feature-based classification method involves the calculation of features. For a new time series, the actual classification

process itself is then almost instantaneous ($\approx 3\times10^{-8}$ s per time series), and is limited by the loading of data into memory ($\approx 5\times10^{-3}$ s per time series). Note that in general, the calculation times quoted for this 'Wafer' dataset will depend on the time-series length, N, the number of features n_{feat} selected during the feature selection process, and the computational time complexity of those selected features.

Performing feature-based classification in this way is suited to applications that value fast, real-time classification of new time series and can accommodate the relatively lengthy training process (or where sufficient distributed computational power can be used to circumvent this lengthy training process). Time is invested into preparing a training dataset, extracting features from it, performing feature selection, and learning an interpretable classification rule that can then be applied rapidly to classify new samples. This could be useful in industrial applications, where measured time series need to be checked on a production line for quality control, for example. Large quantities of medical samples could also be classified quickly using the feature-based method developed in this work.

6.4 Discussion

In this section we discuss the broader implications of the findings presented in this chapter, and how the highly comparative, feature-based approach to time-series classification could be modified and improved upon in future work.

Other work using feature-based classification has emphasized the ability of the method to accommodate missing values. This was possible because only small sets of basic features, like the trend, skewness, etc. were used, which can be modified to tolerate missing values straightforwardly [210]. However, for many of the operations implemented here, the required modification is not obvious or may not be appropriate, e.g., methods from nonlinear time-series analysis, long-range scaling estimates, and model fits. In future work, sets of operations in our library that can accommodate missing values could be labeled so that, rather than applying the full library of over 9 000 operations, just this subset could be applied to datasets containing time series with missing values. In this way, all the analysis methods and procedures in this work

could be applied without alteration, just using a reduced initial feature set.

To demonstrate ideas clearly throughout this work, we have emphasized the interpretability of feature selection and classification methods over their sophistication. To perform feature selection, we used only greedy forward feature selection; and for classification we used only linear or 1-NN classifiers. As described in Sec. 3.4.5, many more sophisticated feature selection [104–106], and classification [82] methods have been developed. Considering combinations of features that are not necessarily the result of a greedy selection process (e.g., classifiers that combine features with poor individual performance have been shown to be very powerful on some datasets [105]), would improve the classification results reported in this chapter. Similarly, considering a wider variety of classifiers, that allow for more robust, nonlinear classification boundaries, for example, should also improve the classification results reported in this chapter. Our cost function for greedy forward feature selection is the classification rate, even though many classes are unbalanced (i.e., with unequal numbers of time series in each class) and with different class proportions in the training and test sets. The use of the classification rate as our cost function biases the performance of classifiers towards those classes containing the greatest number of time series. In future, more subtle cost functions could be investigated, that optimize the mean classification rate across classes, for example, rather than the total number of correct classifications. However, we use the simple classification rate here because it provides an easily-interpretable indication of performance, is simple to compute, and can be compared straightforwardly to existing results in the literature. In summary, the simple methods used in this work were chosen to produce results that are easy to interpret but more sophisticated methods should be investigated in the future to optimize classification accuracies for real applications.

A key advantage of feature-based classifiers is the ability to interpret the behavior of each selected feature in terms of the properties of the dataset: the feature selection process provides information about what types of time-series properties are informative about the class structure in the dataset automatically. However, as we discovered for the 'Synthetic Control' dataset in Sec. 6.3.2.1 for example, the features can be quite difficult to interpret, especially when they are slight variants of much

simpler features. In this work, we have interpreted the methods manually, but in future a variety of approaches could be explored. We could make use of a list of other features that behave similarly, which could be produced automatically using the similarity search method developed in Sec. 4.1.3. We could also label those operations that are easily interpretable with keyword metadata, which could in future be used to help interpret chosen operations. Alternatively, we could evaluate only a small interpretable subset of operations, guaranteeing the construction of simple and interpretable classifiers.

Since some operations are clearly inappropriate for many of the time series—which are short, non-stationary patterns rather than conventional signals representing long streams of recorded data—it is somewhat surprising that useful clustering results are obtained using the full set of operations, as found in Sec. 6.3.1. For example, estimating the correlation dimension of a time-delay embedded time series requires extremely long and precise recordings of a system [13]. Although the output of a correlation dimension estimate on a 60-sample 'time series' will not be a reliable nor meaningful estimate of the correlation dimension, it is nevertheless the result of an algorithm operating on a time series and as such may still contain some useful information about the time series. It may give low values to highly autocorrelated series and higher values to uncorrelated series, for example. In this way, an unsupervised dimensionality reduction by PCA can indeed isolate meaningful reduced dimensions of covarying behavior, even when a large proportion of the operations may not be behaving in a conventional manner. However, because calculating a large number of complex operations on time series that are very short is computationally intensive, in future a reduced subset of features could be compiled as those that can be applied meaningfully to short time-series patterns. In this way, rather than using over 9 000 features, many of which are slow, complicated, and designed for long time series, the feature calculation and feature selection procedures could be dramatically sped up, and the resulting feature sets would be much simpler to interpret.

It is well known that some classification methods are useful for some datasets, but not for others [71]. However, approaches to choosing which classification method to employ for a given dataset are far from systematic; instead, they typically follow the

intuition of a researcher—using the methods with which they are most familiar. In future work, using our feature-based approach, we may be able to discover characteristic properties of time series that make them more suited to particular classification methods over others. For example, we may find that datasets of highly autocorrelated time series are better suited to dynamic time-warping methods than those with different autocorrelation structures. In this way, rather than searching for methods that work 'best' on generic datasets (which is a poorly-defined problem), or producing vague qualifications about the types of data that a method might be appropriate for (e.g., 'this method works well for financial time series'), the search for useful time-series classifiers could instead be framed around an understanding of the classification process and the characteristics of datasets that suit particular classification methods.

6.5 Conclusions

> **The crucial problem is not the classificator function (linear or nonlinear), but the selection of well-discriminating features. In addition, the features should contribute to an understanding [...].**
> –Timmer et al., 1993 [205]

In this work, we introduced a new method of representing time series as sets of features derived from time-series analysis methods to cluster and classify twenty standard datasets of short time-series patterns. Even though these datasets are well-suited to approaches that measure the similarity of two time series using distances in the time domain, we showed that feature-based representations can be used to produce very competitive classifiers in terms of both accuracy and dimensionality reduction. Using our method, features and classification rules are learned automatically using a large database of time-series analysis operations, without requiring any domain knowledge about how the data were generated. Our approach also has a range of computational benefits, including applicability to datasets of time series of variable length, and including time series that are orders of magnitude longer than those that can be accommodated by time-domain approaches. Furthermore, since each test time series can be classified independently, the classification of new data can be achieved

very quickly using feature-based classification. Perhaps most importantly, the results contribute to an understanding of the key differences between different classes of time series in terms of their measured properties.

Chapter 7

Highly comparative feature-based dimensionality reduction for time-series datasets

In this chapter, we develop a feature-based dimensionality reduction framework for time series. The methods we introduce allow us to infer the important sources of variation in time-series datasets from analyzing a diverse set of measurements of their properties. This chapter will be more exploratory in nature than previous chapters, as it represents a precursor to future work. The method we propose uses dependencies between operations acting on a dataset to deduce the important sources of variation in the dataset. After introducing our method, we demonstrate it on a wide range of synthetic datasets, and show that dimensionality reduction techniques can be used to infer the number of free parameters in the model responsible for generating a dataset, and also to produce interpretable estimators of these parameters.

7.1 Introduction

The problem of dimensionality reduction, which aims to find a low-dimensional representation of high-dimensional data, has been studied in detail and applied widely [82, 83]. High-dimensional datasets are in general large sets of features, where dimensions can be the brightness of pixels in an image, or the results of a set of medical tests on a patient, for example. In each case, dimensionality reduction techniques aim to preserve the structure of a dataset using fewer dimensions than in the original representation. Approaches range from linear methods such as Principal Components

Analysis (PCA) and Multidimensional Scaling (MDS) [82] to, more recently, nonlinear methods such as Isomap [83], Locally-Linear Embedding [228], and Laplacian Eigenmaps [229]. Other methods include Independent Components Analysis (ICA) and non-negative matrix factorization [82].

In this chapter, we introduce a method for deducing informative low-dimensional feature-based representations of time-series datasets. Time series, $\mathbf{x} = (x_1, x_2, ..., x_N)$, are represented naturally in their N-dimensional measurement space; dimensionality reduction in this space can be used to transform the time series object into a lower-dimensional form. This can be achieved by a simple coarse-graining (or symbolization) of the measurements in the time-domain, using local interpolants like piecewise polynomial segments, or representing the time series using some set of basis functions, such as a Discrete Fourier Transform (DFT) or wavelet representation [220]. Other approaches to dimensionality reduction use a time-delay embedding of a single time series to reconstruct a nonlinear attractor, for example, to estimate its fractal dimension [13]. For example, one such technique, known as 'singular system analysis', applies PCA in this time-delay embedding space [131]. Other approaches apply dimensionality reduction techniques in measurement space, taking into account the temporal correlations of successive data points [230, 231]. All of these methods explicitly use the sequence of measurements, $\mathbf{x}$, to represent and find low-dimensional structure in time series.

In contrast to these approaches to time-series dimensionality reduction, which aim to exploit redundancy in the sequential time series object, $\mathbf{x}$, the method we present in this chapter exploits redundancy in a high-dimensional feature space representation of time-series datasets. Dimensions of this feature space represent measurements of many different time-series properties, including summaries of their distribution of values, their autocorrelation properties, stationarity, and entropy. Applying dimensionality-reduction methods in this high-dimensional feature space, we can estimate the 'dimensionality' of a time-series dataset. In this case, dimensionality has a somewhat different meaning: as the number of properties one is required to measure to capture the main sources of variation across the dataset. For example, consider a dataset that consists of sinusoids of different frequencies. Time series could be charac-

terized by any number of properties, including their entropy, distribution, goodness of model fits, etc., but by measuring just the frequency, the important source of variation in this (hence 'one-dimensional') dataset is captured fully. In this chapter, we aim to extract this type of information automatically, by investigating low-dimensional structure in the feature spaces of time-series datasets generated from models with small numbers of free parameters. The dimensions of the feature space are interpretable measures of time-series properties, and hence low-dimensional structure in the feature space should provide important and interpretable information about the system that generated it. Analyzing feature-based representations of time series rather than simply their time-domain form, $\mathbf{x}$, we thereby facilitate a more intuitive understanding of their measurable properties rather than simply their measured values. The representation also allows greater flexibility: in analyzing collections of time series of different lengths, for example.

The chapter is structured as follows. First, in Sec. 7.2.1, we develop a theoretical framework for understanding how dependencies measured between operations (and hence low-dimensional structure in the feature space) can be related to constraints in the number of free parameters in a time-series model responsible for generating the time series. A crucial observation is that pairs of operations can share mutual information via two mechanisms: (i) they are algorithmically similar, or (ii) they are informative of an underlying source of variation in the process responsible for generating the time series. We develop a procedure to distinguish these two mechanisms and thereby infer groups of operations that covary with parameters that control the variation in properties across a time-series dataset. The expected structures in pairwise mutual information matrices for datasets generated by time-series models with different numbers of free parameters are explored with reference to numerical experiments in Sec. D.1.1, building some intuition for the method. Methods used to construct feature spaces, compute mutual information, and form appropriate reduced sets of operations are described in Sec. 7.2.2, and the synthetic time-series datasets studied in this chapter are described in Sec. 7.2.3. Results are presented in 7.3, including observed structures in data matrices and pairwise mutual information matrices measured between operations in Sec. 7.3.1, a procedure for filtering mutual

information matrices to gain information about the underlying parameter(s) in 7.3.2, and the use of dimensionality reduction methods in Sec. 7.3.3. Implications of these results, including limitations and directions for future work are discussed in Sec. 7.4 and the overall findings are summarized in Sec. 7.5.

7.2 Theory, methods, and data

7.2.1 Theory

In this section, a theoretical framework is presented that links low-dimensional structure in high-dimensional time-series feature spaces to free parameters in models responsible for generating the time series.

First we introduce an information theoretical framework for understanding time-series datasets as generated by a time-series model. We suppose that datasets are generated by some hypothetical time-series model, $M(\Theta)$, that is a function of d parameters, $\Theta = \{\theta_1, \theta_2, ..., \theta_d\}$, where each parameter, θ_i, is a random variable. Time series, $\mathcal{X} = (X_1, X_2, ..., X_N)$, are generated by $M(\Theta)$, and are ordered sets of N observations (where N, the length of each time series, is also a random variable in general). In this way, probability distributions defined over parameters, Θ, define a probability distribution over $\mathcal{X}$ via M and therefore determine the types of time series that can be generated by M. A given time series, $\mathbf{x} = (x_1, x_2, ..., x_N)$, is a single realization of $\mathcal{X}$, or, in other words, a draw from the distribution of $\mathcal{X}$ specified by Θ through $M(\Theta)$. In general, the model function, $M(\Theta)$, which maps Θ to $\mathcal{X}$, as $M : \mathbb{R}^d \to \mathbb{R}^N$, can also be random, i.e., for any *given* choice of parameters, Θ, M can specify a probability distribution for $\mathcal{X}$ (rather than specifying a unique time series, $\mathbf{x}$).

Now consider a set of n operations, $\mathcal{F} = \{F_1, F_2, ..., F_n\}$, for extracting features from time series, $\mathbf{x}$. Each such algorithm, $F_i(\mathbf{x})$, maps a time series, $\mathbf{x}$, to a single real number as $F_i : \mathbb{R}^N \to \mathbb{R}$, where the function, F_i, is random in general (i.e., for a given time series, $\mathbf{x}$, F_i can specify a probability distribution of outputs rather than a single, deterministic output). In this chapter, we are interested in the relationship between these extracted features, $\mathcal{F}$, and the free input parameters, Θ, to the time-series

model, M.

One can view this process, that maps parameters to time series, and then maps time series to features as a *Markov chain* between random variables [95], as shown in in Fig. 7.1[1]. In this work, the Markov chain formulation helps us to relate measurements of the properties of time series (as features, $\mathcal{F}$), to the underlying free parameters, Θ, that control variation in properties across the dataset. In particular, since $\mathcal{X}(\Theta)$, and $F_i(\mathcal{X})$, we can treat each feature, $F_i(\Theta)$ as a (random) function of Θ. Throughout this chapter, time-series datasets are treated as realizations of the random variable, $\mathcal{X}(\Theta)$, and hence as samples from a probability distribution defined over Θ. By analyzing the properties of time series in a dataset, we thus learn about the distribution over $\mathcal{X}$ and hence gain an understanding of the properties of the generative model, M, and the parameters, Θ, that control it.

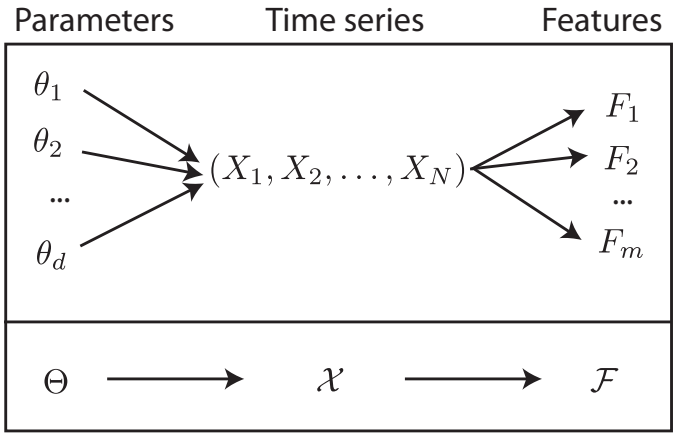

Figure 7.1: **A schematic showing the dependencies between random variables in our model-based formulation of time-series datasets.** A set of d random variables, $\Theta = \{\theta_1, \theta_2, ..., \theta_d\}$, are input parameters to some time-series model, M, that outputs time series, $\mathcal{X}(\Theta)$. A set of operations operate on the time series, $\mathcal{X}$, and generate a set of summary statistics of it: $\mathcal{F} = \{F_1, F_2, ..., F_m\}$. The process forms a Markov chain [95] as $\Theta \rightarrow \mathcal{X} \rightarrow F_i$, for all $F_i \in \mathcal{F}$. In this work, we are interested in the relationships between the features, F_i, extracted from datasets, and how they can inform us about the parameters, Θ, controlling the generative process of a time-series dataset.

[1]An ordered set of random variables, X, Y, and Z form a Markov chain if the conditional distribution of Z depends only on Y and is conditionally independent of X [95]. Since the features, $F_i \in \mathcal{F}$, are each functions of the data, $F_i(\mathcal{X})$, they cannot contain any more information about the parameters, Θ, than what is contained in the data object, $\mathcal{X}$, itself. Thus Markov chains relate the parameters, Θ, time series, $\mathcal{X}$, and resulting features, $\mathcal{F}$, as $\Theta \rightarrow \mathcal{X} \rightarrow F_i$, for all $F_i \in \mathcal{F}$. Viewing the map from model parameters to data objects to extracted statistics as a Markov chain is an established concept in information theory, and leads to related theory including the *Data Processing Inequality* and the idea of a *sufficient statistic* [95].

Having outlined a framework for understanding the generation of time-series datasets from models, we depart for a moment to describe an example to provide some intuition for the method, as shown in Fig. 7.2. The figure illustrates two cases: in the first, shown in Fig. 7.2A, a large number of hypothetical parameters, $\Theta = \{\theta_1, \theta_2, ..., \theta_d\}$, are free to vary in a way that allows a time-series model, M, to produce the range of dynamics observed in this dataset of 1 000 interdisciplinary time series. The resulting feature set, $\mathcal{F}$, extracted from this dataset contains a correspondingly rich diversity of behaviors, as shown in the data matrix plotted in Fig. 7.2A. The second case is shown in Fig. 7.2B, where time series are instead generated from a simple, constrained time-series model:

$$x_t = \sin(2\pi t/30) + n_t, \tag{7.1}$$

where $n_t \sim \mathcal{N}(0, \eta^2)$, $t = 1, 2, ..., N$, and x_t is the time series. Since the length of time series, N, is fixed, time series generated by this (random) time-series model are controlled by the single free parameter, η, which represents the standard deviation of Gaussian-distributed white noise added to periodic signals (with a period of 30 samples). In the formalism developed above, we have a single free parameter $\Theta = \{\eta\}$, and a model, M, given by Eq. (7.1). The parameter, η, is treated as a random variable that controls a probability distribution over time series, $\mathcal{X}$, that can be produced by the model. Defining a distribution over this random variable, η [e.g., taking $\eta \sim U(0, 3)$], and then sampling from it allows us to sample time series from the distribution over $\mathcal{X}$, using Eq. (7.1). Most of the probability mass of the distribution over $\mathcal{X}$ defined by this model lies over time series with a period of 30 samples with additive Gaussian noise; examples of such time series are plotted in Fig. 7.2B. Note that there is a non-zero probability that all possible time series, $\mathbb{R}^N$, can be produced by this model, Eq. (7.1), on account of the random term $n_t \sim \mathcal{N}(0, \eta^2)$. Because of this randomness, it is difficult to detect the low-dimensional structure in the measurement space, but because many features, $F_i \in \mathcal{F}$, vary with the noise amplitude, η (as shown in the data matrix in Fig. 7.2B), this one-dimensional structure is striking in the feature space.

The model described above, i.e., Eq. (7.1), contained a single free parameter, η.

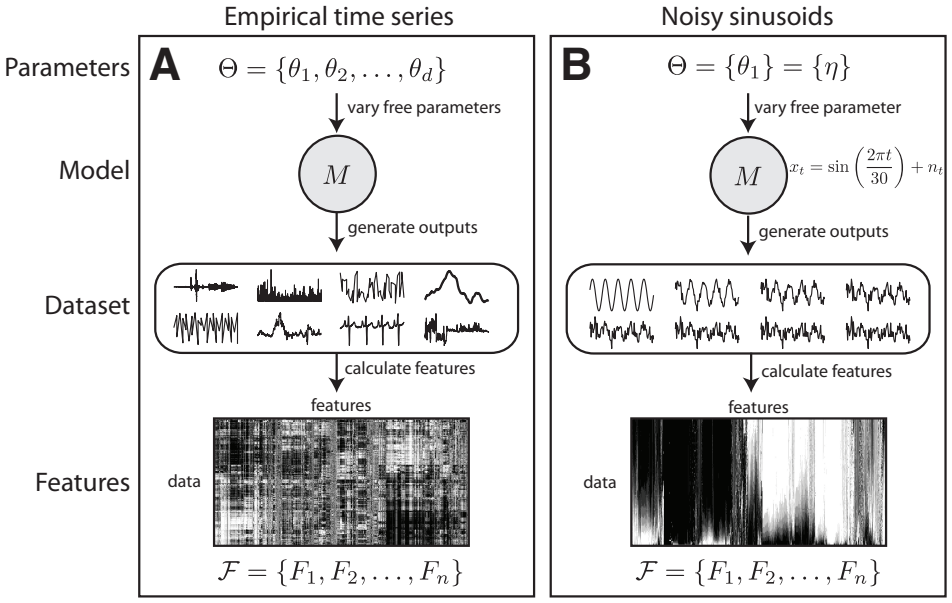

Figure 7.2: **A model-based formulation of time-series datasets.** We consider each time-series dataset as being generated from a time-series model, M, using a set of free parameters, Θ. Measuring time series, $\mathcal{X}$, from a given system then corresponds to sampling this parameter space, Θ. As shown in **A**, a dataset of interdisciplinary empirical time series contains a wide variety of properties and hence requires a large number of features to capture the variation effectively. The data matrix contains rich structure, reflecting this diversity. However, as shown in **B** using the example system of sinusoids with additive Gaussian-distributed white noise, Eq. (7.1), with the single parameter η that defines the standard deviation of additive Gaussian-distributed noise. In this case, most features are informative of η, and hence of each other. Clustered gray-scale color maps show output of the set of features across each dataset as a data matrix.

Because of this, all features can be treated as random functions of η, as $F_i(\eta)$, or, for a general one-parameter system, as $F_i(\theta_1)$. Given a time-series dataset, our goal is to determine which features are informative of this underlying parameter, θ_1. We define a partition of features as follows. Features that share mutual information with θ_1 are grouped as $\mathcal{F}_{\theta_1}$, that is,

$$I(F_i; \theta_1) > 0 \quad \text{for all } F_i \in \mathcal{F}_{\theta_1}. \tag{7.2}$$

All other features will have zero mutual information with θ_1 and are grouped as $\mathcal{F}_0$:

$$I(F_i; \theta_1) = 0 \quad \text{for all } F_i \in \mathcal{F}_0. \tag{7.3}$$

A corresponding partition for a time-series model with two parameters, θ_1 and θ_2, that are independently free to vary, involves four groups, $\mathcal{F}_{\theta_1}, \mathcal{F}_{\theta_2}, \mathcal{F}_{\theta_1,\theta_2}$, and $\mathcal{F}_0$,

defined according to the following conditions:

$$\text{(i) } I(F_i;\theta_1) > 0 \text{ and } I(F_i;\theta_2) = 0 \quad \text{for all } F_i \in \mathcal{F}_{\theta_1}, \tag{7.4}$$

$$\text{(ii) } I(F_i;\theta_2) > 0 \text{ and } I(F_i;\theta_1) = 0 \quad \text{for all } F_i \in \mathcal{F}_{\theta_2}, \tag{7.5}$$

$$\text{(iii) } I(F_i;\theta_1) > 0 \text{ and } I(F_i;\theta_2) > 0 \quad \text{for all } F_i \in \mathcal{F}_{\theta_1,\theta_2}, \tag{7.6}$$

$$\text{(iv) } I(F_i;\theta_1) = 0 \text{ and } I(F_i;\theta_2) = 0 \quad \text{for all } F_i \in \mathcal{F}_0. \tag{7.7}$$

Using arrows to indicate dependencies, these partitions are illustrated graphically in Fig. 7.3 for a single-parameter system (Fig. 7.3A) and for a two-parameter system (Fig. 7.3B). As the number of free parameters increases further, the corresponding number of partitions increases combinatorially. Distinguishing variation due to each θ_i thus becomes increasingly difficult for finite datasets generated from variation in multiple free parameters. In this chapter, we restrict our attention to datasets generated from up to three free parameters; extending the method may be explored in future, as discussed in Sec. 7.4.

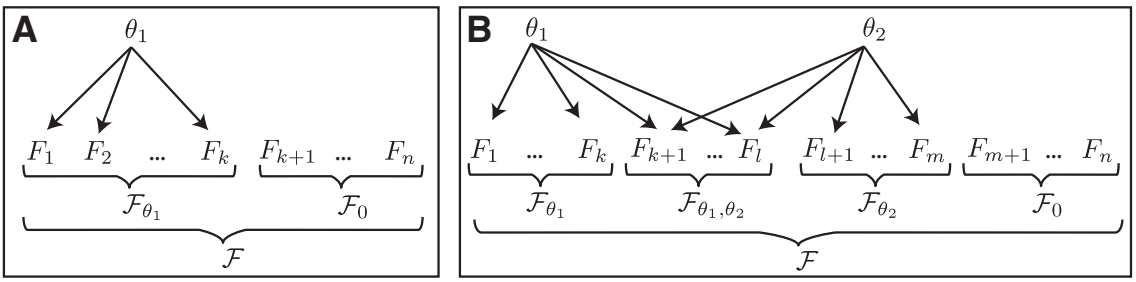

Figure 7.3: **A parameter-based framework for understanding time series allows us to partition features into sets based on their mutual information with parameters.** Arrows are used to connect parameters to the features that they share mutual information with. In **A**, we illustrate a partition of features from a one-dimensional dataset into those that are informative of the underlying parameter, θ_1 as $\mathcal{F}_{\theta_1}$, and those that are not as $\mathcal{F}_0$. In **B**, we consider a two-dimensional model controlled by two free parameters: θ_1 and θ_2. Analogous partitions are made into $\mathcal{F}_{\theta_1}, \mathcal{F}_{\theta_2}, \mathcal{F}_{\theta_1,\theta_2}$, and $\mathcal{F}_0$, as described in the text.

We now consider how such a partition of operations might be made, with access *only* to a set of features, $\mathcal{F}$, extracted from a time-series dataset (i.e., without knowledge of the parameters, Θ, or the generative model, M). First we notice a feature, F_i, can vary across a time-series dataset by three possible mechanisms: (i) M is random, so that a fixed parameter choice, Θ, specifies a distribution over time series

(i.e., the model can generate different time series for a given Θ), (ii) F_i is random, so that a fixed input time series, $\mathbf{x}$, specifies a distribution over F_i (i.e., the feature can produce different outputs for a given input time series, $\mathbf{x}$), and (iii) $I(F_i;\theta_j) > 0$ for some $\theta_j \in \Theta$, that is, F_i is informative of at least one of the parameters controlling variation in the dataset. Central to our task in this chapter is therefore distinguishing the latter, parametric variation in features from variation due to the former effects: randomness in M or F_i. We propose a method that uses, the mutual information measured between pairs of features, $I(F_i;F_j)$, to make this distinction.

Let us first consider the case that the model function, $M(\Theta)$, is non-random (i.e., M defines a deterministic mapping from parameters, Θ, to time series, $\mathcal{X}$). Given this, we can distinguish variation in features due to parameters by noting the three possible cases: (i) non-random features in $\mathcal{F}_0$ will be constant, (ii) random features in $\mathcal{F}_0$ will not share mutual information with any other features: $I(F_i;F_j) = 0$ for all $F_j \in \mathcal{F}$, $(i \neq j)$, and (iii) features, F_i, that depend on one or more of the parameters may share mutual information (i.e., $I(F_i;F_j) > 0$), with other features, F_j, that also depend one or more of the parameters. Thus, when M is non-random, the only mechanism through which $I(F_i;F_j)$ $(i \neq j)$ can be positive is by F_i and F_j both varying with one or more of the underlying free parameters; other features will be constant or have a variation due to randomness in F_i and hence be uninformative of other operations, with $I(F_i;F_j) = 0$ $(i \neq j)$. Therefore, we can infer a partition of features, $F_i \in \mathcal{F}_0$, using the measurable condition: $I(F_i;F_j) \approx 0$ for all $F_j \in \mathcal{F}$ $(i \neq j)$. The remaining features must share mutual information due to a common dependence on one or more of the parameters, Θ.

We have shown how measuring $I(F_i;F_j)$ can distinguish operations that share information due to an underlying parametric variation when the features are allowed to be random but M is not. When M is random, two operations in $\mathcal{F}_0$ *can* share mutual information due to their algorithmic similarity. We demonstrate what is meant by 'algorithmic similarity' using a simple example involving two simple features: the standard deviation and the variance of time series. Because the variance is simply the square of the standard deviation, when both of these features are present in $\mathcal{F}$, these two operations will *always* share mutual information due to their algorithmic

similarity and independent of the properties of the dataset. If the spread of time series does not vary across the dataset, then both operations will yield constant outputs; otherwise they will vary with the variance being the square of the standard deviation. Therefore, when M is random, a crucial challenge involves distinguishing pairs of features that share mutual information due to the variation of some underlying parameter(s) $\theta_i \in \Theta$, and those that share mutual information simply because of trivial similarities in their algorithmic forms. We address this issue in Sec. 7.2.2.3 by first selecting a set of relatively independent operations by applying them to a high-dimensional dataset (i.e., of a large variety of different types of time series with different types of properties), and removing groups of operations that are algorithmically similar using clustering (in a similar way to the reduced set of operations constructed in Sec. 4.1.2). Thus we can assume that $\mathcal{F}$ contains no pairs of operations that are algorithmically similar and hence the only mechanism by which $I(F_i; F_j) > 0$ is when both F_i and F_j are informative of one or more underlying parameters.

As explained above, pinning down operations that are informative of a parametric variation across a dataset involves measuring $I(F_i; F_j)$. We now characterize structures present in square, symmetric mutual information matrices with elements $A_{ij} = I(F_i; F_j)$, calculated between all pairs of operations, $\mathcal{F}$. When very many parameters are allowed to vary independently, all features are independent and there is no redundancy across $\mathcal{F}$. In this case, we find the structure illustrated in Fig. 7.4A, where each feature is informative only of itself, and off-diagonal terms are approximately zero. When only a single parameter, θ_1, is free to vary, we expect to find a group of features, $\mathcal{F}_{\theta_1}$, that share mutual information with each other due to a common dependence on θ_1, but not with those features in $\mathcal{F}_0$. The resulting structure is illustrated in Fig. 7.4B. For a two-parameter system, we expect the structure depicted in Fig. 7.4C, where members of $\mathcal{F}_{\theta_1}$ and members of $\mathcal{F}_{\theta_1,\theta_2}$ share mutual information due to a common dependence on θ_1; and similarly for members of $\mathcal{F}_{\theta_2}$ and $\mathcal{F}_{\theta_1,\theta_2}$ that share a common dependence on θ_2. Note that toy numerical examples exploring how functions of two variables can indeed produce the structures in Fig. 7.4 are in Sec. D.1 of the appendix. Later, in Sec. 7.3, we show how these structures can indeed be observed in feature spaces calculated from finite time-series datasets.

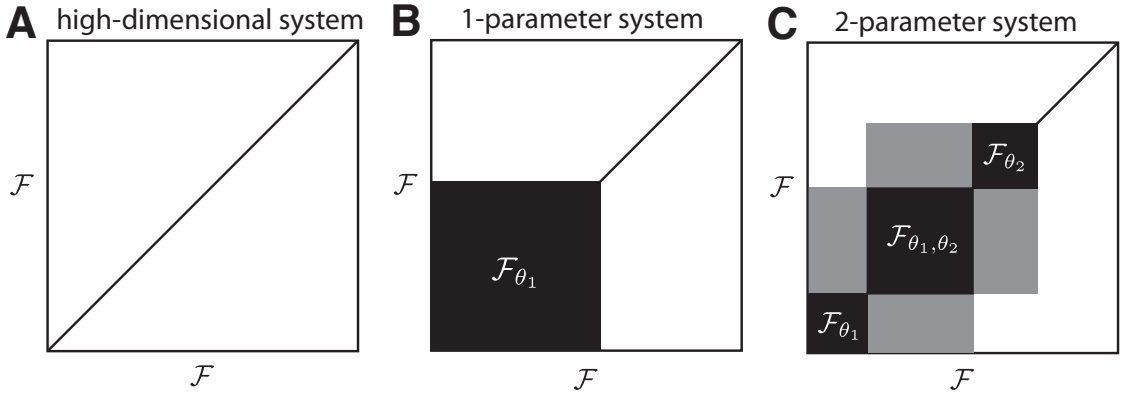

Figure 7.4: **Idealized pairwise mutual information structure between features for one- and two-parameter systems.** Groups of features that vary with the same parameter(s) form correlated blocks (i.e., when similar features are positioned close to one another after clustering them, for example). Elements of these matrices are $A_{ij} = I(F_i; F_j)$, and darker shades are used to indicate higher mutual informations. **A** For a high-dimensional system with many free parameters, features in $\mathcal{F}$ are approximately independent. **B** For a system with a single free parameter, θ_1, those features that vary with this parameter correlate with one another and are grouped as $\mathcal{F}_{\theta_1}$. **B** For a system with two free parameters, θ_1 and θ_2, some features may vary with just θ_1 and not θ_2: $\mathcal{F}_{\theta_1}$, other features may vary with just θ_2 and not θ_1: $\mathcal{F}_{\theta_2}$, and other features vary with both: $\mathcal{F}_{\theta_1,\theta_2}$; the remainder are uncorrelated: $\mathcal{F}_0$.

Note that our method assumes that the set of features, $\mathcal{F}$, is sufficiently diverse to ensure that there exist multiple features $F_i \in \mathcal{F}_{\theta_1}$ with $I(F_i; \theta_1) > 0$ so that mutual information pairs $I(F_i; F_j) > 0$ can be computed. When there is just a single operation, F_1, say, with $I(F_1; \theta_1) > 0$, it is impossible to distinguish its variation from randomness in F_1 or M, in which case all operations are grouped as $\mathcal{F}_0$ and the parametric variation goes undetected. Thus, we assert that parametric variation cannot be inferred simply by analyzing isolated operations; rather, its signature is contained in relationships between algorithmically distinct operations.

7.2.2 Methods

Having introduced a theoretical framework for feature-based dimensionality reduction, in this section we describe and justify the methods used throughout this chapter.

7.2.2.1 The feature-vector representation

In this chapter, we use using the terms 'operation' and 'feature' interchangeably. Normalization of operations is performed as the usual outlier-robust sigmoidal transform, Eq. (3.2). For simplicity, operations that output special values are filtered from data matrices (cf. Sec. 3.2.4). The proportion of operations that output special values is typically less than 10%, but note that this filtering means that some data matrices contain more features than others.

7.2.2.2 Computing mutual information

Crucial to our method is the estimation of mutual information between pairs of features. As described in Sec. 3.4.1, we estimate mutual information using a histogram-based method with equiprobable bins (i.e., with partitions chosen by quantiles of the marginal distributions). Unlike previous sections of this thesis where a fixed number of bins $n_{\text{bins}} = 10$ was used, in this chapter the number of bins is chosen so that each bin contains an average of approximately ten data points: i.e., $n_{\text{bins}} = \text{ceil}\left[\sqrt{N_{\text{data}}/10}\right]$, where N_{data} is the number of data points and the ceil function rounds up to the nearest integer. This alteration allows the mutual information to be estimated more reliably for datasets of different sizes. Note that more complex methods for estimating mutual information, such as k-nearest neighbors and kernel-based methods exist [98], but have not been explored in this work. We do not aim to be sophisticated in this regard, and given the large number of mutual information calculations required to compute pairwise mutual information matrices between hundreds of operations, we value the speed of the histogram-based method over subtle refinements in accuracy that could be explored in future work. Finally we note that when too few data points exist, the mutual information measurements can yield spurious results. To account for this, mutual informations were only calculated between pairs of operations with at least 25 unique data points in the operation-operation space.

Comparing the mutual information measured between different types of random variables, X and Y, directly is inappropriate when the two random variables have different entropies, $H(X)$ and $H(Y)$. Therefore, we compare normalized mutual

informations in this chapter, using the normalization described in Sec. 3.4.1:

$$I_{\text{norm}}(X,Y) = \frac{I(X;Y)}{H(X,Y)} = \frac{I(X;Y)}{H(X) + H(Y) - I(X;Y)}, \tag{7.8}$$

where $0 \leq I_{\text{norm}}(X;Y) \leq 1$ [99]. The denominator, $H(X) + H(Y) - I(X;Y) = H(X,Y)$, is the appropriate quantity to compare, as I_{norm} then quantifies the mutual information $I(X;Y)$ as a proportion of the total, joint entropy $H(X,Y)$. All mutual informations calculated and quoted in this work are normalized, I_{norm}, as Eq. (7.8).

7.2.2.3 A reduced set of operations

Rather than implementing the full library of over 9 000 operations developed for this thesis, we instead use a reduced subset of 500 operations in this chapter. As explained in Sec. 7.2.1 above, the purpose of this reduction is to eliminate groups of algorithmically-similar operations. When the time-series model, M, is random, non-zero mutual information can be measured between two operations simply due to their algorithmic similarity, and not because of their common variation with one or more of the underlying parameters, $\theta_i \in \Theta$. We reduced our library of operations to a reduced set, $\mathcal{F}$, that contains just 500 of these operations. The reduction was achieved using hierarchical average linkage clustering on normalized mutual information distances [i.e., Eq. (3.7)], where each operation is judged according to its behavior across a diverse set of 1 000 empirical time series described in Sec. D.2.1. Rather than k-medoids clustering, which was used to select a reduced set of 200 operations in Sec. 4.1.2, linkage clustering was used in this chapter because our aim is to select operations that are maximally independent, rather than a set that best reproduces the dominant behaviors in the initial set. The result is successful—off-diagonal entries in the pairwise mutual information matrix of $\mathcal{F}$ are approximately zero (on diverse empirical time series): i.e., $I_{\text{norm}}(F_i, F_j) \approx 0$ (for $i \neq j$) with a structure resembling that shown in Fig. 7.4A. Using this reduced set of 500 operations also reduces the computational expense of calculating large pairwise mutual information matrices, which grows quadratically with the number of operations – e.g., with 9 000 operations this amounts to performing 40 495 500 mutual information calculations, but with 500 operations this number is reduced to 124 750. Ideally, the number of operations in

$\mathcal{F}$ should be as large as possible, to encompass as many different types of dynamical behavior as possible, but if too many operations are in $\mathcal{F}$, some pairs of operations will be algorithmically similar. The size of the reduced set of operations used in this section was selected by trialling a range of different sizes of sets, $\mathcal{F}$, produced by linkage clustering. We found that a set of size 500 provided a sufficiently large set, $\mathcal{F}$, with members that are approximately independent: with $I_{\text{norm}}(F_i, F_j) \approx 0$ (for $i \neq j$).

Note that when calculating I_{norm} matrices, we further filter the set of 500 operations to include only those that produced no special-valued outputs (as described in Sec. 7.2.2.1), and at least 25 unique outputs (as described in Sec. 7.2.2.2).

7.2.3 Datasets

A variety of time-series datasets are analyzed in this chapter, and are summarized in this section. Further details of each dataset are in Sec. D.2 of the appendix, including a description of the mechanisms underlying each system, how each dataset was synthesized numerically, and plots illustrating how time series depend on the model parameters.

7.2.3.1 Self-affine time series

A dataset of self-affine time series with a range of scaling exponents, α, was analyzed in detail as a regression task in Sec. 5.1.3. The dataset is analyzed again in this chapter, where we aim to show how most of the variability in the dataset can be understood as the variation in a single summary statistic: the scaling exponent α. More details of this dataset are provided in Sec. D.2.2.

7.2.3.2 Logistic Map time series

Logistic Map time series characterized by their Lyapunov exponents, λ, were analyzed as a regression task in Sec. 5.1.2. In this chapter, we attempt to understand this as a dataset that can be well-summarized by a single parameter that we expect to vary with λ. More details are in Sec D.2.3.

7.2.3.3 Autoregressive time series

Datasets were generated from a simple autoregressive process with an autoregressive coefficient, α, and time lag, τ, according to:

$$x_t = \alpha x_{t-\tau} + n_t, \tag{7.9}$$

where $n_t \sim \mathcal{N}(0, 1)$ are independent random samples from a Gaussian distribution. For a fixed τ, a dataset is generated using uniformly-spaced values of the parameter $\alpha \in [-1, 1]$. Five separate datasets were generated for each of $\tau = 1, 2, 3, 4, 5$. We aim to recover the one-dimensional nature of each dataset, whose variation can be well-summarized by its autocorrelation at lag τ, for example. More details of this model and the time series it produces are in Sec. D.2.4.

7.2.3.4 Correlated, bimodal time series

Correlated, bimodal time series were generated from a random two-parameter time-series model. At each time step, the time series value, x_t, is sampled from either $x_t \sim \mathcal{N}(0, 1)$ or $x_t \sim \mathcal{N}(\delta, 1)$; the parameter δ therefore controls the difference in means between the two states. Correlations are induced between successive time points via an additional parameter, α, that specifies the probability of remaining in the same state at the next time step. When $\alpha = 0.5$, successive states are uncorrelated, when $0.5 < \alpha \leq 1$, successive states are positively correlated, and when $0 \leq \alpha < 0.5$, successive states are negatively correlated. Three datasets are produced by (i) varying δ uniformly across its domain, $0 \leq \delta \leq 6$, while keeping $\alpha = 0.5$ fixed, (ii) varying α uniformly across its domain, $0 \leq \alpha \leq 1$, while keeping $\delta = 3$ fixed, and (iii) allowing $\alpha \sim U(0, 1)$ and $\delta \sim U(0, 6)$ to vary independently. Feature-based dimensionality reduction will be used to extract a measure of autocorrelation that captures the variation in α, and a distributional measure that captures variation in δ. More details of this two-parameter system are provided in Sec. D.2.5.

7.2.3.5 Stochastic sine map time series

Datasets were generated from a stochastic sine map described by Freitas et al. [85], and studied previously in Sec. 4.2.3.2. The model is defined by

$$x_{t+1} = \mu \sin(x_t) + Y_t \eta_t, \tag{7.10}$$

where μ is a parameter, Y_t is a Bernoulli random variable, and $\eta_t \sim U(-b, b)$ are independent random samples from a uniform distribution. The Bernoulli random variable $Y_t = 1$ with probability q, and $Y_t = 0$ otherwise. When $\mu = 2.4$, there are two limit cycles, and the system will attract onto one of them depending on its initial condition; for other values of μ, the dynamics are different, as described in Sec. D.2.6. The parameter b controls the magnitude of random perturbations added to the otherwise deterministic system; such perturbations occur with probability q and can switch the system from one limit cycle to the other. Six datasets of this three-parameter system (i.e., $\Theta = \{\mu, b, q\}$) are produced by varying all combinations of the three parameters: (i) varying each single parameter individually while holding the other parameters fixed, (ii) varying all pairs of parameters while keeping the third parameter fixed, and (iii) varying all three parameters simultaneously. Further details about this system and the datasets generated from it are in Sec. D.2.6.

7.2.3.6 Noisy sine waves with linear trends

Six datasets were generated from a noisy sine wave model with imposed linear trends:

$$x_t = \sin(2\pi t/T) + \eta n_t + mt/N, \tag{7.11}$$

where $n_t \sim \mathcal{N}(0, 1)$, N is the length of the time series, and T, η, and m are parameters: T controls the period of the sine waves, η sets the standard deviation of the white noise process, and m specifies the net displacement of a linear trend. Six datasets are generated from this process, corresponding to allowing all combinations of T, η, and m to vary (while keeping the other parameters fixed). Additional details of datasets generated from this model are in Sec. D.2.7.

7.3 Results

In this section, we present the results of our feature-based dimensionality reduction framework applied to the time-series datasets described in Sec. 7.2.3 above. Our aims are two-fold: (i) to show how the theory developed in Sec. 7.2.1 can be applied to time-series datasets, including the characteristic patterns in mutual information matrices calculated between pairs of operations, and (ii) to demonstrate how important sources of variation in time-series datasets can be extracted by applying dimensionality reduction methods like Isomap to reduced data matrices. The results are divided into three sections. In Sec. 7.3.1, we analyze data matrices and pairwise mutual information matrices for various datasets. In Sec. 7.3.2, we introduce a filtering procedure that enriches the mutual information measured between the remaining set of operations and the parameter of interest, θ_1, thereby demonstrating how features obtained from time-series datasets can be used to understand the mechanisms underlying them. Finally, in Sec. 7.3.3, dimensionality reduction techniques are applied to the feature spaces, $\mathcal{F}$, to estimate the number of free parameters controlling variation in time-series datasets and to deduce interpretable feature-based estimators of those parameters.

7.3.1 Structure in data matrices and pairwise mutual information matrices

In this section, we study the structure in pairwise mutual information matrices, and compare the results to that expected from the theory developed in Sec. 7.2.1. For example, pairwise mutual information matrices are shown in Fig. 7.5 for the correlated, bimodal system (cf. Sec. 7.2.3.4), which has two parameters: α and δ.

Note that the pairwise mutual information matrices are measured between approximately 400 operations (those with sufficiently many unique outputs and no special-valued outputs chosen from an initial set of 500, see Sec. 7.2.2.3) in Figs. 7.5A–C. Since groups of algorithmically similar operations have been removed, the only mechanism through which operations can share mutual information is by varying with one or more of the underlying parameters that control variation across the

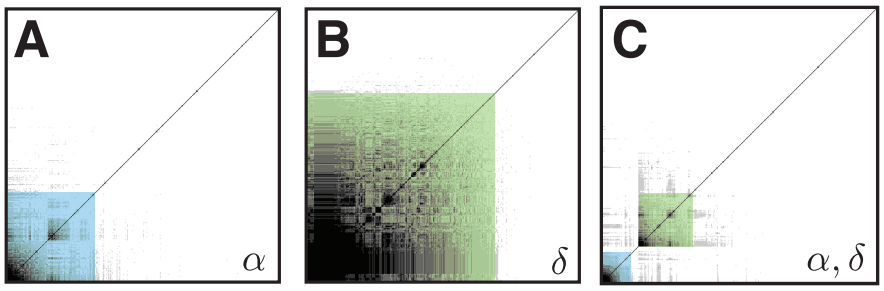

Figure 7.5: **Pairwise mutual information matrices for datasets generated from the correlated bimodal system (Sec. 7.2.3.4).** The correlated bimodal model is controlled by two parameters: the probability of staying in the same state in successive time steps, α, and the separation of the two Gaussian probability distributions, δ. Pairwise mutual information matrices between all features, $\mathcal{F}$, are plotted for **A**: evenly-spaced samples for α in the range $0 \leq \alpha \leq 1$ while $\delta = 3$ is fixed, **B** evenly-spaced samples for δ in the range $0 \leq \delta \leq 6$ with $\alpha = 0.5$ fixed, and **C** independent samples are taken for $\alpha \sim U(0, 1)$ and $\delta \sim U(0, 6)$. Block structures are present for the one-dimensional systems where α and δ are free to vary independently, in **A** and **B**, respectively, and separate into approximately two blocks when α and δ are independently free to vary in **C**. Shading has been added manually to approximately label blocks of operations with $I_{\text{norm}}(F_i; \alpha) > 0.10$ (blue) and those with $I_{\text{norm}}(F_i; \delta) > 0.10$ (green). A grayscale color map is used to represent high values (white) and low values (black) of operations in the data matrices, and low $I_{\text{norm}} = 0$ (white) to high $I_{\text{norm}} > 0.5$ (black).

dataset (or, since this is a finite dataset, mutual information can also be measured between two operations by chance, which we ignore here and note that they could be quantified statistically in future work). In Figs. 7.5A and B, we observe a structure characteristic of one-dimensional datasets: in Fig. 7.5A, the block of operations are informative of α and in Fig. 7.5B, they are informative of δ. The block is larger in Fig. 7.5B because more operations in $\mathcal{F}$ vary with δ than with α. When both α and δ vary independently, two distinct groups of operations are observed, in Fig. 7.5C. The resulting structure resembles that predicted previously in Fig. 7.4. In this case, most operations are predominantly informative of either α (annotated blue in Fig. 7.5), or δ (annotated green); and relatively few operations share mutual information with both α and δ for this system[2].

A pairwise mutual information matrix of features for another two-parameter sys-

[2]Note that this is only an approximate statement: distinguishing operations that vary solely with one parameter and not the other is difficult to verify in practice. Statistical significance thresholds could be determined to quantify this probability in future work, but for now we are satisfied to simply describe the results and note this potential extension that could be explored in future work.

tem is shown in Fig. 7.6 for a dataset of periodic signals with variable period, T, and linear trend displacement, m (cf. Sec. 7.2.3.6). For the purposes of visualization, the matrix has been filtered so as to focus on those operations with prominent changes in mutual information (this was achieved by a simple iterative filtering algorithm explained in Sec. 7.3.2 below, allowing us to 'zoom in' on the structure of the 111 operations with $\langle I_{\text{norm}} \rangle > 0.05$). The two-parameter structure predicted in Fig. 7.4 is reproduced, and the shading in Fig. 7.6 indicates an approximate partition into $\mathcal{F}_m$ (purple), $\mathcal{F}_T$ (green), and $\mathcal{F}_{m,T}$ (orange); operations inferred to be in $\mathcal{F}_0$ have been filtered from Fig. 7.6. As predicted, members of $\mathcal{F}_{m,T}$ share some mutual information with members of $\mathcal{F}_m$ and members of $\mathcal{F}_T$. Note that some of the operations, $F_i \in \mathcal{F}_m$, may have a measured $I(F_i; T) > 0$, but $I(F_i; m) \gg I(F_i; T)$ so that the dependence on m dominates the behavior of F_i, and similarly for some members grouped as $\mathcal{F}_T$; the two cases are difficult to distinguish from finite datasets of finite time series, but making an approximate partition in this way demonstrates our ability to apply the theory of Sec. 7.2.1 to real time-series datasets.

Examples of data matrices and additional pairwise mutual information matrices are in Secs. D.3.1 and D.3.2 of the appendix, respectively.

7.3.2 Filtering operations

We have shown how features extracted from time-series datasets can share mutual information when only a small number of free parameters control variation in the dataset, and how this manifests in the calculated pairwise mutual information matrices measured between operations. In this section, we introduce a simple filtration procedure that allows us to explore the correspondence between operations with high I_{norm} to other operations and their relationship to the parameter $\Theta = \{\theta_1\}$ for one-dimensional systems.

Above, we noticed that operations with increased I_{norm} to other operations form a correlated block in mutual information matrices, and, for one-dimensional systems, are informative of the free parameter, θ_1. To probe this observation further, we introduce a method for removing operations with the lowest mean mutual information to other operations, $\langle I_{\text{norm}} \rangle$, where the mean is calculated across all other operations

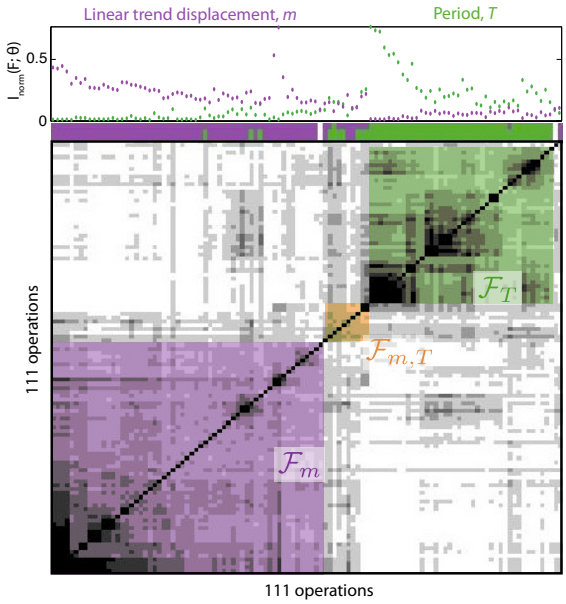

Figure 7.6: **Underlying variation in the period, T, and linear trend displacement, m, of time series can be detected in a pairwise mutual information matrix computed between features extracted from the dataset.** The dataset is described in more detail Sec. 7.2.3.6. A clustered $I_{\text{norm}}(F_i; F_j)$ matrix is plotted between all pairs of a filtered set of 111 operations (those with $\langle I_{\text{norm}} \rangle > 0.05$, see Sec. 7.3.2 below). Black indicates high $I_{\text{norm}} > 0.5$, and white indicates low $I_{\text{norm}} = 0$. Directly above the pairwise mutual information matrix, color is used to label operations that are informative ($I_{\text{norm}} > 0.1$) of the linear trend displacement, m (purple), or the period, T (green); when both conditions are met, both purple and green are annotated. Shading in the matrix is added manually to illustrate these results, whereby blocks of operations are annotated that share mutual information with m ($\mathcal{F}_m$: purple), T ($\mathcal{F}_T$: green), and both m and T ($\mathcal{F}_{m,T}$: orange). In the upper-most plot, I_{norm} is plotted for each operation with each underlying parameter: m and T.

in $\mathcal{F}$. The method is based on the intuition that the remaining operations will be those in the most tightly correlated block and should therefore have increased mutual information with θ_1. The iterative filtering process is as follows: (i) the mean mutual information to all other operations is computed as $\langle I_{\text{norm}} \rangle$ for all operations, (ii) the operation with the lowest $\langle I_{\text{norm}} \rangle$ is removed, (iii) the process is repeated either until no operations remain, or until some specified threshold, $\langle I_{\text{norm}} \rangle_{\text{min}}$, on $\langle I_{\text{norm}} \rangle$ is reached such that all remaining operations have $\langle I_{\text{norm}} \rangle \geq \langle I_{\text{norm}} \rangle_{\text{min}}$. We define this iterative process rather than a simple threshold on the $\langle I_{\text{norm}} \rangle$ computed across the full set of operations because those operations that are filtered are considered not to be informative of θ_1, and hence should not impact the calculation of $\langle I_{\text{norm}} \rangle$ for those remaining operations. However, note that because the operation removed at

each iteration is that with the lowest $\langle I_{\text{norm}} \rangle$, it typically does not impact the relative $\langle I_{\text{norm}} \rangle$ for the other operations appreciably and hence yields broadly similar results.

Applying this method to datasets using a threshold set at $\langle I_{\text{norm}} \rangle_{\text{min}} = 0.3$ returns a set of operations 'enriched' with information about θ_1 (other thresholds produce qualitatively similar results, cf. Sec. D.3.4). In Fig. 7.7, we plot histograms of I_{norm} calculated between θ_1 and this remaining set of operations for the full set of (approximately 400) operations, and for the filtered set of operations (which varies in size for each dataset). Two systems are shown in Fig. 7.7, but similar results were obtained for other one-parameter systems, some of which are shown in Fig. D.13 in the appendix. In all cases, the filtered set of operations has much higher mutual information with θ_1 and thus, by removing those operations with the lowest $\langle I_{\text{norm}} \rangle$ to other operations, we also tend to remove those that are uninformative of θ_1. Analyzing the mutual information between features of time series thereby provides an insight into a common underlying parametric variation across a dataset, and distinguishes it from variability in operations due to randomness in features, F_i, or in the model function, M.

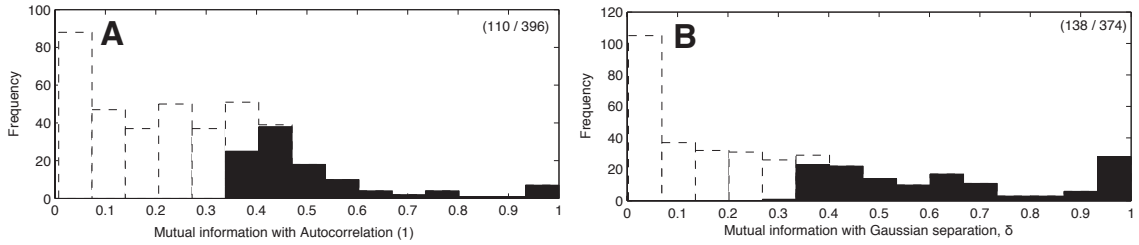

Figure 7.7: **Removing operations with low mean mutual information to other operations also tends to remove those with low mutual information to the free parameter, θ_1.** The effect is shown for two systems in this plot: **A** autoregressive time series with a variable autoregressive parameter, α (Sec. 7.2.3.3) and **B** Correlated, bimodal time series with a variable Gaussian separation, δ (Sec. 7.2.3.4). Histograms with fifteen bins represent the distribution of $I_{\text{norm}}(F_i, \theta_1)$ across all operations (dashed) and after filtering (solid). The iterative filtering process involves removing the operation with the lowest $\langle I_{\text{norm}} \rangle$ at each iteration and is repeated until all remaining operations have $\langle I_{\text{norm}} \rangle \geq 0.3$ (with the remaining operations). Numbers in the upper right of each plot indicate the number of operations retained from an original set of approximately 400.

7.3.3 Dimensionality reduction

So far we have observed the expected structures in pairwise mutual information matrices for real time-series datasets, and demonstrated that correlated blocks of operations correspond to operations that vary with the underlying parameter(s). In this section, dimensionality reduction is used to estimate the number of free parameters controlling variation in a dataset (i.e., its dimensionality in the feature space) and to retrieve estimators of them. Reduced Principal Components projections of datasets have been shown to be informative of structure in various datasets in Chapters 5 and 6. In this chapter, we refine this method in two ways: (i) by using clustering to determine an appropriate reduced set of approximately independent operations with which to apply the dimensionality reduction (cf. Sec. 7.2.2.3), and (ii) by introducing a more sophisticated, nonlinear dimensionality reduction method, Isomap [83]. In light of the theory developed in Sec. 7.2.1, we emphasize the importance of using the reduced set of operations for dimensionality reduction, as correlations between algorithmically similar operations would bias reduced dimensions towards the common variation in these operations, which is informative of the choice of operations in $\mathcal{F}$ rather than being informative of any parametric variation specific to the target dataset. Note that operations are normalized according to Eq. (3.2) prior to performing dimensionality reduction. Results are divided into single-parameter systems, in Sec. 7.3.3.1, and multi-parameter systems, in Sec. 7.3.3.2.

7.3.3.1 Single parameter systems

In this section, Isomap is applied to datasets generated by varying a single parameter, θ_1, in a range of time-series models described in Sec. 7.2.3. As shown in Fig. 7.8, the reduced Isomap dimension (that explains the most variation in the feature space, $\mathcal{F}$) provides an accurate linear estimate of θ_1, for a wide variety of datasets. The technique works well for diverse sources of dynamical variation, including correlation properties, distributional changes, and the period of sinusoidal time series. Surprisingly, the technique also provides a good linear estimate of the control parameter, μ, of the stochastic sine map, Eq. (7.10), as shown in Fig. 7.8F. One might expect this parameter to be difficult to estimate, as its variation controls the dynamical structure

of the sine map system in a nontrivial way (see Sec. 7.2.3.5). The results in Fig. 7.8 thus demonstrate our ability to infer an estimate of the free parameter underlying variation in a variety of time-series datasets, simply by measuring a diverse set of features from the dataset and applying dimensionality reduction in the resulting feature space. Key to the success of the approach is the existence of multiple operations with a sufficiently high mutual information with θ_1—and hence to one another—that can be measured despite randomness in both time-series model, M, and the features, F_i.

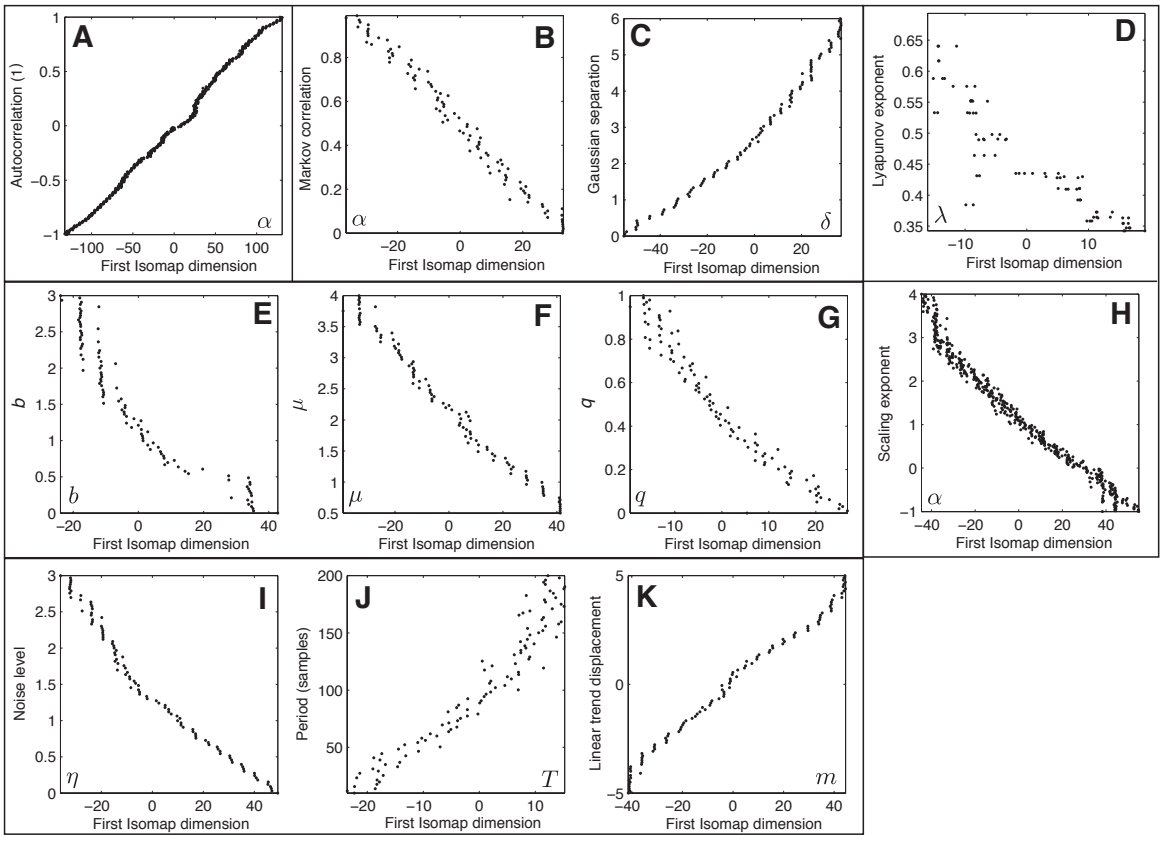

Figure 7.8: **Reduced dimensions produce estimates of free parameters.** The parameter of interest, θ_1, is plotted as a function of the first Isomap dimension in each case. The result is shown for each of the following systems: **A** Autocorrelated time series (only results for $\tau = 1$ are shown, similar results were obtained for $\tau = 2, 3, 4, 5$) (Sec. 7.2.3.3), **B**, **C** Bimodal time series (Sec. 7.2.3.4), **D** Logistic Map time series (Sec. 7.2.3.2), **E**, **F**, **G** Stochastic sine map time series (Sec. 7.2.3.5), **H** Self-affine time series (Sec. 7.2.3.1), **I**, **J**, **K** Noisy periodic time series with added linear trends (Sec. 7.2.3.6). The parameter that is free to vary in each dataset is annotated to a bottom corner in each plot.

The reduced Isomap dimensions analyzed above cannot be interpreted in terms of the time-series properties that they are measuring, as each dimension combines

the outputs from hundreds of different features. Thus, it would be desirable to determine single features that capture a similar variation as the Isomap dimensions. Although sophisticated sparse dimensionality reduction techniques exist, in this work this task is performed in a simpler way: using knowledge of the extracted Isomap dimension to produce a list of features with the highest mutual information to it (as was done for similarity searches in Sec. 4.1.3). Results from this process are shown in Fig. 7.9. For each dataset, we plot an operation with high mutual information to the first Isomap dimension. The results are interpreted in the context of each problem in the following. The Lyapunov exponent, λ, of Logistic Map time series, quantifies the average rate at which predictability in a time series is lost [12], and hence it is appropriate that a prediction error, **FC_prim_median5_meanerr**, was selected as an operation whose variation explains most of the variation in the dataset, as shown in Fig. 7.9A. The separation of two Gaussian distributions, δ, that specify the bimodal distribution for time series can be explained by a distribution-based operation, **olmi_abs_nfexpadjr2**, that analyzes intervals between outliers in a time series, as shown in Fig. 7.9B. The Taken's estimator for the correlation dimension [232], **TSTL_takens_estimator_n1_005_005_1_3**, increases as the time series become more uncorrelated due to an increasing noise probability, q, as shown in Fig. 7.9C. For the system of periodic signals with different standard deviations of added noise, η, a nonlinear autocorrelation-based operation, **AC_nl_023**, is selected that computes $\langle x_i x_{i-2} x_{i-3} \rangle$. Time points become less correlated to their values at two and three time steps in the past as η increases, as we see in Fig. 7.9D; thus, by measuring this autocorrelation-based quantity, this feature is able to estimate the noise standard deviation, η. For the dataset of periodic time series with different periods, T, the selected operation, **NW_visgraphben_horiz_meank**, computes the mean degree of a horizontal visibility graph, as shown in Fig. 7.9E. The horizontal visibility graph represents time series points as vertical bars protruding from a baseline, and the mean degree corresponds to the expected length of a horizontal line extending back and forth horizontally from a point in the time series that terminates when it intersects a vertical bar [28]. Time series with higher periods have longer regions without bars and hence a higher mean degree, as the trend in Fig. 7.9E shows. Finally, we con-

sider a dataset of periodic time series with an imposed increasing or decreasing linear trend, as in Fig. 7.9F. The selected operation, **DV_dynsys_sine_3_05_1_median**, analogous to taking the median of a cumulative sum through time, $S_t = \sum_{i=1}^{t} x_i$, of the (z-scored) time series, x_i. When no linear trend is added ($m = 0$), the cumulative sum will oscillate about zero and will have a resulting mean of zero, but when the trend is increasing ($m > 0$), the cumulative sum will first decrease and then increase, and vice-versa when the trend is decreasing ($m < 0$). In this way, the selected operation captures the variation in linear trend displacement that controls variation across this dataset. Similar results were found for the other one-dimensional systems studied in this chapter. Although interpretation can be difficult, we have demonstrated the automatic selection of operations that capture the main sources of variation in various time-series datasets. If a time-series analyst did not understand the dominant mechanisms underlying a dataset, they could retrieve not just one feature, as shown here, but a set of features that are informative of the reduced dimension, and use that set to understand the types of properties that are influenced by some underlying parametric variability.

Note that these results could not be obtained by analyzing the time series in the measurement space. For example, the addition of white noise to periodic time series, as in Fig. 7.9D, represents an unpredictable perturbation to values in the measurement space. By contrast, this intuitive variation is easily captured by our feature-based approach: we identify the underlying variation in the noise variance and find that a single operation is able to capture this variation across the dataset. The result allows us to provide meaningful explanations of time-series datasets in terms of their properties rather than simply their values.

7.3.3.2 Multiple parameter systems

In this section, we present a systematic and quantitative method for estimating the number of parameters required to explain the variance in time-series datasets using the residual variance measure, V_{resid}, introduced in Sec. 4.1.2, (cf. Eq. (4.1)). The residual variance allows us to quantify how much of the variation in the set of features, $\mathcal{F}$, can be explained by reduced dimensions, which we infer to be estimates of the

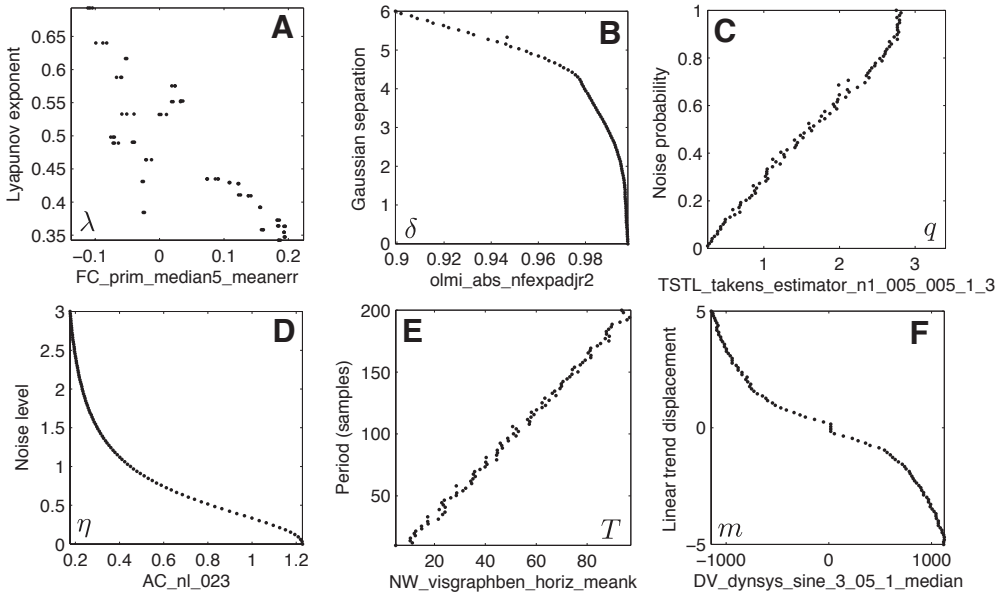

Figure 7.9: **Individual features are retrieved that effectively estimate important free parameters for time-series datasets.** The features were obtained by applying Isomap to the time-series dataset represented in a high-dimensional feature space and selecting an operation with high mutual information with the first Isomap dimension. We plot results for six systems: **A** the Lyapunov exponent, λ, of Logistic Map time series (Sec. 7.2.3.2), **B** the separation, δ, between two Gaussian-distributed states in bimodal time series (Sec. 7.2.3.4), **C** the noise probability, q, in stochastic sine map time series (Sec. 7.2.3.5), and **D** the noise standard deviation, η, **E** period, T, and **F** imposed linear trend, m, of noisy periodic signals (Sec. 7.2.3.6).

parameters, Θ.

Results for Isomap applied to the noisy sinusoidal dataset with imposed linear trends (Sec. 7.2.3.6) are shown in Fig. 7.10. When the standard deviation of additive Gaussian-distributed noise, $\eta \sim U(0, 3)$, is free to vary but the period, $T = 30$ samples, and the linear trend displacement, $m = 0$, are fixed, we obtain the results shown in Fig. 7.10A. In this case, the variation in the features extracted from this dataset can be explained by a single dimension, indicating that the dataset is likely to be generated by the variation of a single parameter, and that the first Isomap dimension is a good measure of this parameter. In Fig. 7.10D, we learn that this first dimension is indeed highly informative of the free parameter η, with $I_{\text{norm}} \approx 0.9$. If the model was unknown, this reduced dimension could be used to retrieve interpretable individual operations with similar behavior (as in Sec. 7.3.3 above).

When a second, independent free parameter is added by taking $m \sim U(-5, 5)$,

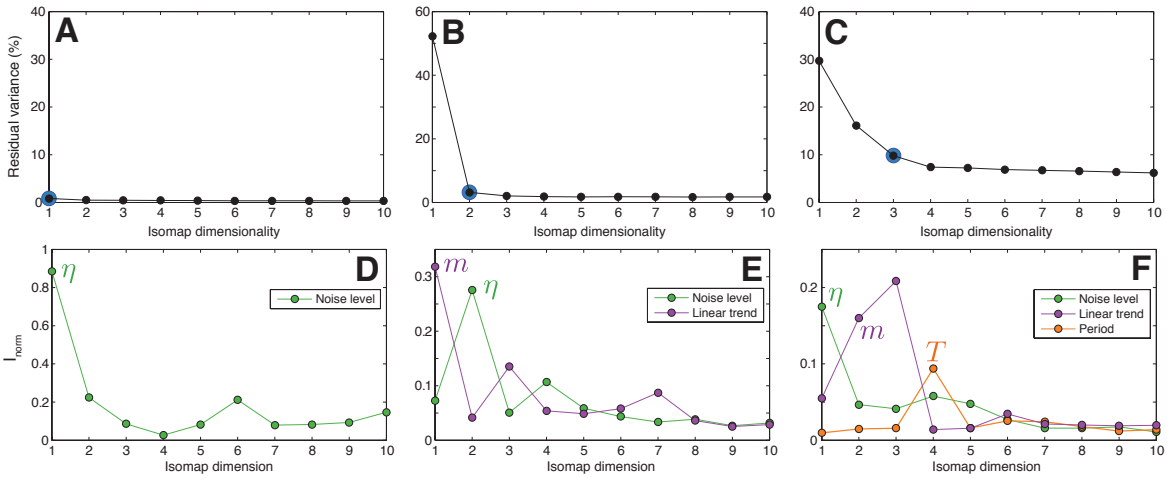

Figure 7.10: **The residual variance can be used to estimate the number of free parameters in a time-series model responsible for generating a time-series dataset.** In this figure we study the noisy, periodic system with added linear trends, Eq. (7.11), with different constraints as follows: (i) only the noise standard deviation, η, is free to vary in **A** and **D**, (ii) both the noise standard deviation, η, and the linear trend displacement, m, are independently free to vary in **B** and **E**, and (iii) and three parameters: η, m, and the period, T, are independently free to vary in **C** and **F**. The residual variance from an Isomap dimensionality reduction is plotted in **A**, **B**, and **C**, and normalized mutual informations I_{norm} between each Isomap dimension and each free parameter are plotted in **D**, **E**, and **F**. Blue circles in Figs. **A**–**C** indicate the number of free parameters used to generate each system.

as in Fig. 7.10B, we see that now two Isomap dimensions are required to explain the variation across this dataset. As shown in Fig. 7.10E, these two dimensions correlate with the linear trend displacement, m ($I_{\text{norm}} \approx 0.3$), and the noise standard deviation, η ($I_{\text{norm}} \approx 0.25$), respectively. We note that I_{norm} between the reduced dimensions and free parameters is reduced when two independent parametric variations are present (Fig. 7.10E), relative to when just a single parameter is free to vary (Fig. 7.10D). This is because it is much more difficult to estimate the standard deviation of additive noise when an unknown linear trend is imposed on a time series; similarly it is much more difficult to estimate the linear trend added to a time series in the presence of an unspecified amount of added noise. To give an example of how this intuition relates to actual operations, consider an operation that measures the standard deviation of a time series. This operation will be informative of η when m is fixed, and it will also be informative of m when η is fixed, but when both vary independently, the standard deviation measure can no longer distinguish

variation due to η from that in m, and as a result exhibits much lower I_{norm} to both. This weakening of I_{norm} between operations and parameters makes it difficult to detect an underlying parametric variation for multi-parameter systems. Despite this, Isomap correctly identifies the two-dimensional nature of this dataset, and retrieves informative estimates of each parameter.

Finally, we add another independent source of parametric variation by taking $T \sim U(10, 100)$. In Fig. 7.10C, we see that now approximately three to four dimensions are required to explain the variation in this dataset. The approach is less successful as for the one- and two-parameter systems, because with three independent sources of parametric variation, each individual variation is much more difficult to identify; a baseline residual variance remains. As shown in Fig. 7.10F, the first Isomap dimension is informative of η ($I_{\text{norm}} \approx 0.2$), the second and third are informative of the linear trend displacement, m ($I_{\text{norm}} \approx 0.15, 0.2$, respectively), and the fourth captures the period, T ($I_{\text{norm}} \approx 0.1$). A clean separation between the three parameters with the three Isomap dimensions is not attained in this case; instead the second and third dimensions appear to behave similarly. This is likely due to the difficulty in estimating the period of time series in the presence of both variable Gaussian noise and an imposed linear trend. Indeed, the linear trend displacement and added noise dominate most operations in this case, making the independent variation in the period difficult to detect. The results may be improved in future by increasing the size of the dataset (in this case 1 000 time series were generated) by increasing length of time series in the dataset (in this case time series were 2 000 samples long), and by improving our algorithm for estimating mutual information.

Space limits our ability to present results for all systems, but we note that all one- and two-parameter systems studied in this chapter (from Logistic Map time series to the correlated bimodal time series) displayed qualitatively similar results to what is shown for the one- and two-dimensional systems in Figs. 7.10. That is, a clear drop in residual variance at the number of free parameters in the generative process (as in Figs. 7.10A and B), and Isomap dimensions that are informative of the free parameters.

In a final example of feature-based dimensionality reduction, we show how the

parameter space representations of time-series datasets can be reproduced using reduced dimensions. To demonstrate this, we used the dataset with the most visually-identifiable relationship between parameters and time series: the noisy periodic dataset, with variable noise standard deviation, η, and sinusoidal period, T. The results are shown in Fig. 7.11 for both Isomap and Principal Components Analysis (PCA). Isomap shows a drop in the residual variance at two dimensions, whereas PCA struggles to identify the two-dimensional nature of this dataset, with the residual variance flattening out at approximately three PCs (as shown in Fig. D.15 in the appendix). Two-dimensional representations of the dataset are explored in Fig. 7.11, and compared to the parameter space used to generate the dataset: $\Theta = \{\eta, T\}$. As shown in Figs. 7.11A and B, the first two Isomap dimensions successfully extract measures of the noise standard deviation, η (with $I_{\text{norm}} = 0.33$), and the sinusoidal period, T (with $I_{\text{norm}} = 0.19$). The corresponding results for PCA are shown in Figs. 7.11C and D. The first PC is a good estimate of η, with $I_{\text{norm}} = 0.51$, but the second PC is a relatively poor estimate of T, with $I_{\text{norm}} = 0.10$. Although the period is relatively easy to measure for periodic signals without added noise, it is much more difficult to estimate in the presence of a variable amount of added noise, i.e., $\eta \sim U(0, 3)$, as is the case for this dataset. Specifically, reliable estimates of the period of time series become more difficult as the noise level increases.

Five hundred samples from the two-dimensional parameter space: $\eta \sim U(0, 3)$ and $T \sim U(10, 100)$, were used to construct this dataset, and are plotted in Fig. 7.11F. The annotated time series in this plot provide a visual representation of the effect of the parameters on the signals: both the standard deviation of added noise, η, and the period, T. As shown in Figs. 7.11A and B, the first and second Isomap dimensions vary monotonically with η and T, respectively, and hence the two-dimensional Isomap projection of the dataset shown in Fig. 7.11E reproduces many aspects of the original parameter space plotted in Fig. 7.11F. The noise level, η, decreases with the first Isomap dimension, and the sinusoidal period, T, increases with the second Isomap dimension. The structure of this space also reflects the influence of η on the estimation of T, noted above: for low η, the signals are well ordered by T, but as η increases, points conglomerate into a single group of noisy time series with high η, where the

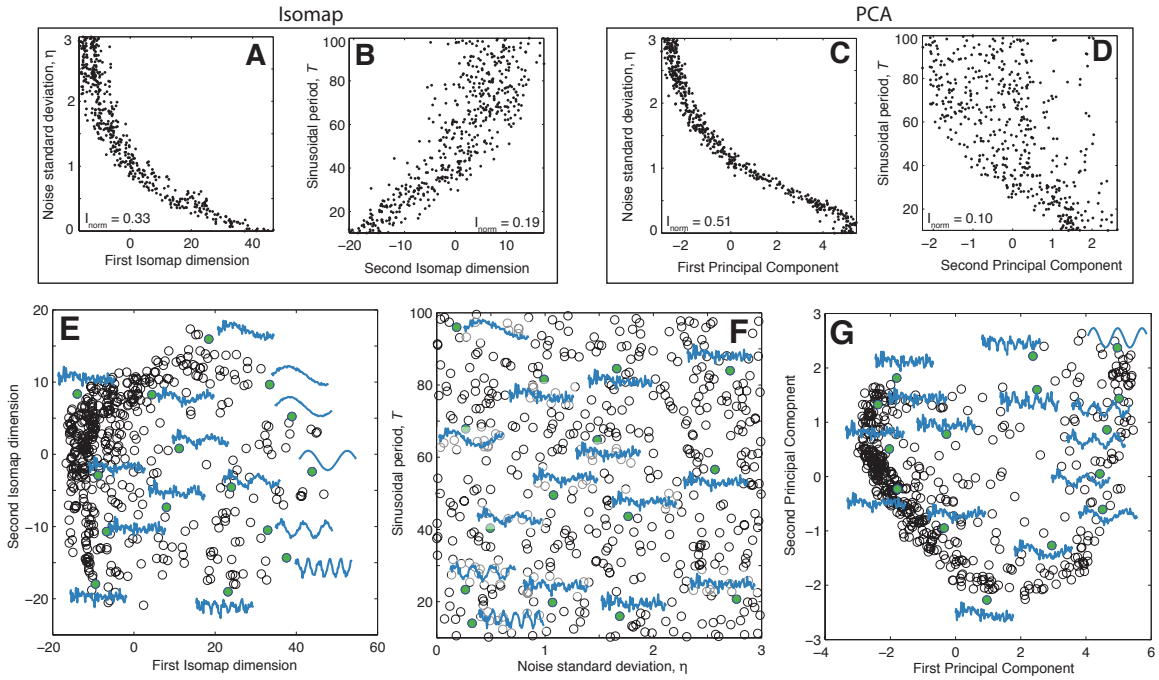

Figure 7.11: **The parameter space of the noisy periodic dataset can be reproduced by applying dimensionality reduction to a set of features extracted from it.** We compare the ability of Isomap and PCA to extract estimators of the standard deviation, η, of added white noise, and the period, T, of the underlying sinusoids. In the upper plots, we show the variation of the first two extracted dimensions with the two parameters η and T, and annotate the normalized mutual information, I_{norm}, calculated between the two variables in each case. In the lower plots, the datasets are projected into the space of the extracted dimensions: **E** the first two Isomap dimensions, **F** the original parameter space used to construct the dataset, **G** the first two PCs. Each of the 500 time series in the dataset are represented by hollow black circles in these plots, and selected data points (green circles) are annotated with an 80-sample segment of the corresponding time series (blue).

time series are so dominated by noise that variation in T has a minimal effect on the measured properties of the time series. By taking only linear combinations of features, the two-dimensional nature of this system is not well captured by PCA, as shown in Fig. 7.11G. In this case, the dataset is represented as being approximately one-dimensional, with most data points lying near a linear manifold in this space, ordered by η, which decreases along the first PC.

In summary, the feature-based dimensionality reduction method introduced in this chapter provides a clear indication of the number of parameters needed to describe the variation in a time-series dataset, and selects interpretable operations that capture this variation. If the sources of variation underlying a system were unknown, these

results could be used to help construct a model for it, simply by analyzing a set of time series generated from the system. For example, if autocorrelation (at a time-lag $\tau = 1$) and stationarity are two properties that explain most of the variation in the dataset, one might proceed to construct a time-series model for the data with parameters that control variation in these two properties. The empirical framework could then be extended to evaluate the model, by comparing the properties of the time series generated from it to the properties measured from the initial dataset. The model could subsequently be refined, or bounds on its parameters could be deduced through this process. Therefore, although model building and evaluation have traditionally been approached in a highly theoretical manner, we have shown that a highly comparative analysis of time-series datasets can be used to motivate and guide this process.

7.4 Discussion

In this chapter, we introduced a theoretical framework that links dependencies between a set of operations acting on a dataset with the variation of free parameters in a time-series model responsible for generating it. The utility of the framework was demonstrated on a range of model systems controlled by a small number of free parameters. In this section we discuss some important consequences of our approach, its limitations, and some future directions.

We first note that the approach introduced in this chapter assumes that the operations, $\mathcal{F}$, used in the analysis is sufficiently rich to capture the underlying sources of variation in each dataset. Further, we require multiple operations to be sensitive to the parametric variations, which we assume will occur for a sufficiently rich set of operations, $\mathcal{F}$. The relatively simple model systems studied in this chapter can be tackled successfully using a relatively small generic set of just 500 time-series analysis operations. Exploiting specific knowledge about a dataset to select a set of suitable operations could be explored in future work.

Although reduced dimensions of the feature spaces studied in this chapter corresponded to the actual parameters of the time-series model that generated it, we

note that there are often multiple ways of defining a model that have similar distributions over time series, $\mathcal{X}$. To provide a most basic example, consider the model $x_t = t + \alpha + \beta$ with parameters $\Theta = \{\alpha, \beta\}$. Even when α and β vary independently, they can always be grouped as a single random variable, $\gamma \equiv \alpha + \beta$, and indeed it would be estimates of γ that would be returned from our method. One can also imagine more complex ways of defining models in terms of different parameters that nevertheless define similar distributions over time series. Noting this, we therefore understand the methods developed in this chapter as identifying reduced sources of variation that could in principle be linked to free parameters in some time-series model. For example, we could find that the variation in a time-series dataset generated by a model containing ten free parameters may be captured effectively by three reduced dimensions. In this case, we might deduce that many of the ten parameters in the original model are redundant (ignoring for the sake of argument the possibility that they are instead controlling subtle aspects of the time series that cannot be detected by operations in $\mathcal{F}$), and a model with just three parameters may be able to capture the observed variation in the dataset. In future work, this framework could be developed to the point that it might be applicable to higher-dimensional systems, or even perhaps real-world datasets. The modeling of real-world systems could thus be aided by analyzing sets of empirical time series in this way.

This chapter represents early work into the problem of feature-based dimensionality reduction for time-series datasets, and there are many straightforward improvements that could be explored in the future. For example, we used Isomap for dimensionality reduction and a simple histogram-based method for calculating mutual informations. These two nonlinear methods were chosen because we observed strong nonlinearity in the dependencies between operations, and between operations and free parameters. For cases in which the parameter variation is straightforward, Principal Component Analysis (PCA) could be used instead of Isomap, and simple linear correlation coefficients calculated between normalized operations [using the outlier-robust sigmoidal transformation, Eq. (3.2)] could be used instead of mutual informations. Both of these options have been explored (e.g., a PC representation is shown in Fig. 7.11), but we found that the presence of nonlinear relationships between opera-

tions can lead to spuriously high dimension estimates, [e.g., two operations can have $I_{\text{norm}}(F_i; F_j) = 0$ when the relationship between them is nonlinear and relatively poor estimates of the free parameters]. In the future, we will investigate sparse dimensionality reduction methods, other types of nonlinear dimensionality reduction methods, and faster and more accurate algorithms for estimating mutual informations.

Finally, we note a connection between this work and Bayesian model selection, which uses similar ideas of distributions across the space of data to compare competing models [9]. As explained above, we consider models as specifying a distribution over time series, $\mathcal{X}$, and hence over $\mathbb{R}^N$ (when N is fixed). When no prior is specified, the most parsimonious model will maximize the probability of the data, $\mathbf{x}$, given a model, M, as $p(\mathbf{x}|M)$ [9]. In future, rather than comparing the distributions over $\mathcal{X}$ specified by different time-series models, we now have a means of representing a given dataset in an appropriate low-dimensional feature space, which informs the most important types of properties that are varying across the dataset—information that can be used to help build an appropriate time-series model. Analyzing time series in terms of their properties is much more intuitive than studying them as time-domain objects, as has been focused on in the past. The method could also be used with Approximate Bayesian Computation (ABC) schemes [233], whereby complex time-series models could be fit to time series using features selected by our method.

7.5 Conclusions

In this chapter, feature-based dimensionality reduction was used to deduce important sources of variation in the properties of time-series datasets generated from constrained time-series models. A theoretical framework that links the structure of pairwise mutual information matrices of features to the parametric freedoms in a model responsible for generating it. The method was applied to datasets generated from various time-series models. For one-dimensional datasets, the structure in the pairwise mutual information matrices of features was interpreted in terms of the previously developed theory, and important sources of variation in these datasets were estimated using Isomap. We also demonstrated how dimensionality reduction

in appropriately-constructed feature spaces could be used to infer the number of free parameters governing variation in a broad range of one- and two-parameter systems. In addition, our method can yield interpretable operations that estimate these parameters.

Automatic, feature-based dimensionality reduction for time-series datasets requires a large and diverse set of time-series analysis operations, which we have assembled. Since such a set of features has never before been available, the results of this chapter therefore represent important first steps in this research direction. In the future, the basic theoretical framework and methodologies developed in this chapter could form the basis of a very powerful way of linking a theoretical understanding of time-series models to the empirical practice of collecting and analyzing time series.

Chapter 8

Future directions and conclusions

In this thesis, a highly comparative framework for time-series analysis was developed and its broad scientific utility demonstrated. Constructing this framework represents a first step towards unifying a vast and fragmented interdisciplinary literature on time-series analysis. As we have shown, much can be learned through a systematic comparison of new methods to those developed in the past and in other disciplines. In the absence of a strong theory to guide the development of methods for time-series analysis, our framework therefore provides an empirical means of evaluating and directing progress in the field. In this brief overview and discussion of the thesis, we focus on three key themes: applications of highly comparative analysis to other data objects in Sec. 8.1, how to approach the interpretation of the results of automated analysis in Sec. 8.2, and the immediate and future relevance of this work to the existing field of time-series analysis in Sec. 8.3.

8.1 Highly comparative data analysis for science

The results presented in this thesis have implications for the analysis of other scientific data objects for which a large number of analysis methods have been developed. For example, the machine learning and data mining literatures contain a large number of classification methods with minimal comparison to alternatives, and cherry-picking of datasets for which the proposed method is successful [71]. In this case, a highly comparative framework for classification methods could be used to group classification methods that behave in similar ways, thereby providing an empirical structure to a collection of apparently disparate methods. As noted in the introductory Chapter 1

of this thesis, this structure is often provided by a common, underlying theory, but in its absence, our approach can be considered to contribute an empirical proxy for this theory that links methods according to their empirical behavior on real datasets. For the example of classification methods, an analogous comparative framework could be developed in which data objects are now labeled classification datasets (rather than time series), operations are now classification methods (rather than time-series analysis methods), and the output statistic could be the 10-fold cross validation classification rate, for example. Patterns in the resulting data matrix would reveal those methods are particularly suited to particular data types, and which pairs of classifiers are complementary (i.e., one classifier performs well when the other performs poorly, and vice-versa), for example . The framework is systematic, transparent, and highly comparative, and explicitly addresses some of the acknowledged problems in the field. Also, unlike the data matrices studied in this work, the classification rate statistic provides an objective measure of the usefulness of a given method on different tasks; in contrast, the outputs of time-series analysis operations simply represent an output of an algorithm, not any measure of quality. However, networks of classification methods could still be constructed by comparing the *behavior* of each method (i.e., rather than the quality of their performance), as was done for operations in this work in Sec. 4.1, by defining distances between classification methods as the mean proportion of objects that were classified in the same way, for example (with the mean taken across datasets). Organizing classifiers in this way would thereby reveal which types of classifiers behave similarly to one another, and the results could be used to subsequently create a unified, reduced set of empirically different types of classifiers.

Another area of data analysis that would benefit from the development of a highly comparative framework is network analysis. Statistics for network analysis are numerous, including summaries of the degree distribution, network centrality, the clustering coefficient, assortativity, frequency of motifs, and so on [234], and although there is often a strong graph theoretical underpinning to the analysis methods, there is no guiding theoretical framework for understanding how all of these diverse types of metrics relate to one another in practical applications, or when it is appropriate to use any particular metric. The situation is a familiar one and we propose that a highly

comparative empirical analysis of network analysis methods would help to structure and guide progress in the field. This task has been attempted on a small scale previously [235], and a more comprehensive treatment analogous to the work presented in this thesis has very recently been completed by a colleague [236].

Finally, we note a possible extension of our work to multivariate time series. In many scientific applications multiple time series are measured simultaneously, as multivariate time series. A highly comparative framework for multivariate time-series analysis could be constructed straightforwardly: by applying the full library of time-series analysis methods to each of the L time series that make up the multivariate time series. In addition to extracting features from each individual time series, correlations between time series in a multivariate recording could also be measured and added as additional features (as cross-correlation coefficients, for example). In this way, the methods developed throughout this thesis could be applied to multivariate data. For example, instead of measuring just the fetal heart rate during labor to predict low cord pH in delivered babies, as in Sec. 5.3, one could take advantage of other physiological recordings measured at the same time, such as the time series of uterine activity (a part of the cardiotocogram) [190] whence features from both fetal heart rate and uterine activity could be combined in resulting classifiers. The main challenge of this approach is addressing the computational cost of evaluating such a large number of features for each multivariate time series, and the resulting ramifications for multiple hypothesis testing, cf. Sec. 3.4.6, for which reduced sets of operations may need to used.

In addition to highly comparative frameworks described above for classifiers, network analysis, and multivariate time-series analysis, there are many other future targets for what is a general highly comparative methodology introduced in this thesis, including other types of machine learning methods (e.g., clustering and regression methods), symbolic string analysis (e.g., musical sequences, weather types, texts), and image analysis. We believe that the time invested by the research community in developing comparative tools such as these will allow new methods to be compared to all others and evaluated immediately, providing an alternative guiding structure to the 'art' of data analysis.

8.2 Automated science and interpretation

In this thesis, the interpretation of operations selected for given classification or regression tasks was left as a separate, more subjective analysis step requiring insight into the dataset, the system that generated it, and its measurement. In this context, it is important to recognize that the data matrix simply represents the results of a set of computations and that any *meaning* of the result must be deduced by a data analyst through further analysis. For example, if our method returns a list of twenty operations that effectively distinguish heart beat interval series measured from patients with congestive heart failure from those of healthy controls, just like with the results obtained from a 'robot scientist' [77], it is up to the time-series analyst to interpret the behavior of the algorithms. The (human) scientist must assess whether the operations: (a) are informative of the generative processes underlying the system, (b) are useful practical measures that summarize sources of difference between the two types of recordings, or (c) might be picking up structure in the target dataset that is peculiar to this dataset and the way it was measured and hence may not generalize to out-of-sample data.

Although the interpretation of results currently requires the expertise of a time-series analyst, in future this step could be guided by empirical tests that measure the quality of an operation's performance on different types of analysis tasks. For example, a new type of data matrix could be constructed in which rows correspond to *tests* on operations, such as a measure of their length-dependence, their robustness to noise, their dependence on the scaling exponent of self-affine time series, their ability to distinguish chaotic time series from random number sequences, their accuracy in predicting the Lyapunov exponent or correlation dimension of time series, and so on. Each of these tests therefore summarizes the performance of an operation on a carefully-constructed time-series dataset. Thus, rather than simply comparing the *behavior* of a new operation to existing operations, we can obtain a set of interpretable diagnostics from a range of synthetic time-series tests which provide information about its useful performance on a range of different time-series analysis tasks.

8.3 Relevance of highly comparative time-series analysis

Looking forward, we hope that the highly comparative framework for time-series analysis developed in this thesis will be developed further using input from the time-series analysis community. To this end, all of the resources developed in this work will be made publicly-available on the internet (excepting some time-series datasets that have been obtained on the condition that they are not distributed). In this way, new time-series analysis methods, whether they be in a physics journal and applied to astrophysical data, or in a biomedical engineering journal and applied to heart beat data, can be compared straightforwardly to methods from across the interdisciplinary time-series analysis literature. Rather than accepting papers that propose a new general method for time-series analysis using only a handful of example datasets and comparisons to just a couple of alternative methods, our comparative framework makes it easy to apply a new method to a large number of different time series, and to compare its behavior to a large collection of alternative methods. If adopted by journal editors, such comparisons could become required of authors that develop new methods for time-series analysis. Ideally, such a resource would be maintained online, with new methods and datasets submitted to it so that the resource can grow into rich repositories of methods coded by experts, and data that has been measured and processed by experts. Perhaps such an enterprise will be recognized as being sufficiently useful to gain the required funding to set-up and maintain in the future; in the meantime, the current set of methods and data are available to the time-series analysis community online and we await their reception.

In conclusion, we have developed a highly comparative framework for time-series analysis and demonstrated its usefulness on a wide variety of problems using both synthetically-generated and real-world datasets. We have shown how a highly comparative approach can be used to reduce redundancy in large collections of time-series analysis operations (Sec. 4.1.2), highlight relationships between different types of operations (Sec. 4.1.3), structure time-series datasets meaningfully (Sec. 4.2), link empirical time-series data with data generated from relevant models (Sec. 4.2.3), and

select useful operations for diverse regression (Sec. 5.1) and classification (Sec. 5.2) tasks automatically. We also introduced feature-based representations of time series, that draw on methods developed across a wide literature on time-series analysis, and showed how they can be successfully applied to problems in temporal data mining (Chapter 6) and dimensionality reduction (Chapter 7): areas in which time series are typically represented and analyzed in their time-domain form. We believe that the highly comparative approach developed in this thesis can be used to direct progress in time-series analysis, and that our results represent first steps in this direction.

References

[1] T. Bollerslev. Generalized autoregressive conditional heteroskedasticity. *J. Econometrics* **31**, 307 (1986).

[2] S. M. Pincus, I. M. Gladstone, and R. A. Ehrenkranz. A regularity statistic for medical data analysis. *J. Clin. Monitor Comp.* **7**, 335 (1991).

[3] C. K. Peng, S. V. Buldyrev, A. L. Goldberger, et al. Statistical properties of DNA sequences. *Physica A* **221**, 180 (1995).

[4] C. K. Peng, S. V. Buldyrev, A. L. Goldberger, et al. Long-range correlations in nucleotide sequences. *Nature* **356**, 168 (1992).

[5] C.-S. Poon and M. Barahona. Titration of chaos with added noise. *P. Natl. Acad. Sci. USA* **98**, 7107 (2001).

[6] L. Lacasa, B. Luque, F. Ballesteros, J. Luque, and J. C. Nuño. From time series to complex networks: The visibility graph. *P. Natl. Acad. Sci. USA* **105**, 4972 (2008).

[7] X. Xu, J. Zhang, and M. Small. Superfamily phenomena and motifs of networks induced from time series. *P. Natl. Acad. Sci. USA* **105**, 19601 (2008).

[8] Y. Dwivedi and S. Subba Rao. A test for second-order stationarity of a time series based on the discrete fourier transform. *J. Time Ser. Anal.* (2010).

[9] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer New York: (2006).

[10] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. MIT Press (2005).

[11] J. P. Crutchfield and D. P. Feldman. Regularities unseen, randomness observed: Levels of entropy convergence. *Chaos* **13**, 25 (2003).

[12] J. C. Sprott. *Chaos and Time-Series Analysis*. Oxford University Press, New York (2003).

[13] H. Kantz and T. Schreiber. *Nonlinear Time Series Analysis*. Cambridge University Press, Cambridge, 2nd edition (2004).

[14] J. Zhang and M. Small. Complex network from pseudoperiodic time series: Topology versus dynamics. *Phys. Rev. Lett.* **96**, 238701 (2006).

[15] Y. Yang and H. Yang. Complex network-based time series analysis. *Physica A* **387**, 1381 (2008).

[16] K. Zhang and W. Fan. Forecasting skewed biased stochastic ozone days: analyses, solutions and beyond. *Knowl. Inf. Syst.* **14**, 299 (2008).

[17] A. H. Shirazi, G. R. Jafari, J. Davoudi, et al. Mapping stochastic processes onto complex networks. *J. Stat. Mech.-Theory E.* **2009**, P07046 (2009).

[18] J. Gao, Z.-y. Xu, and L.-t. Zhang. Approximating long-memory DNA sequences by short-memory process. *Physica A* **388**, 3475 (2009).

[19] M. Small, J. Zhang, and X. Xu. Transforming time series into complex networks (2009).

[20] R. V. Donner, Y. Zou, J. F. Donges, N. Marwan, and J. Kurths. Ambiguities in recurrence-based complex network representations of time series. *Phys. Rev. E* **81**, 015101(R) (2010).

[21] A. S. L. O. Campanharo, M. I. Sirer, R. D. Malmgren, F. M. Ramos, and L. A. N. Amaral. Duality between time series and networks. *PLoS One* (2011).

[22] R. V. Donner, J. Heitzig, J. F. Donges, et al. The geometry of chaotic dynamics – a complex network perspective. *Eur. Phys. J. B* pp. 1–20–20 (2011).

[23] E. Tejera, A. I. Rodrigues, M. J. Areias, I. Rebelo, and J. M. Nieto-Villar. Network centrality and multiscale transition asymmetry in the heart rate variability analysis of normal and preeclamptic pregnancies. *Commun. Nonlinear Sci.* **16**, 1589 (2011).

[24] J. B. Elsner, T. H. Jagger, and E. A. Fogarty. Visibility network of United States hurricanes. *Geophys. Res. Lett.* **36** (2009).

[25] Y. Yang, J. Wang, H. Yang, and J. Mang. Visibility graph approach to exchange rate series. *Physica A* **388**, 4431 (2009).

[26] L. Lacasa, B. Luque, J. Luque, and J. C. Nuno. The visibility graph: A new method for estimating the Hurst exponent of fractional Brownian motion. *EPL-Europhys. Lett.* **86** (2009).

[27] X.-H. Ni, Z.-Q. Jiang, and W.-X. Zhou. Degree distributions of the visibility graphs mapped from fractional Brownian motions and multifractal random walks. *Phys. Lett. A* **373**, 3822 (2009).

[28] B. Luque, L. Lacasa, F. Ballesteros, and J. Luque. Horizontal visibility graphs: Exact results for random time series. *Phys. Rev. E* **80**, 046103 (2009).

[29] S. M. Pincus. Approximate entropy as a measure of system complexity. *P. Natl. Acad. Sci. USA* **88**, 2297 (1991).

[30] P. Grassberger and I. Procaccia. Estimation of the Kolmogorov entropy from a chaotic signal. *Phys. Rev. A* **28**, 2591 (1983).

[31] S. M. Pincus. Approximate entropy (ApEn) as a complexity measure. *Chaos* **5**, 110 (1995).

[32] L. Diambra, J. C. B. de Figueiredo, and C. P. Malta. Epileptic activity recognition in EEG recording. *Physica A* **273**, 495 (1999).

[33] J. W. Sleigh and J. Donovan. Comparison of bispectral index, 95% spectral edge frequency and approximate entropy of the EEG, with changes in heart rate variability during induction of general anaesthesia. *Brit. J. Anaesth.* **82**, 666 (1999).

[34] J. Bruhn, H. Röpcke, and A. Hoeft. Approximate Entropy as an electroencephalographic measure of anesthetic drug effect during desflurane anesthesia. *Anesthesiology* **92**, 715 (2000).

[35] N. Burioka, G. Cornélissen, F. Halberg, et al. Approximate Entropy of human respiratory movement during eye-closed waking and different sleep stages. *Chest*

123, 80 (2003).

[36] U. Grouven, F. Beger, B. Schultz, and A. Schultz. Correlation of narcotrend index, entropy measures, and spectral parameters with calculated propofol effect-site concentrations during induction of propofol-remifentanil anaesthesia. *J. Clin. Monitor Comp.* **18**, 231 (2004).

[37] A. U. Rajendra, O. Faust, N. Kannathal, T. Chua, and S. Laxminarayan. Nonlinear analysis of EEG signals at various sleep stages. *Comput. Meth. Prog. Bio.* **80**, 37 (2005).

[38] D. Abásolo, J. Escudero, R. Hornero, C. Gómez, and P. Espino. Approximate entropy and auto mutual information analysis of the electroencephalogram in Alzheimer's disease patients. *Med. Biol. Eng. Comput.* **46**, 1019 (2008).

[39] I. I. Andreadis, G. A. Giannakakis, C. Papageorgiou, and K. S. Nikita. Detecting complexity abnormalities in dyslexia measuring approximate entropy of electroencephalographic signals. *Eng. Med. Biol. Soc. Ann.* pp. 6292–6295 (2009).

[40] J. A. Palazzolo, F. G. Estafanous, and P. A. Murray. Entropy measures of heart rate variation in conscious dogs. *Am. J. Physiol. Heart Circ. Physiol.* **274**, H1099 (1998).

[41] S. Garde, M. G. Regalado, V. L. Schechtman, and M. C. K. Khoo. Nonlinear dynamics of heart rate variability in cocaine-exposed neonates during sleep. *Am. J. Physiol. Heart Circ. Physiol.* **280**, H2920 (2001).

[42] T. A. Kuusela, T. T. Jartti, K. U. O. Tahvanainen, and T. J. Kaila. Nonlinear methods of biosignal analysis in assessing terbutaline-induced heart rate and blood pressure changes. *Am. J. Physiol. Heart Circ. Physiol.* **282**, H773 (2002).

[43] S. M. Pincus. Approximate entropy as a measure of irregulatiry for psychiatric serial metrics. *Bipolar Disord.* **8**, 430 (2006).

[44] M. E. Torres and L. G. Gamero. Relative complexity changes in time series using information measures. *Physica A* **286**, 457 (2000).

[45] F. Kaffashi, R. Foglyano, C. G. Wilson, and K. A. Loparo. The effect of time delay on Approximate & Sample Entropy calculations. *Physica D* **237**, 3069 (2008).

[46] M. Ferrario, M. G. Signorini, G. Magenes, and S. Cerutti. Comparison of entropy-based regularity estimators: application to the fetal heart rate signal for the identification of fetal distress. *IEEE T. Bio.-Med. Eng.* **53**, 119 (2006).

[47] B. T. Santos, R. A. Martins, J. E. S. Natali, et al. Consistency in approximate entropy given by a volumetric estimate. *Chaos Soliton. Fract.* **42**, 322 (2009).

[48] W. Chen, Z. Wang, H. Xie, and W. Yu. Characterization of surface EMG signal based on Fuzzy Entropy. *IEEE T. Neur. Sys. Reh.* **15**, 266 (2007).

[49] W. Chen, J. Zhuang, W. Yu, and Z. Wang. Measuring complexity using FuzzyEn, ApEn, and SampEn. *Med. Eng. Phys.* **31**, 61 (2009).

[50] J. S. Richman and J. R. Moorman. Physiological time-series analysis using approximate entropy and sample entropy. *Am. J. Physiol. Heart Circ. Physiol.* **278**, H2039 (2000).

[51] D. E. Lake, J. S. Richman, M. P. Griffin, and J. R. Moorman. Sample entropy

analysis of neonatal heart rate variability. *Am. J. Physiol. Regul. Integr. Comp. Physiol.* **283**, R789 (2002).

[52] D. Hoyer, B. Pompe, K. H. Chon, et al. Mutual information function assesses autonomic information flow of heart rate dynamics at different time scales. *IEEE T. Bio.-Med. Eng.* **52**, 584 (2005).

[53] E. Swe and M. Pwint. An efficient approach for classification of speech and music. *Lect. Notes Comput. Sc.* pp. 50–60 (2008).

[54] L. Guzmán-Vargas, A. Ramírez-Rojas, R. Hernández-Pérez, and F. Angulo-Brown. Correlations and variability in electrical signals related to earthquake activity. *Physica A* **388**, 4218 (2009).

[55] Y.-H. Sun, H.-L. Jou, and J.-C. Wu. Auxiliary diagnosis method for lead-acid battery health based on sample entropy. *Energ. Convers. Manage.* **50**, 2250 (2009).

[56] L. Faes, H. Zhao, K. H. Chon, and G. Nollo. Time-varying surrogate data to assess nonlinearity in nonstationary time series: application to heart rate variability. *IEEE T. Bio.-Med. Eng.* **56**, 685 (2009).

[57] J. S. Chang, C. S. Yoo, S. H. Yi, et al. Differential pattern of heart rate variability in patients with schizophrenia. *Prog. Neuro-Psychoph.* **33**, 991 (2009).

[58] P. Micó, M. Mora, D. Cuesta-Frau, and M. Aboy. Automatic segmentation of long-term ECG signals corrupted with broadband noise based on sample entropy. *Comput. Meth. Prog. Bio.* **98**, 118 (2010).

[59] Y. Song and P. Lio. A new approach for epileptic seizure detection: sample entropy based feature extraction and extreme learning machine. *J. Biomed. Sci. Eng.* **3**, 556 (2010).

[60] R. Alcaraz, D. Absolo, R. Hornero, and J. Rieta. Optimal parameters study for sample entropy-based atrial fibrillation organization analysis. *Comput. Meth. Prog. Bio.* **99**, 124 (2010).

[61] R. B. Govindan, J. D. Wilson, H. Eswaran, C. L. Lowery, and H. Preiffl. Revisiting sample entropy analysis. *Physica A* **376**, 158 (2007).

[62] S. Ramdani, F. Bouchara, and J. Lagarde. Influence of noise on the sample entropy algorithm. *Chaos* **19**, 013123 (2009).

[63] E. M. Bollt and J. Skufca. Control Entropy: A complexity measure for nonstationary signals. *Math. Biosci. Eng.* **6**, 1 (2009).

[64] C. Adami. What is complexity? *BioEssays* **24**, 1085 (2002).

[65] I. A. Rezek and S. J. Roberts. Stochastic complexity measures for physiological signal analysis. *IEEE T. Bio.-Med. Eng.* **45**, 1186 (1998).

[66] J. Riihijarvi, P. Mahonen, and M. Wellens. Metrics for characterizing complexity of network traffic. In *Proc. of ICT*. St. Petersburg, Russia (2008).

[67] S. H. Strogatz. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*. Westview Press (1994).

[68] C. Glymour, D. Madigan, D. Pregibon, and P. Smyth. Statistical themes and lessons for data mining. *Data Min. Knowl. Disc.* **1**, 11 (1997).

[69] J. W. Tukey. *Exploratory data analysis*. Addison Wesley (1977).

[70] J. P. A. Ioannidis. Why most published research findings are false. *PLoS Med.* **2**, e124 (2005).

[71] E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: A survey and empirical demonstration. *Data Min. Knowl. Disc.* **7**, 349 (2003).

[72] D. J. Hand. Data mining: New challenges for statisticians. *Soc. Sci. Comput. Rev.* **18**, 442 (2000).

[73] T. Schreiber and A. Schmitz. Discrimination power of measures for nonlinearity in a time series. *Phys. Rev. E* **55**, 5443 (1997).

[74] D. J. Hand. Deconstructing statistical questions. *J. Roy. Stat. Soc. A Sta.* **157**, 317 (1994).

[75] C. C. Chatfield. Avoiding statistical pitfalls. *Statist. Sci.* **6**, 240 (1991).

[76] D. Stekel. *Microarray Bioinformatics*. Cambridge University Press (2003).

[77] R. D. King, K. E. Whelan, F. M. Jones, et al. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature* **427**, 247 (2004).

[78] R. D. King, J. Rowland, S. G. Oliver, et al. The automation of science. *Science* **324**, 85 (2009).

[79] A. Turing. Computing machinery and intelligence. *Mind* **59**, 433 (1950).

[80] M. Schmidt and H. Lipson. Distilling free-form natural laws from experimental data. *Science* **324**, 81 (2009).

[81] B. Dumouchel and J. Demaine. Knowledge discovery in the digital library: access tools for mining science. *Inform. Serv. Use* **26**, 39 (2006).

[82] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd edition (2009).

[83] J. B. Tenenbaum, V. d. Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science* **290**, 2319 (2000).

[84] P. Samuelson. Rational theory of warrant pricing. *Ind. Manag. Rev.* **6**, 13 (1965).

[85] U. S. Freitas, C. Letellier, and L. A. Aguirre. Failure in distinguishing colored noise from chaos using the "noise titration" technique. *Phys. Rev. E* **79**, 035201(R) (2009).

[86] R. G. Andrzejak, K. Lehnertz, F. Mormann, et al. Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state. *Phys. Rev. E* **64**, 061907 (2001).

[87] A. Subasi and M. Ismail Gursoy. EEG signal classification using PCA, ICA, LDA and support vector machines. *Expert Syst. Appl.* **37**, 8659 (2010).

[88] M. Malik, J. T. Bigger, A. J. Camm, et al. Heart rate variability: Standards of measurement, physiological interpretation, and clinical use. *Eur. Heart J.* **17**, 354 (1996).

[89] S. G. Cecchetti and P.-s. Lam. Variance-ratio tests: Small-sample properties with an application to international output data. *J. Bus. Econ. Stat.* **12**, 177 (1994).

[90] H.-O. Peitgen, D. Saupe, Y. Fisher, et al. *The Science of Fractal Images*.

Springer (1988).

[91] M. A. Little, P. E. McSharry, E. J. Hunter, J. Spielman, and L. O. Ramig. Suitability of dysphonia measurements for telemonitoring of Parkinson's Disease. *IEEE T. Bio.-Med. Eng.* **56**, 1015 (2009).

[92] R. Hegger, H. Kantz, and T. Schreiber. Practical implementation of nonlinear time series methods: The TISEAN package. *Chaos* **9**, 413 (1999).

[93] C. Chatfield. *The Analysis of Time Series.* CRC Press LLC (2004).

[94] X. Wang, A. Wirth, and L. Wang. Structure-based statistical features and multivariate time series clustering. In *IEEE Data Mining*, pp. 351–360. IEEE Computer Society (2007).

[95] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* Wiley-Interscience, second edition (2006).

[96] R. Steuer, J. Kurths, C. O. Daub, J. Weise, and J. Selbig. The mutual information: Detecting and evaluating dependencies between variables. *Bioinformatics* **18**, S231 (2002).

[97] I. Priness, O. Maimon, and I. Ben-Gal. Evaluation of gene-expression clustering via mutual information distance measure. *BMC Bioinformatics* **8**, 111 (2007).

[98] A. Kraskov, H. Stögbauer, and P. Grassberger. Estimating mutual information. *Phys. Rev. E* **69**, 066138 (2004).

[99] A. Kraskov, H. Stögbauer, R. G. Andrzejak, and P. Grassberger. Hierarchical clustering using mutual information. *EPL-Europhys. Lett.* **70**, 278 (2005).

[100] Z. Dawy, J. Hagenauer, P. Hanus, and J. C. Mueller. Mutual information based distance measures for classification and content recognition with application to genetics. In *IEEE ICC* (2005).

[101] G. Gan, C. Ma, and J. Wu. *Data Clustering. Theory, Algorithms, and Applications.* SIAM, Philadelphia, Pennsylvania (2007).

[102] R. Xu and D. Wunsch II. Survey of clustering algorithms. *IEEE T. Neural Networ.* **16**, 645 (2005).

[103] L. Kaufman and P. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis.* John Wiley & Sons, Ltd., New York (1990).

[104] A. K. Jain, R. P. W. Duin, and J. Mao. Statistical pattern recognition: a review. *IEEE T. Pattern. Anal.* **22**, 4 (2000).

[105] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.* **3**, 1157 (2003).

[106] I. Guyon, C. Aliferis, and A. Elisseeff. Causal feature selection (2007). Http://www.clopinet.com/isabelle/Papers/causalFS.pdf.

[107] R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Roy. Stat. Soc. B Met.* **58**, 267 (1996).

[108] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *J. Roy. Stat. Soc. B* **67**, 301 (2005).

[109] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using Support Vector Machines. *Mach. Learn.* **46**, 389 (2002).

[110] D. Donoho and J. Jin. Higher criticism thresholding: Optimal feature selection

when useful features are rare and weak. *P. Natl. Acad. Sci. USA* **105**, 14790 (2008).

[111] J. Jin. Impossibility of successful classification when useful features are rare and weak. *P. Natl. Acad. Sci. USA* **106**, 8859 (2009).

[112] C. Ambroise and G. J. McLachlan. Selection bias in gene extraction on the basis of microarray gene-expression data. *P. Natl. Acad. Sci. USA* **99**, 6562 (2002).

[113] P. Smialowski, D. Frishman, and S. Kramer. Pitfalls of supervised feature selection. *Bioinformatics* **26**, 440 (2010).

[114] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Inform. Process. Lett.* **31**, 7 (1989).

[115] D. Smith. Network visualization and energy optimization (2012). Private communication.

[116] H. Hotelling. Analysis of a complex of statistical variables into Principal Components. *J. Educ. Psychol.* **24**, 417 (1933).

[117] E. Koscielny-Bunde, A. Bunde, S. Havlin, et al. Indication of a universal persistence law governing atmospheric variability. *Phys. Rev. Lett.* **81**, 729 (1998).

[118] P. Talkner and R. O. Weber. Power spectrum and detrended fluctuation analysis: Application to daily temperatures. *Phys. Rev. E* **62**, 150 (2000).

[119] Y. Liu, P. Cizeau, M. Meyer, C. K. Peng, and H. Eugene Stanley. Correlations in economic time series. *Physica A* **245**, 437 (1997).

[120] H. Stanley, L. Amaral, A. Goldberger, et al. Statistical physics and physiology: monofractal and multifractal approaches. *Physica A* **270**, 309 (1999).

[121] R. B. Davies and D. S. Harte. Tests for Hurst effect. *Biometrika* **74**, 95 (1987).

[122] D. Delignieres, S. Ramdani, L. Lemoine, et al. Fractal analyses for 'short' time series: A re-assessment of classical methods. *J. Math. Psychol.* **50**, 525 (2006).

[123] S. M. Pincus, T. R. Cummins, and G. G. Haddad. Heart rate control in normal and aborted-SIDS infants. *Am J. Physiol. Regul. Integr. Comp. Physiol.* **264**, R638 (1993).

[124] D. Kwiatkowski, P. C. B. Phillips, P. Schmidt, and Y. Shin. Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *J. Econometrics* **54**, 159 (2002).

[125] P. C. Ivanov, L. A. N. Amaral, A. L. Goldberger, et al. Multifractality in human heartbeat dynamics. *Nature* **399**, 461 (1999).

[126] P. C. Ivanov, Q. D. Y. Ma, R. P. Bartsch, et al. Levels of complexity in scale-invariant neural signals. *Phys. Rev. E* **79**, 041920 (2009).

[127] A. L. Goldberger, L. A. N. Amaral, J. M. Hausdorff, et al. Fractal dynamics in physiology: Alterations with disease and aging. *P. Natl. Acad. Sci. USA* **99**, 2466 (2002).

[128] H. D. Jennings, P. C. Ivanov, A. d. M. Martins, P. C. da Silva, and G. M. Viswanathan. Variance fluctuations in nonstationary time series: a comparative study of music genres. *Physica A* **336**, 585 (2004).

[129] A. Lempel and J. Ziv. On the complexity of finite sequences. *IEEE T. Inform.*

Theory **22**, 75 (1976).

[130] M. Small. *Applied Nonlinear Time Series Analysis: Applications in Physics, Physiology, and Finance*, volume 52 of *Nonlinear Science Series A*. World Scientific (2005).

[131] D. S. Broomhead and G. P. King. Extracting qualitative dynamics from experimental data. *Physica D* **20**, 217 (1986).

[132] A. M. Fraser and H. L. Swinney. Independent coordinates for strange attractors from mutual information. *Phys. Rev. A* **33**, 1134 (1986).

[133] J. Kohlmorgen, K. R. Müller, J. Rittweger, and K. Pawelzik. Identification of nonstationary dynamics in physiological recordings. *Biol. Cybern.* **83**, 73 (2000).

[134] F. Burkhardt, A. Paeschke, M. Rolfes, W. Sendlmeier, and B. Weiss. A database of German emotional speech. In *Interspeech*, pp. 1517–1520. Lisbon, Portugal (2005).

[135] L. Wesserman. *All of Statistics*. Springer (2004).

[136] W. Weibull. A statistical distribution function of wide applicability. *J. Appl. Mech.* **18**, 293 (1951).

[137] V. P. Singh. On application of the Weibull distribution in hydrology. *Water Resour. Manag.* **1**, 33 (1987).

[138] D. J. Fenn, M. A. Porter, M. McDonald, et al. Dynamic communities in multichannel data: An application to the foreign exchange market during the 2007–2008 credit crisis. *Chaos* **19**, 033119 (2009).

[139] J. P. Onnela, A. Chakraborti, K. Kaski, J. Kertész, and A. Kanto. Dynamics of market correlations: Taxonomy and portfolio analysis. *Phys. Rev. E* **68**, 056110 (2003).

[140] F. Black and M. Scholes. The pricing of options and corporate liabilities. *J. Polit. Econ.* **81**, 637 (1973).

[141] R. S. Tsay. *Analysis of Financial Time Series*. John Wiley & Sons, Ltd., Hoboken, New Jersey (2005).

[142] T. G. Anderson, R. A. Davis, J.-P. Kreiß, and T. Mikosch, eds. *Handbook of Financial Time Series*. Springer (2009).

[143] J. J. Jiang, Y. Zhang, and J. Stern. Modeling of chaotic vibrations in symmetric vocal folds. *J. Acoust. Soc. Am.* **110**, 2120 (2001).

[144] M. A. Little. *Biomechanically informed nonlinear speech signal processing*. Ph.D. thesis, Oxford University, Oxford, UK (2007).

[145] M. Zaiki, G. P. Können, T. Tsukahara, et al. Recovery of nineteenth-century Tokyo/Osaka meteorological data in Japan. *Int. J. Climatol.* **26**, 399 (2006).

[146] C. Heneghan and G. McDarby. Establishing the relation between detrended fluctuation analysis and power spectral density analysis for stochastic processes. *Phys. Rev. E* **62**, 6103 (2000).

[147] B. D. Malamud and D. L. Turcotte. Self-affine time series: measures of weak and strong persistence. *J. Stat. Plan. Infer.* **80**, 173 (1999).

[148] M. S. Taqqu, V. Teverovsky, and W. Willinger. Estimators for long-range

dependence: An empirical study. *Fractals* **3**, 785 (1995).

[149] D. C. Caccia, D. Percival, M. J. Cannon, G. Raymond, and J. B. Bassingthwaighte. Analyzing exact fractal time series: evaluating dispersional analysis and rescaled range methods. *Physica A* **246**, 609 (1997).

[150] S. Cerutti, F. Esposti, M. Ferrario, R. Sassi, and M. G. Signorini. Long-term invariant parameters obtained from 24-h Holter recordings: A comparison between different analysis techniques. *Chaos* **17**, 015108 (2007).

[151] A. Clauset, C. R. Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *arXiv* p. 0706.1062 (2007).

[152] M. Little, P. McSharry, S. Roberts, D. Costello, and I. Moroz. Exploiting nonlinear recurrence and fractal scaling properties for voice disorder detection. *Biomed. Eng. Online* **6**, 23 (2007).

[153] R. H. Shumway and D. S. Stoffer. *Time Series Analysis and Its Applications*. Springer-Verlag, New York (2000).

[154] H.-Y. Huang, H. Ombao, and D. S. Stoffer. Discrimination and Classification of Nonstationary Time Series Using the SLEX Model. *J. Am. Stat. Assoc.* **99**, 763 (2008).

[155] R. Blandford. Discrimination between earthquakes and underground explosions. *Annu. Rev. Earth Pl. Sc.* **5**, 111 (1977).

[156] Y. Kakizawa, R. H. Shumway, and M. Taniguchi. Discrimination and clustering for multivariate time series. *J. Am. Stat. Assoc.* **93**, 328 (1998).

[157] S. Chandrakala and C. Sekhar. Classification of multi-variate varying length time series using descriptive statistical features. *Lect. Notes Comput. Sc.* pp. 13–18 (2009).

[158] D. Kim, S.-Y. Lee, and S.-i. Amari. Representative and discriminant feature extraction based on NMF for emotion recognition in speech. *Neu. Inf. Pro.* pp. 649–656 (2009).

[159] S. Chandrakala and C. C. Sekhar. Support vector regression based autoassociative models for time series classification. In *NCC*, pp. 1–5 (2010).

[160] Z. Xiao, E. Dellandréa, W. Dou, and L. Chen. Hierarchical Classification of Emotional Speech. Technical report, École Centrale de Lyon (2007).

[161] B. Schuller, D. Arsic, F. Wallhoff, and G. Rigoll. Emotion recognition in the noise applying large acoustic feature sets. In *Speech Prosody, Dresden*, pp. 276–289 (2006).

[162] M. M. H. El Ayadi, M. S. Kamel, and F. Karray. Speech emotion recognition using Gaussian mixture vector autoregressive models. In *ICASSP*, volume 4, pp. IV–957. IEEE (2007).

[163] V. Nigam and D. Graupe. A neural-network-based detection of epilepsy. *Neurol. Res.* **26**, 55 (2004).

[164] A. Subasi. EEG signal classification using wavelet feature extraction and a mixture of expert model. *Expert Syst. Appl.* **32**, 1084 (2007).

[165] N. F. Güler, E. D. Übeyli, and I. Güler. Recurrent neural networks employing Lyapunov exponents for EEG signals classification. *Expert Syst. Appl.* **29**, 506 (2005).

[166] P. E. McSharry, L. A. Smith, and L. Tarassenko. Prediction of epileptic seizures: are nonlinear methods relevant? *Nat. Med.* **9**, 241 (2003).

[167] L. Cnockaert, J. Schoentgen, P. Auzou, et al. Low-frequency vocal modulations in vowels produced by Parkinsonian subjects. *Speech Commun.* **50**, 288 (2008).

[168] K. M. Rosen, R. D. Kent, A. L. Delaney, and J. R. Duffy. Parametric quantitative acoustic analysis of conversation produced by speakers with dysarthria and healthy speakers. *J. Speech. Lang. Hear. R.* **49**, 395 (2006).

[169] D. A. Rahn, M. Chou, J. J. Jiang, and Y. Zhang. Phonatory impairment in Parkinson's Disease: Evidence from nonlinear dynamic analysis and perturbation analysis. *J. Voice* **21**, 64 (2007).

[170] A. Tsanas, M. Little, P. McSharry, and L. Ramig. Accurate telemonitoring of Parkinson's Disease progression by noninvasive speech tests. *IEEE T. Bio.-Med. Eng.* **57**, 884 (2010).

[171] A. Tsanas, M. A. Little, P. E. McSharry, and L. O. Ramig. Nonlinear speech analysis algorithms mapped to a standard metric achieve clinically useful quantification of average Parkinson's Disease symptom severity. *J. Roy. Soc. Interface* **8**, 842 (2010).

[172] P. Boersma. Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound. In *Proc. Instit. Phonet. Sci.*, volume 17, pp. 97–110 (1993).

[173] M. Little, P. McSharry, I. Moroz, and S. Roberts. Nonlinear, biophysically-informed speech pathology detection. In *IEEE T. Acoust. Speech*, volume 2 (2006).

[174] J. E. Mietus, C.-K. Peng, I. Henry, R. L. Goldsmith, and A. L. Goldberger. The pNNx files: re-examining a widely used heart rate variability measure. *Heart* **88**, 378 (2002).

[175] S. Guzzetti, M. G. Signorini, C. Cogliati, et al. Non-linear dynamics and chaotic indices in heart rate variability of normal subjects and heart-transplanted patients. *Cardiovasc. Res.* **31**, 441 (1996).

[176] M. Costa, A. L. Goldberger, and C. K. Peng. Broken asymmetry of the human heartbeat: Loss of time irreversibility in aging and disease. *Phys. Rev. Lett.* **95**, 198102 (2005).

[177] N. Wessel, C. Ziehmann, J. Kurths, et al. Short-term forecasting of life-threatening cardiac arrhythmias based on symbolic dynamics and finite-time growth rates. *Phys. Rev. E* **61**, 733 (2000).

[178] J. S. Chang, C. S. Yoo, S. H. Yi, et al. Changes in heart rate dynamics of patients with schizophrenia treated with risperidone. *Prog. Neuro-Psychoph.* **34**, 924 (2010).

[179] A. Voss, J. Kurths, H. J. Kleiner, et al. The application of methods of non-linear dynamics for the improved and predictive recognition of patients threatened by sudden cardiac death. *Cardiovasc. Res.* **31**, 419 (1996).

[180] L. A. N. Amaral, A. L. Goldberger, P. C. Ivanov, and H. E. Stanley. Scale-independent measures and pathologic cardiac dynamics. *Phys. Rev. Lett.* **81**, 2388 (1998).

[181] D. Makowiec, A. Dudkowska, R. Galaska, and A. Rynkiewicz. Multifractal estimates of monofractality in RR-heart series in power spectrum ranges. *Physica A* **388**, 3486 (2009).

[182] Y. Ashkenazy, P. C. Ivanov, S. Havlin, et al. Magnitude and sign correlations in heartbeat fluctuations. *Phys. Rev. Lett.* **86**, 1900 (2001).

[183] J. Kurths, A. Voss, P. Saparin, et al. Quantitative analysis of heart rate variability. *Chaos* **5**, 88 (1995).

[184] A. Voss, J. Kurths, H. J. Kleiner, A. Witt, and N. Wessel. Improved analysis of heart rate variability by methods of nonlinear dynamics. *J. Electrocardiol.* **28**, 81 (1995).

[185] M. Brennan, M. Palaniswami, and P. Kamen. Do existing measures of Poincaré plot geometry reflect nonlinear features of heart rate variability? *IEEE T. Bio.-Med. Eng.* **48**, 1342 (2001).

[186] S. M. Duarte Queirós and L. G. Moyano. Yet on statistical properties of traded volume: Correlation and mutual information at different value magnitudes. *Physica A* **383**, 10 (2007).

[187] J. Faust. When are variance ratio tests for serial dependence optimal? *Econometrica* **60**, 1215 (1992).

[188] J. Durbin and G. S. Watson. Testing for serial correlation in least squares regression: I. *Biometrika* **37**, 409 (1950).

[189] C. K. Peng, J. Mietus, J. M. Hausdorff, et al. Long-range anticorrelations and non-Gaussian behavior of the heartbeat. *Phys. Rev. Lett.* **70**, 1343 (1993).

[190] A. Georgieva, S. Payne, M. Moulden, and C. W. G. Redman. Artificial neural networks applied to fetal monitoring in labour. *Neural Comput. Appl.* (2011). In press, doi: 10.1007/s00521-011-0743-y.

[191] J. Westgate. *Medical informatics in obstetrics and gynecology. Medical Info Science Reference.*, chapter x: Computerizing the Cardiotocogram (CTG), pp. 151–158. Medical Information Science Reference (2009).

[192] M. Cesarelli, M. Romano, and P. Bifulco. Comparison of short term variability indexes in cardiotocographic foetal monitoring. *Comput. Biol. Med.* **39**, 106 (2009).

[193] G. Georgoulas, D. Gavrilis, I. G. Tsoulos, et al. Novel approach for fetal heart rate classification introducing grammatical evolution. *Biomed. Signal Proces.* **2**, 69 (2007).

[194] J. Spilka, V. Chudácek, M. Koucký, et al. Using nonlinear features for fetal heart rate classification (in press). *Biomed. Signal Proces.* (2011).

[195] C. U. Lee and A. G. Dorffner. Application of artificial neural networks for detection of abnormal fetal heart rate pattern: a comparison with conventional algorithms. *J. Obstet. Gynecol.* **19**, 482 (1999).

[196] L. C. Pello, S. K. Rosevear, G. S. Dawes, M. Moulden, and C. W. G. Redman. Computerized fetal heart rate analysis in labor. *Obstet. Gynecol.* **78** (1991).

[197] B. K. Strachan, D. S. Sahota, W. J. van Wijngaarden, D. K. James, and A. M. Z. Chang. Computerised analysis of the fetal heart rate and relation to acidaemia at delivery. *BJOG-Int. J. Obstet. Gy.* **108**, 848 (2001).

[198] P. A. Warrick, E. F. Hamilton, D. Precup, and R. E. Kearney. Identification of the dynamic relationship between intrapartum uterine pressure and fetal heart rate for normal and hypoxic fetuses. *IEEE T. Bio.-Med. Eng.* **56**, 1587 (2009).

[199] P. A. Warrick, E. F. Hamilton, D. Precup, and R. E. Kearney. Classification of normal and hypoxic fetuses from systems modeling of intrapartum cardiotocography. *IEEE T. Bio.-Med. Eng.* **57**, 771 (2010).

[200] M. Jezewski, J. Wrobel, P. Labaj, et al. Some practical remarks on neural networks approach to fetal Cardiotocograms classification. *Eng. Med. Biol. Soc. Ann.* pp. 5170–5173 (2007).

[201] A. E. Georgieva, S. J. Payne, M. Moulden, and C. W. G. Redman. Automated fetal heart rate analysis in labor: Decelerations and overshoots. *AIP Conf. Proc.* **1293**, 255 (2010).

[202] J. Pardey, M. Moulden, and C. W. G. Redman. A computer system for the numerical analysis of nonstress tests. *Am. J. Obstet. Gynecol.* **186**, 1095 (2002).

[203] X. Wang, H. Ding, G. Trajcevski, P. Scheuermann, and E. J. Keogh. Experimental comparison of representation methods and distance measures for time series data. *arXiv* p. 1012.2789 (2010).

[204] T. W. Liao. Clustering of time series data – a survey. *Pattern Recogn.* **38**, 1857 (2005).

[205] J. Timmer, C. Gantert, G. Deuschl, and J. Honerkamp. Characteristics of hand tremor time series. *Biol. Cybern.* **70**, 75 (1993).

[206] K. Deng, A. W. Moore, and M. C. Nechyba. Learning to recognize time series: combining ARMA models with memory-based learning. In *IEEE CIRA*, pp. 246–251 (1997).

[207] A. Nanopoulos, R. Alcock, and Y. Manolopoulos. *Feature-based classification of time-series data*, pp. 49–61. Nova Science Publishers, Inc., Commack, NY, USA (2001).

[208] P. Siirtola, P. Laurinen, E. Haapalainen, J. Roning, and H. Kinnunen. Clustering-based activity classification with a wrist-worn accelerometer using basic features. In *CIDM*, pp. 95–100. IEEE (2009).

[209] F. Mörchen. Time series feature extraction for data mining using DWT and DFT. Technical report (2003).

[210] X. Wang, K. Smith, and R. Hyndman. Characteristic-based clustering for time series data. *Data Min. Knowl. Disc.* **13**, 335 (2006).

[211] P. Siirtola, H. Koskimäki, V. Huikari, P. Laurinen, and J. Röning. Improving the classification accuracy of streaming data using SAX similarity features. *Pattern Recogn. Lett.* **32**, 1659 (2011).

[212] D. Wolpert and W. Macready. No free lunch theorems for optimization. *IEEE T. Evolut. Comput.* **1**, 67 (1997).

[213] T. Mitsa. *Temporal Data Mining*. Chapman & Hall/CRC Press (2010).

[214] E. Keogh, X. Xi, L. Wei, and C. A. Ratanamahatana. The UCR Time Series Classification/Clustering Homepage (2006).

[215] E. Keogh, Q. Zhu, B. Hu, et al. The UCR Time Series Classification/Clustering Homepage (2011).

[216] Olszweski. *Generalized Feature Extraction for Structural Pattern Recognition in Time-Series Data.* Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA (2001).

[217] D. Eads, D. Hill, S. Davis, et al. Genetic algorithms and support vector machines for time series classification. *P. Soc. Photo.-Opt. Ins.* **4787**, 74 (2002).

[218] O. J. O. Söderkvist. *Computer vision classification of leaves from Swedish trees.* Master's thesis (2001).

[219] L. Wei and E. Keogh. Semi-supervised time series classification (2006).

[220] J. Lin, E. Keogh, L. Wei, and S. Lonardi. Experiencing SAX: a novel symbolic representation of time series. *Data Min. Knowl. Disc.* **15**, 107 (2007).

[221] C. A. Ratanamahatana and E. Keogh. Making time-series classificaiton more accurate using learned constraints. In *SIAM SDM* (2004).

[222] T. Subba Rao and M. M. Gabr. *An Introduction to Bispectral Analysis and Bilinear Time Series Models, Lecture Notes in Statistics (Vol. 24).* Springer, New York (1984).

[223] C. Diks, J. C. van Houwelingen, F. Takens, and J. DeGoede. Reversibility as a criterion for discriminating time series. *Phys. Lett. A* **201**, 221 (1995).

[224] T. Schreiber and A. Schmitz. Surrogate time series. *Physica D* **142**, 346 (2000).

[225] K. Chakrabarti, E. Keogh, S. Mehrotra, and M. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Trans. Database Syst.* **27**, 188 (2002).

[226] J. Shieh and E. Keogh. *i*SAX: Indexing and mining terabyte sized time series. *SIGKDD* (2008).

[227] J. Shieh and E. Keogh. *i*SAX: disk-aware mining and indexing of massive time series datasets. *Data Min. Knowl. Disc.* **19**, 24 (2009).

[228] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science* **290**, 2323 (2000).

[229] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.* **15**, 1373 (2003).

[230] O. C. Jenkins and M. J. Matarić. A spatio-temporal extension to Isomap nonlinear dimension reduction. In *ICML*, p. 56 (2004).

[231] R. Li, T. Tian, and S. Sclaroff. Simultaneous learning of nonlinear manifold and dynamical models for high-dimensional time series. In *IEEE I. Conf. Comp. Vis.* IEEE (2007).

[232] F. Takens. Detecting strange attractors in turbulence. *Lect. Notes Math.* **898**, 366 (1981).

[233] T. Toni, D. Welch, N. Strelkowa, A. Ipsen, and M. P. H. Stumpf. Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems. *J. Roy. Soc. Interface* **6**, 187 (2009).

[234] L. d. F. Costa, F. A. Rodrigues, G. Travieso, and P. R. Villas Boas. Characterization of complex networks: A survey of measurements. *arXiv* (2006).

[235] V. Filkov, Z. M. Saul, S. Roy, R. M. D'Souza, and P. T. Devanbu. Modeling and verifying a broad array of network properties. *EPL-Europhys. Lett.* **86**,

28003 (2009).

[236] S. Agarwal, G. Villar, and N. S. Jones. High throughput network analysis (in preparation) (2012).

[237] J. Theiler. Spurious dimension from correlation algorithms applied to limited time-series data. *Phys. Rev. A* **34**, 2427 (1986).

[238] Y. Ashkenazy, S. Havlin, P. C. Ivanov, et al. Magnitude and sign scaling in power-law correlated time series. *Physica A* **323**, 19 (2003).

[239] P. S. P. Cowpertwait and A. V. Metcalfe. *Introductory Time Series with R*. Springer (2009).

[240] D. T. Pham and A. B. Chan. Control chart pattern recognition using a new type of self-organizing neural network. *Proc. Inst. Mech. Eng. I-J. Sys.* **212**, 115 (1998).

[241] N. Saito. *Local Feature Extraction and Its Applications Using a Library of Bases*. Ph.D. thesis, Yale University (1994).

[242] C. A. Ratanamahatana and E. Keogh. Everything you know about Dynamic Time Warping is wrong. In *KDD/TDM* (2004).

[243] A. Gandhi. Content-based image retrieval: Plant species identification. *Master's thesis, Oregon State University* (2002).

[244] T. Rath and R. Manmatha. Word image matching using Dynamic Time Warping. *CVPR* **2**, 521 (2003).

[245] D. Roverso. Multivariate temporal classification by windowed wavelet decomposition and recurrent neural networks. In *NPIC & HMIT*. Washington, DC, USA (2000).

[246] P. Geurts. *Contributions to decision tree induction: bias/variance tradeoff and time series classification*. Ph.D. thesis, Department of Electrical Engineering, University of Liege, Belgium (2002).

[247] A. Jalba, M. Wilkinson, J. Roerdink, M. Bayer, and S. Juggins. Automatic diatom identification using contour analysis by morphological curvature scale spaces. *Mach. Vision Appl.* **16**, 217 (2005).

[248] D.-J. Lee, R. B. Schoenberger, D. Shiozawa, X. Xu, and P. Zhan. Contour matching for a fish recognition and migration-monitoring system. In *Proc. SPIE*, volume 5606, pp. 37–48 (2004).

[249] O. Al-Jowder, E. K. Kemsley, and R. H. Wilson. Detection of adulteration in cooked meat products by mid-infrared spectroscopy. *J. Agr. Food Chem.* **50**, 1325 (2002).

[250] R. Briandet, E. K. Kemsley, and R. H. Wilson. Discrimination of Arabica and Robusta in instant coffee by Fourier transform infrared spectroscopy and chemometrics. *J. Agr. Food Chem.* **44**, 170 (1996).

[251] H. S. Tapp, M. Defernez, and E. K. Kemsley. FTIR spectroscopy and multivariate analysis can distinguish the geographic origin of extra virgin olive oils. *J. Agr. Food Chem.* **51**, 6110 (2003).

[252] D. Guarín, E. Delgado, and A. Orozco. Band-phase-randomized surrogates to asses nonlinearity in non-stationary time series. *arXiv* p. 1101.6063 (2011).

[253] J. Timmer. Power of surrogate data testing with respect to nonstationarity.

Phys. Rev. E **58**, 5153 (1998).

[254] L. Cao. Practical method for determining the minimum embedding dimension of a scalar time series. *Physica D* **110**, 43 (1997).

[255] J. M. Hausdorff, L. Zemany, C. K. Peng, and A. L. Goldberger. Maturation of gait dynamics: stride-to-stride variability and its temporal organization in children. *J. Appl. Physiol.* **86**, 1040 (1999).

[256] A. A. Priplata, J. B. Niemi, J. D. Harry, L. A. Lipsitz, and J. J. Collins. Vibrating insoles and balance control in elderly people. *The Lancet* **362**, 1123 (2003).

[257] A. Neumaier and T. Schneider. Estimation of parameters and eigenmodes of multivariate autoregressive models. *ACM Trans. Math. Softw.* **27**, 27 (2001).

[258] T. Schneider and A. Neumaier. Algorithm 808: ARFIT—a Matlab package for the estimation of parameters and eigenmodes of multivariate autoregressive models. *ACM Trans. Math. Softw.* **27**, 58 (2001).

[259] M. A. Little, B. C. Steel, F. Bai, et al. Steps and bumps: Precision extraction of discrete states of molecular machines. *Biophys. J.* **101**, 477 (2011).

[260] M. A. Little and N. S. Jones. Sparse Bayesian step-filtering for high-throughput analysis of molecular machine dynamics. In *Proc. ICASSP* (2010).

[261] K. I. Goh and A. L. Barabási. Burstiness and memory in complex systems. *EPL-Europhys. Lett.* **81**, 48002 (2008).

[262] E. G. Altmann, S. Hallerberg, and H. Kantz. Reactions to extreme events: Moving threshold model. *Physica A* **364**, 435 (2006).

[263] S.-J. Kim, K. Koh, S. Boyd, and D. Gorinevsky. l_1 trend filtering. *SIAM Review* **51**, 339 (2009).

[264] M. J. Cannon, D. B. Percival, D. C. Caccia, G. M. Raymond, and J. B. Bassingthwaighte. Evaluating scaled windowed variance methods for estimating the Hurst coefficient of time series. *Physica A* **241**, 606 (1997).

[265] J. Alvarez-Ramirez, E. Rodriguez, and J. C. Echeverria. Using detrended fluctuation analysis for lagged correlation analysis of nonstationary signals. *Phys. Rev. E* **79**, 057202 (2009).

[266] C. Bandt and B. Pompe. Permutation entropy: A natural complexity measure for time series. *Phys. Rev. Lett.* **88**, 174102 (2002).

[267] P. Grassberger and I. Procaccia. Characterization of strange attractors. *Phys. Rev. Lett.* **50**, 346 (1983).

[268] N. Marwan, N. Wessel, U. Meyerfeldt, A. Schirdewan, and J. Kurths. Recurrence-plot-based measures of complexity and their application to heart-rate-variability data. *Phys. Rev. E* **66**, 026702 (2002).

[269] A. Wolf, J. B. Swift, H. L. Swinney, and J. A. Vastano. Determining Lyapunov exponents from a time series. *Physica D* **16**, 285 (1985).

[270] D. Kugiumtzis. Surrogate data test for nonlinearity including nonmonotonic transforms. *Phys. Rev. E* **62**, R25 (2000).

[271] D. L. Guarín López, A. A. Orozco Gutierrez, and E. Delgado Trejos. A new surrogate data method for nonstationary time series. *arXiv* p. 1008.1804 (2010).

[272] T. Nakamura, M. Small, and Y. Hirata. Testing for nonlinearity in irregular fluctuations with long-term trends. *Phys. Rev. E* **74**, 026205 (2006).

Appendix A

Additional material: Empirical structure of time series and their methods

In this appendix, we present additional information on a broad clustering of operations into four classes, to give a global perspective on the database that compliments the local similarity searches performed in Sec. 4.1.3.

A.1 Broad classes of operations

Clustering time-series analysis methods at a highly coarse-grained level allows us to obtain a reduced number of broad classes of algorithmic behaviors from time-series analysis methods. In this section we consider a clustering of all operations (excluding length-dependent operations) using k-medoids with $k = 4$ to form four clusters. The result of this clustering was shown in Fig. 2.2 of the summary chapter, Chapter 2. We consider this an initial exercise to test the ability of our framework to cluster operations using their behavior on empirical signals; the results will evidently provide a very broad grouping that is insufficient to capture the rich behaviors represented in the set of $\approx 9\,000$ different time-series analysis operations. Nevertheless, we obtain a meaningful clustering into just four groups, that correspond to qualitatively distinct sets of operations for time-series analysis. Summaries of each cluster are obtained by examining the 200 operations nearest to its center; a simple visual representation of the results was plotted in Fig. 2.2A.

The first cluster contains 2 247 operations and is dominated by correlation and

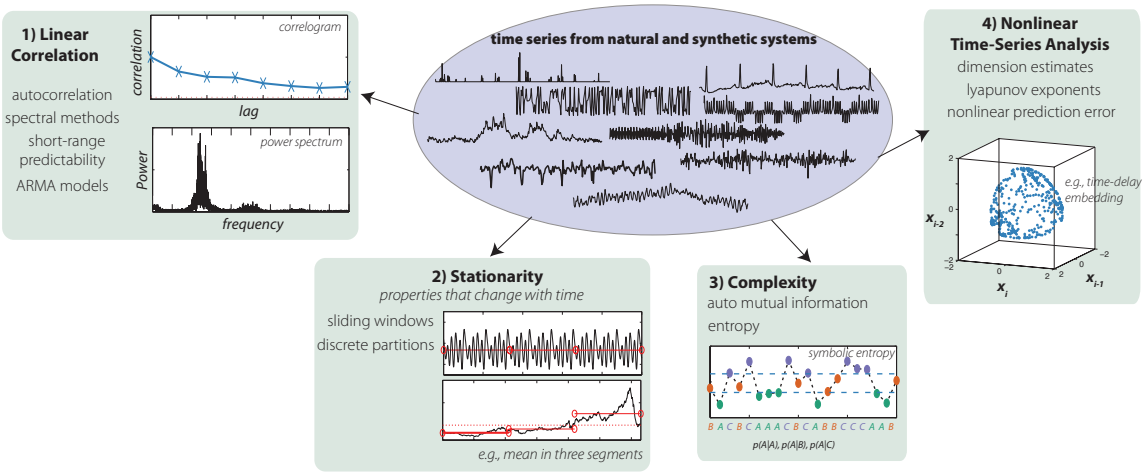

Figure A.1: **Four main classes of time-series analysis operations**, as determined by a k-medoids clustering of 8 651 operations into four groups. Each cluster can be summarized briefly as: (1) methods based linear correlation-based methods, (2) stationarity measures, (3) entropy-based metrics, (4) methods based on nonlinear-time series analysis. Note that clustering all methods into just 4 classes yields a highly simplistic grouping of the time-series analysis literature, however it does provide an interpretable overview of the main classes of operations in our database.

predictability measures, such as the autocorrelation at lag 1, the accuracy of simple local mean forecasters, the frequency of local repetitive motifs (such as the probability of three consecutive low values of a time series, the probability of four consecutive increases in a time series, etc.), the proportion of mean-crossings, and summaries of the power spectrum. The second cluster, which contains 1 616 operations, contains many stationarity and scaling measures, including: sliding window and StatAv measures, KPSS stationarity tests from Econometrics [124], scaling exponent estimates from fluctuation analysis, the first zero-crossing of the autocorrelation function, and measures of low-frequency Fourier spectral power density. In this case, it appears as though the scaling operations are broadly behaving like stationarity measures of signals, a relationship that is examined further in Sec. 4.1.3. The third cluster contains 2 514 operations, and is dominated by complexity-type operations, including Lempel-Ziv complexities, motif entropies, SampEn, ApEn, automutual information, AR and state-space model prediction errors, and nonlinear model cross-prediction errors. The final cluster contains 2 018 operations, and is centered near methods from nonlinear time-series analysis, including correlation dimension estimates, higher-order distributional moments, and outlier measures. In this case, the presence of extreme

outliers, which strongly affect measures of higher-order moments of the distribution are affecting many dimension estimate routines from nonlinear time-series analysis, which are known to be very sensitive techniques [13].

Placing every operation in our database into one of just four groups is evidently a highly coarse-grained representation, and the members of each cluster are a diverse mix of different operations. However, as we have shown, this clustering yields a meaningful empirical grouping of the main types of time-series analysis methods represented in our database. Further, it can also be considered to provide an (albeit overly-simplified) representation of the main types of ways that scientists have proposed to quantify the empirical properties of time series: by measuring simple correlations, fitting statistical models, checking temporal variation in properties (stationarity), quantifying disorder/entropy, and employing techniques derived from nonlinear time-series analysis.

Appendix B

Additional material: Applications of highly comparative time-series analysis

In this appendix, we present additional information on operations described throughout Chapter 5. The description of large number of operations disturbs the flow of the text in these sections and so has been summarized in Chapter 5, with more details presented here for interested readers. In addition to these lists and descriptions of operations, we also show selected additional results for each case study that could not be included in the main text due to space constraints.

B.1 Regression

B.1.1 Noisy periodic time series

The most successful operations for predicting η are operations measuring spread in various ways, including the standard deviation, plotted in Fig. 5.3(a). The reduction in the periodic component of the signal with η is also measured by various statistics summarizing the power spectrum, including the operation plotted in Fig. 5.3(b), which outputs the logarithm of the maximum value in the power spectrum, as computed using a fast Fourier transform (via the MATLAB function **fft**). The power in the peak in the normalized power spectrum is initially maximal, and decreases as the power in the white noise (which has a flat power spectrum) increases with η. The time scale over which consecutive values in the time series are correlated can be estimated by the first zero crossing of the autocorrelation function—periodic time series have

a large value of this statistic, while uncorrelated processes will have a value close to zero. The operation plotted in Fig. 5.3(c) measures this quantity in 100 50-sample segments of the time series and returns the mean. It decreases with the standard deviation of the added noise, as expected. An exponential smoothing model [93] is fitted to a 'training set' of the first 50% of the time series in Fig. 5.3(d), where the optimal smoothing parameter, α, is returned. A high α implies greater autocorrelation in the time series – this fitted parameter is able to estimate the variation in η approximately linearly. A simple nonlinear autocorrelation, $\langle x_i^3 x_{i-1} \rangle$, varies monotonically with η in Fig. 5.3(e). Finally, we find a metric based on the idea of *noise titration* [5], which incrementally adds Gaussian-distributed noise to a given input time series, and estimates the exponential decay rate in the automutual information at lag one, AMI_1 (calculated using a histogram-based method with 20 equiprobable bins). For the highly periodic signals, the AMI_1 is initially very high and decays quickly with the addition of uncorrelated noise. As η increases, however, the autocorrelations are already weak in the time series, and do not decay significantly with the further addition of noise.

B.1.2 Logistic Map time series operations

In this section we describe a selection of operations plotted in Fig. 5.6 that exhibit strong linear correlations to the Lyapunov exponent, λ, of Logistic Map time series analyzed in Sec. 5.1.2.

Largest Lyapunov Exponent As could be expected, a Lyapunov exponent estimator, shown in Fig. 5.6(a), performs the best for this task, although it only yields valid outputs for $\lambda \geq 0$. This operation is based on the *TSTOOL* method `largelyap`, which is publicly available[1]. The algorithm uses all possible reference points, a maximum prediction length equal to 10% of the time series length (i.e., 1 000 samples for these time series that are 10 000 samples long), a Theiler window [13, 237] equal to 1% of the time series length (i.e., 100 samples), and three nearest neighbors for a time-delay embedding of the time series with time-delay $\tau = 1$ and embedding di-

[1] Code available at `http://www.physik3.gwdg.de/tstool/index.html`

mension $m = 4$. A scaling range is found by varying the maximum prediction length to get the best fit whilst penalizing shorter scaling ranges, and returns the gradient of the prediction error versus the prediction length across this optimal scaling range.

Local Density Another operation derived from nonlinear time-series analysis exhibits strong correlations to λ for $\lambda \geq 0$, and is plotted in Fig. 5.6(b). It uses the routine `localdensity` from the *TSTOOL* package with five nearest neighbors, and a Theiler window [13, 237] of 40 for a time-delay embedding of the signal using a time-delay $\tau = 1$ and embedding dimension $m = 3$. The algorithm returns the mean of a set of local density estimates obtained at each point in the three-dimensional time-series trajectory.

Correlation Entropy The algorithm plotted in Fig. 5.6(c) exhibits a strong linear correlation to λ for $\lambda \geq 0$ is and is based on the routine `d2` from the popular *TISEAN* package for nonlinear time series analysis[2] [92]. This operation uses a time-delay $\tau = 1$, maximum embedding dimension $m = 10$, and no Theiler window. Since a flat region in the scaling curves of $h2$ is a signature of deterministic chaos, this algorithm searches for such an intermediate-scale flat region and returns the value that this flat region approaches as the embedding dimension m increases.

Box-Counting Dimension The operation plotted in Fig. 5.6(d) is a box-counting dimension estimate implemented through the `dimensions` algorithm from the *TSTOOL* package. We use 50 bins across each axis for a time-delay embedding of the time series with time-delay τ equal to the first zero-crossing of the autocorrelation function (a standard choice [13]) and embedding dimension $m = 3$. The algorithm returns the mean of $\log(N)$, across the full domain of $\log(r)$, where N is the number of boxes of length r arranged in a regular grid that contain at least one point [13].

Correlation Dimension The operation plotted in Fig. 5.6(e) is very similar to the box-counting dimension estimate plotted in Fig. 5.6(d). This is another *TSTOOL* algorithm that creates a time-delay embedding choosing the time-delay τ as the first

[2]Publicly-available code is here: `http://www.mpipks-dresden.mpg.de/~tisean/Tisean_3.0.1/index.html`

zero-crossing of the autocorrelation function for an embedding dimension $m = 5$ and 50 partitions per axis. Like the similar method in Fig. 5.6(d), this algorithm uses a box-counting approach and returns the mean of the outputs of `corrdim` at embedding dimension $m = 4$.

Approximate Entropy The operation plotted in Fig. 5.6(f) calculates the Approximate Entropy, ApEn(m, r) [2], of the time series for $m = 1$ and $r = 0.1$. As expected, the regular periodic time series with $\lambda \leq 0$ have an approximate entropy of zero, and as λ increases, ApEn(1,0.1) quantifies this variation approximately linearly.

Automutual Information Decay Like ApEn, the operation plotted in Fig. 5.6(g), is an entropy-based operation, implemented by us based on the idea of 'noise titration' [5]. In this algorithm, Gaussian-distributed noise with standard deviation η is incrementally added to the z-scored time series, over the range $\eta = 0, 0.1, ..., 2$. At each stage, the automutual information at lag one, AMI_1, is calculated using a histogram-based method with 20 equiprobable bins. The AMI_1 decays with the standard deviation of added noise, and we fit a (decaying) exponential to the variation, returning the decay rate as b from a fit $\mathrm{AMI}_1(\eta) = a \exp(b\eta)$, where η represents the standard deviation of added noise. The original noise titration paper [5] suggested their method as an empirical means of quantifying 'chaos intensity'. By implementing a similar idea here using a simpler method, high correlations are indeed observed between the output of our algorithm and the relevant 'chaos intensity' measure for these Logistic Map time series: the Lyapunov exponent λ. Here, the exponential decay in AMI_1 is greater for time series with higher λ. The behavior of this operation for this dataset is demonstrated in Fig. B.1.

Spectral Flatness Somewhat surprisingly, we find a linear time-series analysis method that performs relatively well at this task: a spectral flatness measure plotted in Fig. 5.6(h). This algorithm first estimates the power spectrum as a periodogram using the `periodogram` function from MATLAB's *Signal Processing Toolbox*, with a Hamming window, and then returns the spectral flatness measure, SFM:

$$\mathrm{SFM} = 10 \log_{10} \left[\frac{\text{geometric mean}(S)}{\text{mean}(S)} \right], \tag{B.1}$$

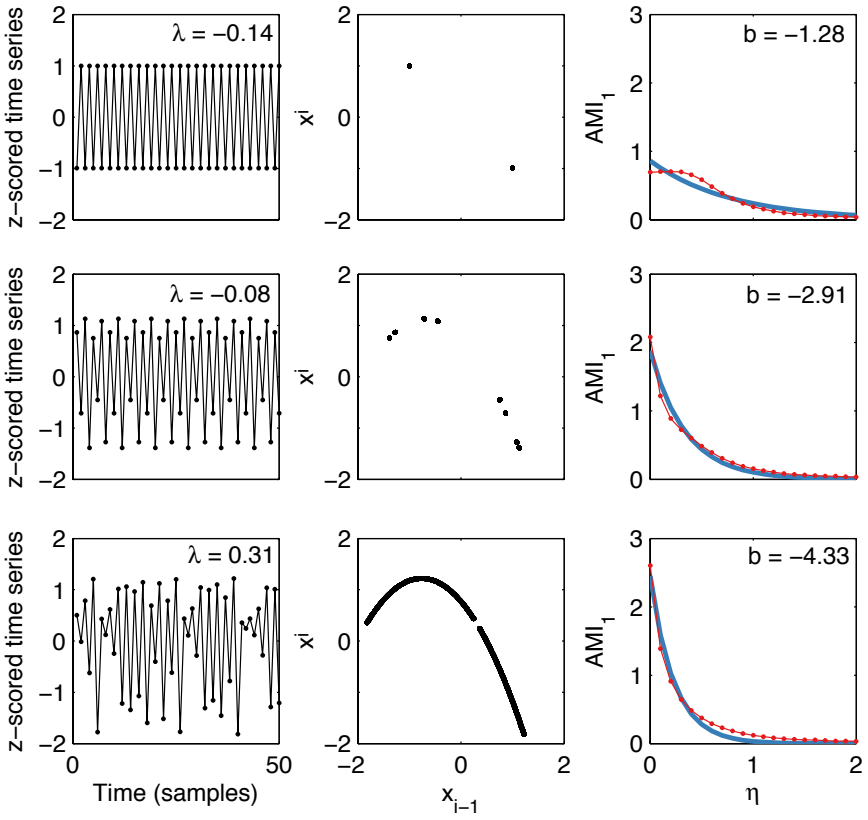

Figure B.1: **An operation that adds uncorrelated Gaussian-distributed noise to Logistic Map time series successfully predicts its Lyapunov exponent**. The operation, based on the idea of 'noise titration' [5], incrementally adds noise of variance η^2 to time series, measuring the automutual information at lag 1, AMI_1 at each stage, and fits a decaying exponential to the trend. This is demonstrated in the figure for three target Logistic Map time series, where λ is annotated to each plot on the left, that shows a 50-sample segment of the time series. Middle plots show the dependence of a time-series value x_i on the previous sample x_{i-1} across the time-series length. The estimation of AMI_1 is with histograms with 20 bins on each axis of these plots. The panels on the right show the measured AMI_1 as a function of η in red, along with the best fit of $\mathrm{AMI}_1(\eta) = a\exp(b\eta)$ in blue, where the fitted value of b is annotated. Because axis partitions are done by quantile, AMI_1 increases with λ, as there are more data points to infer the monotonic relationship Eq. (5.2) and hence the decay rate with added noise is increased. Note that the outputs from this operation are plotted against λ in Fig. 5.6(g).

where S is the power spectral density vector obtained from the periodogram. Although we did not expect linear time-series analysis operations to detect the variation in λ, the observed trend in the output of this metric makes sense: chaotic time series with high λ will produce disordered time series with many frequency components and hence a flatter spectrum with the geometric mean much closer to the arithmetic mean. Although this operation is sensitive to the variation in λ, the dependence is

non-monotonic, and time series with $\lambda \leq 0$ are not distinguished by this measure.

B.1.3 Unsupervised analysis of Logistic Map time series

As shown in Fig. B.2, we find that many operations are sensitive to the Lyapunov exponent of Logistic Map time series. In this figure, all time series are ordered by the control parameter, A, producing the familiar qualitative variation of λ as a function of A for the Logistic Map shown in Fig. B.2(a) (note that A is not evenly spaced through its domain: $A = 3, 3.2, 3.4$ and $A = 3.50, 3.51, ..., 4.00$). The reduced data matrix plotted in Fig. B.2(b), contains 45 operations (with less than 20% special-valued outputs and non-zero interquartile ranges for this dataset) from the set of 50 chosen by k-medoids on a representative set of empirical time series, excluding location-dependent, length-dependent, and spread-dependent operations (see Sec. 4.1.2). The qualitative results below hold for sets of operations as small as 10 and up to the usual 200, but here we choose 50 to demonstrate that these representative sets of operations can be chosen to be different sizes and still produce meaningful results. Changes in dynamical behavior of the time series produced from different parameters of the map, as indicated by λ in Fig. B.2(a), are clearly visible in the data matrix; indeed, the majority of the operations in this data matrix are sensitive to changes in λ. The similarity matrix in Fig. B.2(c) reflects the structure in the data matrix by plotting pairwise distances between all time series as the Euclidean distance between their 45-element feature vectors of operation outputs–columns in Fig. B.2(b). The qualitative changes in dynamical behavior with the control parameter A of these Logistic Map time series is visually evident in the data matrix, which shows characteristic signatures of both simple periodic ($\lambda \leq 0$) and chaotic ($\lambda > 0$) for these time series.

B.1.4 Self-affine operations

In this section we describe selected operations whose outputs vary with the scaling exponent, α, of self-affine time series. The main analysis is carried out in Sec. 5.1.3, and the variation of outputs from operations described in this section with the scaling exponent, α, is plotted in Fig. 5.9.

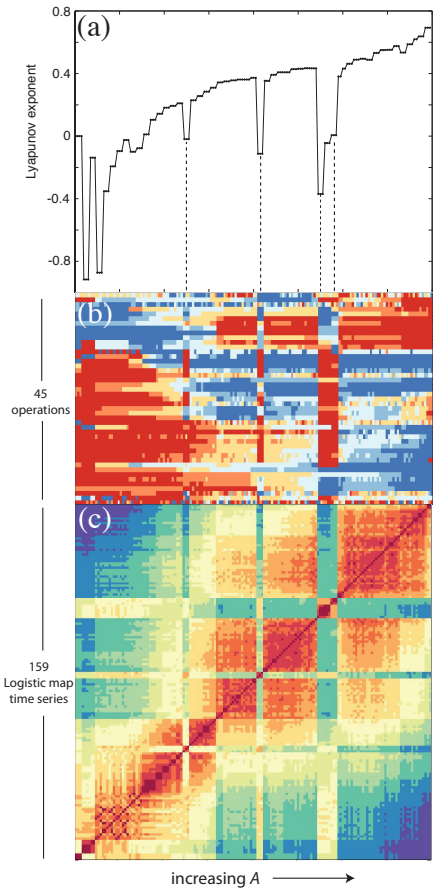

Figure B.2: **Unsupervised analysis of Logistic Map time series**. Changes in the dynamics of Logistic Map time series with the control parameter A can be seen in the data matrix using 45 representative operations. Time series have been ordered by the control parameter A [cf. Eq. (5.2)], which runs along the horizontal axis in all plots. (a) The Lyapunov exponent exhibits dips (of periodic behavior) with the increasing control parameter A. (b) Many of the operations are sensitive to the changes in dynamical behavior—changes in the Lyapunov exponent are reflected by characteristic changes in outputs of operations in the data matrix. Note that this data matrix is rotated relative to normal (i.e., time series are columns and operations are rows) so as to align with the other plots. Red indicates high values, and blue indicates low values of each operation. (c) A pairwise distance matrix between time series, from red (low) to blue (high). The main dynamical changes shown in (a) are also evident in the structure of this matrix.

Fluctuation analysis Figs. 5.9(a)–(e) correspond to operations based on fluctuation analysis. The operations shown in (a), (b), and (d) are implemented by us, and seek root-mean-square fluctuations at 25 logarithmically-spaced scales (from 5 samples to $N/4$, a quarter the length of the time series). Each of the operations shown in Figs. 5.9(a)–(d) return the estimated scaling exponent β as the gradient in the $\log\tau$–$\log F$ plot (where F represents fluctuations at a time-scale τ); methods (a), (b), and (d) estimate it using an iteratively re-weighted least squares linear fit implemented through the `robustfit` function from MATLAB's *Statistics Toolbox*. The detrended fluctuation analysis (DFA) methods in (a) and (d) use local quadratic detrending. The method shown in (c) is a widely-used publicly-available algorithm for measuring the detrended fluctuation analysis (DFA) exponent using local linear detrending[3]. All of these fluctuation analysis-based operations use the z-scored time series as input, and in (d) we have considered scaling in just the *sign* of the z-scored time series (i.e., in whether the time series is above the mean – encoded by 1, or below the mean – encoded by 0). Scaling in the magnitude and sign of time-series have been studied previously [182, 238]. The strongest linear correlation, $R^2 = 0.997$, to α is found for our implementation of DFA in (a), using quadratic detrending (although using cubic detrending yields the same $R^2 = 0.997$, and using linear detrending yields a similar $R^2 = 0.996$). The DFA scaling exponent, β, is related to the power spectrum scaling exponent, α, by $\beta \equiv 2\alpha - 1$ [146]. This theoretical linear relationship is confirmed empirically here in (a), (b), and (c). Apart from a flattening at low α and high variance at high α, the DFA exponent calculated from the sign of the z-scored time series shows good performance in Fig. 5.9(d), despite a dramatic reduction in the information contained in the time series: converting it to a binary string. The wavelet-based fluctuation analysis estimator of α, implemented using the `wfbmest` function from MATLAB's *Wavelet Toolbox*, performs very well on this task . The third output of this function, plotted in Fig. 5.9(e), uses linear regression on the log-log plot of detail vs. level variance.

Apart from these plotted fluctuation analysis methods, we also find good perfor-

[3]Coded by an author of this paper, Max A. Little. The code, *FastDFA*, is available for download `http://www.physics.ox.ac.uk/users/littlem/software/index.html`

mance for variants of these methods: e.g., using linear or cubic detrending for DFA, or using linearly-spaced scales (e.g., stepping τ in fixed increments of 25 samples), rather than logarithmically-spaced scales (e.g., stepping τ over 25 logarithmically-spaced values). It is apparent that operations measuring the scaling exponent using a range of fluctuation analysis-based methods exhibit strong linear variation with the known scaling exponent of 5 000 sample self-affine time series. Whether using detrended fluctuation analysis, a wavelet-based approach, using different orders of local polynomial detrendings, or the spacing of scales, τ, this family of algorithms is recognized as important for this task.

Model Fits Aside from fluctuation analysis-based methods, methods from many other areas of time-series analysis also perform well on this task. In particular, a number of model fit statistics, are plotted in Figs. 5.9(f)–(i): a summary statistic from an autoregressive model, shown in Fig. 5.9(f), a local mean predictor, shown in Fig. 5.9(g), a state-space model, shown in Fig. 5.9(h), and a Gaussian Process model, shown in Fig. 5.9(i).

In Fig. 5.9(f), the first parameter of a third-order autoregressive, AR(3) model, is estimated by the covariance method using the `arcov` function in MATLAB's *Signal Processing Toolbox*, and shows a strong linear correlation with α. As a linear time-series analysis measure, it performs particularly well on time series generated deterministically from the power spectrum (plotted as dots in Fig. 5.9). A consequence of the Wiener-Kintchine theorem is that the autocorrelation function of any real-valued stationary stochastic process is related to its spectrum by a Fourier transform [93]. Note that the first parameter of AR(1), AR(2), AR(3), AR(4), and AR(5) models are all very similar, and all perform similarly well on this task.

As shown in Fig. 5.9(g), a very simple predictor exhibits an approximately linear variation with α. This model uses the mean of the previous four values of the time series to predict the next value. By measuring the difference between the prediction and the actual time series value at that point, a time series of residuals is constructed. The operation plotted in Fig. 5.9(g) returns the linear autocorrelation coefficient at lag 1 of this time series of residuals. Residuals are positively-autocorrelated for fBM,

uncorrelated for $\alpha \approx 0$, and negatively-correlated for $\alpha < 0$, with an approximately monotonic variation across $-1 < \alpha < 3$.

Another example of a model fit-based metric is shown in Fig. 5.9(h), and involves fitting a state-space model to the time series using the function `n4sid` from MATLAB's *System Identification Toolbox*. This operation returns the coefficient k for noise input in an order 1 model. Highly monotonic variation is observed, but the variation is different for time series generated by the two methods, and the linear relationship breaks down at high α.

In Fig. 5.9(i), the output of a metric based on Gaussian Process regression [10] is plotted. This operation uses a covariance function that is the sum of squared exponential and noise terms. Twenty different contiguous 10-sample segments are selected uniformly from the time series (i.e., starting points of the segments are spaced linearly through the time series), and the natural logarithm of the best fit length-scale parameter of the squared exponential covariance term is calculated in each segment. The mean of this log-transformed hyperparameter across all 20 segments is returned as the output of this operation. Publicly-available code is used to perform the Gaussian Process regression[4]. Using segments longer than just 10 samples may improve the performance of this algorithm by reducing its variance, but it is interesting how well the outputs of this very simple local Gaussian process regression fit the scaling exponent, α, of the underlying time series.

Other Methods As well as fluctuation analysis and various time-series model-based metrics, other types of operations also perform well on this task. The operation shown in Fig. 5.9(j) takes 100 different contiguous 50-sample segments of the input time series, chosen at random, and calculates the autocorrelation at lag 1 in each segment. Its output is the standard deviation of this set of autocorrelation values. The variability in local estimates of autocorrelation increases with the scaling exponent α of these self-affine time series.

The next method, plotted in Fig. 5.9(k), is a new operation implemented by us

[4]Code written by Carl Edward Rasmussen and Hannes Nickisch. We use version 2.0 of the code; this, and newer versions are available at `http://www.gaussianprocess.org/gpml/code/matlab/doc/`

and involves summarizing the trajectory of an imaginary particle that moves in the time domain in response to the time series. The particle starts at zero and moves towards the value of the (z-scored) time series by an amount equal to 0.1 times the magnitude of the difference between the walker and the time series. This operation returns the number of times, as a fraction of the time series length, that the particle's trajectory crosses the time series. Intuitively, the highly-correlated fBM processes should have fewer crossings than the fGN series, but that the variation is monotonic, and could be used to reliably predict α for these time series, is an interesting result.

As a final example, we consider a measure based on time-series motifs, plotted in Fig. 5.9(l). This operation takes incremental differences of the input time series, and then converts it to a symbolic string using an equiprobable alphabet of three symbols: 'A', 'B', and 'C'. Partitioning of the time series is done by quantile such that 'A' refers to a point in the bottom third of time-series values, 'C' a point in the upper third, and 'B' a point in the intermediate third. An algorithm that measures the frequency of the string 'CACB' in the symbolic string is plotted in Fig. 5.9(l). Although the word 'CACB' is expected to be more probable in fGN time series than in highly-autocorrelated fBM time series, that the variation with α is linear across this range is an interesting result. Note that other similar motif measures, not plotted, also exhibit strong linear correlations with α: e.g., 'BCAC', 'ACAB', 'BACA', as do other types of motif-based metrics obtained from different symbolization methods (i.e., different ways of converting the time series to a symbolic string).

One might suspect that we are simply detecting trivial fixed-scale autocorrelation structure with this analysis. This is not the case. While the fixed-scale correlation structure does vary across this range of α, as shown in Fig. B.3 for the linear autocorrelation at lag 1, it saturates at $\alpha \approx 1$ and is therefore a relatively poor predictor of α.

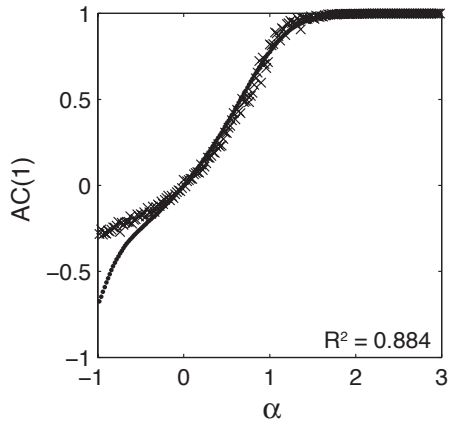

Figure B.3: **The autocorrelation at lag 1, AC(1), varies with** α. Each time series is represented as a point in this plot, and encompass two different generative procedures: the Fourier filtering method (dots) and the random midpoint displacement method (crosses) [90]. The variation of this simple quantity is approximately linear in the range $0 \lesssim \alpha \lesssim 1$, and is monotonic for each type of noise in the range $-1 \lesssim \alpha \lesssim 0$, but saturates at $\alpha \gtrsim 1.5$. The coefficient of determination, R^2, is labeled.

B.2 Classification

B.2.1 Uncorrelated and periodic time series

B.2.1.1 Data

In this section we describe in more detail the time series included in the dataset studied in Sec. 5.2.1. Random number sequences in the database were generated from different distributions (using the MATLAB *Statistics Toolbox* function indicated in parentheses in the following list): Beta distribution (`betarnd`), Binomial distribution (`binornd`), χ^2 distribution (`chi2rnd`), Extreme Value distribution (`evrnd`), F-distribution (`frnd`), Gamma distribution (`gamrnd`), Geometric distribution (`geornd`), Generalized Extreme Value distribution (`gevrnd`), Generalized Pareto Random Numbers (`gprnd`), Hypergeometric distribution (`hygernd`), Log-Normal distribution (`lognrnd`), Normal distribution (`normrnd`), Poisson distribution (`poissrnd`), Student's t Distribution (`trnd`), discrete Uniform distribution (`unidrnd`), continuous Uniform distribution (`unifrnd`), Weibull distribution (`wblrnd`), and Exponential distribution (`exprnd`). In each case a range of parameters for the distribution were used and sequences are of length 1 000, 5 000, or 10 000 samples. Time series labeled as 'periodic' in the database are mostly outputs from flows (i.e., sets of differential

equations), and sums of sinusoids. Those selected (at random from the database) for this task are: pure sinusoids, sums of two or three sinusoids (some with very small amounts of additive, Gaussian-distributed noise), outputs from the 'Rössler system', outputs from the 'Van der Pol Oscillator', outputs from the 'driven harmonic oscillator', Sprott's 'Jerk Equation 1' ($\dddot{x} + A\ddot{x} + \dot{x} - |x| + 1 = 0$), Sprott's 'Jerk Equation 2' ($\dddot{x} + A\ddot{x} + \dot{x} - |x| + 1 = 0$), Sprott's 'Flow G' ($\dot{x} = 0.4x + z$, $\dot{y} = xz - y$, $\dot{z} = -x + y$), Sprott's 'Flow I' ($\dot{x} = -0.2y$, $\dot{y} = x + z$, $\dot{z} = x + y^2 - z$), Sprott's 'Flow O' ($\dot{x} = y$, $\dot{y} = x - z$, $\dot{z} = x + xz + 2.7y$), the 'Duffing-van der Pol Oscillator', and 'Hadley circulation'. Details of all of these systems can be found in [12][5]. In each case, parameters and initial conditions that produce periodic outputs from these systems have been used (hence the 'periodic' label for them in the database).

B.2.1.2 Individual operations

In this section, selected operations that most effectively separate the noisy and periodic time series are described in detail. The operations described are those plotted in Fig. 5.12.

AMI decay The first algorithm, plotted in Fig. 5.12(a), progressively adds Gaussian-distributed noise of standard deviation η ranging from $\eta = 0, 0.1, ..., 2$ to the z-scored input time series, and measures the exponential decay of the auto-mutual information at lag 1, AMI_1, by fitting a function $\text{AMI}_1(\eta) = Ae^{B\eta}$ to the variation of AMI_1 across the range of η and returning B. This algorithm was implemented by us, using ideas from a method called 'noise titration' [5], and was discussed in Sec. 5.1.2.1 above. The random number sequences are uncorrelated and have a small AMI_1 before noise is added, and hence show minimal change with added noise and returning an output that is approximately zero. The periodic processes, on the other hand, exhibit a significant decay in AMI_1 with the addition of uncorrelated noise.

Sample Entropy change A metric with a similar idea is shown in Fig. 5.12(b), where we measure the mean change in the Sample Entropy, $\text{SampEn}(m, r)$ [50] for

[5]Three-dimensional chaotic flows are listed in Table 4.1, the simple chaotic Jerk systems are in Table 4.2, and all other systems are summarized in Appendix A

embedding dimensions $m = 1, 2, 3, 4$ and r fixed at $r = 0.15$. The random number sequences show minimal dependence on the embedding dimension, whereas the highly-correlated periodic signals show a decrease in SampEn as the embedding dimension increases.

Cross-prediction error The operation plotted in Fig. 5.12(c), uses the `nstatz` routine from the TISEAN package[6] to calculate cross-forecast errors of zeroth-order models for the z-scored time series embedded in a time-delay embedding space using a time-delay $\tau = 1$ and embedding dimension $m = 3$. The model is fit in each of 4 equally-spaced segments of the time series, and used to predict the other time series segments. The mean cross-forecast error is returned and plotted in Fig. 5.12(c). The random number sequences exhibit high prediction errors approximately equal to 1 (i.e., the variance of the z-scored input time series), whereas the periodic processes feature much lower average cross-prediction errors due to their stationary and informative autocorrelation structure.

AR(3) model fit In Fig. 5.12(d) we plot distributions of outputs from a variance estimate of white noise input to an AR(3) model fit using the `arcov` function in MATLAB's *Signal Processing Toolbox*. This algorithm very sharply distinguishes the purely periodic processes (a sharp peak in the plot at 0) from the uncorrelated processes (an output near 1).

Sinusoid fit Using MATLAB's *Curve-Fitting Toolbox* to fit a sum of 3 sine waves (via the `fit` function with the `'sin3'` input) and returning the goodness of fit, R^2, yields outputs near 1 for periodic processes and outputs near zero for the random number sequences, as shown in Fig. 5.12(e).

Power Spectrum Maximum In Fig. 5.12(f), we show the distribution of outputs from a Fourier spectrum-based measure. This particular metric returns the logarithm of the maximum in the periodogram spectrum (using the `periodogram` function in MATLAB's *Signal Processing Toolbox*), calculated using a Hamming window. The

[6] A popular publicly-available nonlinear time-series analysis package: `http://www.mpipks-dresden.mpg.de/~tisean/Tisean_3.0.1/index.html`

periodic time series show large peaks in the Fourier amplitude spectrum, whereas the uncorrelated random number sequences have flatter spectra and hence lower values for this metric.

B.2.2 EEGs

In this section we demonstrate that the structure in the reduced representations of the EEG dataset shown in Fig. 5.22 is not dependent on using large numbers of operations. Reduced sets of operations could be used instead: as representative sets of the full database using the methods described in Sec. 4.1.2. To demonstrate this, network representations of the dataset were plotted using reduced sets of operations of different sizes, as shown in Fig. B.4. Even using a few as ten operations, the structure of the dataset, which groups healthy EEGs, epileptic EEGs, and separates seizure recordings, is maintained.

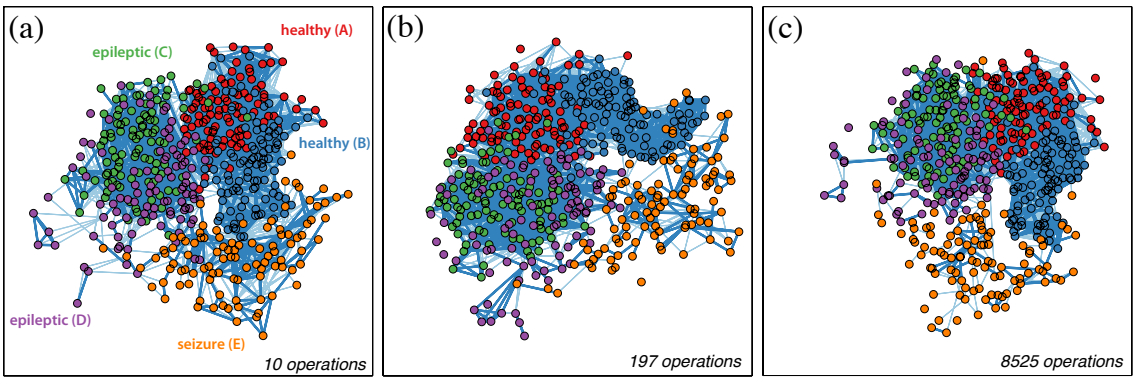

Figure B.4: **Structure in the EEG dataset is maintained using different reduced sets of operations.** Sets of operations (excluding length-dependent operations) were constructed using k-medoids clustering of operations based on their behavior on an interdisciplinary set of signals, as described in Sec. 4.1.2, for : (a) 10 operations, (b) 200 operations (of which 197 had fewer than 20% special-valued outputs on this dataset), and (c) all 8 525 operations with fewer than 20% special-valued outputs on this dataset. The colors representing the five classes of EEGs are labeled in (a), with the set label in parentheses, and are described in more detail in Tab. 5.1. Pairwise distances between all time series were calculated in each case and used to construct a fully-weighted network, where the strongest 2% of links are plotted with thick dark links, and the next strongest 3% of links are plotted with light lines. In each case, the qualitative structure between the five classes of data is maintained: sets A and B are grouped, sets C and D are grouped, and set E is separated.

B.2.3 Parkinsonian speech signals

In this section we present additional results relating to the material presented in Sec. 5.2.5.

B.2.3.1 Principal Components projection

A two-dimensional Principal Components projection of the data matrix (containing the 8 126 operations with no special-valued outputs and a nonzero interquartile range on this dataset) is shown in Fig. B.5. Although it has been useful in separating different classes of time series in classification tasks examined above, the two classes are not distinguished here: the $50 \pm 6\%$ 10-fold cross-validation linear classification error rate in this two-dimensional space is consistent with chance. As may be visually expected from the lower panels of Fig. 5.25, the distinction between these healthy and Parkinsonian speech recordings is subtle and requires the use of supervised learning methods to determine which types of algorithms, if any, are able to effectively distinguish the dynamics present in these two types of signals.

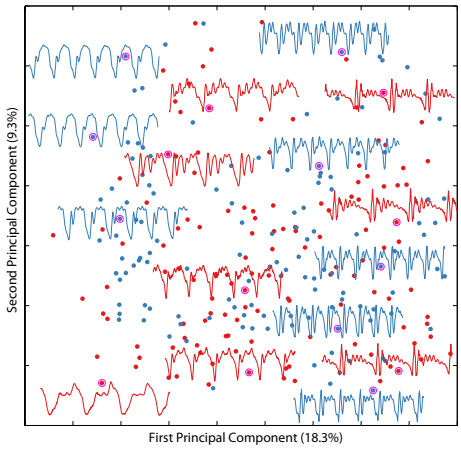

Figure B.5: **A two-dimensional Principal Components projection of the speech data shows no clustering structure indicative of the known classification of the signals.** The two groups: patients with Parkinson's disease and healthy controls, are plotted as red and blue dots, respectively. Selected time series segments of length 500 samples (or 31 ms) are annotated. The 10-fold cross-validation linear misclassification rate is $50 \pm 6\%$ in this space.

B.2.3.2 Individual operations

In this section we describe the most successful operations for distinguishing Parkinsonian speech phonemes from those of healthy controls. The operations are plotted in the main text, in Fig. 5.27.

Forced particle motion The operation in our library with the lowest classification error on this dataset, $20 \pm 7\%$, is a new type of time-series analysis operation introduced by us in this work; distributions of its outputs across both classes are plotted in Fig. 5.27(a). This operation involves simulating a particle that starts at the same initial position as the time series and moves in response to the time-series values at each point in time. Note that the time series is first z-scored to remove any mean or variance dependence of this operation. The discrete equation of motion for the particle is

$$w_{i+1} = \left[w^0_{i+1} + \frac{1}{m}(x_{i+1} - w^0_{i+1}) \right] \frac{\tilde{\sigma}_x}{\tilde{\sigma}_w}, \tag{B.2}$$

where w represents the particle position, x represents the time series, m is proportional to the 'mass' of the particle, $\tilde{\sigma}$ represents the local standard deviation, and w^0_{i+1} is the inertial position of the particle:

$$w^0_{i+1} = [w_i + (w_i - w_{i-1})] = 2w_i - w_{i-1}. \tag{B.3}$$

This quantity, w^0, corresponds to the position of the particle in the absence of any forcing or $\tilde{\sigma}$-scaling of the particle from the time series. The quantity within the square brackets of Eq. (B.2) is derived from Newtownian dynamics: $s = u\Delta t + 1/2 a \Delta t^2$ with $\Delta t = 1$ and $a = F/\tilde{m}$, for a displacement, s, velocity u, acceleration a, and force F, and mass $\tilde{m}$ (note that we absorb the factor of two into the mass m above). It corresponds to the motion of a particle under the influence of a force equal to the difference between the time series value and the position of the particle at each point in time; the first term in Eq. (B.2) is the inertial term (i.e., in the absence of any forcing: $F = 0$), the second is the forcing term due to the influence of the time series.

For this operation, we use $m = 3/2$ (which approximately pushes the particle from its inertial position towards the time series, to narrow the gap by $1/m = 2/3$). The additional factor expressed as a ratio, $\tilde{\sigma}_x/\tilde{\sigma}_w$, scales the motion according to the local

standard deviation of the time series, $\tilde{\sigma}_x$, measured over the past q values of the time series, x. This operation uses $q = 50$ samples. Thus, if the local standard deviation of the walker, $\tilde{\sigma}_w$, is different to that of the time series, $\tilde{\sigma}_x$, this factor, $\tilde{\sigma}_x/\tilde{\sigma}_w$, rescales the particle's motion in an attempt to equalize them. The algorithm therefore has two parameters: the mass of the particle, $m = 3/2$, and the window length, $q = 50$, for calculating local standard deviations $\tilde{\sigma}$. The output of this operation is the mean of the particle's trajectory across the whole time series. The particle trajectory can be thought of as a type of smoothing filter for the time series.

Since this algorithm takes the z-scored time series as input, a particle that perfectly followed the time series would have a mean of zero, like the z-scored time series itself. As shown in the annotated distributions of Fig. B.6, particle trajectories following Parkinson's disease speech signals have higher means that those following phonemes recorded from healthy controls. A detailed analysis of the mechanism of this operation is illustrated in Fig. B.7. It is clear that the particle's trajectory is sensitive to the local shape of the time series, which cause it to overshoot local extremums. Small overshoots are compounded across the length of the time series and give rise to a net change in the mean of the particle compared to that of the original time series. It appears that the local shape of healthy control speech recordings are such that the particle tends to overshoot sudden dips in the time series, resulting in a negative mean across the trajectory, whereas for patients with Parkinson's disease, the particle is more likely to overshoot peaks in the time series, resulting in a net positive mean across the trajectory.

Other particle trajectory-based operations in our library also show good performance on this dataset – one simply returns the maximum value of the trajectory of a particle that moves 10% closer to the time series at each point in time, which has a misclassification rate of $24 \pm 6\%$. Another operation considers a particle that moves 50% closer to the time series when it increases, and just 10% when it decreases, and outputs the sum of absolute differences across 200-sample kernel-smoothed distributions of values for the time series and for the particle's trajectory ($26 \pm 13\%$ error). The particle motion defined by Eq. (B.2), and described above, however, performs considerably better than these two variants, with a misclassification rate of $20 \pm 7\%$

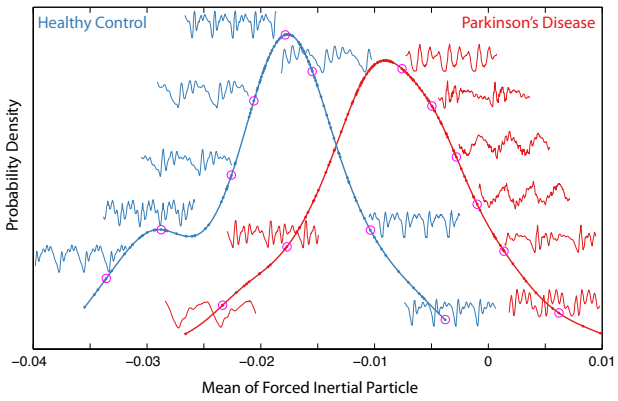

Figure B.6: **Annotated distribution of outputs from a forced particle motion operation**. The algorithm returns the mean of a simulated forced inertial particle following the time series, as defined by Eq. (B.2), and produces higher outputs in response to phonemes recorded from patients with Parkinson's disease than for healthy controls. The plot is a more detailed version of Fig. 5.27(a), in which selected points have been annotated with a 300-sample (19 ms) subsegment of the corresponding time series. Differences between time series with low and high values of this metric are subtle, but involve the particle overshooting different parts of the signal due to the signal shape, as demonstrated in Fig. B.7.

– the best individual performance of all algorithms in our library.

The success of these particle simulation-based operations indicate that differences in the shape of local patterns are important for distinguishing Parkinsonian speech signals. It is striking that the best performing individual operation for this task, by a robust margin, involves simulating the forced motion of an imaginary particle – a novel operation introduced by us in this work. This example demonstrates how the construction of our database of operations allows us the freedom to be creative with new types of operations, whose utility are automatically highlighted on particular, and often unexpected, time-series analysis tasks.

Asymmetry measures Algorithms that measure time-series asymmetry above and below the mean are also useful for this task, including a measure of local extreme values, plotted in Fig. 5.27(f), and a simple outlier-trimmed mean, plotted in Fig. 5.27(h). The first of these measures, plotted in Fig. 5.27(f), first takes 100 non-overlapping windows equally spaced throughout the time series and returns the maximum and the minimum value in each window. The output of the operation is the ratio of the median of the windowed maximums to the median of the magnitude of

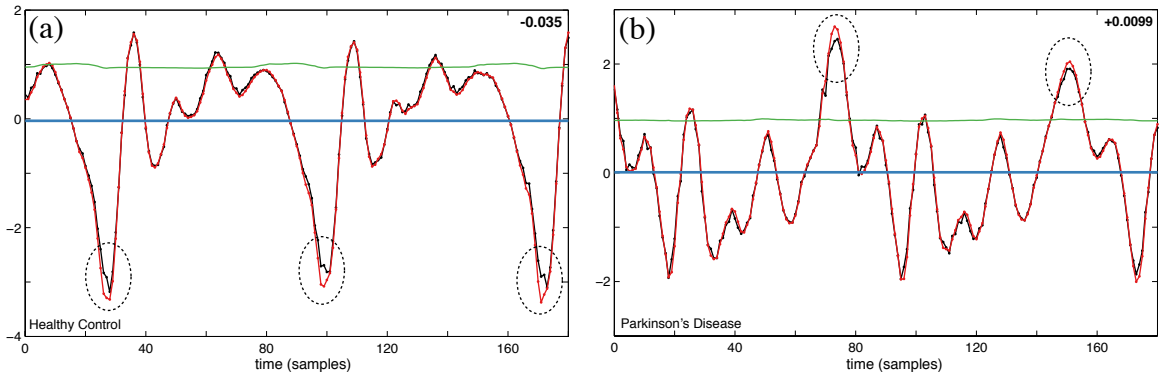

Figure B.7: **Forced inertial particle trajectories**. A forced inertial particle operation, defined by Eq. (B.2), distinguishes the Parkinson's disease speech signals from those of healthy controls with high accuracy. We show (a) the time series in this dataset with the lowest output from this operation (a healthy control), and (b) the time series with the highest output for this metric (a patient with Parkinson's disease). We plot the original time series (black), the forced particle's trajectory (red), and the local standard deviation factor $\tilde{\sigma}_x/\tilde{\sigma}_w$ (green), for a 180-sample (11.25 ms) subsegment of each time series. In (a), as highlighted by dashed ellipses, the particle overshoots large dips in the time-series, decreasing its mean value compared to the original time series. In (b), the phenome shape is such that these decreases do not occur; instead the particle slightly overshoots local maximums more than local minimums, causing a slight increase in its mean value compared to the original time series. By simulating these forced particles, this operation is able to quantify these subtle differences in shape between phonemes recorded from healthy controls and patients with Parkinson's disease.

the windowed minimums. Other similar measures perform well, including a measure that takes 25 windows instead of 100 ($28 \pm 9\%$ error), and another that takes 50 windows and returns the ratio of means rather than medians ($28 \pm 7\%$ error).

Another simple asymmetry measure, which is plotted in Fig. 5.27(h), takes in the z-scored time series, removes 2% of the highest values as well as 2% of the lowest values, and returns the resulting mean, which ought to be zero if extreme positive deviations match extreme negative deviations. As shown in Fig. 5.27(h), both Parkinsonian and healthy speech signals are biased towards positive values, indicating that the most extreme negative deviations are further from the mean than the most extreme positive deviations across time series in this dataset. This result is intuitive given the patterns in the time series shown in Fig. 5.25, for example. The selection of this operation shows that the effect is more pronounced for the healthy controls than for subjects with Parkinson's disease. A similar metric that removes the highest

and lowest 5% of values performs similarly, with a cross-validation error of $30 \pm 7\%$.

Nonlinear time-series analysis Some operations from nonlinear time-series analysis perform well on this dataset. One such operation, plotted in Fig. 5.27(b), computes local density estimates using the routine `localdensity` from the *TSTOOL* MATLAB Toolbox for nonlinear time-series analysis[7]. The operation, which achieves a misclassification rate of $25 \pm 6\%$ acts on a z-scored time series by first embedding it in a three-dimensional space, $m = 3$, using a time-delay embedding with a time delay $\tau = 1$, and then uses five nearest neighbors with a Theiler window of 40 samples to perform a local density estimate at each embedded point. The output of the operation is the first zero-crossing of the autocorrelation function of the constructed sequence of local densities.

Other measures derived from local density estimation, that return the spread of these local density estimates, also perform well. These measures use a different embedding: choosing the time delay, τ, from the first zero-crossing of the autocorrelation function, which is a standard choice [13], and the embedding dimension, m, using Cao's method (via the `cao` routine in *TSTOOL*). Returning either the standard deviation or interquartile range results in similar outputs, and both have a misclassification rate of $28 \pm 8\%$. Although derived from the same `localdensity` algorithm, these measures of spread return values that are relatively independent to the first zero-crossing metric plotted in Fig. 5.27(b). Note that the time-scale operation is labeled as 'local density: scale' in Fig. B.8, whereas these spread-based operations are labeled as 'local density: spread'.

Another nonlinear time-series analysis measure, plotted in Fig. 5.27(g), is based on the TISEAN[8] routine `c2g`, which uses a Gaussian kernel to compute the correlation integral of the time-delay embedded time series in spaces of dimension $m = 1, 2, 3, .., 10$, with a time delay, τ, given by the first zero-crossing of the autocorrelation function, and a Theiler window equal to 1% of the time-series length. From the scaling region, the algorithm returns the maximum correlation dimension from those estimated at

[7]This package is freely available at `http://www.physik3.gwdg.de/tstool/index.html`

[8]A popular nonlinear time-series analysis package: `http://www.mpipks-dresden.mpg.de/~tisean/Tisean_3.0.1/index.html`

embedding dimensions $m = 1, 2, 3, ..., 10$, which is greater for recordings obtained from patients with Parkinson's disease. This result is consistent with a previous report showing that a related dimension estimate, D_2, is greater in speech recorded from patients with Parkinson's disease [169], although in another study by Little et al. [91], the measure was found to be unreliable due to its high sensitivity to added noise.

Linear model fits An illuminating model fit for this data involves the ARMA (autoregressive moving average) model [239]. An operation summarizing a fit to this model with a $23 \pm 6\%$ misclassification rate is shown in Fig. 5.27(c). This operation uses the `armax` function in MATLAB's *System Identification Toolbox* to fit an ARMA(2,2) model to the first half of the time series. This fitted model is then used to incrementally predict the second half of the time series using one-step-ahead forecasting. Autoregressive models, AR(p), of order $p = 1, 2, ..., 10$, are then fitted to the residuals from this prediction phase using the *ARfit* package[9]. The output of the operation is the minimum Schwarz's Bayesian Criterion (SBC) [93] across these ten AR models. If residuals are well-fit by AR models, this indicates that the model is not fitting the time-series well, as basic linear structure remains in the residuals, which are ideally uncorrelated. This metric is therefore measuring the appropriateness of ARMA(2,2) models for these time series – it appears that there is much more linear structure remaining in residuals of the model fit to Parkinsonian speech signals, which feature lower SBC values of AR fits to ARMA(2,2) residuals. A similar operation, that fits ARMA(3,1) models, instead of ARMA(2,2) models, yields similar results, with a misclassification rate of $28 \pm 7\%$. Also, for both models, measuring Akaike's Final Prediction Error (FPE) [93] instead of the SBC results in the same misclassification rates.

Stationarity A simple stationarity measure, StatAv, is plotted in Fig. 5.27(d). This operation divides the z-scored time series into non-overlapping segments of length 150 samples (9.4 ms), and calculates the mean in each segment. It then returns the standard deviation of this set of local means. Since the period of each oscillation of these speech time series is approximately 75 samples, or 4.7 ms (see Fig. B.7),

[9] Available at `www.gps.caltech.edu/~tapio/arfit/`

the mean in most segments is close to that across a period of oscillation, which is close to zero such that this operation outputs low values for most time series. The non-stationarity/irregularity in oscillation period for the Parkinson's disease data is picked up by this operation, which assigns greater StatAv values on average to the Parkinson's disease speech signals. Note that this operation is qualitatively similar to traditional speech analysis measures known as *jitter* and *shimmer*, that measure variation in fundamental frequency and amplitude between vocal cycles, respectively [91]. Jitter was found to be a useful operation in a number of studies of this data [91, 169, 171].

Summaries of the Fourier spectrum Some operations that measure scaling in fluctuations of the time series about its mean are effective for this classification task. An example of such an operation is plotted in Fig. 5.27(e). This operation takes magnitudes of logarithmic deviations from the mean of the z-scored time series, produces a power spectrum from this transformed time series as a periodogram using a Hamming window, then returns the gradient of an iteratively re-weighted linear fit to the log-log spectrum. Thus, this operation measures long-range scaling in (logarithmic) absolute fluctuations of time series about their mean. Related operations also exhibit strong classification performance, including those that perform the linear fit on only the middle 50% of frequencies ($26 \pm 5\%$ classification error), and others that return the ratio of the root-mean-square error of the linear fit from ordinary least squares regression to that obtained from iteratively-reweighted least-squares regression ($26 \pm 9\%$ classification error). Note that operations measuring fractal scaling have been used with success on this task in an existing study [152]; here the relevance of this type of operation is detected automatically through our highly-comparative framework.

Another Fourier-spectrum-based method involves a measure of low-frequency amplitudes in the spectrum of logarithmic magnitudes of deviations from the mean. This measure is plotted in Fig. 5.27(i), and returns the sum of amplitudes in the lowest 20% of represented frequencies. A similar measure that retrieves the lowest 25% of amplitudes behaves similarly, with a 10-fold cross-validation error of $29 \pm 8\%$. Note

that our measures have no knowledge of the sampling rate of these recordings, and hence cannot produce estimates of fundamental frequency or use standard estimates of power in different frequency bands, which are known to be useful for this task [167, 168, 171], as are the cepstrum coefficients [171]. Nevertheless, the usefulness of spectral measures is identified automatically from our library of operations via our highly-comparative analysis.

B.2.3.3 Structure in the set of top operations

In the Sec. 5.2.5.2, operations that are successful at distinguishing Parkinsonian speech signals were automatically retrieved from our library of time-series analysis operations. These results could be used as a starting point for a time-series analyst working on this problem to understand how to link the behavior of these operations to physiological knowledge of the processes affecting speech in individuals with Parkinson's disease. Alternatively, the methods could remain heuristics whose parameters could be optimized for the clinical aim of classifying new recordings. However, given this large assortment of different time-series analysis operations, which of them are independently measuring unique and useful properties of the time series, and which are simply reproducing the behavior of others? In this section, we structure this set of operations in a way that allows us to visualize the correlations between them.

Linear correlation coefficients measured between all pairs of the 23 individual operations with a mean 10-fold misclassification rate less than 28% over 10 repeats are shown in Fig. B.8. The various types of operations were summarized in Sec. 5.2.5.2 above (and in more detail in Sec. B.2.3), and include ARMA model goodness of fit statistics, scaling in the power spectrum of logarithmic deviations from the mean, extreme value structure, mean asymmetry, stationarity, and nonlinear time-series analysis methods. Unlike the top operations for the heart rate variability dataset, analyzed in Sec. 5.2.6.4, most of these operations are linearly uncorrelated to outputs of other successful operations, suggesting that there are a variety of different ways of probing informative structure in this dataset. This result is consistent with existing findings [144]. We consequently expect that building classifiers using combinations of operations will allow us to draw on these multiple sources of difference between

the two classes simultaneously and hence improve classification performance on this dataset, as will be explored in the next section.

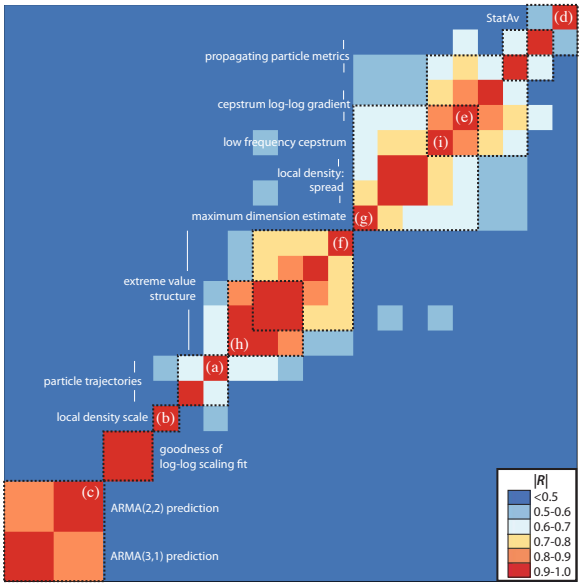

Figure B.8: **Linear correlation coefficients between pairs of successful operations for distinguishing Parkinsonian phoneme recordings.** We consider the 23 operations with less than a 28% mean 10-fold linear misclassification rate. Color indicates the magnitude of the linear correlation coefficient between pairs of features, from red, $|R| > 0.9$, to blue, $|R| < 0.5$. Groups of similar operations are highlighted manually with dotted squares, and operations are briefly labeled to illustrate the rich variety of approaches to time-series analysis with utility for this task. These operations are described in detail in Sec. 5.2.5.2. Operations whose distributions are plotted in Fig. 5.27 are labeled here by their corresponding subplot in Fig. 5.27.

B.2.3.4 Classifiers using pairs of operations

In this section, we construct linear classifiers combining pairs of operations using an exhaustive search across all possible pairs of operations, in the same way that all individual features were compared in Sec. 5.2.5.2. All possible combinations of 8 399 individual features amount to $8\,399 \times (8\,399\text{-}1)/2 = 35\,267\,401$ pairs. Linear classification errors were calculated for all of these combinations, as well as for shuffled permutations of them, and the results are shown in Fig. B.9(b).

Classifiers that combine pairs of operations are easily visualized in a two-dimensional space. Some of the most successful two-feature linear classifiers are shown in Fig. B.10. These are obtained from the left tail of the (red) distribution in Fig. B.9(b), which extends down to a misclassification rate of 13.4%. In principle, it is possi-

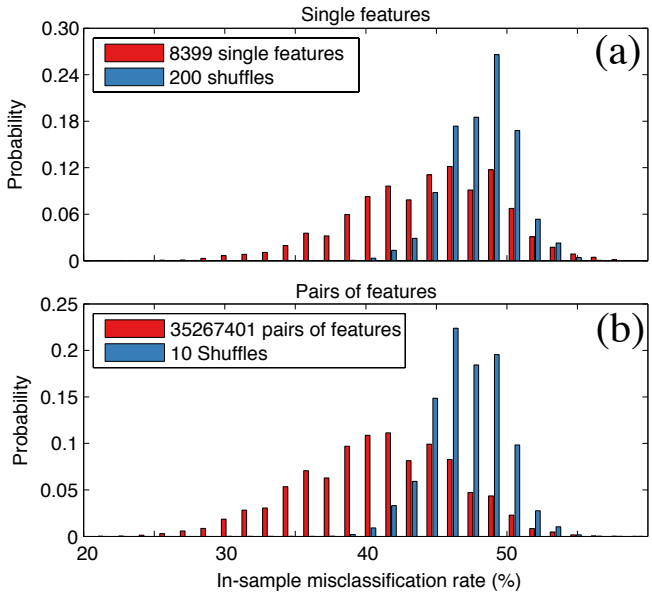

Figure B.9: **Histograms of misclassification rates for single-feature and two-feature classifiers.** Misclassification rate distributions are plotted for (a) all 8 399 individual operations, and (b) all 35 267 401 pairwise combinations of them. The in-sample bias of linear classifiers is slightly increased for pairs of features over single features, as seen by the mean of the shuffled distribution (blue), which decreases from approximately 48% for single operations to approximately 47% for pairs of operations. It is clear in both cases that many linear classifiers measure structure in these phoneme recordings that corresponds to real differences between Parkinsonian and healthy signals.

ble to produce successful classifiers from pairs of operations that are individually unsuccessful but work well together [105], but for this case study we observe that the most successful two-feature classifiers contain at least one operation that is itself individually successful.

An example of a linear classifier that combines the outputs of two operations is plotted in Fig. B.10 and contains the successful ARMA(2,2) model fit operation shown in Fig. 5.27(c) and explained above, which has a 10-fold linear misclassification rate of $23 \pm 5\%$. This operation is combined with the output of a Taken's estimator of the correlation dimension[10], which has an individual classification error of $34 \pm 12\%$. By using these two features in combination, the 10-fold cross-validation classification

[10]Calculated using the *TSTOOL* package: a publicly-available set of MATLAB code for nonlinear time-series analysis available at `http://www.physik3.gwdg.de/tstool/index.html`. This operation using all reference points, a maximum search radius equal to 5% of the length of the time series, a Theiler window also equal to 5% of the length of the time series, the time-delay τ chosen as the first minimum of the automutual information function, and an embedding dimension $m = 3$.

error is reduced to $15 \pm 6\%$. In words the plot can be interpreted as follows: when the minimum SBC of the AR fit to ARMA(2,2) residuals is high and the Taken's dimension estimate is low, speech recordings are classified as healthy.

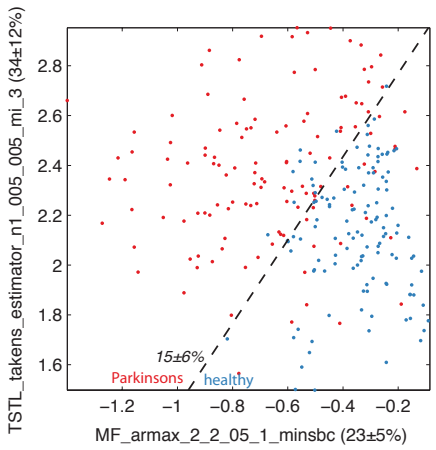

Figure B.10: **An example of a successful linear classifiers combining a pair of operations**. Parkinsonian speech signals are plotted as red dots and healthy controls are represented as blue dots. The optimal linear classification boundary is plotted as a dashed line and the 10-fold cross validation linear classification error is annotated. Both operations are denoted by its database name, with its individual 10-fold cross validation classification error in parentheses.

Other classifiers that combine two operations include one that combined a particle trajectory-based operation with a motif frequency operation to achieve a 10-fold cross validation misclassification rate of $14 \pm 6\%$, and another that combined a stationarity measure with a measure of the spread of local extremums in the time series which achieves a liner 10-fold cross validation misclassification rate of $15 \pm 7\%$. Our method allows operations developed in different areas of time-series analysis to be combined automatically to produce classifiers that achieve high classification accuracies. In some cases operations are combined that have low individual classification accuracies. Successful two-feature classifiers for this problem combined all sorts of operations, including a particle-based trajectory measure, local motif, higher-order distributional moments, and stationarity measures at different time scales. The most accurate classifiers tend to measure multiple different properties of the time series simultaneously and combine the result to produce a more accurate and robust classifier.

B.2.4 RR intervals

In this section we present additional information relating to the RR-interval classification dataset studied in Sec. 5.2.6.

B.2.4.1 Individual operations

In this section we describe operations that are successful in classifying normal sinus rhythm RR interval series from those of congestive heart failure patients. The problem is analyzed in detail in 5.2.6.3, and the operations described here are plotted in Fig. 5.33.

Measures of location As shown in Fig. 5.33(a), measures of location like the mean are powerful at distinguishing the classes; average heart beat intervals are greater for normal sinus rhythms. A large number of related metrics exhibit similar performance, including trimmed means (means after removing some proportion of outliers from the distribution), the harmonic mean, and the median. This basic finding, that mean RR intervals are significantly larger for healthy patients is established [57].

Outliers In Fig. 5.33(b), a novel measure introduced by us in this work is plotted that exhibits a high linear classification accuracy, $89 \pm 7\%$ for this dataset. The operation removes from the time series 10% of points that are furthest from the mean, and then calculates the linear autocorrelation at lag 3, AC(3). The output of this operation is the AC(3) of this outlier-stripped series as a fraction of the AC(3) of the original RR interval series. This is a strange measure because for the truncated time series, the time-ordering of points is distorted according to the temporal distribution of outliers in the time series. In Fig. 5.33(b), we see that the AC(3) after removing outliers is generally higher for the congestive heart failure time series. The magnitude and temporal distribution of anomalous beats is a significant source of difference between the two classes of recorded RR interval sequences that is effectively captured by this algorithm.

Entropy Existing studies in the literature have found increased entropy in RR interval sequences measured from healthy patients than in those obtained from patients

with congestive heart failure [57, 61, 178]. The output of our highly-comparative analysis reiterates this finding, by automatically retrieving a number of entropy-based operations.

An algorithm that estimates the Sample Entropy [50] of the differenced time series using an embedding dimension $m = 4$ and radius $r = 0.1$, is plotted in Fig. 5.33(c). Applying Sample Entropy to differenced time series has been suggested previously in the literature under the name 'Control Entropy' [63]. The success of this approach for these HRV time series is demonstrated with this algorithm, which has a misclassification rate of $15 \pm 8\%$, and other control entropies with different parameter sets, including $m = 3$, $r = 0.1$, and $m = 2, r = 0.1$ perform similarly (both with a $15 \pm 7\%$ misclassification rate).

The operations plotted in Figs. 5.33(f) and 5.33(l) measure predictability of the sequences by first converting them to a binary string of incremental increases (coded as 1) and decreases (coded as 0). Increases in this case correspond to RR intervals that are longer than the interval that preceded it. The operation in Fig. 5.33(f), measures the entropy of three letter words in this alphabet. The operation plotted in Fig. 5.33(l) records the lengths of strings of consecutive 0s (i.e., successive decreases of RR intervals), and outputs the standard deviation of this set of lengths. The spread of these lengths is less in congestive heart failure series, indicating a greater degree of predictability in them. Complexity measures derived from symbolic dynamics have been used with success to analyze RR interval sequences in the past [40, 177, 179, 183, 184].

Each of these entropy measures judges normal sinus rhythm as having higher entropy than congestive heart failure series, consistent with prior studies [57, 61, 178]. At this point, we also notice a theme appearing: operations acting on the differences between successive RR intervals, ΔRR, frequently occur in our list of useful operations. This trend will be discussed at the end of this section.

Model Fits In Fig. 5.33(d), an algorithm based on a nonlinear prediction algorithm developed by Michael Small[11] is plotted that has a 10-fold cross-validation misclassifi-

[11]The original code on which this algorithm is based, `nlpe.m`, is publicly-available at `http://small.eie.polyu.edu.hk/matlab/`

cation rate of $13 \pm 7\%$. The operation uses an embedding dimension $m = 3$ and time delay τ determined by the first zero-crossing of the linear autocorrelation function (a standard choice for τ [13]) to construct a time-delay embedding up to the first 3 000 samples of the RR interval sequences. Its output measures the Gaussianity of residuals from the nonlinear prediction routine by returning the test statistic from a Kolmogorov-Smirnov test. The test statistic approaches zero for Gaussian-distributed residuals, and hence we discover that residuals from congestive heart failure heart beat interval series are less Gaussian than those from normal sinus rhythms.

In Fig. 5.33(i), the distributions of outputs from an operation that measures the first coefficient of an AR(2) model fit to the time series (using the `arcov` function in MATLAB's *Signal Processing Toolbox*) are plotted. The fitted coefficients are more negative for normal sinus rhythms than congestive heart failure RR intervals.

Scaling Operations that estimate the long-range scaling behavior of RR interval series are effective for this task. Examples are plotted in Fig. 5.33(g), which using the second order discrete derivative to estimate the scaling exponent (via the MATLAB *Wavelet Toolbox* function `wfbmest`), and Fig. 5.33(k), which uses an iteratively re-weighted least squares linear fit (using MATLAB's *Statistics Toolbox* function `robustfit`) to a log-log plot of the power spectrum estimated using a periodogram with a Hamming window. This is consistent with many papers in the literature that use differences in measured scaling exponents to classify pathological RR interval series [127].

Autocorrelation Operations measuring various autocorrelation-based properties can distinguish these two classes of heart beat intervals. The operation in Fig. 5.33(j) calculates the *generalized self-correlation function* [186] using $\alpha = 1$, $\beta = 10$ and $\tau = 5$. Another example of this is shown in Fig. 5.33(e), where the target operation simply returns the autocorrelation at lag 1 of the differenced time series.

pNNx As well as these various general time-series analysis operations, a classic HRV statistic, PNNx, is plotted in Fig. 5.33(h). This operation calculates the proportion of successive RR intervals that differ by at least a duration x (typically measured

in milliseconds) [174]. Many PNNx measures show good behavior for this dataset, the one plotted in Fig. 5.33(h) is a slight variant of PNNx that z-scores the RR interval series and then applies the metric, using the threshold $x = 0.005$. The same misclassification rate, $17 \pm 7\%$ is obtained from other PNNx operations (also operating on the z-scored interval series such that x now represents a fraction of the standard deviation, σ) with $x = 0.01, 0.02, 0.03$, and 0.04. The classic PNNx measures, that operate on the non-z-scored RR interval sequences, also exhibit good performance, e.g., $17 \pm 7\%$ for $x = 5$, and $18 \pm 8\%$ for $x = 10$.

B.2.4.2 Building classifiers

Having used our library of operations to understand informative sources of difference between the two classes of RR interval sequences, we now show how operations can be combined to construct useful classifiers. We report results using greedy forward feature selection using 500 repeats, as described in Sec. 3.4.5. The feature selection was performed twice, using two different sets of operations: (i) all 7 672 operations, and (ii) excluding location-dependent operations, leaving a set of 7 560 operations. Test set misclassification rates across the 500 iterations of this procedure [using different partitions of the data into training (80%) and test (20%) sets] are plotted in Fig. B.11. Misclassification rates are reduced when location-dependent operations are included, reflecting the usefulness of these operations for this task. However, trends for the two operation sets are similar: a modest improvement in classification accuracy is gained by using pairs of operations in combination over single operations, but subsequent features provide minimal out-of-sample improvement. When location-dependent operations were included, a location-dependent operation was selected first for all 500 repeats of the procedure. Whereas, when location-dependent operations were excluded, our novel measure based on the autocorrelation ratio from removing outliers, plotted in Fig. 5.33(b), and described in Secs. 5.2.6.3 and 5.2.6.4, was selected in 411 out of 500 runs of the procedure, or 82.2% of the time, confirming its robust usefulness for this dataset in the absence of location-dependent operations.

In each iteration of the feature selection process, a classifier is learned. Two-feature classifiers obtained from this process (with location-dependent operations excluded),

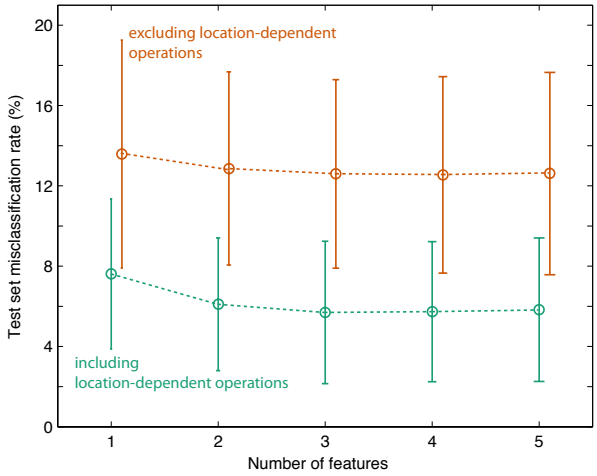

Figure B.11: **Misclassification rate as a function of the number of features for greedy forward feature selection with a linear classifier**. Distributions are obtained from 500 repeats of the procedure, using 20% of the data for testing, and are summarized in this plot by their mean (circles) and standard deviations (error bars). We show results for two sets of operations: all 7 672 operations with no special-valued outputs on this dataset (green), and another set that excludes location-dependent operations: leaving 7 560 operations (brown). A modest improvement in classification accuracy is obtained by using pairs of operations in combination, but subsequent features provide minimal improvement.

are plotted and described in Sec. B.2.4.3.

B.2.4.3 Two-operation classifiers

In this section we describe some classifiers for the RR interval dataset that combine two operations. The two-dimensional classifiers were selected during individual iterations of the procedure described in Sec. B.2.4.2, and are plotted in Figs. B.12(a)–(c).

The first pair, plotted in Fig. B.12(a) consists of an operation that returns the mean change in RMS prediction errors across prediction lengths of 1 up to 6 samples, using an autoregressive, AR(p) model of order p chosen by Schwarz's Bayesian Criterion[12], to a maximum order $p = 10$. The performance of this measure is enhanced by adding a symbolic complexity measure: the probability the duration of successive RR intervals decreasing, then increasing, then increasing again. This measure performs in a similar way to other binary motif measures: 'DII', 'IID', 'DIII', and 'IIID' where 'I' stands for an increase, and 'D' represents an incremental decrease in the time series

[12]Code used is publicly-available: the *ARfit* package, available at `www.gps.caltech.edu/~tapio/arfit/`

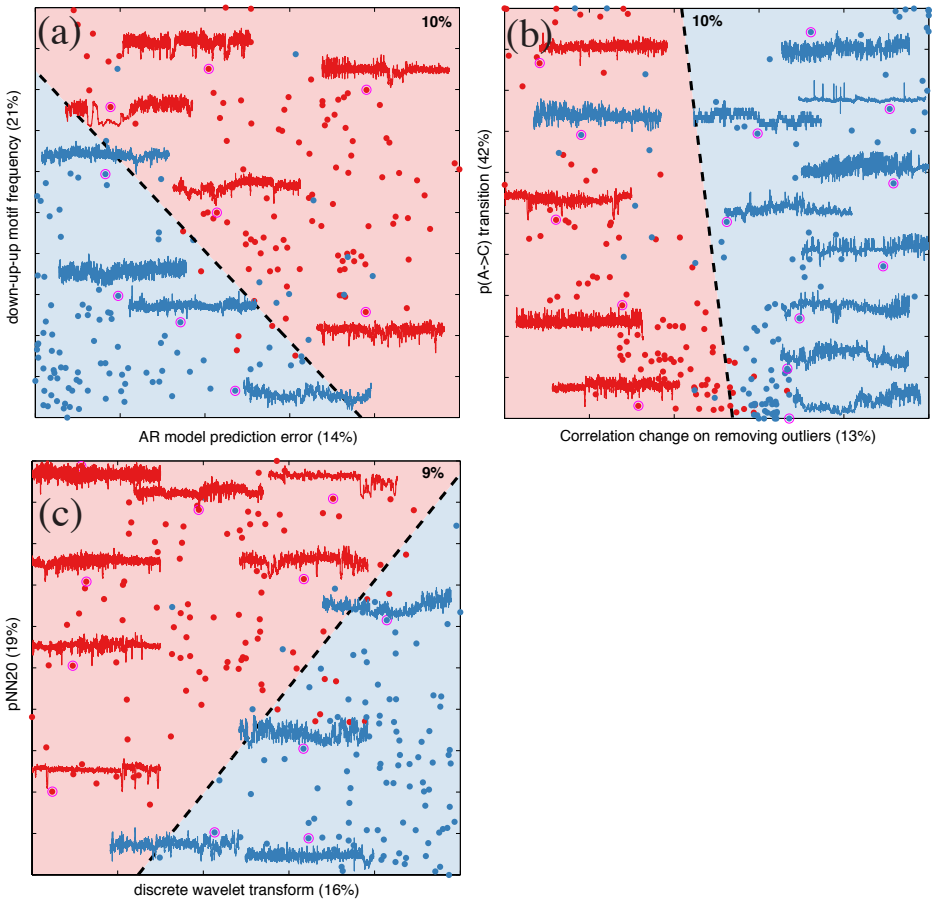

Figure B.12: **Examples classifiers combining the outputs of two operations selected by greedy forward selection.** Normal sinus rhythm signals are plotted in red and congestive heart failure in blue, a linear boundary fitted using all the points in this space is plotted in each case, mean 10-fold cross-validation linear classification errors for individual features are quoted on the axis labels, and the combined mean error is annotated on each plot. Note that outputs have been normalized for visualization purposes using a scaled sigmoidal transformation, Eq. (3.2), although quoted errors are for the unnormalized outputs. RR interval sequences are annotated to selected points in each point.

value. The combination of these operations provides a good classification ability on this dataset (a mean 10-fold cross-validation misclassification rate of 10%); when RR interval sequences contain few 'DII' patterns and a low AR model prediction error, they are classified as congestive heart failure.

In Fig. B.12(b), we show another pair with a high classification rate. In this case, however, the optimal linear discriminant boundary is close to vertical, indicating that most of the differences are explained by the unique operation introduced above – the autocorrelation change (at lag 3) from decimating outliers from the time series. An

improvement in the mean 10-fold cross-validation classification rate of three percentage points can be gained by combining a motif quantity – the probability of the time series going from the lowest third of its values to the highest third. Representing the three equiprobable quantiles of values in the time series as 'A', 'B', and 'C', this operation (which measures the probability of transitioning from A–C) behaves in a similar way to other probabilities, including the words 'BAC' and 'BBAC'.

An interesting pair of measures forms the classifier plotted in Fig. B.12(c), which consists of a discrete wavelet transform measure, which receives a significant improvement when coupled with the pNN20 measure (applied to the un-normalized time series). The wavelet transform measure returns the minimum detail coefficient for a level five discrete wavelet transformation of the RR interval series using a second order *Symlet* wavelet. RR interval sequences with high outputs from this operation combined with low values of the pNN20 statistic are classified as congestive heart failure, with a mean 10-fold cross-validation linear misclassification rate of 9%.

In this section we have demonstrated our ability to automatically construct simple and interpretable classifiers using a large database of time-series operations representing a diverse and interdisciplinary knowledge on the subject. Classifiers tend to group multiple different types of operations together, providing a more robust way of distinguishing the two classes.

B.3 Fetal heart rate analysis

In this section we describe additional information pertaining to the fetal heart rate time-series analysis performed in Sec. 5.3.

B.3.1 Data processing and partition

Because operations in our library are not designed to tolerate missing values in time series, we devised a scheme for processing the time series accordingly. First, contiguous periods of missing values of length less than 240 samples (or 1 min) were linearly interpolated. After this interpolation process, the longest contiguous period of time-series data from the original recording was obtained and saved if it is at least 3 600 samples long (i.e., at least 15 min). Recordings for which the longest contiguous

period of time-series data after processing is still less than this minimum length of 15 min are not included in our analysis. Thus, at most one processed time series is produced for each patient, with a length ranging from 3 600 samples (15 min) to 7 200 samples (30 min) and is assumed to represent the dynamics of that patient's fetal heart rate in the final 30 mins before birth. This scheme is illustrated in Fig. B.13. Of the 7 568 recordings in the original dataset, 7 221 (over 95%) produced a valid time-series output of at least 15 min of continuous measurements after local linear interpolation. The percentage of interpolated values was recorded and annotated to each time series, as an indicator of signal quality. Note that 3 445 time series, or approximately half, are over 7 000 samples (> 29 min) long.

For the purposes of retrieving features that measure relevant dynamical characteristics of fetal heart rate time series, we used a designated training and testing set of time series, as per a previous study using this data [190]. Note that, because some time series contained too many bad values to be treated in this work, this training set of time series is a subset of the original training set, containing those time series that matched the pre-processing criteria. From the original training set, we have 60 time series with pH > 7.2, and 59 time series with pH < 7.2. To create a balanced set, one time series (chosen at random) was removed from the normal pH group to form a balanced training set containing 59 time series in each class, that is analyzed in this section. For the remainder of this section, we refer to the two groups as: *normal pH*, and *low pH*. Some time series had too many missing values to be included in this study and were excluded, leaving 117 time series in each class for this testing set. The distributions of pH are similar in both the training and testing sets.

B.3.1.1 Multiple features

As well as investigating the performance of individual operations, as described in Sec. , we also explored the performance of linear classifiers that combine outputs from multiple operations. The results of greedy forward feature selection (described in Sec. 3.4.5) up to 10 features are shown in Fig. B.14. Interestingly, we find that combining multiple features in this way provides no improvement in the out-of-sample classification accuracy, i.e., linear classifiers containing more than one feature

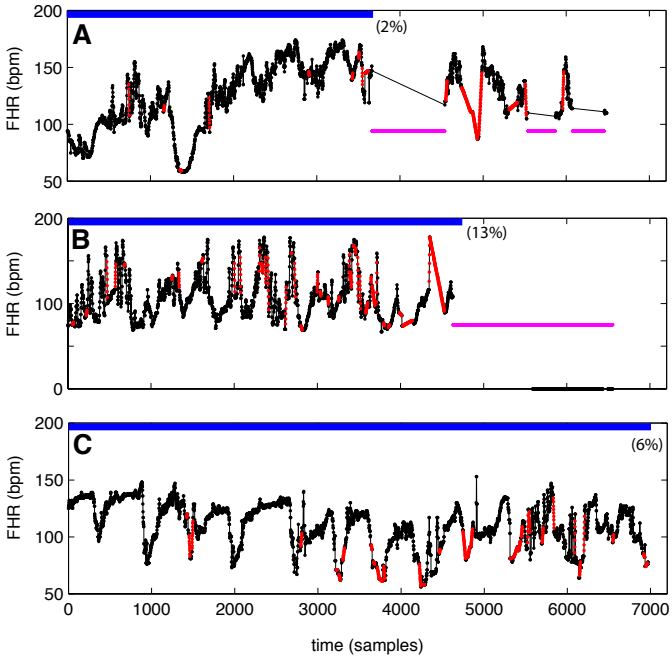

Figure B.13: **Pre-processing method for time-series**. We demonstrate our automatic pre-processing method on three example time series. Measured time-series values (after artifact removal) are indicated in black, contiguous periods of missing values of length less than 240 samples are indicated in red, contiguous periods of missing values of length greater than 240 samples are indicated with pink lines. Blue lines indicate the longest section of contiguous time-series values of the processed time series (either raw, or linearly-interpolated measurements), and form the time series that are analyzed in this study. The proportion of interpolated values in the blue region of each time series is indicated in parentheses; this value is annotated to saved recording. In **A**, the second half of the recording contains many time intervals containing no signal, and approximately the first half of the recording is retrieved. In **B**, the last approximately 2 000 samples contain a large missing segment, and a signal of zeros, which are removed from the recording, and first approximately 4 800 samples are analyzed. In **C**, consecutive periods of missing signal are short, and can be reliably linear interpolated such that a long signal of approximately 7 000 samples is obtained.

are over-fitting to the structure in the training data, without a corresponding out-of-sample classification improvement. This is likely the result of the high variability in the correspondence between training and test set classification rates.

B.3.2 Clustering operations for regression

The 50 operations with the strongest measured linear correlation coefficients, $|R|$, to cord pH are represented in Fig. B.15. Hierarchical average linkage clustering of the absolute linear correlation distances between operation outputs produce the

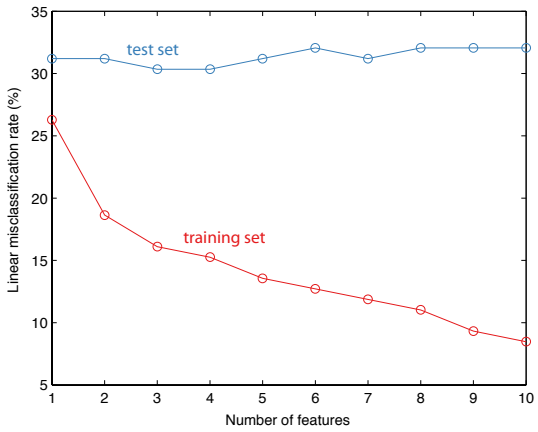

Figure B.14: **Misclassification rates on the training and test sets as a function of the number of features in a linear classifier.** Classifiers are constructed using greedy forward feature selection, as described in Sec. 3.4.5. As more features are added, the classifiers over-fit to characteristics of the training set that are not present in the test set.

dendrogram plotted at the top of Fig. B.15 whence clustering to five clusters yields the groups plotted with black squares in the figure. The member in each group with the strongest linear correlation to arterial cord pH is considered to represent its group. Thus we formed a set of five operations representative of those that exhibit the strongest correlations to cord pH from a set of over 7 000 different time-series analysis algorithms.

B.3.3 Description of selected operations

In this section we describe in more detail the operations selected for the fetal heart rate classification and regression problems described in Sec. 5.3.

B.3.3.1 Classification

CO_trev_mi_num This operation calculates the following quantity:

$$\frac{1}{N-\tau}\sum_{t=1}^{N-\tau}(x_{t+\tau}-x_t)^3, \tag{B.4}$$

where the time-delay τ is chosen as the first minimum of the auto-mutual information function of the time series, x, which has length N. This third-order quantity is related to the asymmetry of the time series under time reversal, which itself can be an indicator of time-series nonlinearity [224]. Note that the usual quantity for

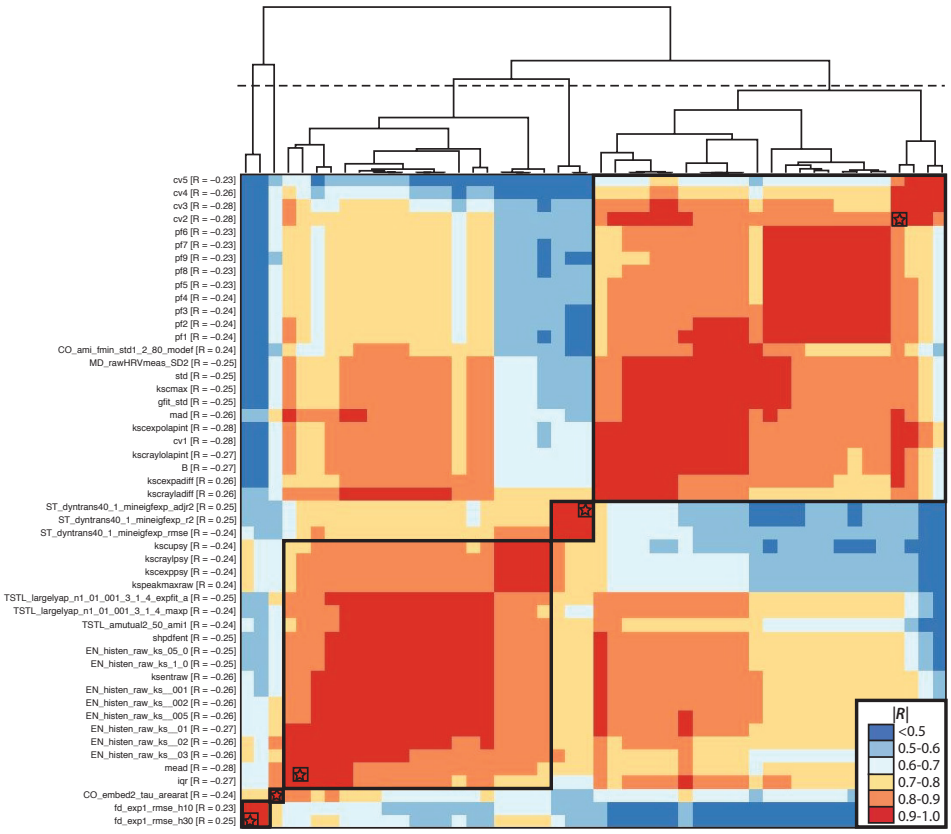

Figure B.15: **Pairwise correlations between the 50 operations with the strongest linear correlations to cord pH across the dataset.** Correlations are represented as color, according to the legend shown in the bottom right corner of the plot. We use average linkage clustering to group the 50 operations into 5 clusters, and represent each cluster by the member that is closest to that cluster center (defined as the mean of all feature vectors representing operations in that cluster). The dendrogram between these 50 operations is plotted above the plot, and the threshold at which we divide the operations into five groups is annotated. The five representative operations are plotted as stars in the plot and are investigated in detail in the main text.

measuring time-series asymmetry is a normalized version of Eq. (B.4) that involves division by $(N-\tau)^{-1}\sum_{t=1}^{N-\tau}(x_{t+\tau}-x_t)^{3/2}$. The output of this very simple statistic is able to distinguish compromised from healthy FHR recordings in both the training (28% error) and test sets (35% error), as shown in Figs. 5.39A and B, respectively.

OLbs2 This operation returns the following statistic: $\sigma(\tilde{x})/\sigma(x)$, where $\sigma(\cdot)$ calculates the standard deviation, and $\tilde{x}$ is the time series with the lowest 2% and the highest 2% of values removed. Thus, this quantity represents a ratio of standard deviations of the time series with its most extreme values removed to the original time series. Removing 'outliers' from a time series will reduce its standard deviation, so

the output of this statistic must be less than 1. The more extreme the 4% of outliers are, the smaller $\sigma(\tilde{x})$ will be relative to $\sigma(x)$, and hence the lower the output of this statistic will be. The distributions of this statistic on the training and test datasets are shown in Fig. 5.39C and D, respectively. In each case, the healthy fetal heart rhythms have more extreme outliers and lower values of this statistic compared to the compromised fetal heart rates.

dynpick200_meanapen1_02 This operation, shown in Figs. 5.39E and F, combines local measures of entropy. The operation takes 100 different contiguous 200-sample (50 s) segments at random from the z-scored time series and calculates the Approximate Entropy, ApEn(m,r) [2, 29], using an embedding dimension, $m = 1$, and threshold $r = 0.2$. The output of this operation is the result of averaging these 100 local ApEn(1,0.2) entropy estimates. As shown in Figs. 5.39E and F, this operation produces higher outputs for healthy fetal heart rates (i.e., local 25 s time-series segments recorded from healthy patients have higher entropy). Of the five top operations chosen from the training set, this operation shows the best performance on the training set, with a 26% classification error, and also shows the best performance on the test set, which it classifies with only 31% error.

ST_dyntrans40_1_mineigfexp_adjr2 This operation calculates 1-step transition probabilities between different states in a discretized transformation of the time series, and measures how they change as the size of the alphabet increases. The transformation assigns one of an alphabet of symbols to each point in the time series according to its value, where each letter in the alphabet is equiprobable. For example, for an alphabet size of three, we transform the time series into a string using the letters 'A', 'B', 'C', where the lowest third of time-series values are assigned to 'A', the middle third to 'B', and the highest third to 'C'. We then calculate the probabilities of transitioning from a given letter at time t to the letter at time $t + 1$, and thereby assemble an empirical transition probability matrix. For this algorithm, we construct transition probability matrices using alphabets of size 2, 3, ..., and 40 and calculate the minimum real part of the set of eigenvalues of each transition probability matrix, $[\mathrm{Re}(\lambda)]_{\mathrm{min}}$. A decaying exponential of the form $f(x) = A\exp(Bx)$ is fitted to the

variation in $[\mathrm{Re}(\lambda)]_{\mathrm{min}}$ as the alphabet size increases. The output from this operation is the adjusted R^2 goodness of fit measure from this exponential fit, and is plotted for the training and test sets of data in Fig. 5.39G and H, respectively. Despite being difficult to interpret, this operation exhibits a good ability to classify the training set (in-sample error of 28%), and also shows strong out-of-sample performance on the testing set (34% error).

dynpick200_stdsampen1_02 The final operation we consider, which is similar to the *dynpick100_meanapen1_02* measure described above, measures the variation in Sample Entropy [50], SampEn(m,r), estimates across local segments of the time series. We use $m = 1$, and $r = 0.2$, and calculate SampEn(1,0.2) in 100 randomly-chosen contiguous 200-sample (50 s) subsegments of each time series, which is first z-scored. The output of this operation is the standard deviation of this set of local entropy estimates, and is plotted in Figs. 5.39I and J for the training and test sets, respectively. Healthy fetal heart rate recordings (plotted as blue in the figure), have a greater variability in SampEn(1,0.2) values calculated in local 50 s segments of the time series than do compromised babies (red in the figure).

B.3.3.2 Regression

In this section we describe five operations that are representative of those with the strongest linear correlations to cord pH across a database of 7 221 fetal heart rate time series, as described in Sec. 5.3.3.

cv2 This is the coefficient of variation statistic of 'order' k: $[\sigma(x)/\mu(x)]^k$, where $\sigma(\cdot)$ calculates the standard deviation, and $\mu(\cdot)$ calculates the mean of the time series, x. In this case, we use $k = 2$, but as can be seen from the cluster in Fig. B.15, the coefficient of variation with $k = 1, 3, 4, 5$ all give very similar results for this dataset. In Fig. 5.41A, we show that the outputs from this simple statistic have a linear correlation coefficient $R = -0.28$ with the cord pH data.

mead This operation returns the median absolute deviation of the time series as

$$\langle |x - \mathrm{median}(x)| \rangle, \tag{B.5}$$

where $\langle\cdot\rangle$ indicates a mean across the values in the time series, x. As shown in Fig. B.15, many other measures of spread from the distribution show similar performance to this measure, including the inter-quartile range. The median absolute deviation of the time series exhibits a negative correlation with arterial cord pH, as $R = -0.28$, as shown in Fig. 5.41B. This indicates that the greater this measure of signal variability, the more likely it is for a baby to have a low pH.

ST_dyntrans40_1_mineigfexp_adjr2 This operation, which fits exponential decay to the minimum real part of eigenvalues of transition probability matrices was described above as one of the representative measures for separating the two classes of the training dataset. Although difficult to interpret, this measure shows strong performance on both the training set and as a linear predictor of arterial cord pH across the full dataset of over 7 000 patients. The scatter plot of its output and arterial cord pH is in Fig. 5.41C, where a linear correlation coefficient of $R = 0.25$ is calculated.

fd_exp1_rmse_h30 This operation estimates the distribution of the time series using histograms with 30 bins (using MATLAB's **hist** function), and then fits an exponential distribution using the 'exp1' input to the **fit** function of MATLAB's *Curve Fitting Toolbox*. The output of the function is the root mean square error of the fit. As shown in Fig. 5.41D, FHR recordings associated with a higher arterial cord pH have distributions of data values that are closer to exponential, on average.

CO_embed2_tau_arearat This operation embeds a z-scored transformation of the time series in a two-dimensional time-delay embedding with time-delay τ equal to the first zero-crossing of the autocorrelation function, which is a standard choice [13]. Once embedded in this two-dimensional space, the rectangular area spanned by all datapoints is calculated as the produce of ranges in each dimension: $A_x = \text{range}(s_x) \times \text{range}(s_y)$, where the embedded time series, $\mathbf{s}$ is a sequence with two-components $s_x = x_{1:N-\tau}$ and $s_y = x_{\tau+1:N}$. The rectangular area spanned by the 50% of data points that are the closest to the origin is also calculated as $A_x^{50\%}$. The output of this operation is the ratio of these two areas: $A_x^{50\%}/A_x$. In Fig. 5.41E, we see that

high arterial cord pH is associated with low values of this statistic.

Appendix C

Additional material: Highly comparative time-series analysis for temporal data mining

In this appendix, we present additional information related to Chapter 6.

C.1 Data

In this section we present a table, Table C.1, describing the twenty data mining datasets analyzed in Chapter 6.

C.2 Principal Components Projections

In Fig. C.1, two-dimensional PC projections of four datasets are plotted.

The 'Coffee' dataset contains 'time series' that each represent a Fourier transform infrared spectrum from either an *Arabica* or *Robusta* freeze-dried coffee sample [250]. In the two-dimensional PC space of this dataset, shown in Fig. C.1A, the two classes are well separated, even though the spectra are difficult to separate by eye. Note that a more detailed unsupervised analysis of this dataset, including the analysis of confusion matrices constructed from the results of k-means clustering in this space, can be found in Sec. C.3.1. In Fig. C.1B, for the 'Trace' dataset, the four classes form four clusters in the two-dimensional PC space for this dataset. Indeed, it is the first PC that contributes to informing the differences between these four classes. The pink and purple classes, which are very similar patterns (differing only by a small deviation towards the end of each time series), are close in this representation.

Table C.1: The twenty time-series datasets analyzed in this work, as obtained from *The UCR Time Series Classification/Clustering Homepage* [214]. For each dataset, we list the number of classes, n_{classes}, the number of training examples, n_{train}, the number of test examples, n_{test}, and the length of each time series, N (number of samples). References are given to the first published work to use each dataset.

Set name	n_{classes}	n_{train}	n_{test}	N (samples)
Synthetic Control [240]	6	300	300	60
Gun point [221]	2	50	150	150
CBF [241]	3	30	900	128
Face (all) [242]	14	560	1690	131
OSU leaf [243]	6	200	242	427
Swedish leaf [218]	15	500	625	128
50 words [244]	50	450	455	270
Trace [245]	4	100	100	275
Two Patterns [246]	4	1 000	4 000	128
Wafer [216]	2	1 000	6 164	152
Face (four) [242]	4	24	88	350
Lightning (two) [217]	2	60	61	637
Lightning (seven) [217]	7	70	73	319
ECG [216]	2	100	100	96
Adiac [247]	37	390	391	176
Yoga [219]	2	300	3 000	426
Fish [248]	7	175	175	463
Beef [249]	5	30	30	470
Coffee [250]	2	28	28	286
Olive oil [251]	4	30	30	570

For the 'Gun Point' dataset, plotted in Fig. C.1C, the first PC does not contribute to separating the two classes, but the second PC does. Finally, for the 'Synthetic Control' dataset in Fig. C.1D, the six classes cluster into four identifiable groups. The oscillation pattern (orange) and noise series (green) are distinguished as separate clusters, with the step-up (lime green) and step-down (yellow) time series forming a group, and the increase (purple) and decrease (pink) time series forming another group. Step-up (lime green) and step-down (yellow) time series would be considered very different in the time domain, in which similarity is judged according to actual values of the time series; and similarly for the increase (purple) and decrease (pink) time series. However, both are grouped in our feature-based framework because a step change has a greater effect on the measured properties of the time series, including distributional, autocorrelation, and stationarity-based operations. The leading PCs

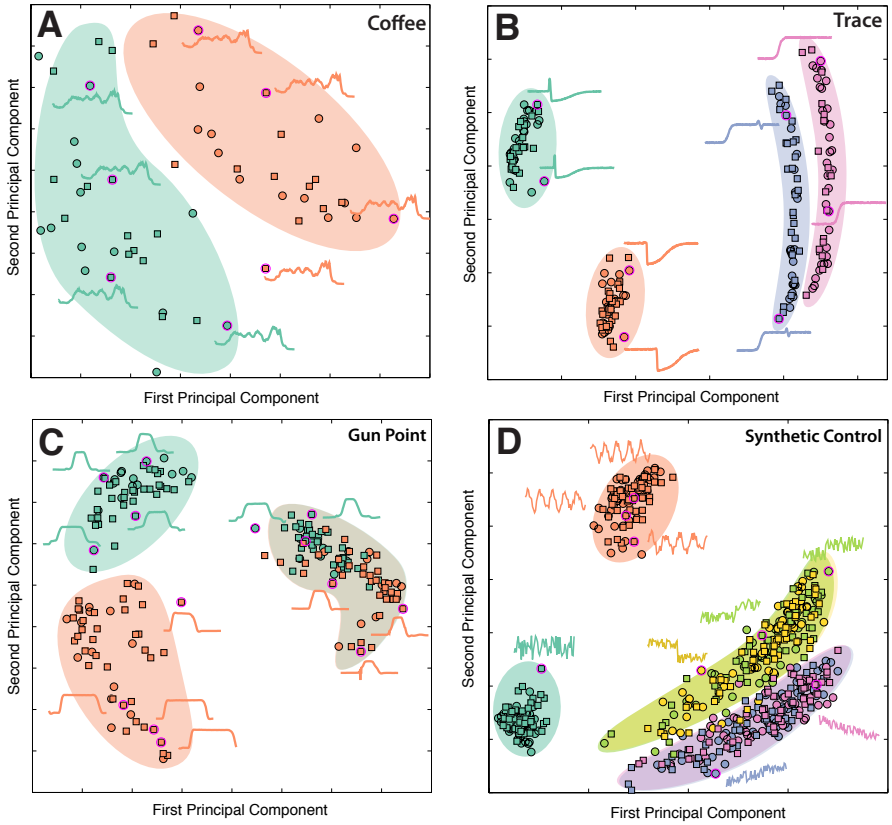

Figure C.1: **Principal Components projections of large, diverse feature spaces can reveal structure in time-series datasets.** The training (circles) and test (squares) data are embedded in a two-dimensional Principal Components (PC) space for four datasets: **A** 'Coffee', **B** 'Trace', **C** 'Gun Point', and **D** 'Synthetic Control'. The PCs were calculated on the normalized training data using all operations with no special-valued outputs from our extensive library of operations. In each plot, background shading is added manually to guide the eye to the clustering of classes, and selected time series are annotated. Note that this is an unsupervised analysis: the class labels were not used in constructing these projections (nor were the test data).

detect this strong dependence of a step change on many operations and produce the corresponding representation of this dataset shown in Fig. C.1D.

Although the structuring of the time series classes in these examples is interpretable and intuitive, the unsupervised representation of the 'Synthetic Control' dataset in Fig. C.1D does not cluster the six labeled classes of the dataset separately. In Sec. 6.3.2.1, we show how the known class labels can be used to select those features that effectively measure differences between labeled classes of time series, rather than relying on this purely unsupervised representation.

C.3 Additional Classification results

In this section, additional results for the temporal data mining datasets studied in Chapter 6 are presented.

C.3.1 Coffee

In Fig. C.2, we show the results of an unsupervised analysis of the 'Coffee' dataset, where each 'time series' in the dataset represents a Fourier transform infrared spectrum from either an *Arabica* or *Robusta* freeze-dried coffee sample [250]. For this example, we explicitly perform a clustering of the test data in the reduced PC spaces. The clustering is performed in spaces of increasing dimensionality using k-medoids clustering for $k = 2$. As shown in Fig. C.2A, for any number of Principal Components in the range 1–10, this unsupervised clustering of the test data into 2 groups gives a better agreement to the actual known class labels than 1-NN classification using either Euclidean distances between time-series objects or dynamic time warping, despite the fact that the class labels are not used in our unsupervised clustering. The two-dimensional PC projection of the full dataset, including training and test data is shown in Fig. C.2B, where a natural clustering of the dataset into two groups is evident. Confusion matrices for k-medoids clustering in a two-dimensional PC space, and for a 1-NN classifier using Euclidean distances between time-series objects are shown in Figs. C.2C and D. The k-medoids clustering confuses one example from each class, whereas the Euclidean 1-NN classifier confuses five examples from the first class, and two in the second.

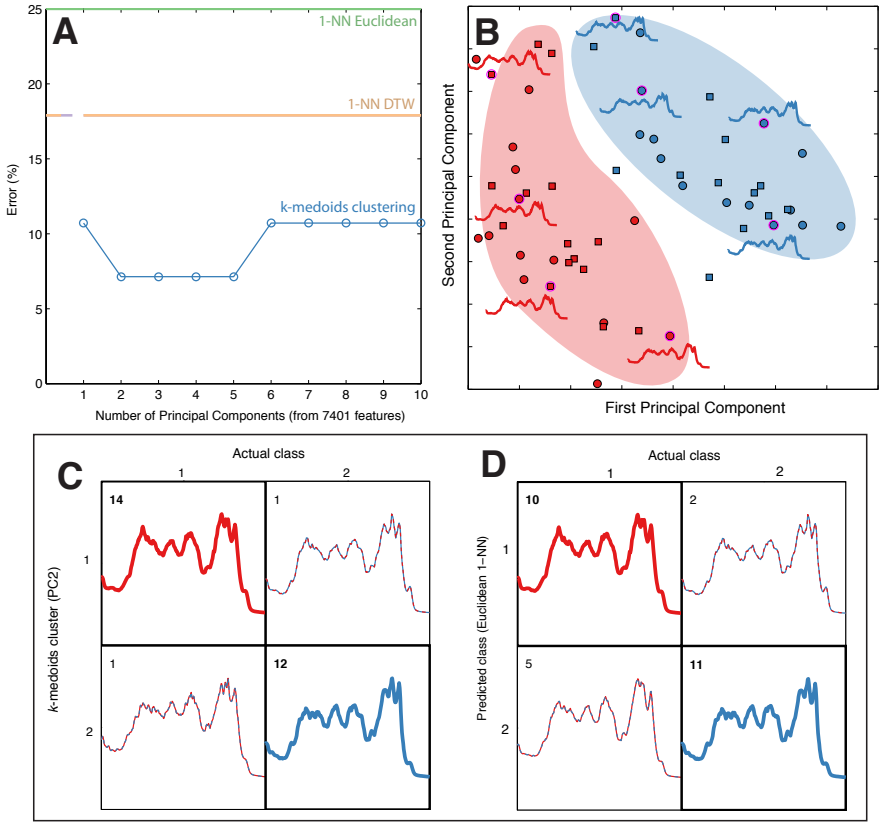

Figure C.2: **Clustering of the 'Coffee' dataset in Principal Components feature spaces**. The PC spaces are formed from the 7 401-dimensional feature spaces obtained from the normalized training data, in which the test data are then projected. **A**: Unsupervised k-medoids clustering (with $k = 2$) of the test data gives a better correspondence with the labeled classes than any of the supervised time-domain methods. Note that 1-NN DTW using the best warping window produces the same misclassification error as without using a warping window. **B**: The two-dimensional space containing both the training (circles) and test (squares) data. Shading has been added manually to show the two clusters in this space. Finally, confusion matrices are shown for **C**: k-medoids clustering in the two-dimensional PC space, and **D**: Euclidean 1-NN classification using distances between time-series objects. Curves in each box represent an appropriate example time series, and numbers indicate the number of time series in each element of the confusion matrix.

C.3.2 Synthetic Control

Confusion matrices are shown in Figs. C.3A and C.3B. As we discovered above, the first cumulative sum-based feature has trouble separating the random and oscillatory time series, as shown in Fig. C.3A. This is resolved by adding the autocorrelation-based feature, as shown in Fig. C.3B, which also helps correct some of the other misclassifications between other classes. Extracting features to summarize the difference in dynamical properties between these time series is evidently appropriate for this dataset. The results from instead calculating Euclidean distances between time-series objects are shown in the confusion matrix of Fig. C.3C. With only 60 samples in each time series, the random number sequences (class 1) often by chance have nearest neighbors corresponding to the other classes, contributing to a higher classification error for this 1-NN classifier, that operates on 60-dimensional feature vectors instead of the one-, or two-dimensional feature-based 1-NN classifiers shown in Figs. C.3A and C.3B, respectively.

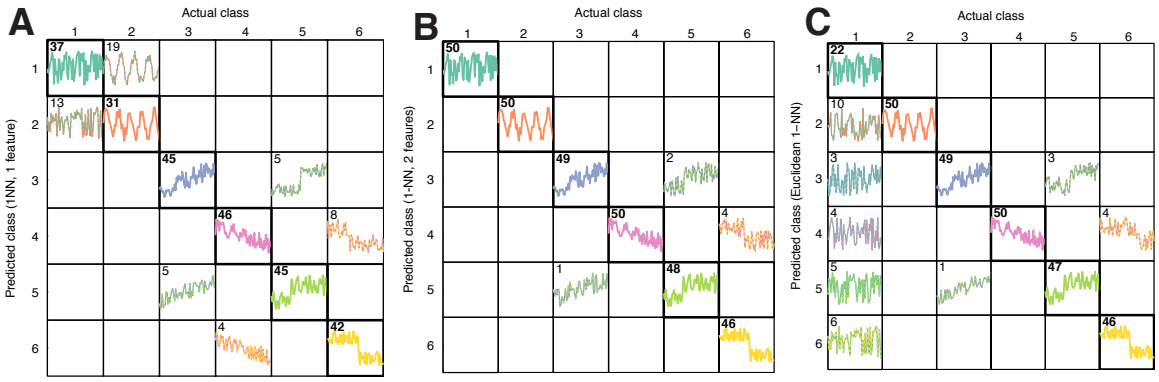

Figure C.3: **Confusion matrices for the 'Synthetic Control' dataset.** We plot confusion matrices for three classifiers: **A** 1-NN with one feature, **B** 1-NN with two features, and **C** 1-NN using Euclidean distances in the time domain.

C.3.3 Two Patterns

An example of a two-feature representation of a dataset is shown for the 'Two Patterns' dataset in Fig. C.4A. The first selected feature calculates the quantity $\langle (x_{t+\tau} - x_t)^3 \rangle$, where τ is chosen as the first minimum of the autocorrelation function of each time series, x, and the average, $\langle \cdot \rangle$, is performed across the time series. The

output of this operation is very good at distinguishing the green and pink classes, but the blue and orange classes remain very similar in this representation. The second selected operation is very poor at discriminating the green and pink classes, but effectively discriminates the blue and orange classes and is therefore a good complement to the first operation. This second operation computes the mean in one hundred different 100-sample segments from the z-scored time series and returns the mean of this set of local means. Since the time series themselves are only 128 samples long, this amounts to averaging the means in different contiguous 78% segments of the time series. We can understand why this operation works by inspecting example time series in Fig. C.4B. The time series in class 2 (orange) are all of the same pattern: 'dip, peak, peak, dip', and therefore the time series values that are above average occur near the middle of the time series so that most 78% segments of the time series will include these two peaks and have a mean that is on average greater than zero. The time series in class 3 (blue), however, have the opposite pattern: 'peak, dip, dip, peak', and hence most 78% segments of these time series will include both dips and yield a mean that is less than zero. Together, these operations have just a 7.4% linear classification error on the test set using a linear classifier learned on the training set.

C.3.4 OSU Leaf

As an example of a feature-based classifier containing more than two features, we show results for the 'OSU Leaf' dataset in Fig. C.5. The iterative feature selection process for this six-class classification problem using a 1-NN classifier and normalized features is shown in Fig. C.5A. We see that the misclassification rate decreases in both the training and test sets up to seven features, after which it remains approximately stable in the test set (and continues to modestly decrease in the training set on which it is optimized). With just two features, the performance of the feature-based classifiers exceeds the conventional time series distance-based classifiers, which have misclassification rates over 38.4% (see Tab. 6.1). The performance is further increased when additional features are added, with the test set misclassification rate reaching just 16% with seven features.

However, in general we do not have access to the test set to choose an optimal

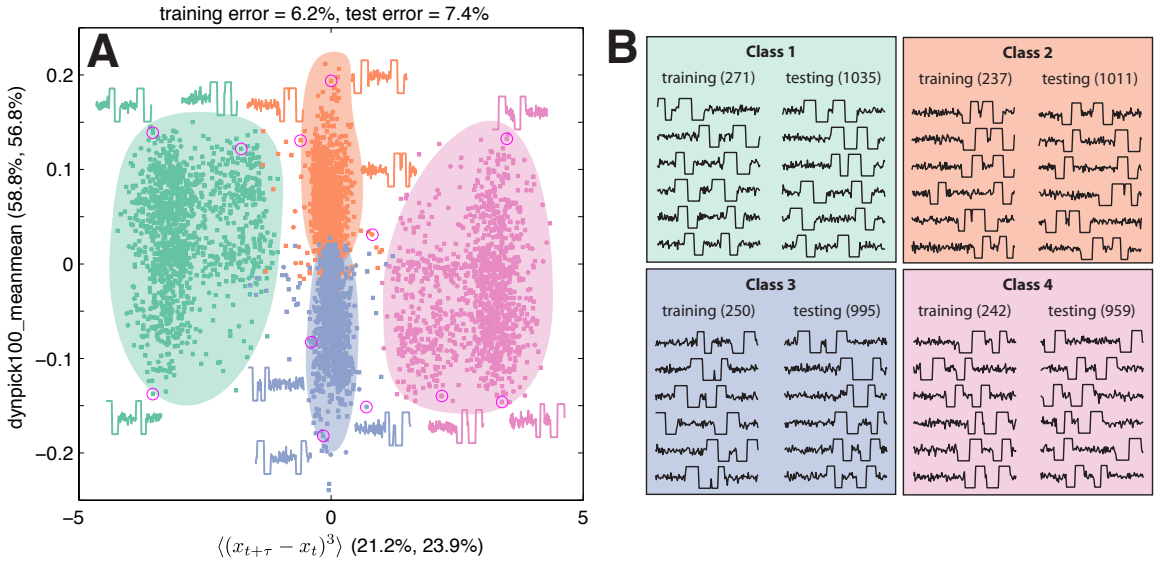

Figure C.4: **A two-feature representation of the 'Two Patterns' dataset**. These two features were chosen by a greedy forward selection method using a linear classifier (on unnormalized features). **A** The two features separate the classes well, with a 92.6% linear classification rate on the test set using a linear classifier learned on the training set. Each feature is labeled with misclassification rates on the training and test sets in parentheses. Selected time series are annotated to the plot, as indicated by pink circles, and manual background shading has been added to guide the eye. Due to the large number of time series in this dataset, points in the plot are small, with the 1 000 training data points are plotted as circles, and the 4 000 test data points are plotted as squares. The features are explained in the main text. Numbers in parentheses indicate linear misclassification rates on the training and test sets of each individual feature in isolation. **B** Example time series from the four classes are shown, separated into training and test sets. Numbers in parentheses indicate the number of time series in each class and training/test partition.

number of features, which must be decided using the training data alone. Our proposed method for choosing this number is the point at which the improvement in the training set misclassification rate drops below 3% (cf. Sec. 6.2.3). For this dataset, that corresponds to a classifier with five features for this dataset with an error of 21.1% (cf. Table 6.1). Because it is difficult to plot five-dimensional spaces, we visualize how the datasets are structured in these spaces using networks, as shown in Figs. C.5C and C.5D for the training and test sets, respectively. The plots are of fully-weighted networks where each node is a time-series in the dataset, and links represent similarity between pairs of time series, as a normalized Euclidean distance between their (in this case five-dimensional) feature vectors. The networks suggest that with five features, the time series in this dataset are indeed being organized into

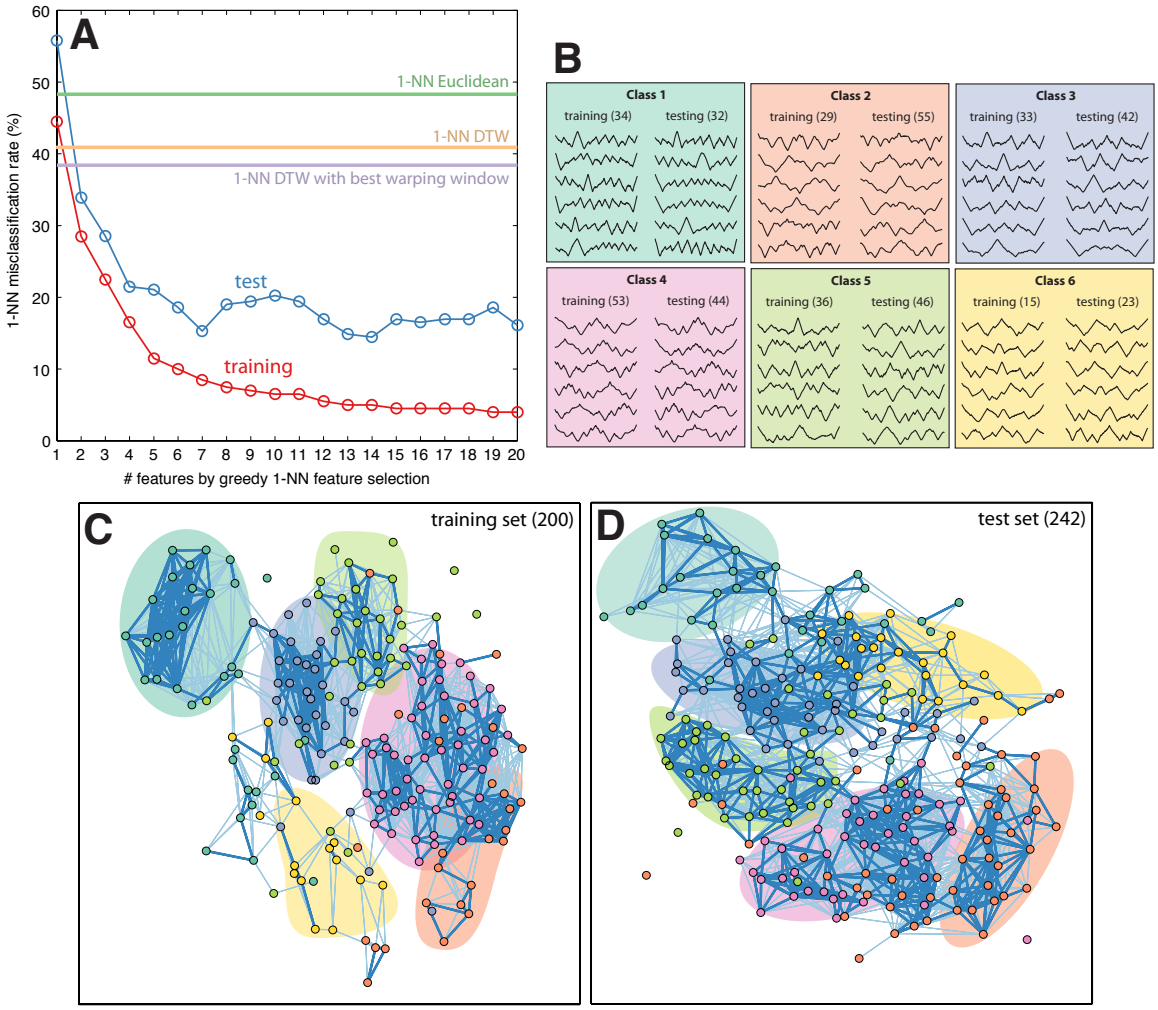

Figure C.5: **Feature selection on the 'OSULeaf' dataset using a 1-NN classifier on normalized features.** **A** The training and test set errors as a function of the number of features learned for a greedy forward feature selection procedure. Misclassification rates for conventional 1-NN classifiers using distances between time-series objects are shown using horizontal lines for comparison. **B** Example training and test time series in each of the six classes. Colors assigned to each class match plots **C** and **D**, where networks of the training and test sets, respectively, are plotted using five features chosen by greedy 1-NN feature selection. Each node in the network is a time series in the dataset, and distances between all pairs of time series are calculated as Euclidean distances between their five-element feature vectors, forming a fully-weighted network. The strongest 7% of links are drawn, and the strongest 3% are distinguished using thick blue lines. Background shading has been added manually to guide the eye.

sensible classes.

C.4 Sensitivity to partitions

Throughout Chapter 6, we used the partition of the data into training and testing sets as prescribed in the *The UCR Time Series Classification/Clustering Homepage* [214]. But how sensitive are the results of classifiers to this choice of partition? That is, are successful classifiers simply learning this (often arbitrary) training/test partition, or are they really measuring informative characteristics of the class structure in the dataset? To probe this question, we performed a temporal-based classification using a 1-NN Euclidean classifier using 1 000 random partitions of the data into training and testing sets, while keeping the same class proportions in the training and test sets as the original partition. Note that we used a 1-NN classifier in the time domain in this case because it is very quick to train. For feature-based classifiers, the same plots could be produced by repeating the feature selection procedure for each partition (potentially choosing different features for each partition). The results are shown in Fig. C.6 for all twenty datasets.

In Fig. C.6, distributions of (in-sample) training and (out-of-sample) test errors across these 1 000 partitions are shown along with the training and test errors for the specified partition used in this work. For many cases, the test error for the given partition is approximately equal to the mean of the distribution: i.e., the test error obtained from the specified training/test partition is a fair representation of the range of possible partitions that could have been chosen. For other datasets, this is not the case: the test error is higher than would be expected across 1 000 random partitions and hence optimistic, or lower and hence pessimistic. For the 'Face (all)' dataset, in the top right panel of Fig. C.6, the test error is extremely pessimistic due to a very specific training/test partition of the data. Thus, for cases in which the partition into training and test sets is done arbitrarily, simply quoting a single test set error on a classification task is misleading, as it could be optimistic: as the result of a 'lucky' partition, or pessimistic: as the result of an 'unlucky' partition. Rather, we argue that plots such as those shown in Fig. C.6 should be constructed to show the distribution of errors for a given classifier, or these distributions (which are approximately Gaussian here) could also be summarized by their mean and standard

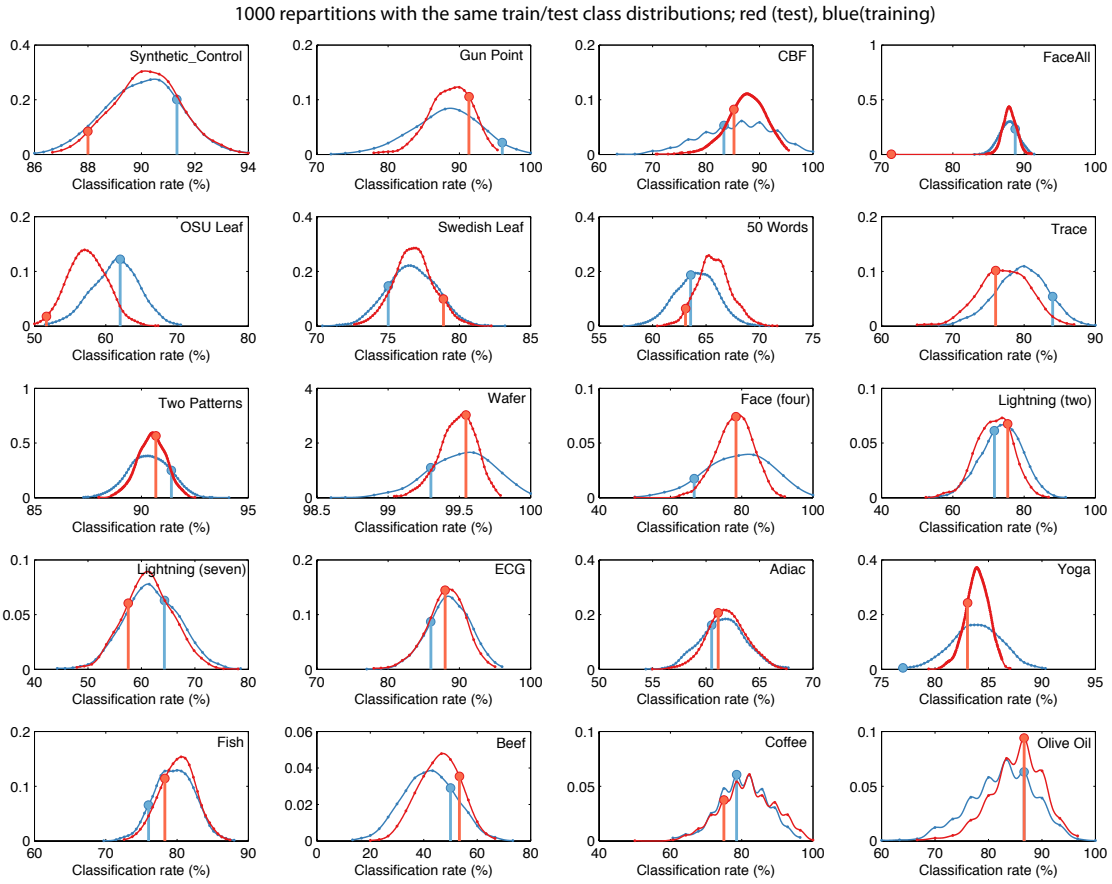

Figure C.6: **Sensitivity of results to given training and testing partitions.** Each plot shows the distribution of in-sample training (blue) and out-of-sample test (red) errors for a 1-NN classifier using Euclidean distances in the time domain (i.e., *not* using a feature-based representation) across 1 000 repartitions of each dataset into training and testing portions while keeping the same training and testing proportions. The training and test errors for the given partition, specified from the *The UCR Time Series Classification/Clustering Homepage* [214], are shown with large circles and stems.

deviation.

Appendix D

Additional material: Highly comparative feature-based dimensionality reduction for time-series datasets

In this appendix, we provide additional information for the material presented in Chapter 7.

D.1 Numerical examples

In this section we describe synthetic, numerical case studies exploring the theoretical ideas outlined in Sec. 7.2.1. Throughout this section we compute an appropriately normalized mutual information measure, $I_{\mathrm{norm}}(X;Y)$ described in Sec. 7.2.2.2.

D.1.1 Example: Twelve functions

Rather than extracting features from a time-series dataset generated from a model that depends on parameters $\Theta = \{\theta_1, \theta_2\}$, functions of θ_1 and θ_2 are instead defined explicitly. The result allows us to gain some intuition for the process that links the mutual information measured between features to the parameters that they depend on.

For this example, twelve functions are examined, four of which depend on θ_1 only, four depend on both θ_1 and θ_2, and the final four depend on θ_2 only. The functions

are as follows. The first four depend on θ_1:

$$F_1(\theta_1) = \sin(\theta_1), \tag{D.1}$$

$$F_2(\theta_1) = \cos(\theta_1/2) + 0.1\eta_2, \tag{D.2}$$

$$F_3(\theta_1) = \theta_1^2/2, \tag{D.3}$$

$$F_4(\theta_1) = -\theta_1^5/10, \tag{D.4}$$

the next four depend on both θ_1 and θ_2:

$$F_5(\theta_1, \theta_2) = \theta_1^3 - \theta_2^2, \tag{D.5}$$

$$F_6(\theta_1, \theta_2) = -\theta_1 + \theta_2, \tag{D.6}$$

$$F_7(\theta_1, \theta_2) = 2\exp(-\theta_1^2) + \sin(\theta_2) + 0.1\eta_7, \tag{D.7}$$

$$F_8(\theta_1, \theta_2) = \sin(\theta_1/3) + \cos(\theta_2), \tag{D.8}$$

and the final four depend only on θ_2:

$$F_9(\theta_2) = 5\exp(-\theta_2) + \eta_9, \tag{D.9}$$

$$F_{10}(\theta_2) = \theta_2, \tag{D.10}$$

$$F_{11}(\theta_2) = -\theta_1^2 + \theta_2/10 + 5\eta_{11}, \tag{D.11}$$

$$F_{12}(\theta_2) = \sin(\theta_2/10). \tag{D.12}$$

Note that each of $\eta_2, \eta_7, \eta_9, \eta_{11} \sim U(0, 1)$ each represent independent draws from a uniform distribution over the unit interval and add randomness to the functions F_2, F_7, F_9, and F_{11}, respectively.

The mutual information, $I_{\text{norm}}(F_i; F_j)$, measured between pairs of these twelve functions will depend on the distribution over parameters θ_1 and θ_2. The relationship between variation in these parameters and the resulting dependences between these twelve functions is explored by studying different constraints on θ_1 and θ_2. In Fig. D.1, we plot normalized mutual information, I_{norm}, matrices calculated between all pairs of the twelve functions: F_1, F_2, ..., F_{12}, defined in Eqs. (D.1)–(D.12) above, for different domains of θ_1 and θ_2, using 1 000 independent samples from the parameter space in each case.

First we study the case in which the random variables $\theta_1 \sim U(-3, 3)$ and $\theta_2 \sim U(0, 6)$ are both independently free to vary across their respective domains, as shown

in Fig. D.1B. The functions that depend on θ_1 only: F_1–F_4, form a correlated block as $\mathcal{F}_{\theta_1}$ (labeled red), with high I_{norm} between members of this group. Similarly, the functions that depend on both θ_1 and θ_2 form a correlated block as $\mathcal{F}_{\theta_1,\theta_2}$ (i.e., F_5–F_8, labeled yellow), as do those that very with θ_2 only as $\mathcal{F}_{\theta_2}$ (i.e., F_9–F_{12}, labeled blue). Positive mutual information is also measured between $\mathcal{F}_{\theta_1}$ and $\mathcal{F}_{\theta_1,\theta_2}$ (due to their common dependence on θ_1), and between $\mathcal{F}_{\theta_2}$ and $\mathcal{F}_{\theta_1,\theta_2}$ (due to their common dependence on θ_2). Thus, the basic mutual information structure expected from the analysis performed in Sec. 7.2.1 above, and illustrated in Fig. 7.4B, is reproduced in this synthetic example. The variation of I_{norm} within each group (i.e., within each of $\mathcal{F}_{\theta_1}$, $\mathcal{F}_{\theta_1,\theta_2}$, and $\mathcal{F}_{\theta_2}$) is related to whether the functions are monotonic with the parameters (which increases $I_{\text{norm}}(F_i;F_j)$), and whether there is an additional stochastic component to the random functions (which decreases $I_{\text{norm}}(F_i;F_j)$).

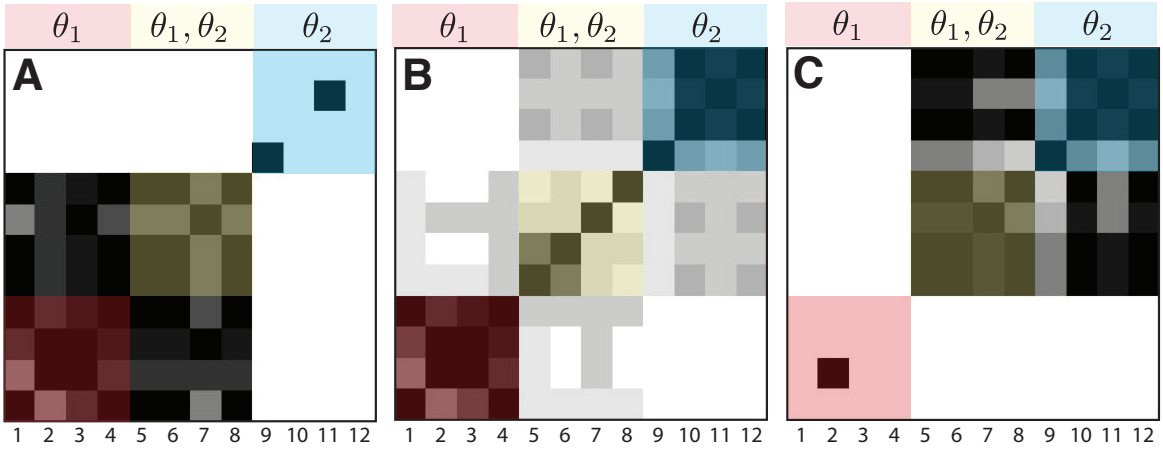

Figure D.1: **Normalized pairwise mutual information matrices between a set of twelve functions**. Four of the functions depend on θ_1 only (F_1–F_4, Eqs. (D.1)–D.4, labeled red), four depend on θ_1 and θ_2 (F_5–F_8, Eqs. (D.5)–(D.8), labeled yellow), and four depend on θ_2 only (F_9–F_{12}, Eqs. (D.9)–(D.12), labeled blue). **A** $\theta_1 \sim U(-3,3)$ is free to vary but $\theta_2 = 0$ is fixed, **B** both $\theta_1 \sim U(-3,3)$ and $\theta_2 \sim U(0,6)$ are free to vary, and **C** $\theta_1 = 0$ is fixed and $\theta_2 \sim U(0,6)$ is free to vary. These mutual information matrices reveal the characteristic structure for a one-parameter system in **A** and **C**, and for a two-parameter system in **B**, as predicted in Fig. 7.4. Calculations are performed using 1 000 points sampled independently from the distributions over each random variable, θ_1 and θ_2 in each case. A greyscale colormap is used to represent $I_{\text{norm}}(F_i;F_j)$: using black for $I_{\text{norm}} \geq 0.5$, and white for $I_{\text{norm}} = 0$.

Next, we investigate mutual information measured between the twelve functions when $\theta_2 = 0$ is fixed and $\theta_1 \sim U(-3,3)$ is free to vary. The functions F_1–F_8 vary with the only free parameter, θ_1, while F_9–F_{12} do not. As shown in Fig. D.1A, those

functions that depend on θ_1 form a correlated block, as $\mathcal{F}_{\theta_1}$, and those that do not are distinguished as $\mathcal{F}_0$. The same effect is shown in Fig. D.1C where now $\theta_1 = 0$ is fixed and only $\theta_2 \sim U(0, 6)$ is free to vary. Again, we correctly identify a correlated block of operations, $\mathcal{F}_{\theta_2}$, as those informative of θ_2: the parameter whose variation controls variation in these functions. Thus, the expected mutual information structure depicted in Fig. 7.4A is reproduced for these one-parameter datasets.

Note that the random variables η_2, η_7, η_9, and η_{11} add a stochastic component to the random functions F_2, F_7, F_9, and F_{11} that is distinct from a third random variable, θ_3, say, because their outputs are independent for each function. If instead they were generated from a common random variable, θ_3, they would exhibit measurable correlations with one another in a similar way to θ_1 and θ_2. This basic observation is the key to the general applicability of this approach to time-series datasets: even though time-series analysis operations could depend on any number of possible parameters, when only a small number of parameters are free to vary and all others are fixed, the only mechanism through which operations can share mutual information is by varying with one or more of the free parameters, Θ.

We also note the reduction in mutual information measured between operations when both parameters, θ_1 and θ_2, are simultaneously free to vary, compared to when one of them is fixed. This reduction is due to the presence of multiple independent sources of variation, making it increasingly difficult to predict the value of one function given knowledge of another. We will find this effect limits our ability to scale up the approach to datasets controlled by large numbers of parameters, especially for finite time-series datasets (see Sec. 7.3.3.2).

D.1.2 Example: Five functions of two variables

The this example we consider involves five functions of two variables, θ_1 and θ_2. The functions are defined as follows:

$$F_1(\theta_1, \theta_2) = \sin(\theta_1), \tag{D.13}$$

$$F_2(\theta_1, \theta_2) = \theta_1^3 + \theta_2^2, \tag{D.14}$$

$$F_3(\theta_1, \theta_2) = -\theta_1 + \theta_2, \tag{D.15}$$

$$F_4(\theta_1, \theta_2) = 2\exp(-\theta_1^2) + \sin(\theta_2) + 0.1\eta_4, \quad \text{where } \eta_4 \sim U(0,1), \tag{D.16}$$

$$F_5(\theta_1, \theta_2) = 5\exp(-\theta_2) + \eta_5, \quad \text{where } \eta_5 \sim U(0,1). \tag{D.17}$$

Note that F_1 depends on θ_1 only so that $I(F_1; \theta_2) = 0$, F_5 depends on θ_2 only so that $I(F_5; \theta_2) = 0$, and F_4 and F_5 are random functions due to the small additive uniformly-distributed random variables η_4 and η_5. In this example, we define $\theta_1 \sim U(-3, 3)$ and $\theta_2 \sim U(0, 6)$ as uniformly-distributed random variables. In the remainder of this section, we explore the relationships between these five functions and the random variables θ_1 and θ_2.

First consider the case where θ_1 and θ_2 are both free to vary within their respective domains, as shown in Fig. D.2B. In this figure, scatter plots are plotted for F_1, F_2, ..., F_5 as functions of θ_1 and as functions of θ_2 individually. The plots are the result of 1 000 uniformly-distributed random samples of θ_1 and θ_2 from their respective domains.

Because F_2, F_3, and F_4 vary with both θ_1 and θ_2, have $I(F; \theta_1) > 0$ and $I(F; \theta_2) > 0$, but in each plot the variation of the other parameter is unaccounted for and introduces a stochastic component to the random functional relationship, which decreases the calculated mutual information, $I_{\text{norm}}(\theta_i; F_j)$; similarly for the random components η_4 and η_5 to F_4 and F_5. For the function F_1, which is a sinusoidal function of θ_1, $I(F_1; \theta_2) \approx 0$, and $F_1(\theta_1)$ shows the expected functional relationship. Similarly for F_5, $I(F_5; \theta_1) \approx 0$ and the noisy exponential decay is shown for $F_5(\theta_2)$ (the relationship remains noisy because F_5 is a random function through η_5, Eq. (D.17)). The pairwise I_{norm} plots in the bottom of D.2B (weakly) group F_1, F_2, F_3, and F_4 (on account of their common dependence on θ_1), and place F_2, F_3, F_4, and F_5 in a second group

(on account of their common dependence on θ_2). Although we have only a single function that varies with θ_1 alone (F_1) and only a single function that varies with θ_2 alone (F_2), we can still make out the idealized structure shown in 7.4B above, with $F_1 \in \mathcal{F}_{\theta_1}$, $F_2, F_3, F_4 \in \mathcal{F}_{\theta_1,\theta_2}$, and $F_5 \in \mathcal{F}_{\theta_2}$. Note that $I_{\text{norm}}(F_2; F_3)$ is particularly large, due to their similar monotonic dependences on θ_1 and θ_2.

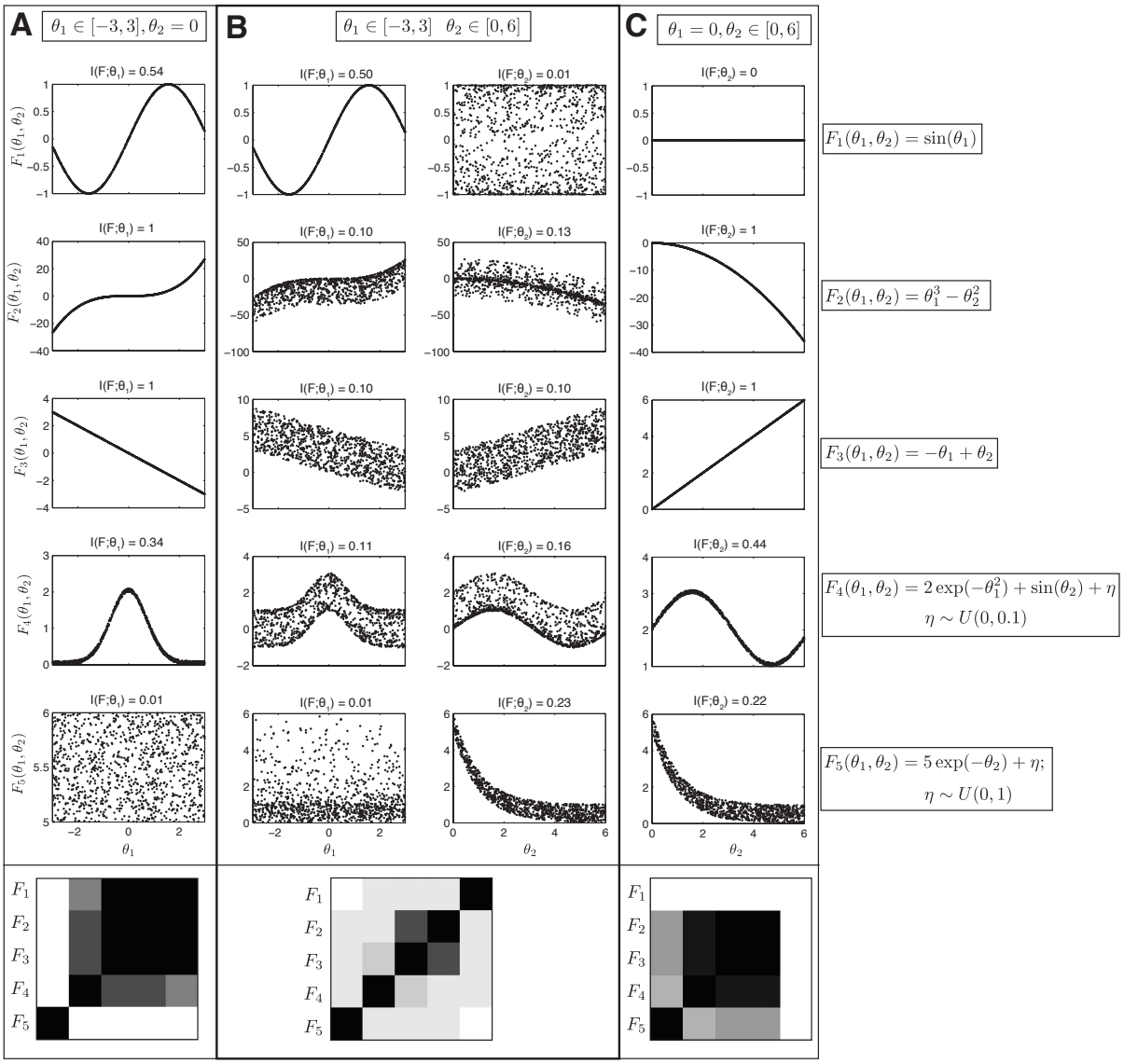

Figure D.2: **Mutual information structure between five functions of random variables.** In this example, five functions of two random variables, θ_1 and θ_2, Eqs. (D.13)–(D.17), are shown, as labeled to the right of the plot and explained in the main text. Results are computed from 1 000 uniform random samples of the parameter space (θ_1, θ_2). In **A**, $\theta_1 \in [-3, 3]$, but $\theta_2 = 0$ is held constant; in **B**, $\theta_1 \in [-3, 3], \theta_2 \in [0, 6]$, and in **C**, $\theta_1 = 0$ is held constant while $\theta_2 \in [0, 6]$. Plots of the five functions as a function of θ_1, θ_2 are shown in the upper portions of the plots, along with corresponding calculations of I_{norm}, as labeled. In the lower portion of each plot, I_{norm}, between all pairs of the five functions is plotted, using black for $I_{\text{norm}} \geq 0.5$, and white for $I_{\text{norm}} = 0$.

Now, let us suppose that we constrain variation in $\theta_2 = 0$ so that only θ_1 is free to vary in its usual domain. This situation is shown in Fig. D.2A, for F_1 through F_5 as a function of θ_1 (variation of these functions with θ_2 is not shown, as θ_2 is fixed). For the functions F_1, F_2, F_3 and F_4, the variation with θ_1 is revealed without the contaminating 'noise' from the unaccounted-for variation with θ_2 (apart from the random function F_4, which retains the stochastic relationship from η_4). Importantly, these functions that depend on θ_1, now exhibit high mutual information with one another, forming a strong group at the bottom of Fig. D.2A. In this plot, F_5 is distinguished, because with θ_2 fixed, F_5 is simply a uniform random number generator such that its outputs that do not correlate with the free parameter of interest, θ_1, nor hence any of the other functions. This demonstrates the theory developed in Sec. 7.2.1 above: the common variation in F_1, F_2, F_3, and F_4 due to their dependence on θ_1 is picked up by measuring pairwise mutual informations. In this case, the structure is similar to that shown in Fig. 7.4A: measuring this pairwise mutual information matrix, allows us to discern a set of operations with a common underlying variation as $F_1, F_2, F_3, F_4 \in \mathcal{F}_{\theta_1}$, and $F_5 \in \mathcal{F}_0$.

For the final regime, in which $\theta_1 = 0$ is fixed and $\theta_2 \sim U(0, 6)$, we find similar results, as shown in Fig. D.2C. In this case, those operations that vary with θ_2: F_2, F_3, F_4, and F_5, are grouped in the pairwise mutual information matrix, and the deterministic F_1, which outputs a constant, $F_1 = \sin(0) = 0$, is distinguished. Note that the stochastic component of the random function F_5 is large (compared with the variation in F_5 due to θ_2), and weakens the I_{norm} between F_5 and the other operations F_2, F_3, and F_4. This effect simulates the randomness in the functional mapping for real, finite time series.

In this first simple example, we have demonstrated how functional dependencies give rise to mutual information structures that can be computed empirically. From these structure, we can infer how many parameters are varying in the underlying model, and which functions contain information about them.

D.1.3 Example: A family of functions

Until now we have not explicitly considered the relative strength of dependencies on θ_1 and θ_2. For example, in the case that $I(F_1;\theta_1) \gg I(F_1;\theta_2)$ when both θ_1 and θ_2 are free to vary, the dependence of F_1 on θ_2 may be too weak to measure. To investigate this phenomenon, as a final example we investigate the structure of mutual information matrices for two-parameter systems for functions with a range of relative dependencies on θ_1 and θ_2. We consider a parametrized family of functions, $F(\theta_1, \theta_2; k)$, where each feature in this class is a deterministic function of two random variables, θ_1 and θ_2 as:

$$F(\theta_1, \theta_2; k) = k \sin(\theta_1) + (1-k)\theta_2^2, \tag{D.18}$$

where k parameterizes the degree to which F depends on θ_1 and θ_2: when $k = 1$, $F(\theta_1, \theta_2; k = 1) = \sin(\theta_1)$ depends on θ_1 only, and when $k = 0$, $F(\theta_1, \theta_2; k = 0) = \theta_2^2$ depends on θ_2 only. Both $\theta_1 \sim U(0, 2\pi)$ and $\theta_2 \sim U(0, 2)$, are uniformly-distributed random variables in their respective domains.

First we sample 200 points from this parameter space, as shown in Fig. D.3B, to generate the plots shown in Fig. D.3A. Relationships between F (for $k = 0, 0.25, 0.5, 0.75,$ and 1) and each of θ_1 and θ_2 are plotted in Fig. D.3A. As k changes, the changes in the relative mutual informations $I(F;\theta_1)$ and $I_{\text{norm}}(F;\theta_2)$ can be seen visually. When $k = 0$, $I(F;\theta_1) = 0$ and $I(F;\theta_2)$ is maximal; conversely, when $k = 1$, $I(F;\theta_2) = 0$ and $I(F;\theta_1)$ is maximal. When $k > 0$, $I(F;\theta_1) > 0$, and as k increases, $I(F;\theta_1)$ increases with it. The unexplained variation in F due to the other random variable θ_2 reduces $I(F;\theta_1)$ as k decreases. Similarly, when $k < 1$, $I(F;\theta_2) > 0$, and $I(F;\theta_2)$ decreases as k increases due to the incorporation of variation due to the random variable θ_1 that is unaccounted for by θ_2. This variation in I_{norm} was computed and is plotted in Fig. D.3C, this time using 2 000 points in the parameter space to yield more reliable estimates of this quantity. Note that the maximal $I_{\text{norm}}(F;\theta_1)$ is lower than that of $I_{\text{norm}}(F;\theta_2)$ because the sinusoidal variation, $\sin(\theta_1)$, is not a one-to-one correspondence, unlike the parabolic variation of θ_2^2.

Finally we investigate measurements of pairwise mutual information between members of this family of functions, as shown in Fig. D.3D for fifty equally-spaced

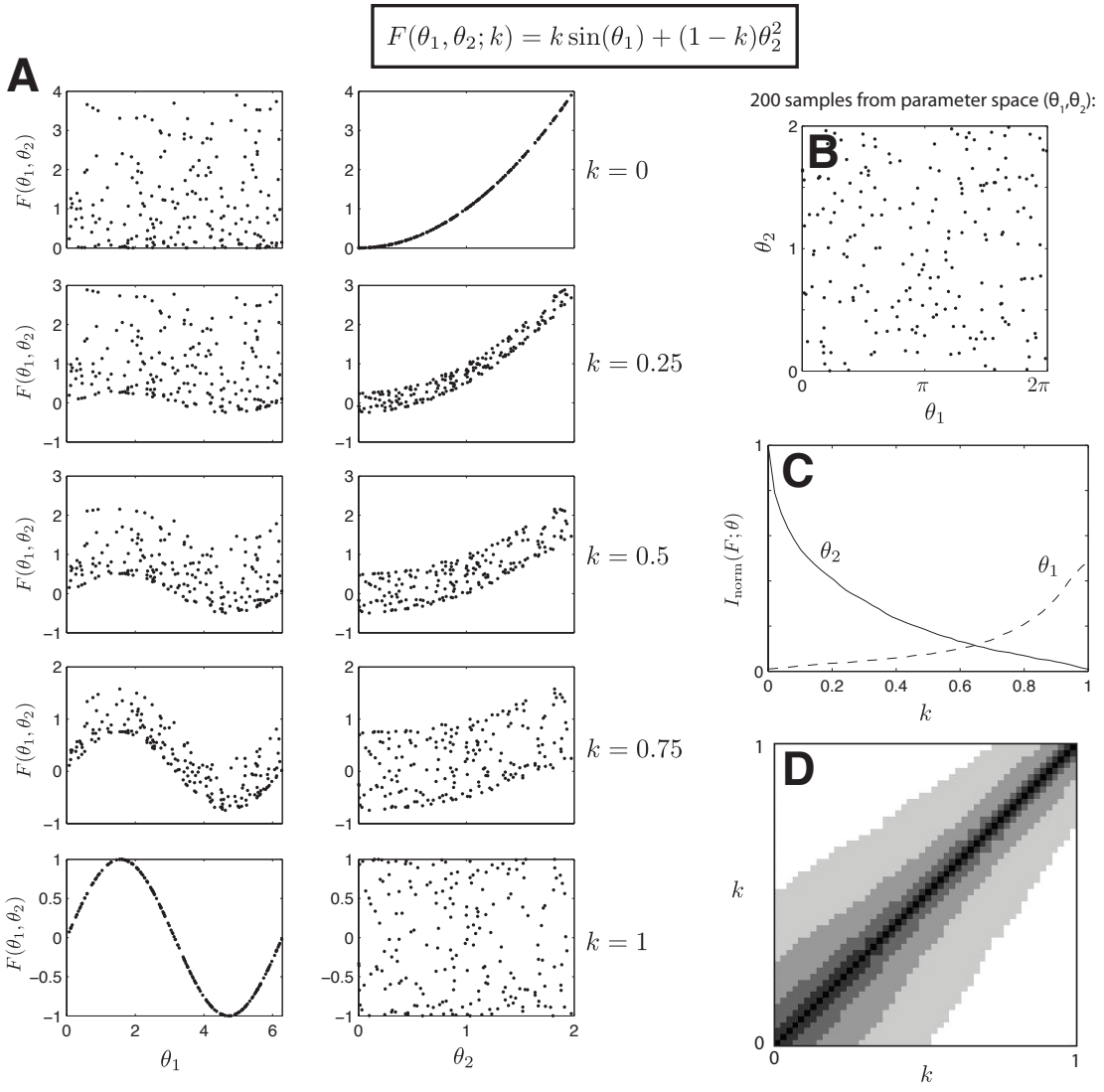

Figure D.3: **Example of correlations between parameters θ_1 and θ_2 and a simple function $F(\theta_1, \theta_2; k) = k\sin(\theta_1) + (1-k)\theta_2^2$, for $k \in [0, 1]$.** When k is low, most of the variation in F can be explained by θ_1 and, alternatively, when k is high, most of the variation in F can be explained by θ_2. By setting $k = 0$, we simulate the case where F becomes a function of θ_2 alone, and by setting $k = 1$, we simulate the opposite case: where F is a function of just θ_1. When both vary together ($0 < k < 1$), estimating just θ_1 or just θ_2 gives a relatively weak correlation to F. As shown in **B**, the figures in **A** were generated from 200 uniformly-distributed random samples in the parameter space (θ_1, θ_2) for $\theta_1 \in [0, 2\pi]$ and $\theta_2 \in [0, 2]$, and varying over $k = 0, 0.25, 0.5, 0.75$, and 1 gives the plots of $F(\theta_1, \theta_2)$ as a function of θ_1, and of θ_2 in **A**. In **C**, we computed a normalized mutual information measure, I_{norm}, between F and θ_1 and θ_2 as a function of k for 50 increments of k and 2 000 points in the parameter space (θ_1, θ_2). In **D**, pairwise I_{norm} were calculated between F as a function of $k \in [0, 1]$ for 2 000 points in the parameter space (θ_1, θ_2). Black indicates high values ($I_{\text{norm}} = 1$), and white low values ($I_{\text{norm}} = 0$).

increments of $k \in [0, 1]$, and hence fifty different functions in this family. The plot shows measurements $I_{\text{norm}}(F_i, F_j)$, measured between all pairs of the functions. The diagonal corresponds to self-informations: $I_{\text{norm}}(X, X) = I(X; X)/H(X, X) = 1$, and we see a characteristic pattern whereby functions with similar values of k exhibit high I_{norm} to one another. This smooth variation is in contrast from the expected block structure plotted in 7.4, to partition $\mathcal{F}_{\theta_1}$, $\mathcal{F}_{\theta_1,\theta_2}$, and $\mathcal{F}_{\theta_2}$. We hence discover that even for these very simple, deterministic functional dependencies on θ_1 and θ_2, that mutual informations calculated between functions of multiple variables vary with the strength and type of relationship between the function and the parameters. Functions that vary in different ways with different parameters, like for one function with low k and another with high k, the common variation is too weak to be estimated accurately, so that $I_{\text{norm}} \approx 0$.

D.1.4 Summary

Two further numerical examples, exploring the relationships between the variation of parameters and the mutual information measured between functions of them, are presented in Sec. D.1 of the appendix. In the first example, in Sec. D.1.2, we focused on understanding how scatter plots between five functions of θ_1 and θ_2 were related to the free variation of parameters and resulting pairwise mutual information matrices. For example, note that each square in the mutual information matrices shown in Fig. D.1 represents a relationship between two functions that can be visualized as a scatter plot of one function against the other. These scatter plots allowed us to visualize how randomness in functions can affect the calculation of mutual information between functions with common underlying dependencies and thereby give an insight into the actual plots from which the mutual information measures are calculated. As a second example, in Sec. D.1.3, we studied a family of functions parameterized by their relative dependence on θ_1 and θ_2 and showed that those functions with similar functional dependencies on θ_1 and θ_2 have high mutual information.

The numerical example explored in this section (and those in Sec. D.1 of the appendix) hint at some key practical limitations of our method. Firstly, measuring the mutual information between operations (using the histogram-based method described

in Sec. 7.2.2.2) can be difficult for datasets containing less than 1 000 samples. The amount of data required to make a reliable mutual information measurement depends on the level of randomness in a given function relative to its variation with one of the underlying parameters, and the number of parameters. For example, a key challenge involves detecting positive mutual information between two functions that vary with multiple parameters simultaneously and in different ways.

To recapitulate: by interpreting parameters as random variables and constraining their variation, characteristic mutual information structures can be measured between sets of features that depend on these parameters. This structure is relatively obvious for a one-parameter system, but the patterns can be more subtle for higher-dimensional systems. Results for features extracted from real time-series datasets are presented in Sec. 7.3.

D.2 Datasets

D.2.1 Empirical time series

One thousand empirical time series were selected from our database of real-world and synthetic time series, including iterative maps, outputs from systems of ordinary and stochastic differential equations, meteorological time series, astrophysical signals, biomedical recordings, and audio signals (Appendix E). This set of 1 000 time series was formed using average linkage clustering on the full time-series database represented using a representative interdisciplinary set of 200 time-series analysis operations (see Sec. 4.1.2). Time series in this set range in length from 1 000 samples through to 27 345 samples. Some excerpts from some time series in this dataset are shown in Fig. D.4. Because these empirical time series cover such a wide variety of dynamics, from heartbeats to outputs from dynamical systems, will be used to construct a representative set of 500 time-series analysis operations in Sec. 7.2.2.3.

D.2.2 Self-affine time series

Self-affine time series [90], each containing 5 000 samples, were generated with a range of scaling exponents, α. This dataset was used in Sec. D.2.2 to automatically retrieve

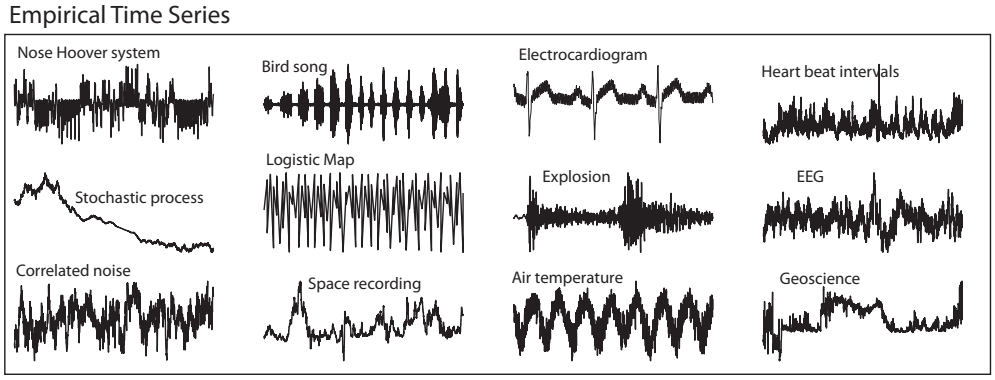

Figure D.4: **Empirical Time Series**. Excerpts from time series taken from the *empirical time series* dataset, which includes a diverse range of synthetic and real-world systems. This reduced set of 1 000 time series was taken from the time-series database developed in this thesis.

good estimators of the scaling exponent, α, of self-affine time series. Self-affine time series can be divided into two categories: (i) *fractional Gaussian noise* (fGN) time series, which have $-1 < \alpha < 1$, and (ii) *fractional Brownian motion* (fBM) time series, which have $1 < \alpha < 3$ [147]. The *Hurst exponent*, H, relates to the scaling exponent by $H = (\alpha + 1)/2$ for fGN and $H = (\alpha - 1)/2$ for fBM. It can take values between 0 and 1; if successive increments for fBM (or successive values for fGN) are positively correlated, $H > 0.5$ (*persistence*), and if they are anti-correlated, $H < 0.5$ (*anti-persistence*) [13]. In this dataset, time series were generated using both the *Fourier filtering method* and the *random midpoint displacement method* across a range $-1 \leq \alpha \leq 3$ [90, 152]. This dataset contains 451 time series in total. Since each time series can be characterized by a single scaling exponent, α, we aim to automatically uncover this one-dimensional structure with our method.

D.2.3 Logistic Map time series

We generated a dataset of time series from the *Logistic Map* of length $N = 10\,000$ (after removing an initial transient of 20 000 samples). The Logistic Map is defined by the following iterative relation:

$$x_{n+1} = Ax_n(1 - x_n), \tag{D.19}$$

which contains a single control parameter, A [12, 67]. Time series were synthesized using a range of A: $A = 3.0, 3.2, 3.4$ and $A = 3.50$, 3.51, 3.52, ..., 4.00. In total,

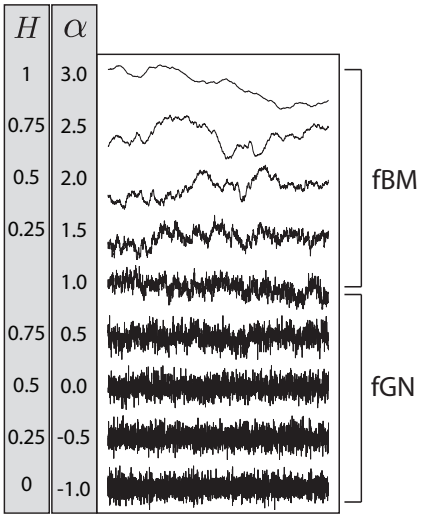

Figure D.5: **Self-affine time series**. Self-affine time series can be characterized as either fractional Gaussian noise (fGN), which have scaling exponents $-1 < \alpha < 1$, or fractional Brownian motion (fBM), which have scaling exponents $1 < \alpha < 3$. The related *Hurst exponent*, H, characterizes the self-similarity of a time series, from anti-persistent, $H = 0$, to random $H = 0.5$, to persistent $H = 1$.

the dataset contains 159 time series and was used in Sec. 5.1.2 to select time-series analysis methods that are good estimators of λ. For each A, we generated three time series using a different initial condition chosen randomly as $x_1 \sim U(0, 1)$. In this way, the model, Eq. (D.19), which maps from a single parameter A to a time series $\mathcal{X}$, is random. The Lyapunov exponent, λ, quantifies the amount of chaos in the system: a bounded dynamical system with $\lambda > 0$ is chaotic, and λ encapsulates the average rate at which predictability is lost [12]. For a given A, the Lyapunov exponent λ was estimated numerically according to

$$\lambda \approx \log A \left\langle \log(A|1 - 2x|) \right\rangle_x , \tag{D.20}$$

where the average $\langle \cdot \rangle_x$ is performed over a 20 000-sample simulated Logistic Map trajectory, which was adequate to obtain a sufficiently converged estimate for our purposes [12]. Example time series in this dataset are shown in Fig. D.6. In this work, we aim to reproduce the one-dimensional structure in these time series with the appropriate dynamical quantity, the Lyapunov exponent, λ.

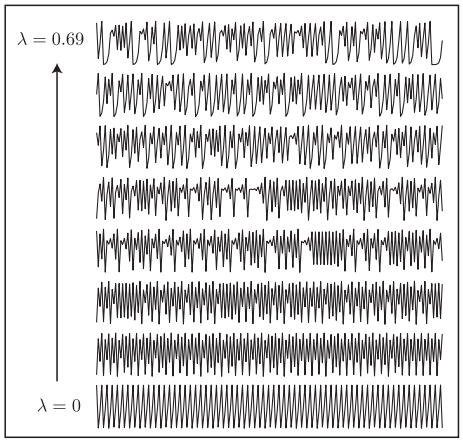

Figure D.6: **Time series generated by the Logistic Map are characterized by their Lyapunov exponent,** λ. The Lyapunov exponent is 0 for periodic series, while more chaotic time series have higher λ. For a given value of the control parameter of a Logistic Map, the value of λ can be estimated by simulating long trajectories using Eq. (5.3).

D.2.4 Fixed-lag autoregressive time series

Time series datasets were generated according to a simple autoregressive process with autoregressive parameter α and time lag τ according to:

$$x_t = \alpha x_{t-\tau} + n_t, \tag{D.21}$$

where $n_t \sim \mathcal{N}(0, 1)$ are independent random samples from a Gaussian distribution. For a fixed τ, a dataset is generated by sampling $\alpha \in [-1, 1]$. We generated time series with 5 000 samples by taking $t = 1, 2, ..., 5\,000$. For $\tau = 1$, we use an initial condition $x_1 = 0.1$ and generate 500 time series, where α is varied using equally-spaced values across its domain from -1 to 1.

We also generated subsequent datasets containing 100 time series for each of $\tau = 2$, 3, 4, and 5, we generated 100 time series, again using an equally-spaced range for $\alpha \in [-1, 1]$. For these datasets, the initial conditions $x_1, x_2, ..., x_\tau$ were chosen as the values of the noise process $n_1, n_2, ..., n_\tau$. Example time series generated using Eq. (D.21) and their autocorrelation structure for $\alpha = -1, -0.5, 0, 0.5, 1$ are shown in Figs. D.7A and D.7B for $\tau = 1$ and $\tau = 2$, respectively. The autocorrelation at lag τ is approximately α for these time series. Changing α clearly has a dramatic effect on the time series produced by this model, Eq. (D.21), and we therefore expect many of the features in our library of time-series analysis operations to vary with α across

this dataset.

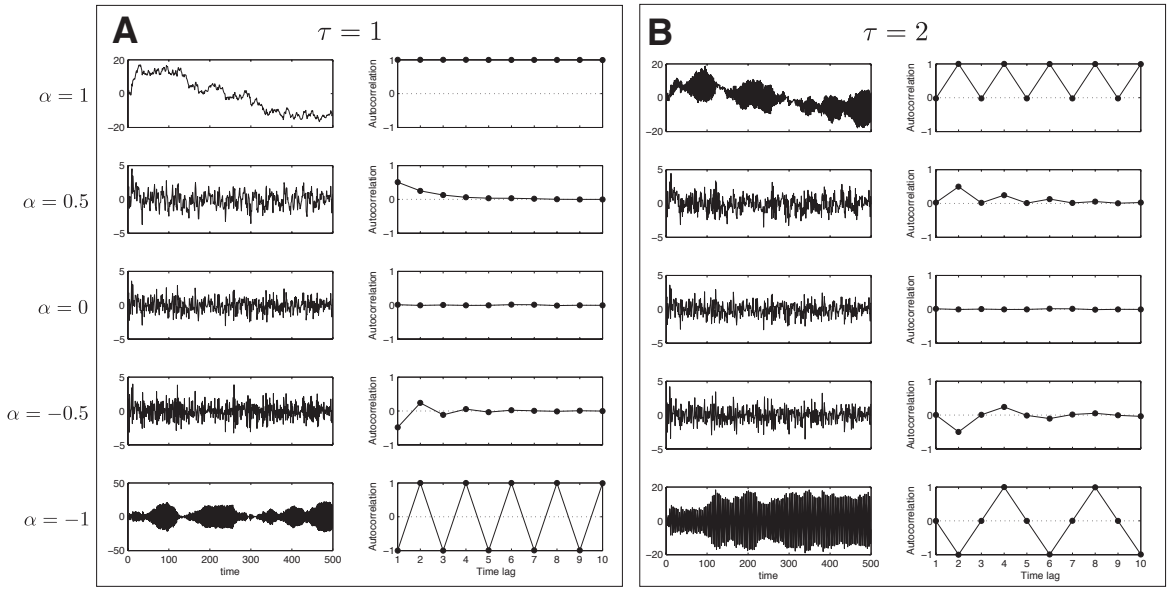

Figure D.7: **Autoregressive time series**. These time series are produced by Eq. (D.21), for a given time lag τ. Here we show some time series for **A** $\tau = 1$, and **B** $\tau = 2$. In each plot, time series are shown in the left panels and autocorrelation functions in the right panels for each of $\alpha = -1, -0.5, 0, 0.5$, and 1. In each case, the extreme autocorrelation pattern at $\alpha = \pm 1$ is dampened as $|\alpha|$ decreases to a function of zeros at $\alpha = 0$. Only the first 500 samples of time series are plotted for clarity, and autocorrelation functions are calculated from the full 5 000-sample time series.

D.2.5 Correlated, bimodal time series

Time series of 5 000 samples were generated according to a two-parameter stochastic time-series model. At each time step, the time series value, x_t, is sampled from either $x_t \sim \mathcal{N}(0, 1)$ or $x_t \sim \mathcal{N}(\delta, 1)$; the parameter δ thus controls the difference in means of the two states. Defining a binary state vector v, where $v = 0$ (state 1) or $v = 1$ (state 2), this process can be written as

$$x_t = n_t + v_t \delta, \tag{D.22}$$

where $n_t \sim \mathcal{N}(0, 1)$. In this way, the parameter δ acts as a fixed offset to the uncorrelated n_t when $v = 1$. We can introduce correlations between successive states by defining a simple Markov process:

$$v_t = Y_t v_{t-1} + (1 - Y_t)(1 - v_{t-1}), \tag{D.23}$$

where Y_t is a random variable from a Bernoulli process, which takes a value of 1 with probability α, and a value of 0 with probability $1-\alpha$. Thus, the state v_t remains the same as at the previous time step, v_{t-1}, with probability α, and changes otherwise. When $\alpha = 0.5$ successive states are uncorrelated, when $0.5 < \alpha \leq 1$ successive states are positively correlated, and when $0 \leq \alpha < 0.5$, successive states are negatively correlated.

Time series are generated from this stochastic model by setting the fixed offset, δ, between the two states, and the probability, α, of remaining in the same state. There is a somewhat complicated relationship between these two variables: for example, when $\delta = 0$, changing α has no effect on the time series (since both states draw values from the same distribution), and the value of α has more of an effect on the resulting time series as δ increases (as the two states become increasingly distinct). In this work, we generate three time-series datasets from this model as follows: (i) varying α (spaced equally in the range $0 \leq \alpha \leq 1$) while keeping $\delta = 3$ fixed, (ii) varying δ (spaced equally in the range $0 \leq \delta \leq 6$) while keeping $\alpha = 0.5$ fixed, and (iii) varying both α and δ, in which case (instead of equally-spaced values of α and δ) random samples for $\alpha \sim U(0,1)$ and $\delta \sim U(0,6)$ were taken. In each case, the values of α and δ were rounded to three decimal places. Note that the random process n_t was generated from a fixed random seed so that the model in fact maps from the parameter space to time series instances deterministically. A schematic of this process and the time series it generates is illustrated in Fig. D.8.

D.2.6 Stochastic sine map

Time series of 5 000 samples were generated from a stochastic sine map described by Freitas et al. [85]. The model is defined by

$$x_{t+1} = \mu \sin(x_t) + Y_t \eta_t, \tag{D.24}$$

where μ is a parameter, Y_t is a random variable from a Bernoulli process, and η_t are independent random samples from a uniform distribution between $-b$ and b. The Bernoulli random variable $Y_t = 1$ with probability q and $Y_t = 0$ otherwise, with probability $1-q$. As shown in Figs. D.9B and D.9E, when $\mu = 2.4$, there are two

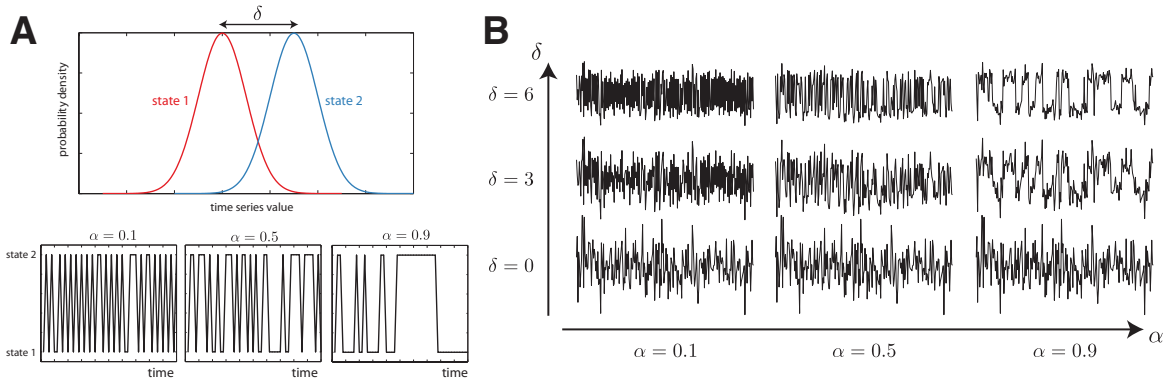

Figure D.8: **Correlated, bimodal time series**. **A** The generative process of these time series consists of two states, with probability densities specified by Gaussian distributions of unit variance, centered at 0 and δ, respectively. The parameter α specifies the probability of staying in the same state at two consecutive time steps and hence controls correlations in the time series: when $0 < \alpha < 0.5$, successive states are anti-correlated; when $\alpha = 0.5$, successive states are uncorrelated; and when $0.5 < \alpha < 1$, successive states are correlated. In the lower plots, we show representative time series of 50 points for the state vectors for this process for $\alpha = 0.1$, $\alpha = 0.5$, and $\alpha = 0.9$. **B** A grid of (the first 250 samples of z-scored) time series generated by this correlated, bimodal time-series model for $\alpha = 0.1, 0.5, 0.9$ and $\delta = 0, 3, 6$. When $\alpha = 0.1$, successive values are anti-correlated, and when $\alpha = 0.9$, successive values are positively correlated. As δ increases, changes in α have a greater effect on the resulting time series because the probability distributions of values from each state are more separated.

limit cycles, and the system will attract onto one of them depending on its initial condition. The parameter b controls the magnitude of random perturbations added to the otherwise deterministic system; such perturbations occur with probability q and can switch the system from one limit cycle to the other.

The choice $\mu = 2.4$ is a sensible parameter for this system, but we also studied variations across μ, b, and q, in all combinations. Parameters are allowed to vary across the following ranges: $0.5 \leq \mu \leq 4$, $0 \leq b \leq 3$, and $0 \leq q \leq 1$. When parameters are fixed, they take the following values: $\mu = 2.4$, $b = 1$, and $q = 0.5$. First, we generated datasets of 100 time series each by varying just a single parameter through 100 equally-spaced increments across its range: for μ, b, and q. Second, while keeping one parameter fixed, all pairs of parameters were varied together by sampling uniformly at random from the ranges of each, producing three datasets, each containing 200 time series. Finally, a dataset was generated in which all three parameters are allowed to vary together by taking 500 uniformly-distributed random

samples in the range of each parameter to produce a dataset of 500 time series. In total, therefore, we generated seven datasets from this stochastic sine map system, Eq. (D.24).

Schematics of the generative process underlying each dataset and the time series it generates are shown in Fig. D.9. For low μ, as in Figs. D.9A and D.9D, there is a single stable fixed point at $x_{t-1} = x_t = 0$ that trajectories attract onto. As the noise probability, q, and amplitude, b, increase, trajectories have less time to relax to equilibrium and the resulting time series become noisier. The nominal value of μ for this map is $\mu = 2.4$ [85], as shown in Figs. D.9B and D.9E. In this case, the map has two stable limit cycles (plotted in green in Fig. D.9E) separated by the unstable fixed point at $x = 0$. In this case, additive noise can switch the system from one stable limit cycle to the other. Finally, for $\mu = 4.0$, shown in Figs. D.9C and D.9F, there are no stable limit cycles. In this work, we investigate cases in which μ, b, and q in all combinations are free to vary.

D.2.7 Noisy sine waves with linear trends

Time series of 2 000 samples were generated from the following time-series model:

$$x_t = \sin(2\pi t/T) + \eta n_t + mt/N, \tag{D.25}$$

where $n_t \sim \mathcal{N}(0, 1)$. The model has three free parameters: (i) the period of sine waves, T, (ii) the standard deviation, η, of additive uncorrelated Gaussian-distributed noise, and (iii) the net displacement, m, of a linear trend added to the signal. Time-series datasets are constructed for which all combinations of parameters are varied. We vary η in the range $0 \leq \eta \leq 3$, m in the range $-5 \leq m \leq 5$, and T in the range $10 \leq T \leq 100$ samples. When holding parameters fixed, we used $\eta = 0$, $m = 0$, and $T = 30$ samples. We formed three datasets containing 100 time series each by varying each parameter at equally-spaced increments throughout its range while holding the other two parameters fixed. Another three datasets containing 500 time series each were generated by keeping one parameter fixed and varying the other two by taking uniformly-distributed random values within their respective domains. Finally we generated a dataset containing 1 000 time series in which all three parameters are free

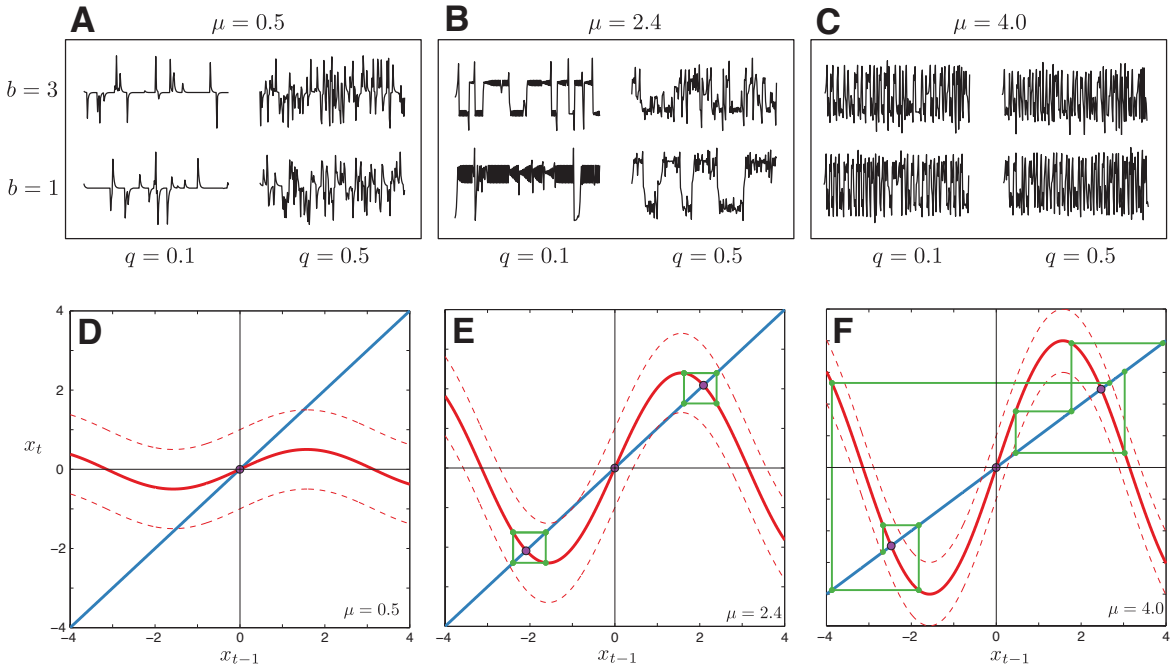

Figure D.9: **Stochastic sine map time series.** The stochastic sine map is defined by Eq. (D.24), as described by Freitas et al. [85]. The original map used $\mu = 2.4$; here we allow μ to vary. The first 200 samples of time series are plotted for $b = 1, 3$ and $q = 0.1, 0.5$ in **A**, **B**, and **C** for $\mu = 0.5, 2.4$, and 4.0, respectively. The relationship between successive time points, (x_{t-1}, x_t), is plotted for $\mu = 0.5, 2.4$, and 4.0 in **D**, **E**, and **F**, respectively. In these plots, the sinusoidal recurrence relationship is plotted as a red curve, equality: $x_t = x_{t-1}$ is plotted as a blue line, fixed points are plotted as purple circles, sample limiting trajectories are plotted as green lines, and the deviation of the theoretical sinusoidal recurrence relationship by noise is shown for $b = 1$ as dashed red curves.

to vary; values for each parameter are taken uniformly at random from their domains. Examples of time series generated by this process are shown in Fig. D.10.

D.3 Results

In this section we present additional results relating to the feature-based dimensionality framework studied in Chapter 7.

D.3.1 Data matrices

In this section we present a selection of data matrices for four one-dimensional datasets studied in Chapter 7: Logistic Map time series (Sec. 5.1.2) in Fig. D.11A, correlated bimodal time series with α free and δ fixed (Sec. D.2.5) in Fig. D.11B, self-affine series (Sec. D.2.2) in Fig. D.11C, and noisy sinusoids (Sec. D.2.7) in Fig. D.11D.

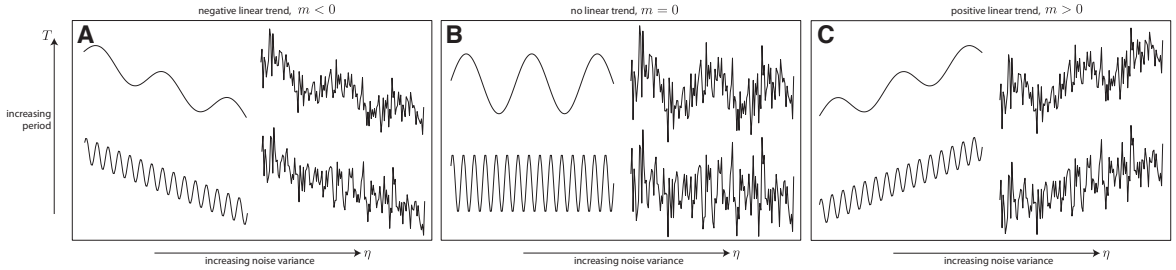

Figure D.10: **Time series generated from the noisy sine waves with linear trends model**. Time series in this dataset are controlled by three free parameters: the period of the sinusoid, T, the variance of added noise, η, and the net displacement of an added linear trend, m, as per Eq. (D.25). We illustrate: **A** a negative trend: $m < 0$, **B** no trend: $m = 0$, and **C** a positive trend $m > 0$. Four examples in each plot illustrate the types of time series that can be obtained from this model, which allows independent variation in linear trend, noise standard deviation, and sinusoidal period.

Columns of the data matrices contain all of the 9 000 operations with no special-valued outputs, and rows contain all the time series in the dataset. As shown using the time series examples adjacent to each data matrix, simply reordering the rows and columns of data matrices using hierarchical clustering (cf. Sec. 3.4.2) yields a sensible ordering of the time series: by their approximate Lyapunov exponent in Fig. D.11A, by their autocorrelation in Fig. D.11B, by their scaling exponent in Fig. D.11C, and noise standard deviation in Fig. D.11D.

As explained above, common variation of sets of operations across the dataset are due to a common dependency on an underlying model parameter, θ_1, and are easily detected by inspecting these plots because the columns have been reordered to place similar operations next to one another. For example, a steady increase (black to white) or decrease (white to black) across the dataset is a common feature of these data matrices. Also, as can be seen for some sets of operations in Fig. D.11B, an increase and then decrease (black to white to black) or vice-versa is another type of functional dependency. Sets of operations that do not vary with the trend are also detectable as 'noisy' columns, at the left of the data matrix in Fig. D.11B for example, or towards the middle of Fig. D.11C. This partition into operations that share mutual information due to a common dependency on an underlying parameter θ_1, i.e., $\mathcal{F}_{\theta_1}$, and those with purely stochastic outputs that are uninformative of this important variation, i.e., $\mathcal{F}_0$, is already evident from the structure of these clustered

data matrices.

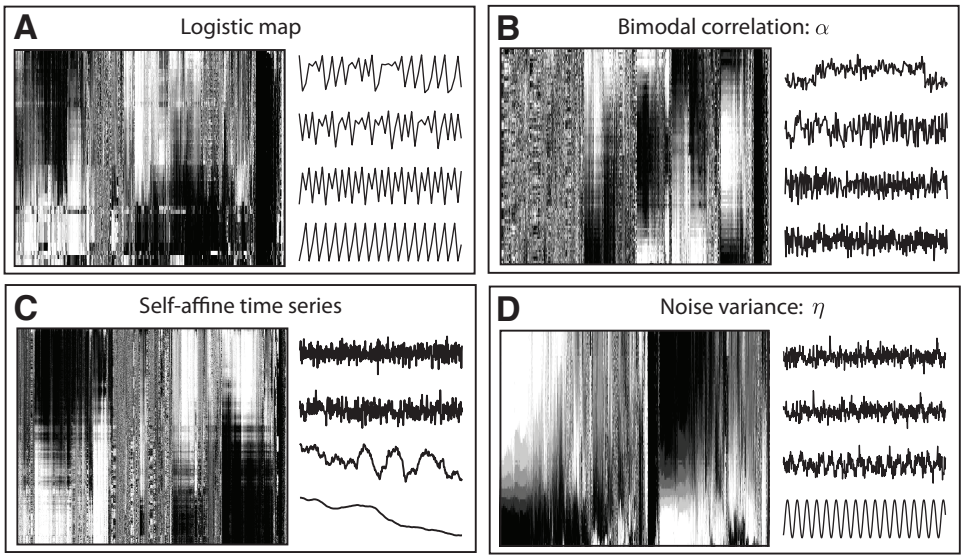

Figure D.11: **Example data matrices with annotated time series from four selected one-dimensional systems.** **A**: Logistic Map time series with variable control parameter A (Sec. 5.1.2), **B**: The state-change probability, α, for successive draws from a mixed bimodal Gaussian model (Sec. D.2.5), **C**: Self-affine time series generated with different scaling exponents, α (Sec. D.2.2), **D**: Noisy sine waves with a variable standard deviation of added noise, η (Sec. D.2.7). In each figure, each time series in the dataset occupies a row, and each column represents an operation; dark indicates low outputs of an operation on a time series, and white indicates high values. Each column has been normalized to the unit interval using the scaled sigmoidal transform, Eq. (3.2). Both rows and columns in these data matrices have been reordered by hierarchical linkage clustering using Euclidean distances (rather than absolute correlation distances for operations) to aid visualization. In each plot, four selected time series have been annotated adjacent to their corresponding row in the data matrix.

D.3.2 Pairwise operation mutual information matrices

In this section we plot pairwise mutual information matrices calculated between pairs of operations across the initial set of 500 operations described in Sec. 7.2.2.3 (note that fewer operations will be represented after removing those with special-valued outputs). Results for ten selected one-parameter systems are shown in Fig. D.12. These plots use reduced sets of 500 operations to aid calculation times, as described in Sec. 7.2.2.3. In each case, groups of operations with high I_{norm} to one another can be seen, that we infer to be varying with the free parameter in each case. This pattern persists across a wide variety of dynamical changes, from the Lyapunov exponent, λ,

of Logistic Map time series (Fig. D.12A), the scaling exponent, α, of self-affine time series (Fig. D.12B), the autocorrelation of simple linear auto-regressive processes (Fig. D.12C), the parameters of the mixed bimodal correlated time-series model, α and δ (Figs. D.12D and E), the parameters of a stochastic sine map, μ and q (Figs. D.12F and G), and the noise standard deviation, η, the period of an underlying sinusoid, T, and net displacement of a linear trend, M (Figs. D.12H–J). Note that only 10 of the 15 systems are shown here; other systems showed similar behavior to that represented in Fig. D.12. We also emphasize the fact that for the dataset of 1 000 empirical time series (described in Sec. D.2.1) and also for a dataset of 1 000 random time series generated from a $x_t \sim U(0, 1)$ process, that these pairwise operation mutual information matrices are almost completely devoid of significant off-diagonal entries (not shown); hence the mutual information present between operations in these systems are characteristic of the strong constraints on these systems (as demonstrated for the twelve functions studied in Sec. D.1.1, for example). The results show that the expected correlations between operations can indeed be detected through the calculation of mutual information. Therefore, given a diverse time-series library such as that used in this work, a wide range of often subtle variation in dynamics can be detected by multiple operations and produce informative structures in these mutual information matrices.

D.3.3 Iterative filtering with a threshold

In this section we show a figure containing more examples of iterative filtering with a threshold, as shown in Fig. 7.7.

D.3.4 Iterative filtering without a threshold

The mean mutual information of operations that have not yet been removed through the course of an iterative filtration process described in Sec. 7.3.2 are shown in Fig. D.14. Results are shown for all fifteen one-dimensional systems studied in this chapter: the fixed-lag autoregressive time series in Figs. D.14A–E for $\tau = 1$–5 (Sec. D.2.4), the correlated bimodal time series for α and δ in Figs. D.14F and G (Sec. D.2.5), the stochastic sine map in Figs. D.14H–J for b, μ, and q (Sec. D.2.6), the Logistic

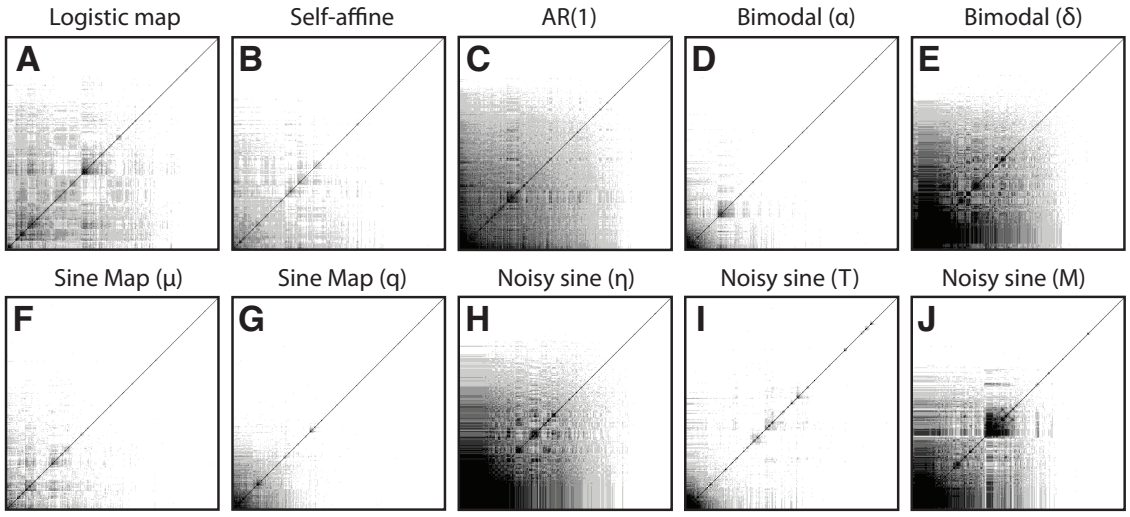

Figure D.12: **Normalized pairwise mutual information matrices between operations for a range of single parameter systems show blocks of operations that vary with the target parameter.** Positive mutual information exists between operations that are sensitive to the variation in the parameter θ_1, across a range of systems representing diverse types of dynamics: **A** the Lyapunov exponent, λ, of the Logistic Map, **B** the scaling exponent, α, of self-affine time series, **C** the lag-1 autocorrelation of an autoregressive process, Eq. (7.9), **D** the correlation, α, and **E** the separation, δ, for a bimodal Gaussian mixed model, **F** the parameter μ and **G** the noise probability, q, of a stochastic sine map, Eq. (7.10), and **H** the noise standard deviation, η, **I** period, T, and **J** linear trend, m, of a noisy sine wave model, Eq. (7.11). Pairwise mutual information calculations are performed across 500 operations, and each matrix has been ordered using hierarchical linkage clustering to place operations with similar behavior close to one another.

Map in Fig. D.14K (Sec. 5.1.2), self-affine time series in Fig. D.14 (Sec. D.2.2), and noisy periodic signals in Figs. D.14M–O for η, T, and m (Sec. D.2.7). Plots show the mean mutual information of operations that have not yet been filtered with the parameter of interest, θ_1, as $\langle I_{\text{norm}}(F_i;\theta_1)\rangle$, across the entire iterative procedure (until just a single operation remains). In most cases, the trend is approximately monotonic and increasing, indicating that at all iterations, an operation is removed with $I_{\text{norm}}(F_i;\theta_1) < \langle I_{\text{norm}}(F_i;\theta_1)\rangle$. Thus, although knowledge of this underlying parameter, θ_1, is not used in the filtration process, in most cases, removing operations in this order strengthens the mean mutual information of those remaining operations to θ_1. In Figs. D.14A–E, we see that, due to the abundance of operations in the database that are sensitive to the basic auto-correlation properties of time series, finding operations with a high mutual information with the autocorrelation at lag $\tau = 1, 2, 3, 4, 5$

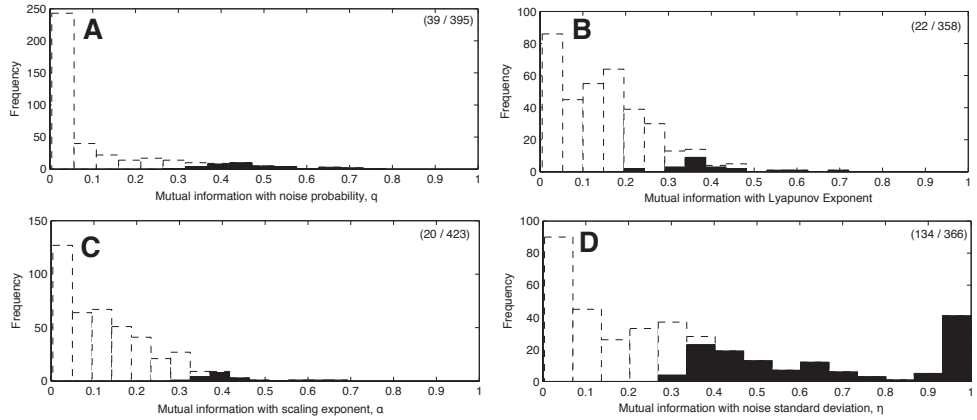

Figure D.13: **Removing operations with low mean mutual information to other operations also tends to remove those with low mutual information to the parameter of interest.** We show the effect for six selected single-parameter systems in this plot, where histograms with fifteen bins are used to represent the distribution of $I_{\text{norm}}(F_i, \theta_1)$ across all operations (dashed) and after filtering (solid). The iterative filtering process involves removing the operation with the lowest $\langle I_{\text{norm}} \rangle$ at each iteration and is repeated until all remaining operations have a mean $\langle I_{\text{norm}} \rangle \geq 0.3$ with other remaining operations. Numbers in the upper right of each plot indicate the number of operations kept from an original set or approximately 500 (this original set contains less than 500 operations because it has first been filtered of constant operations, those with few unique outputs, and those that produced special-valued outputs, see Sec. 7.2.2.3).

in the autoregressive datasets is very successful. Similarly for the state change probability, α (Fig. D.14F), which induces correlations in the bimodal time series, and the noise standard deviation, η (Fig. D.14M), which destroys autocorrelation structure in otherwise periodic time series. Indeed, broad increasing trends are observed for all systems, indicating that removing operations with low mutual information to other operations tends to increase the mean mutual information of remaining operations to the parameter of interest, θ_1.

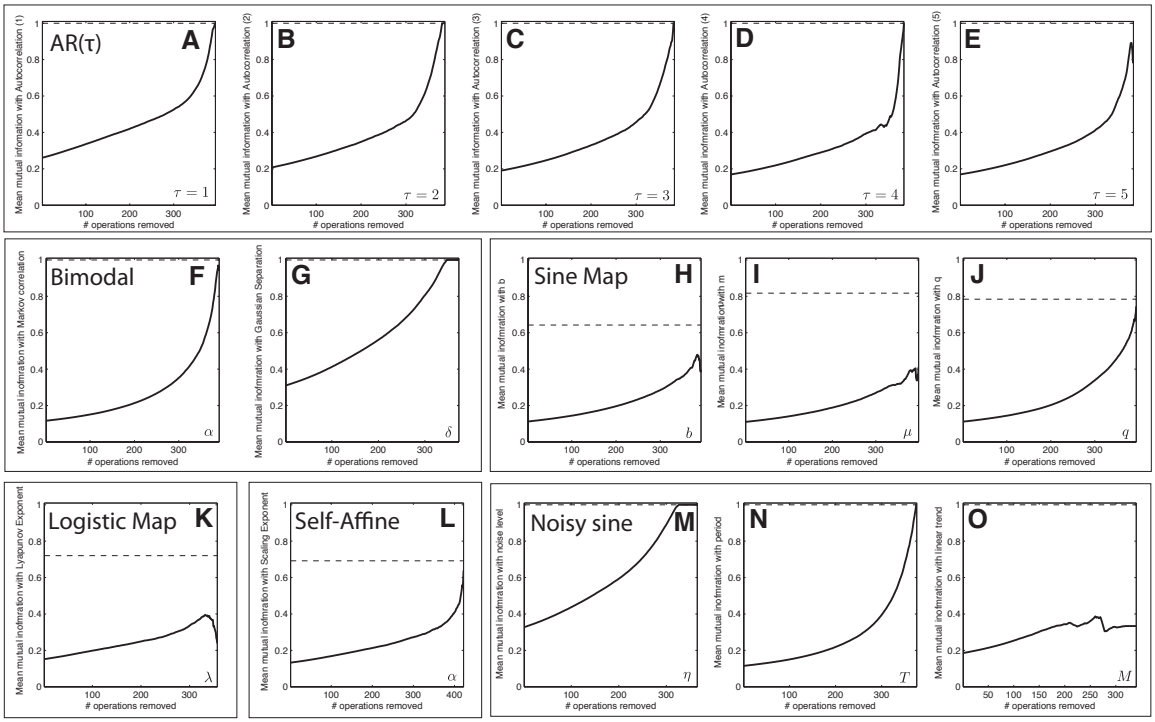

Figure D.14: **Mean mutual information of operations to the parameter of interest, θ_1, across an iterative filtering process for selected one-dimensional systems.** At each iteration of the filtering process (described in the text), the operation with the lowest mean mutual information to other operations is removed and we plot $\langle I_{\text{norm}}(F_i;\theta_1)\rangle$ calculated across the remaining operations, F_i. A dashed line indicates the operation that is most informative of θ_1: the maximum, $\max[I_{\text{norm}}(F_i;\theta_1)]$, across all operations F_i in the initial set.

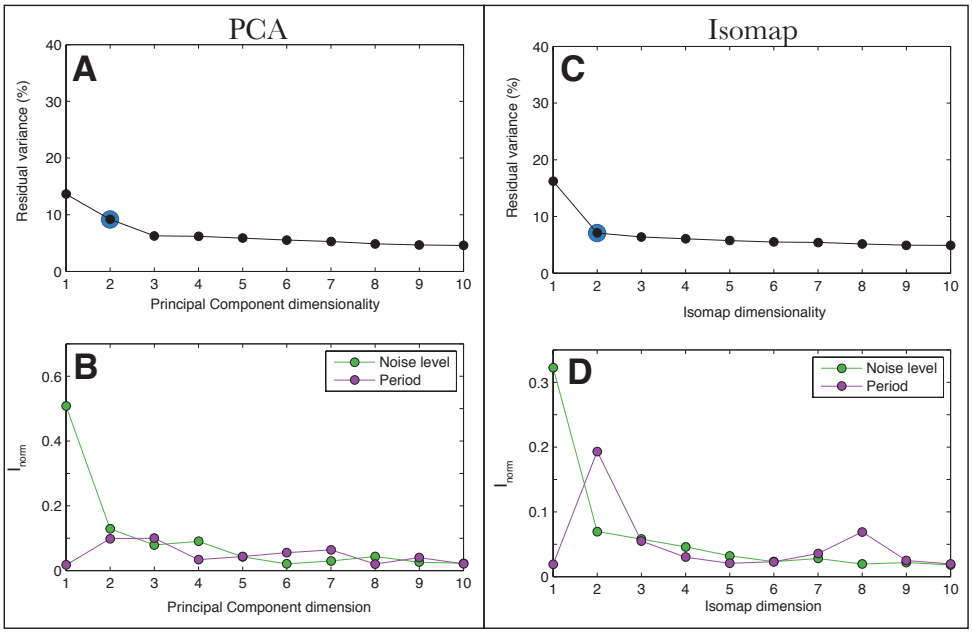

Figure D.15: **Residual variance plots for Principal Components Analysis and Isomap applied to a noisy periodic dataset.** Analogous to Fig. 7.10, we plot the residual variance as a function of the dimensionality of the reduced feature space (in **A** and **C**), and the normalized mutual information between each dimension and the free parameters, η and T (in **B** and **D**). Isomap more accurately detects the two-dimensional natures of this dataset, whereas PCA is dominated by the effect of noise on the time series controlled by the parameter, η.

D.3.5 Dimensionality reduction

In this section we show a figure related to Fig. 7.11, that shows explicitly the residual variance as a function of the dimensionality of the feature spaces, $\mathcal{F}$, using both PCA and Isomap. We find that results obtained from Isomap provide a more accurate reflection of the two free parameters (η and T) whose variation controls the variation of properties across this dataset.

Appendix E

Summary of included time series

This appendix summarizes the time series assembled in our database. For specific applications, only certain subsets of time series are used as data. However, when searching for nearest neighbors of time series, for example, the full database of time series is used. This database, or 'library' of time series, including synthetic time series that we have generated, and real-world time series that we have obtained from various publicly-available resources, is briefly described in this document. We use the following basic notation: x_0 is used to designate an initial condition when simulating a time series, and N is used to represent the time-series length measured in a number of samples.

E.1 Synthetic time series

E.1.1 Uncorrelated noise sequences

In this section, we describe uncorrelated noise sequences generated via various random number generators implemented in MATLAB's *Statistics Toolbox*. Lengths of series are N =1 000, 5 000, and 10 000.

- Beta-distributed noise created using **betarnd**. Parameters used were $a = 0.5, 1, 2, 3, 4, 5, 10$ and $b = 0.5, 1, 2, 3, 4, 5, 10$. Two instances of each combination of parameters were synthesized.
- Binomially-distributed random number sequences generated using **binornd** for parameters $n = 20, 50, 200$ and $p = 0.05, 0.5, 0.95$. Two instances of each combination of parameters were generated.

- χ^2-distributed random number sequences generated using **chi2rnd** for parameter $\nu = 1, 2, 3, 4, 5, 20, 50, 200$. Two instances of each combination of parameters were generated.

- Random number sequences from the extreme value distribution were synthesized using **evrnd**. We used parameters $\mu = 0$ and $\sigma = 1, 10$, and generated three occurrences of each combination.

- F-distributed noise using **frnd**. Parameters are $\nu_1 = 5, 50, 100$ and $\nu_2 = 5, 50, 100$, where two time series are generated for each combination of parameters.

- Γ-distributed noise using **gamrnd**. Parameters used were $a = 1, 10$, $b = 0.2, 2$, and two time series with each combination of parameters were synthesized.

- Random samples from the geometric distribution using **geornd**. Parameters are $p = 0.01, 0.1, 0.2$ and two instances of each parameter and length values are produced.

- Random number sequences from the generalized extreme value distribution generated using **gevrnd**. Parameters are $k = 0.01, 0.1, 0.2$, $\sigma = 1, 100$, $\mu = 0$, and two sequences are generated for each combination.

- Random samples from a generalized Pareto distribution generated using **gprnd**. Parameters are $k = 0.01, 0.2$, $\sigma = 10, 100$, and $\theta = 0$, and two sequences are generated for each combination.

- Random samples from a hypergeometric distribution generated using **hygernd**. Parameters are $M = 0.01, 0.1, 0.3$, $K = 1, 100$, and $N = 0$, with one sequence generated for each parameter combination.

- Random samples from a log-normal distribution generated using **lognrnd**. Parameters are $\mu = 0$, and $\sigma = 0.1, 0.4, 0.8$, with two sequences generated for each parameter combination.

- Random samples from a Gaussian distribution generated using **normrnd**. Parameters are $\mu = 0$, and $\sigma = 1, 10$, with five sequences generated for each parameter combination.

- Random samples from a Poisson distribution generated using **poissrnd**. Parameters are $\lambda = 0.1, 1, 10$, with three sequences generated for each parameter combination.

- Random samples from a Rayleigh distribution generated using **raylrnd**. Parameters are $b = 0.1, 1, 10$, with two sequences generated for each parameter combination.

- Random samples from a Rayleigh distribution generated using **trnd**. Parameters are $\nu = 2, 5, 50$, with two sequences generated for each parameter combination.

- Random samples from a discrete uniform distribution generated using **unidrnd**. Parameters are $N = 50$, with three sequences generated for each parameter combination.

- Random samples from a continuous uniform distribution generated using **unifrnd**. Parameters are $a = 0$, $b = 10$, with three sequences generated for each parameter combination.

- Random samples from a Weibull distribution generated using **wblrnd**. Parameters are $a = 1, 10$, $b = 0.5, 1, 10, 100$, with two sequences generated for each parameter combination.

- Random samples from an Exponential distribution generated using **exprnd**. Parameters are $\lambda = 0.01, 0.1, 1, 10, 100$, with two sequences generated for each parameter combination.

E.1.2 Periodic time series

- Various sinusoidal time series were generated according to the following model: $y_i = \sum_{j=1}^{J} A_j \sin(2\pi f_j t + \theta_j) + Bn_i$, for J components with amplitudes A_j,

frequencies f_j, and phases θ_j, and noise amplitude B, where $n_i \sim \mathcal{N}(0,1)$.

- A dataset of 131 sinusoids with added Gaussian-distributed noise was generated according to the following simple model: $x_t = \sin(2\pi ft) + n_t$, where $n \sim \mathcal{N}(0, \eta^2)$ represents independent draws from a normal distribution with zero mean and standard deviation η. All signals were 5 000 samples long, and had a frequency $f = 0.05$ /sample. The standard deviation of added noise was increased from $\eta = 0$, 0.01, ..., 1 and then from 1.1, 1.2, ..., 4.0.

- A dataset of time series generated from the same model as above: $x_t = \sin(2\pi ft) + n_t$, with different frequencies, f, and noise standard deviations, η. Time series of length $N = 5\,000$ were generated for $f = 1/3200$, $1/1600$, $1/800$, $1/400$, $1/200$, $1/100$, $1/50$, $1/25$, and for $\eta = 0.0$, 0.2, ..., 2.0.

E.1.3 Self-affine time series

- *Fourier filtering method* [90] was used to construct time series with the desired scaling behavior in the frequency domain and then converting to time domain series via the inverse Fourier transform [152]. Series were generated in this way with $N = 5\,000$, and a scaling exponent $\alpha = -1, -0.08, ..., 3.98, 4.00$.

- *The random midpoint displacement method* is also used to generate 5 000-sample time series. The method is described in Peitgen et al. [90]. We implement publicly available code [152] to generate the time series across the range $\alpha = -1.00, -0.98, ..., 3.0$.

E.1.4 Stochastic differential equations (SDEs)

- Random walks with Gaussian-distributed increments of length 1 000, 5 000, 7 500, 10 000, and 20 000, with 20 instances of each. Implemented as the cumulative sum of Gaussian-distributed white noise.

The remaining time series in this section were produced using the *SDE Toolbox* for MATLAB by Umberto Picchini, which is available at `http://sdetoolbox.sourceforge.net`. All series are labeled by their model code in the *SDE Toolbox* and were evaluated using the Milstein approximation.

- Model *M1a*: $dX_t = -aX_t dt + \sigma dW_t$. Parameters $a = -0.02, 2$, $x_0 = 0.01, 1$, $\sigma = 0.2, 2$, and time series lengths $N = 1\,000$, $5\,000$, and $10\,000$. We evaluate each parameter combination six times.

- Model *M2d*: $dX_t = (aX_t + b)dt + \sigma dW_t$ for $a = -0.1$, $b = -1, 1$, $x_0 = 0.01$, $\sigma = 0.2$, and time-series lengths $N = 1\,000$, $5\,000$, and $10\,000$. We create six instances of each parameter combination, removing an initial transient of 500 samples.

- Model *M3a*: $dX_t = (a - \sigma^2/2)dt + \sigma dW_t$. We evaluate this for $a = 0.1$, $x_0 = 1$, $\sigma = 0.1, 1, 2$, and time-series lengths $N = 1\,000$, $5\,000$, and $10\,000$. With each parameter combination, we produced six instances of time series.

- Model *M6a*: $dX_t = [0.5a(a-1)X_t^{1-2/a}]dt + aX_t^{1-1/a}dW_t$. We considered parameters $a = 1, 2.5, 5, 10$, $x_0 = 1$, and time-series lengths $N = 1\,000$, $5\,000$, and $10\,000$. Six instances of each parameter combination are produced.

- Model *M7a*: $dX_t = [-0.5a^2 X_t]dt + a\sqrt{1 - X_t^2}dW_t$. We use parameters $a = 2, 4, 6, 8, 10$, $x_0 = 1$, and time-series lengths $N = 1\,000$, $5\,000$, and $10\,000$. Six instances of each parameter combination are produced.

- Model *M8a*: $dX_t = [a^2 X_t(1 + X_t^2)]dt + a(1 + X_t^2)dW_t$. We implement this using parameters $a = -0.3, -0.2, -0.1, -0.05, 0.05, 0.1, 0.2, 0.3$, $x_0 = 1$, and for time-series lengths $N = 1\,000$, $5\,000$, and $10\,000$. Some solutions with these parameters are unstable (and blow up to $\pm\infty$), and are not kept, but otherwise six occurrences of each parameter combination are evaluated.

- Model *M10a*: $dX_t = a(b - X_t)dt + \sigma\sqrt{X_t}dW_t$. We used parameters $a = 0.1, 1, 10$, $b = 0.1, 1, 10$, $\sigma = 0.1, 1$, and time-series lengths $N = 1\,000$, $5\,000$, and $10\,000$. Six occurrences of each combination are evaluated and added to the database.

E.1.5 Correlated noise processes

Outputs from Autoregressive (AR) models We generated 935 time series from AR processes: $x_n = \sum_{j=1}^{p} a_j x_{n-j} + Nn_j$, where p is the order of the model and

$n_j \sim \mathcal{N}(0,1)$ are random samples from a Gaussian distribution with standard deviation N. Initial conditions were chosen at random in in the interval $-5 \leq x_1 \leq 5$, and each AR coefficient, a_j, was chosen uniformly at random in the interval $[-1,1]$. We consider model orders from $p = 2,3,...,10$, variable noise standard deviations $N = 0, 0.2, ..., 1.0$, and save outputs of different lengths: $N = 500, 1\,000, 5\,000, 7\,500$. For each combination of parameters, we simulate five time series (using different coefficients a_j) and retain and save solutions that do not blow up to $\pm\infty$ or shrink rapidly to 0.

Outputs from Moving Average (MA) models Outputs are produced from equations of the form $x_n = \sum_{j=1}^{p} a_j r_{n-j}$, where r_j are random samples from a uniform distribution in the range $[-1,1]$. For each order $p = 1,...,10$, we produced 10 sequences in this way for a range of lengths: $N = 500, 1\,000, 5\,000, 7\,500$.

Nonlinear observation functions Inspired by the basic process described in Eqs. (1) and (2) of Schreiber and Schmitz [224], e.g.,

$$s_n = s(x_n); \qquad \{x_n\} : \mathrm{AR}(p), \tag{E.1}$$

where the output time series s_n is some nonlinear function $s(\cdot)$ of the autoregressive process x_n, of order p. We generate series x_n according to an AR(1) process with autoregressive coefficients $a_1 = -0.9, -0.5, -0.1, 0.1, 0.5, 0.9$. We use measurement functions $s(x) = x^2$, $s(x) = x^3$, $s(x) = x^4$, $s(x) = x^5$, $s(x) = \exp(x)$, and $s(x) = \sin(x)$. Time series of length $N = 10\,000$ are generated, treating the first $1\,000$ as a transient to be removed (i.e., time series are generated with $11\,000$ samples, and the latter $10\,000$ are kept as the time series). For each parameter combination, four realizations are generated, for different random initial conditions.

Freitas's nonlinear moving average filter A nonlinear moving average filter of uniformly-distributed random noise u: $x_n = au_n + bu_{n-1}(1-u_n)$, from a paper by Freitas et al. [85]. We selected parameters a and b at random in the interval $[-5,5]$, and with a random initial condition in the range $[-5,5]$. For each parameter set (a,b), we produced two different time series from two different random initial conditions.

Faes nonlinear AR process I found this process mentioned as Eq. (3) in Guarín et al. [252]. The model is defined as:

$$x_t = a_1 x_{t-1}(1 - x_{t-1}^2)\exp(-x_{t-1}^2) + a_2 x_{i-2}. \tag{E.2}$$

We generated time series with $N = 10\,000$ samples from this process using $a_1 = 3.2$, 3.4, 3.6, and $a_2 = -0.1$, 0.5, 0.8, 0.9. For each parameter combination we outputted five different realizations for different initial conditions, and treated the first 1 000 samples as a transient (i.e., 11 000 samples were generated, and the time series treated as the latter 10 000 samples).

Nonstationary Autoregressive processes These processes were described and analyzed in [253]. The generating equation is

$$x(t) = a_i(t)x(t-1) + a_2(t)x(t-2) + a_3(t) + \epsilon(t), \tag{E.3}$$

where $\epsilon(t) \sim \mathcal{N}(0, \sigma^2)$. This is Eq. (6) in Timmer [253]. In the first instance, the coefficients are constant: $a_1 = 2\cos(2\pi/T)\exp(-1/\tau)$ and $a_2 = -\exp(-2/\tau)$, and the resulting time series is stationary. We generate series with $N = 10\,000$, $T = 10, 20, 50$, $\tau = 20, 50, 100$, $\sigma = 1$, and create five realizations of each parameter combination, removing the first 1 000 samples as 'transient'. Initial conditions are chosen randomly as the noise process ϵ.

Next, we implement realizations of the same model, but with amplitude-modulated periodic nonstationarity [253]:

$$x_{\text{amp}}(t) = [1 + \mathcal{M}_{\text{amp}}\sin(2\pi/T_{\text{mod}}t)]x_0(t), \tag{E.4}$$

where $x_0(t)$ is derived from Eq. (E.3). The parameter $\mathcal{M}$ parametrizes the violation of the null hypothesis (of stationarity) and T_{mod} determines the modulation period. We generate $N = 10\,000$ realizations of this 'cyclostationary' process for $\mathcal{M}_{\text{amp}} = 0.3, 0.6, 0.9$ and $T_{\text{mod}} = 100, 250$. For each parameter combination we generate five examples with random initial conditions (chosen as the noise process ϵ), where the first 1 000 samples are treated as a transient.

Finally, we consider a period-modulated AR(2) process, which takes the form of Eq. (E.3), but for $T(t) = T_{\text{mean}} + \mathcal{M}_T\sin(2\pi/T_{\text{mod}}t)$ and $a_1(t) = 2\cos[2\pi T(t)]\exp(-1/\tau)$,

where $\mathcal{M}_T$ parametrizes the violation of the null hypothesis [253]. To keep the variance of the process constant, σ is adjusted according to

$$\sigma^2(t) = \frac{\sigma^2}{1 - a_1^2 - a_2^2 - 2a_1^2 a_2/(1 - a_2)} \times \left(1 - a_1^2 - a_2^2 - \frac{2a_1^2 a_2}{1 - a_2}\right). \tag{E.5}$$

We generated time series of $N = 10\,000$ samples according to this process for $T_{\text{mod}} = 100, 250$, $\mathcal{M}_T = 1$, 3, 5, $T_{\text{mean}} = 10, 20$, and $\tau = 20, 50$. Five examples of each parameter combination were simulated (with different random initial conditions chosen by ϵ).

E.1.6 Maps

Maps are defined by iterative recurrence relations. We implement many of those listed in Sprott [12]. Throughout this section, N refers to the number of iterations of the map to produce the series, and equals the length of series. Initial 'transients' are removed of length 100 samples unless otherwise specified (i.e., $N + 100$ samples are generated and the latter N are used to represent the simulated time series).

E.1.6.1 Noninvertible maps

Time series listed in this section were generated according to the iterative recurrence relations in Appendix A.1 of Sprott [12].

- *Logistic Map*: $x_{n+1} = Ax_n(1 - x_n)$ for $N = 300, 1\,000, 5\,000$, and across a range of A [65]. Initial condition $x_1 = 0.1$.
- *More Logistic Map Series*: $x_{n+1} = Ax_n(1 - x_n)$ for $A = 3.0, 3.2, 3.4$ and $A =$ 3.50, 3.51, 3.52, ..., 4.00 with $N = 10\,000$, and removing an initial transient of 20 000 samples. Three time series were generated for each A using a different initial condition chosen uniformly at random in the range [0,1].
- *Sine Map*: $x_{n+1} = A \sin \pi x_n$, with $A = 0.9, 1.0, ..., 1.4$, $N = 300, 1\,000, 5\,000$, and $x_1 = 0.1$.
- *Tent Map*. $x_{n+1} = A \min(x_n, 1 - x_n)$, with $A = 1.80, 1.82, ..., 2$ (replacing 2 by 1.999999 for numerical reasons suggested in [12]), $N = 300, 1\,000, 5\,000$ and $x_1 = 1/\sqrt{2}$.

- *Linear congruential generator.* $x_{n+1} = Ax_n + B \mod C$, where $A = 7\,141, B = 54\,773, C = 259\,200$, $N = 300, 1\,000, 5\,000$, and $x_1 = 0, 0.1, ..., 1$.

- *Cubic Map*: $x_{n+1} = Ax_n(1 - x_n^2)$, using $A = 2.40, 2.45, ..., 3.00$ for $N = 300, 1\,000, 5\,000$, $x_1 = 0.1$.

- *Ricker's population model*: $x_{n+1} = Ax_n \exp(-x_n)$, with $A = 15, 16, ..., 30$, $N = 300, 1\,000, 5\,000$, and $x_1 = 0.1$.

- *Gauss map*: $x_{n+1} = 1/x_n \mod 1$, for $N = 300, 1\,000, 5\,000$ and random initial conditions x_1 in the range $0 \leq x \leq 3$.

- *Cusp map*: $x_{n+1} = 1 - A\sqrt{|x_n|}$, for $A = 1.90, 1.95, 2.00$, $N = 300$, $1\,000$, $5\,000$, $10\,000$, and x_1 are uniformly-distributed random numbers in the range $0 \leq x_1 \leq 1$.

- *Gaussian white chaotic map*: implemented as $x_{n+1} = |x_n|^{-1/4} - 1/2 - |x_n|$, with $N = 300, 1\,000, 5\,000, 10\,000, 20\,000$, and uniformly-distributed initial conditions in the range $0 \leq x_1 \leq 1$.

- *Pinchers map*: $x_{n+1} = |\tanh[s(x_n - c)]|$, for $s = 1.9, 2.0, 2.1$, $c = 0.40, 0.45, ..., 0.60$, $N = 300, 1\,000, 5\,000, 10\,000$, $x_1 = 0$.

- *Spence Map*: $x_{n+1} = |\ln x_n|$, for $N = 300, 1\,000, 5\,000, 10\,000$, with x_1 uniformly distributed in the interval [0,1].

- *Sine-Circle Map*: $x_{n+1} = x_n + \Omega - \frac{K}{2\pi} \sin 2\pi x_n \mod 1$, for $N = 300$, $1\,000$, $5\,000$, $10\,000$, x_1 is chosen randomly from a uniform distribution with $0 \leq x_1 \leq 1$, $K = 2$, and $\Omega = 0.5$.

E.1.6.2 Dissipative maps

Time series listed in this section were generated according to the iterative recurrence relations in Appendix A.2 of Sprott [12].

- *Hénon map*: $x_{n+1} = 1 - ax_n + by_n; y_{n+1} = x_n$, for $N = 300, 1\,000, 5\,000, 10\,000$, $x_1 = 0, 0.1$, $y_1 = 0.8, 0.9$.

- *Lozi map*: $x_{n+1} = 1 - a|x_n| + by_n; y_{n+1} = x_n$, using $a = 1.7$, $b = 0.5$, $N = 300$, 1 000, 5 000, 10 000, and initial conditions $x_1 = -0.2, -0.1$, $y_1 = 0.10, 0.15$.

- *Delayed Logistic Map*: $x_{n+1} = Ax_n(1 - y_n); y_{n+1} = x_n$, for $A = 2.27$, $N = 300$, 1 000, 5 000, 10 000, $x_1 = 0.001$, $y_1 = 0.001, 0.002$.

- *Tinkerbell Map*: $x_{n+1} = x_n^2 - y_n^2 + ax_n + by_n; y_{n+1} = 2x_n y_n + cx_n + dy_n$, for $a = 0.9$, $b = -0.6$, $c = 2$, $d = 0.5$, $N = 300, 1\,000, 5\,000, 10\,000$, $x_1 = 0, 0.05, 0.10$, $y_1 = 0.40, 0.45, 0.50$.

- *Burgers' Map*: $x_{n+1} = ax_n - y_n^2; y_{n+1} = by_n + x_n y_n$ using $a = 0.75, b = 1.75$, $N = 300, 1\,000, 5\,000, 10\,000$, $x_1 = -0.2, -0.1$, $y_1 = 0.1, 0.2$.

- *Holmes Cubic Map*: $x_{n+1} = y_n; y_{n+1} = -bx_n + dy_n - y_n^3$, for $b = 0.2, d = 2.77$, $N = 300, 1\,000, 5\,000, 10\,000$, $x_1 = 1.6, 1.7, 1.8$, $y_1 = 0$.

- *Kaplan-Yorke Map*: $x_{n+1} = ax_n \mod 1; y_{n+1} = by_n + \cos(4\pi x_n) \mod 1$, for $a = 1.99999999$, $b = 0.2$, $N = 300$, 1 000, 5 000, 10 000, $x_1 = 1/\sqrt{2}$, $y_1 = -0.4$.

- *Dissipative Standard Map*: $x_{n+1} = x_n + y_{n+1} \mod 2\pi; y_{n+1} = by_n + k \sin x_n \mod 2\pi$, for $b = 0.1, k = 8.8$, $N = 300, 1\,000, 5\,000, 10\,000$, initial conditions $x_1 = 0.05, 0.10, 0.20$, $y_1 = 0.05, 0.10, 0.20$.

- *Ikeda Map*: $x_{n+1} = \gamma + \mu(x_n \cos\phi - y_n sin\phi); y_{n+1} = \mu(x_n \sin\phi + y_n \cos\phi)$, where $\phi = \beta - \alpha/(1 + x_n^2 + y_n^2)$, for $\alpha = 6$, $\beta = 0.4$, $\gamma = 1$, $\mu = 0.9$, $N = 300, 1\,000, 5\,000, 10\,000$, and initial conditions $x_1 = 0, 0.05, 0.10$, $y_1 = 0$.

- *Sinai Map*: $x_{n+1} = x_n + y_n + \delta \cos(2\pi y_n) \mod 1; y_{n+1} = x_n + 2y_n \mod 1$, where $\delta = 0.08, 0.10, 0.12$, $N = 300, 1\,000, 5\,000, 10\,000$, and initial conditions $x_1 = 0.50, 0.55$, $y_1 = 0.50, 0.55$.

- *Discrete predator-prey map*: $x_{n+1} = x_n \exp[r(1 - x_n/K) - \alpha y_n]; y_{n+1} = x_n[1 - \exp(-\alpha y_n)]$, for $r = 3, K = 1, \alpha = 5$, $N = 300, 1\,000, 5\,000, 10\,000$, and initial conditions $x_1 = 0.50, 0.55$, $y_1 = 0.50, 0.55$.

E.1.6.3 Conservative maps

Time series listed in this section were generated according to the iterative recurrence relations in Appendix A.3 of Sprott [12].

- *Chirikov (standard) map*: $x_{n+1} = x_n + y_{n+1}; y_{n+1} = y_n + k \sin x_n$, for $k = 1$, $N = 300,\ 1\,000,\ 5\,000,\ 10\,000,\ 20\,000$, and initial conditions $x_1 = 0, 0.1, 0.2$, $y_1 = 6.0, 6.1, 6.2$.

- *Hénon area-preserving quadratic map*: $x_{n+1} = x_n \cos\alpha - (y_n - x_n^2)\sin\alpha,\ y + n + 1 = x_n \sin\alpha + (y_n - x_n^2)\cos\alpha$, where $\alpha = \cos^{-1}(0.24)$, $N = 300,\ 1\,000,\ 5\,000,\ 10\,000$, and initial conditions $x_1 = 0.6, y_1 = 0.13$.

- *Arnold's cat map*: $x_{n+1} = x_n + y_n \mod 1; y_{n+1} = x_n + k y_n \mod 1$ for $k = 2$, $N = 300, 1\,000, 5\,000, 10\,000$, and initial conditions $x_1 = 0$, $y_1 = 1/\sqrt{2}$.

- *Gingerbreadman map*: $x_{n+1} = 1 + |x_n| - y_n; y_{n+1} = x_n$, for $N = 300,\ 1\,000,\ 5\,000,\ 10\,000,\ 20\,000,\ 40\,000$, and initial conditions $x_1 = 0.50,\ 0.51,\ ...,\ 0.54$, $y_1 = 3.7$.

- *Chaotic web map*: $x_{n+1} = x_n \cos\alpha - (y_n + k \sin x_n)\sin\alpha; y_{n+1} = x_n \sin\alpha + (y_n + k \sin x_n)\cos(\alpha)$, using $\alpha = \pi/2, k = 1$ for $N = 300,\ 1\,000,\ 5\,000,\ 10\,000,\ 20\,000,\ 40\,000,\ 60\,000$ (although the output is still not stationary after $N = 60\,000$ samples!), using $x_1 = 0$, $y_1 = 2.98,\ 2.99,\ 3.00,\ 3.01,\ 3.02$.

- *Lorenz three-dimensional chaotic map*: $x_{n+1} = x_n y_n - z_n, y_{n+1} = x_n, z_{n+1} = y_n$, for $N = 300, 1\,000, 5\,000, 10\,000$, initial conditions $x_1 = 0.50, 0.51$, $y_1 = 0.50, 0.51$, $z_1 = -1.0, -0.99$.

E.1.6.4 Other maps

- *Cao's Periodic Map*: $x_{n+4} = \sin(x_n + 5) + \sin(2x_{n+1} + 5) + \sin(3x_{n+2} + 5) + \sin(4x_{n+3} + 5)$, Eq. (8) in Cao [254]. It is evaluated for for $N = 300,\ 1\,000,\ 5\,000,\ 10\,000,\ 20\,000$.

- *Freitas's Stochastic Sine Map*, described by Freitas et al. [85]: $x_{n+1} = \mu \sin(x_n) + Y_n \eta_n$, where Y_n is Bernoulli (implemented in MATLAB as **binornd(1,q,[])**), which has a probability q of 1, and probability $1 - q$ of 0. The noise variable η

is uniformly-distributed random samples in the range $[-b, b]$ (using **unifrnd(-b,b,[])**). We use parameters $\mu = 2.4$, $b = 1, 2, 3$, $q = 0.01, 0.05, 0.10, 0.20, 0.50$, and take four realizations of each parameter combination. Initial condition is $x_1 = 2$, and we remove the first 100 points as a 'transient'.

E.1.7 Dynamical systems

Systems of ordinary differential equations (ODEs) define certain classes of dynamical behavior that are often described as 'dynamical systems' or 'flows'. We implement many of these as described in [12]. In this section we refer to L, the time-span of the run in time, t; and N, the number of points to sample from the flow and hence the length of the time series. The time series is evaluated on a uniformly-spaced grid in time using the MATLAB code **deval**.

- *Lorenz attractor*. We return the x, y, and z co-ordinates of the trajectory for $L = 200, 500, 1\,000$, and $N = 1\,000, 5\,000, 10\,000$.
- *Duffing two-well oscillator* for $L = 100, 200, 400$, $N = 500, 1\,000$.
- *Rössler Attractor* with $a = b = 0.2$, $c = 4, 5.5, 5.6, 5.7$, $x_0 = (0, 0, 0)$. Removes the initial transient for 500 samples. Time series with $L = 500, 1\,000, 2\,000$ and $N = 1\,000, 5\,000, 10\,000$.
- *Van der Pol Oscillator*, undriven with $b = 1, 2, 3$, and initial condition $x = 1, \dot{x} = 0$. For $N = 500, 1\,000, 2\,000$.
- *Driven pendulum*: parameters b (dissipative term), A (driving amplitude), and Ω (driving frequency). We implement for $b = 0, 0.05$, $A = 0, 0.3, 0.6, 0.9$, $\Omega = 0.35, 0.7, 1.05$, $L = 20\,000, 80\,000, 160\,000$ with $N = 1\,000, 5\,000, 10\,000$ and initial condition $x = 2, \dot{x} = 0$. An initial transient of 800 samples is removed.
- *Driven harmonic oscillator*. Solutions lie on a torus. We choose the damping term $b = 0$, natural frequency $\omega = 0.35, 0.6, 1.1$, driving amplitude $A = 0.3, 0.6, 0.9, 1.2$, and driving frequency $\Omega = 0.35, 0.7, 1.05$ with initial condition $x = 2, \dot{x} = 0$, and a transient of $t = 0, ..., 500$ removed. We use $L = 2\,000, N = 5\,000$.

E.1.7.1 Jerk systems

These simple chaotic Jerk systems are in Table 4.2 of Sprott [12]. All systems are evaluated in MATLAB using a tolerance 10^{-7}, and in each case an initial transient of 200 time points is removed. ODE systems are evaluated over a time span of $L = 300, 1\,000, 4\,000$, and are evaluated on a uniformly-spaced grid of $N = 1\,000, 5\,000, 10\,000$ points using **deval**.

- JD_1. $\dddot{x} = a\ddot{x} + bx + x\dot{x} - 1$ with $a = -1.8, b = -2$.
- JD_2. $\dddot{x} = a\ddot{x} + b\dot{x} + x^2 - 1$ with $a = -0.5, b = -1.9$.
- JD_3. $\dddot{x} = a\ddot{x} + b\dot{x} + cx^2 + x\dot{x} - 1$ with $a = -0.6, b = -3, c = 5$.
- JD_4. $\dddot{x} = a\ddot{x} + b\dot{x} + cx^2 + x\dot{x} - 1$ with $a = -0.6, b = -2, c = 3$.
- JD_6. $\dddot{x} = a\ddot{x} + b\dot{x} + cx^2 + d\dot{x}^2 + x\ddot{x} - 1$ with $a = -1, b = -1, c = 2, d = 2$.

E.1.7.2 Sprott's three-dimensional flows

These flows are equations listed in Table 4.1 of Sprott [12]. In each case we use an initial condition $(x, y, z) = (0.05, 0.05, 0.05)$, and remove an initial transient of length 200 samples. The differential equations are solved over time periods $L = 300, 1\,000, 4\,000$ and evaluated on uniformly-spaced grids of length $N = 1\,000$, $5\,000$, and $10\,000$.

- Flow A: $\dot{x} = y, \dot{y} = -x + yz, \dot{z} = 1 - y^2$.
- Flow B: $\dot{x} = yz, \dot{y} = x - y, \dot{z} = 1 - xy$.
- Flow C: $\dot{x} = yz, \dot{y} = x - y, \dot{z} = 1 - x^2$.
- Flow D: $\dot{x} = -y, \dot{y} = x + z, \dot{z} = xz + 3y^2$.
- Flow E: $\dot{x} = yz, \dot{y} = x^2 - y, \dot{z} = 1 - 4x$. E
- Flow F: $\dot{x} = y + z, \dot{y} = -x + 0.5y, \dot{z} = x^2 - z$.
- Flow G: $\dot{x} = 0.4x + z, \dot{y} = xz - y, \dot{z} = -x + y$.

- Flow H: $\dot{x} = -y + z^2, \dot{y} = x + 0.5y, \dot{z} = x - z$.
- Flow I: $\dot{x} = -0.2y, \dot{y} = x + z, \dot{z} = x + y^2 - z$.
- Flow J: $\dot{x} = 2z, \dot{y} = -2y + z, \dot{z} = -x + y + y^2$.
- Flow K: $\dot{x} = xy - z, \dot{y} = x - y, \dot{z} = x + 0.3z$.
- Flow L: $\dot{x} = y + 3.9z, \dot{y} = 0.9x^2 - y, \dot{z} = 1 - x$.
- Flow M: $\dot{x} = -z, \dot{y} = -x^2 - y, \dot{z} = 1.7 + 1.7x + y$.
- Flow N: $\dot{x} = -2y, \dot{y} = x + z^2, \dot{z} = 1 + y - 2z$.
- Flow O: $\dot{x} = y, \dot{y} = x - z, \dot{z} = x + xz + 2.7y$.
- Flow P: $\dot{x} = 2.7y + z, \dot{y} = -x + y^2, \dot{z} = x + y$.
- Flow Q: $\dot{x} = -z, \dot{y} = x - y, \dot{z} = 3.1x + y^2 + 0.5z$.
- Flow R: $\dot{x} = 0.9 - y, \dot{y} = 0.4 + z, \dot{z} = xy - z$.
- Flow S: $\dot{x} = -x - 4y, \dot{y} = x + z^2, \dot{z} = 1 + x$.

E.1.7.3 Driven dissipative flows

These iterative recurrence relations are used to generate time series in this section and are listed in Appendix A.4 of Sprott [12]. The integer N refers to the number of iterations of the map to produce the series, and equals the length of series. Equations are evaluated using **ode45** with a tolerance of 10^{-6}, and evaluated on an evenly-spaced grid using **deval**. Initial 'transients' are removed of length 100 samples.

- *Damped driven pendulum*: $\dot{x} = y; \dot{y} = -\sin x - by + A\sin(\Omega t)$, where $b = 0.05$, $A = 0.6$, $\Omega = 0.7$, $L = 300, 1\,000, 5\,000, 10\,000$, $N = 1\,000, 5\,000, 10\,000$, and initial conditions $x_0 = 0, 0.1, 0.2$, $y_0 = 0, 0.1, 0.2$.
- *Driven van der Pol oscillator*: $\dot{x} = y; \dot{y} = -x + b(1 - x^2)y + A\sin(\Omega t)$, where $b = 3$, $A = 5$, $\Omega = 1.788$, $L = 300, 1\,000, 5\,000$, $N = 1\,000, 5\,000, 10\,000$, and initial conditions $x_0 = -1.9, -1.7$, $y_0 = 0, 0.2$.

- *Shaw-van der Pol oscillator*: $\dot{x} = y + A\sin(\Omega t); \dot{y} = -x + b(1 - x^2)y$, for $b = 1$, $A = 1$, $\Omega = 2$, $L = 300, 1\,000, 5\,000$, $N = 1\,000, 5\,000, 10\,000$, and initial conditions $x_0 = 1.3, 1.4$, $y_0 = 0, 0.1$.

- *Forced Brusselator*: $\dot{x} = x^2y-(b+1)x+a+A\sin(\Omega t); \dot{y} = -x^2y+bx$, for $a = 0.4$, $b = 1.2$, $A = 0.05$, $\Omega = 0.8$, $L = 300, 1\,000, 5\,000$, $N = 1\,000, 5\,000, 10\,000$, and initial conditions $x_0 = 0.3, y_0 = 2$.

- *Ueda Oscillator*: $\dot{x} = y; \dot{y} = -x^3 - by + A\sin\Omega t$, with $b = 0.05$, $A = 7.5$, $\Omega = 1$, $L = 300, 1\,000, 5\,000$, $N = 1\,000, 5\,000, 10\,000$, and initial conditions $x_0 = 2.5, y_0 = 0$.

- *Duffing two-well oscillator*: $\dot{x} = y; \dot{y} = -x^3 + x - by + A\sin\Omega t$, for $b = 0.25$, $A = 0.4$, $\Omega = 1$, $L = 300, 1\,000, 2\,000$, $N = 1\,000, 5\,000, 10\,000$, and initial conditions $x_0 = 0.20, 0.25, 0.30$, $y_0 = 0$.

- *Duffing-van der Pol oscillator*: $\dot{x} = y; \dot{y} = \mu(1-\gamma x^2)y-x^3+A\sin\Omega t$, where $\mu = 0.2$, $\gamma = 8$, $A = 0.35$, $\Omega = 1.02$, $L = 300, 1\,000, 2\,000$, $N = 1\,000, 5\,000, 10\,000$, and using initial conditions $x_0 = 0.20, 0.25$, $y_0 = -2.5, -2.0$.

- *Rayleigh-Duffing Oscillator*: $\dot{x} = y; \dot{y} = \mu(1 - \gamma y^2)y - x^3 + A\sin\Omega t$, where $\mu = 0.2$, $\gamma = 4$, $A = 0.3$, $\Omega = 1.1$, $L = 300, 1\,000, 2\,000$, $N = 1\,000, 5\,000, 10\,000$, $x_0 = 0.30, 0.35, 0.40$, and $y_0 = 0$.

E.1.7.4 Autonomous dissipative flows

Flows in this section are listed in Appendix A.5 of Sprott [12].

- *Lorenz Attractor*: $\dot{x} = \sigma(y - x); \dot{y} = -xz + rx - y; \dot{z} = xy - bz$, where $\sigma = 10$, $r = 28$, $b = 8/3$, $L = 250, 500, 1\,000$, $N = 1\,000, 5\,000, 10\,000$, and initial conditions $x_0 = 0$, $y_0 = -0.01$, $z_0 = 9, 9.1$.

- *Rössler attractor*: $\dot{x} = -y - z; \dot{y} = x + ay; \dot{z} = b + z(x - c)$, where $a = 0.2$, $b = 0.2$, $c = 5.7$, $L = 250, 1\,000, 2\,000$, $N = 1\,000, 5\,000, 10\,000$, and using initial conditions $x_0 = -9.1, -9.0$, $y_0 = 0$, $z_0 = 0$.

- *Diffusionless Lorenz Attractor*: $\dot{x} = -y - x$, $\dot{y} = -xz$, $\dot{z} = xy + R$, where $R = 1$, $L = 250, 1\,000, 2\,500$, $N = 1\,000, 5\,000, 10\,000$, and using initial conditions $x_0 = 1.0, 1.1$, $y_0 = -1.1, -1.0$, $z_0 = 0.01$.

- *Complex Butterfly*: $\dot{x} = a(y - z); \dot{y} = -z\text{sgn}(x); \dot{z} = |x| - 1$, where $a = 0.55$, $L = 2\,000, 5\,000, 10\,000$, $N = 5\,000, 10\,000, 20\,000$, and initial conditions: $x_0 = 0.2, y_0 = 0, z_0 = 0$. I had some problems evaluating this for longer time scales, e.g., $L = 20\,000$ I run out of memory.

- *Chen's System*: $\dot{x} = a(y - x); \dot{y} = (c - a)x - xz + cy; \dot{z} = xy - bz$, with $a = 35$, $b = 3$, $c = 28$, $L = 100, 250, 500$, and $N = 1\,000, 5\,000, 10\,000$.

- *Hadley Circulation*: $\dot{x} = -y^2 - z^2 - ax + aF; \dot{y} = xy - bxz - y + G; \dot{z} = bxy + xz - z$, with $a = 0.25$, $b = 4$, $F = 8$, $G = 1$, $L = 25, 50, 100$, $N = 1\,000, 5\,000, 10\,000$, and initial conditions $x_0 = 0$, $y_0 = 0$, $z_0 = 1.3$. Because the variation is on such small length scales, I had to subtract the mean, multiply by 1×10^6, and then add the mean again to amplify the fluctuations.

- *ACT attractor*: $\dot{x} = \alpha(x - y); \dot{y} = -4\alpha y + xz + \mu x^3; \dot{z} = -\delta\alpha z + xy + \beta z^2$, with $\alpha = 1.8$, $\beta = -0.07$, $\delta = 1.5$, $\mu = 0.02$, $L = 100, 250, 1\,000$, $N = 1\,000, 5\,000, 10\,000$.

- *Rabinovich-Fabrikant attractor*: $\dot{x} = y(z - 1 + x^2) + \gamma x; \dot{y} = x(3z + 1 - x^2) + \gamma y; \dot{z} = -2z(\alpha + xy)$, where $\gamma = 0.87$, $\alpha = 1.1$, $L = 100, 250, 1\,000$, $N = 1\,000, 5\,000, 10\,000$, and initial conditions $x_0 = -1, y_0 = 0, z_0 = 0.5$.

- *Chua's circuit*: $\dot{x} = \alpha[y - x + bx + 0.5(a - b)(|x + 1| - |x - 1|)]; \dot{y} = x - y + z; \dot{z} = -\beta y$, $\alpha = 9$, $\beta = 100/7$, $a = 8/7$, $b = 5/7$, $L = 250, 500, 1\,000, 2\,000$, $N = 1\,000, 5\,000, 10\,000$, and initial conditions $x_0 = 0$, $y_0 = 0$, $z_0 = 0.6$.

- *Moore-Spiegel oscillator*: $\dot{x} = y$, $\dot{y} = z$, $\dot{z} = -z - (T - R + Rx^2)y - Tx$, where $T = 6$, $R = 20$, $L = 100, 250, 500, 750$, $N = 1\,000, 5\,000, 10\,000$.

- *Thomas' cyclically symmetric attractor*: $\dot{x} = -bx + \sin(y), \dot{y} = -by + \sin(z), \dot{z} = -bz + \sin(x)$, $b = 0.18$, $L = 1\,000, 2\,500, 5\,000, 10\,000$, $N = 1\,000, 5\,000, 10\,000$, and initial conditions $x_0 = 0.1$, $y_0 = 0$, $z_0 = 0$.

- *Halvorsen's cyclically symmetric attractor*: $\dot{x} = -ax - 4y - 4z - y^2, \dot{y} = -ay - 4z - 4x - z^2, \dot{z} = -az - 4x - 4y - x^2$, setting $a = 1.27$, $L = 100, 500, 1\,000$, $N = 1\,000, 5\,000, 10\,000$, and initial conditions $x_0 = -5$, $y_0 = 0$, $z_0 = 0$.

- *Rucklidge attractor*: $\dot{x} = -\kappa x + \lambda y - yz, \dot{y} = x, \dot{z} = -z + y^2$, with $\kappa = 2, \lambda = 6.7$, $L = 100, 500, 1\,000, 2\,000$, $N = 1\,000, 5\,000, 10\,000$, and initial conditions $x_0 = 1$, $y_0 = 0$, $z_0 = 4.5$.

- *WINDMI attractor*: $\dot{x} = y, \dot{y} = z, \dot{z} = -az - y + b - \exp(x)$, for $a = 0.7$, $b = 2.5$, $L = 200, 500, 1\,000, 2\,000, 4\,000$, $N = 1\,000, 5\,000, 10\,000$, and initial conditions $x_0 = 0$, $y_0 = 0.8$, $z_0 = 0$.

- *Simplest quadratic chaotic flow*: $\dot{x} = y, \dot{y} = z, \dot{z} = -az + y^2 - x$, where $a = 2.017$, $L = 200, 500, 1\,000, 2\,000, 4\,000$, $N = 1\,000, 5\,000, 10\,000$, and initial conditions $x_0 = -0.9$, $y_0 = 0$, $z_0 = 0.5$.

- *Simplest cubic chaotic flow*: $\dot{x} = y, \dot{y} = z, \dot{z} = -az + xy^2 - x$, with $a = 2.028$, $L = 200, 500, 1\,000, 2\,000, 4\,000$, $N = 1\,000, 5\,000, 10\,000$, and initial condition $x_0 = 0$, $y_0 = 0.96$, $z_0 = 0$.

- *Simplest piecewise linear chaotic flow*: $\dot{x} = y, \dot{y} = z, \dot{z} = -az - y + |x| - 1$, with $a = 0.6$, $L = 200, 500, 1\,000, 2\,000, 4\,000$, $N = 1\,000, 5\,000, 10\,000$, and initial condition $x_0 = 0$, $y_0 = -0.7$, $z_0 = 0$.

- *Double Scroll*: $\dot{x} = y, \dot{y} = z, \dot{z} = -a[z + y + x - \mathrm{sgn}(x)]$, setting $a = 0.8$, $L = 200,\ 500,\ 1\,000,\ 2\,500,\ 5\,000$, $N = 1\,000,\ 5\,000,\ 10\,000$, and initial condition $x_0 = 0.01$, $y_0 = 0.01$, $z_0 = 0$.

E.1.7.5 Conservative flows

Flows in this section are listed in Appendix A.6 of Sprott [12].

- *Driven Pendulum*: $\dot{x} = y, \dot{y} = -\sin x + a \sin \Omega t$, for $A = 1, \Omega = 0.5$ and initial condition $x_0 = 0$, $y_0 = 0$. The x co-ordinate is evaluated for $L = 1\,000, 2\,000, 5\,000, 10\,000, 20\,000, 50\,000$, while its derivative, y, is evaluated for $L = 200, 500, 1\,000, 2\,500, 5\,000$. The solutions are evaluated for $N = 1\,000$,

5 000, 10 000. We used **ode45** to solve the equations with a tolerance of 10^{-7}, but the solutions weren't terrifically stable: changes in the length L produced different trajectories.

- *Simplest Driven Chaotic Flow*: $\dot{x} = y, \dot{y} = -x^3 + \sin \Omega t$, with $\Omega = 1.88$, $L = 250$, 500, 1 000, 2 000, 5 000, 7 000, $N = 1\,000$, 5 000, 10 000, and initial condition $x_0 = 0$, $y_0 = 0$.

- *Nosé-Hoover Oscillator*: $\dot{x} = y, \dot{y} = -x + yz, \dot{z} = a - y^2$, with $a = 1$, $L = 250$, 500, 1 000, 2 000, 5 000, 7 500, $N = 1\,000$, 5 000, 10 000, and initial condition $x_0 = 0, y_0 = 5, z_0 = 0$.

- *Labyrinth Chaos*: $\dot{x} = \sin(y), \dot{y} = \sin(z), \dot{z} = \sin(x)$, for $L = 250$, 500, 1 000, 5 000, 10 000, 20 000, 40 000, 80 000, $N = 1\,000$, 5 000, 10 000, and initial condition $x_0 = 0.1, y_0 = 0, z_0 = 0$. There is some quite long-scale structure in this flow, but it cannot be resolved due to memory constraints.

- *Hénon-Heiles System*: $\dot{x} = v, \dot{y} = w, \dot{v} = -x - 2xy, \dot{w} = -y - x^2 + y^2$, for $L = 200, 500, 1\,000, 2\,500, 5\,000$, $N = 1\,000, 5\,000, 10\,000$, and initial condition $x_0 = 0.499$, $y_0 = 0$, $v_0 = 0$, $w_0 = 0.03160676$.

E.1.8 Other models

MIX(p) processes . We produce a set of signals based on the MIX(P) model described in Pincus [29]. We generated periodic signals, $x_t = \sin(x/10)$, and replaced a proportion p of the data points (chosen uniformly at random) with a random value in the range $[-1, 1]$. Note that the original definition is similar, but differs in some minor details [29]. We produce signals with $p = 0.1, 0.2, ..., 0.9$, and for signal lengths $N = 1\,000, 5\,000$.

E.2 Real-world time series

Data from a Keogh database A large collection of time-series datasets was retrieved from `http://www.cs.ucr.edu/~eamonn/tutorials.html`. Datasets include tide data, soil temperature, speech, earthquake tremor, wind, data from an industrial

drier, rainfall, a buoy sensor, wing flutter, sunspots, shuttle, ECG, pH data from a controlled stirred tank reactor, sensors attached to a telephone, EEG, air balloon, chaotic laser output, leaf shape data, event-related potentials (ERPs) in a visual spatial attention task, evaporator data, fluid dynamics, fetal ECG, gait intervals, glass furnace heating and cooling inputs from temperature sensors in a cross section of the furnace, rainfall at the great lakes, light curve data from the variable white dwarf star PG1159-035, network packet round trip times, data from a robot's arm, gene expression time series (PGT alpha), daily spot exchange rates, S&P500 from 1926–1993, variables from a model of a steam generator, infrasound from explosions, speed of an industrial winding process, electrooculograms (EOGs), and respiration recordings. In total we used 773 time series obtained from this source in our time-series database.

Time-Series Data Library (TSDL) Many time series are available at `http://robjhyndman.com/TSDL/`, from the following categories: agriculture, chemistry, crime, demography, ecology, finance, health, hydrology, industry, labour market, macro-economic, meteorology, micro-economic, physics, production, sales, sport, transport and tourism, tree-rings, synthetically-generated, and utilities. Note that most of these are relatively short recordings (most are less than 200 samples). In total, 765 time series in our database were obtained from this source.

Google trends Time series were obtained from *Google Trends*, `http://www.google.com/trends`, with a variety of keywords, including 'amazon', 'ballroom dancing', 'beethoven', 'bikini models', 'calculus', 'dark side of the moon', 'holy trinity', 'jesus', 'mahler', 'tibet', 'vegetarian', and other such keyword realizations that stem from one's stream of consciousness. Time series were exported using the option 'CSV with relative scaling' and are all 296 samples long. In total we retrieved 212 such time series from this source.

Emory website The following website contains a set of time-series for download: `http://www.physics.emory.edu/~weeks/research/tseries1.html`. We retrieved experimental data that many be chaotic, periodic, quasi-periodic, and a Henon, Lorenz, and Rössler time series.

Data provided by Max A. Little Max A. Little provided a variety of time-series data for use in this project. This included various ECGs, EEGs, rainfall, and a river flow time series.

Astronomy Various time series obtained from `http://astro.uni-tuebingen.de/groups/time/time_series_EXO/`.

The Santa Fé time series competition Data includes *Data Set A*: Laser generated data, *Data Set B*: physiological data, *Data Set C*: currency exchange rate data, *Data Set D*: computer-generated series, *Data Set E*: astrophysical data, and *Data Set F*: Bach's last (unfinished) fugue. Data is available at `http://www-psych.stanford.edu/~andreas/Time-Series/SantaFe.html`.

StatLib *StatLib* is an online time-series data resource: `http://lib.stat.cmu.edu/`. Datasets we implemented include a synthesized univariate time series, monthly river flow series, wire wave gauge ocean wave time series, wind speed time series, atomic clock time series, ocean noise time series, ice profile series, hourly water level time series, government security yields, monthly sunspot numbers, and air temperatures.

Time-series analysis and its applications, with R examples Time series from this book are available at `http://lib.stat.cmu.edu/general/stoffer/tsa2/`.

Free recall Inter-recall times provided by Malia Mason for associative recall, episodic recall, recalls of people, and semantic recalls.

Text We retrieved a number of texts from Project Gutenberg: `http://www.gutenberg.org/wiki/Main_Page`. For each text, the information about the terms and conditions of Project Gutenberg, and other irrelevant parts of the text, like '[Illustration]', although captions of images are usually kept. Tables of contents and footnotes are removed, as are forewords by other authors, publishing information, etc. For each text, we generated time series of the lengths of consecutive sentences in the text: measured as the number of characters or the number of words. In total we analyzed approxi-

mately 266 books, including *Illustrated History of Furniture*, by Frederick Litchfield, *The Kama Sutra of Vatsyayana* by Vatsyayana, *Alice's Adventures in Wonderland* by Lewis Carroll, *Improvement of the Understanding* by Spinoza, *Poems*by T. S. Eliot, and books from the King James version of *The Holy Bible*.

DNA pore Signals of current from pores as DNA strands pass through them. Data provided by Beth Jelfs is a signal from 'adenine', 'cytosine', 'guanine', or 'thymine'. All signals are 1 000 samples long. There are approximately 160 different time series for each base pair.

E.2.1 Medical

E.2.1.1 Gait

- A selection of gait intervals were provided by Max A. Little labeled in groups: 'als' 'control', 'huntingtons', and 'parkinsons'.

- A selection of data from the *Gait Maturation Database* was obtained from the PhysioNet website in October 2009, as explained in Hausdorff et al. [255]. Data are from 50 healthy children aged from 40–163 months. Data is available at `http://physionet.org/physiobank/database/gait-maturation-db/`.

- *Gait in Parkinson's Disease*: `http://www.physionet.org/pn3/gaitpdb/`. Data obtained and reformatted by Max A. Little, October 2009. Original time series were longer than 20 000, so we just analyze subsegments of the original series here.

- *Postural sway measurements* from the noise enhancement of sensorimotor function data source on PhysioNet: `http://www.physionet.org/physiobank/database/nesfdb/`, as described in Priplata et al. [256]. Data are of 15 healthy young male volunteers, and 12 healthy elderly male volunteers. The data are 1 800 samples long and were retrieved by Max A. Little, October 2009.

- *Tremor Database* contains data measured to study the effect of deep brain stimulation on Parkinsonian tremor, available at `http://www.physionet.org/physiobank/database/tremordb/`.

E.2.1.2 Electrocardiograms (ECGs)

- *The BIDMC Congestive Heart Failure Database* contains ECG signals sampled at 250 Hz with 12-bit resolution over a range ±10,mV measured at Boston's Beth Israel Hospital and is available at `http://physionet.org/physiobank/database/chfdb/`. The recordings are of 15 subjects with congestive heart failure. We took (non-overlapping) segments of recordings from these signals ranging from 1 000–15 000 samples.

- *The MIT-BIH Normal Sinus Rhythm Database* contains 18 long-term ECG recordings of subjects referred to the Arrhythmia Laboratory at Boston's Beth Israel Hospital and is available here: `http://physionet.org/physiobank/database/nsrdb/`. Subjects include 5 men and 13 women that had no significant arrhythmias. Segments of time series of length between 1 000–15 000 samples were taken from the long recordings to analyze.

- *The Massachusetts General Hospital/Marquette Foundation (MGH/MF) Waveform Database* is a comprehensive collection of electronic recordings of hemodynamic and electrocardiographic waveforms of patients in critical care units available at `http://physionet.org/pn3/mghdb/`. The database contains recordings from 250 patients containing three ECG leads, arterial pressure, pulmonary arterial pressure, central venous pressure, respiratory impedance, and airway CO_2 waveforms. Some recordings also include intra-cranial, left atrial, ventricular and/or intra-aortic-balloon pressure waveforms. We analyzed segments of these physiological time series, where we randomly took two (non-overlapping) segments from each full time series of lengths chosen randomly in the range 1 000–15 000 samples.

E.2.1.3 EEGs

- *Epilepsy Dataset* obtained via Max A. Little from Michel Le Van Quyen, based at `http://cogimage.dsi.cnrs.fr/`. Subsegments of time series were taken from each EEG channel. Data is from three subjects in three different areas: inter-critique periods, recordings where electrodes are near the focus in the same

Hippocampal area, and at the same period of focus but in a different area of the brain. Segments from these recordings are taken as examples of EEGs.

- *Bonn Dataset.* In this section, we analyze a publicly-available dataset of electroencephalogram (EEG) time series available at `http://epileptologie-bonn.de/cms/front_content.php?idcat=193&lang=3&changelang=3` [86]. Each time series was sampled at 173.61 Hz and contains a 23.6 second (i.e., 4 097 sample) segment of a single-channel EEG recording. The segments were selected so as not to contain artifacts, and to fulfill a weak stationarity criterion [86]. We have applied a low-pass filter at 40 Hz to the data, as suggested in the original paper [86]. The recordings are categorized into five sets, labeled *A*, *B*, *C*, *D*, and *E*, where each set contains 100 time series. These data are analyzed in detail in Sec. S5.2.4 of the supplementary document.

E.2.1.4 Heart rate

- *Physionet 2002 Challenge Dataset*: `www.physionet.org/challenge/2002/dataset/`. Contains 50 RR intervals. Data formatted and provided by Max A. Little. These RR intervals were very long series (all longer than 80 000 beats), so shorter segments obtained from them were analyzed instead of the full recordings.

- *RR intervals* Were provided from Max A. Little, and come in two forms: normal sinus rhythm ('nsr') and congestive heart failure ('chf'). The age of each patient is also annotated to each time series. Data was obtained from `http://www.physionet.org/physiobank/database/chf2db/`, and also from `http://www.physionet.org/physiobank/database/nsr2db/`. We took (non-overlapping) subsegments from each RR interval series in the range 800–20 000 samples. This dataset is studied in detail in Sec. S5.2.6 of the supplementary online document to this report.

- *Fetal Heart Rate Recordings.* Fetal heart rate time series were provided by Antoniya Georgieva.

E.2.1.5 Miscellaneous

- *Synthetic Datasets*: `http://www.physionet.org/physiobank/database/synthetic/tns/`. These time series were very long; all were reduced to segments from the full series.

E.2.2 Meteorology

- *Daily meteorological data*: `http://www.sci.usq.edu.au/staff/dunn/Datasets/applications/climatology/qldrain.html`. Data is daily: minimum/maximum temperature, rainfall, radiation, pan evaporation, and maximum vapour pressure deficit. Series were over 40 000 samples long and converted to shorter segments for analysis.

- *River discharge* series available at `http://daac.ornl.gov/RIVDIS/rivdis.html`. Time series are of monthly average discharges in m^3/s. I did not include series that contained NaNs or that were shorter than 100 samples long.

- *Lamb/Jenkins Weather Types.* Based on Professor H. H. Lamb's method of subjectively classifying each day's weather over the British Isles from 1861–1997. These 'symbolic' time series are available at `http://www.cru.uea.ac.uk/cru/data/lwt.htm`.

- *Madras Monthly Sea Level.* Available at `http://www.cru.uea.ac.uk/cru/data/madrasmslp.htm`.

- *Mediterranean Oscillation Index (MOI)* data available at `http://www.cru.uea.ac.uk/~andrewh/moi.html`. We used series of the normalized pressure difference between Algiers and Cairo, and also between Israel and Gibraltar.

- *North Sea Caspian Pattern (NCP)* data is available at `http://www.cru.uea.ac.uk/~andrewh/ncp.html`.

- *North Atlantic Oscillation (NAO)* data is available at `http://www.cru.uea.ac.uk/cru/data/nao.htm`. Modifications were calculated at `http://www.cru.`

uea.ac.uk/cru/data/vinther/ and also included, as are reconstructions available at http://www.cru.uea.ac.uk/cru/data/naojurg.htm.

- *Southern Oscillation Index (SOI)*, defined as the normalized pressure difference between Tahiti and Darwin; data available at http://www.cru.uea.ac.uk/cru/data/soi.htm.

- *Trans Polar Index (TPI)* is the normalized pressure difference between Hobart and Stanley. Three time series of this variation were analyzed.

- *Surface Air Temperature*: data taken on a 73x37 grid of the globe, obtained from http://www.cru.uea.ac.uk/cru/data/ncep/. Data is recorded daily from 1948–2007. All series are 21 915 points long.

- *Rainfall*: daily precipitation rate at the surface of the earth measured in $kg/m^2/s$ (or mm/s) obtained at http://www.cru.uea.ac.uk/cru/data/ncep/qs_eurasia/daily/sflux/prate.sfc/.

- *Relative Humidity* measured at grid spots (73x37) on the earth: http://www.cru.uea.ac.uk/cru/data/ncep/qs_eurasia/daily/surface/air.sig995/. Data is daily from 1948–2007. Data also includes sea level pressure.

- *Zooplankton* data obtained from http://hahana.soest.hawaii.edu/hot/hot-dogs/.

- *Austrian river flow* data (in m^3/s) obtained from Max A. Little. Data are of mean daily flows during the period 1950–2001. Another river flow series was also obtained from Max A. Little – Churn at Perrots Brook (an environment agency gauging station); Churn is a tributary of the Thames that flows through Cirencester.

- *NIMROD atmospheric water droplet density* data available at http://badc.nerc.ac.uk/view/badc.nerc.ac.uk__ATOM__dataent_nimrod. Frames are taken every 5 mins over a 164x175 grid of the earth. I used 20 points on the spatial grid to form time series, points labeled: 15, 6540, 13763, 14296, 15145, 16089, 16493, 16727, 17581, 19995, 22999, 23402, 24257, 24416, 24839, 25228, 25855, 26680,

28382, 28412. Time series were for the year 2007 and are generated daily (by summing over days to create 365-long time series), hourly (by summing over hours, to create 8 760-long series), and the full 5-minutely series, taken over different (randomly-chosen) periods of time, from 1 day up to 2 months of data.

- *MIDAS rainfall data* is available at `http://badc.nerc.ac.uk/view/badc.nerc.ac.uk__ATOM__dataent_ukmo-midas`. First, we took data from the period 2000–2007, and then we took the first 2750 points to avoid NaNs at the end of the record, and then selected the spatial positions in the global grid that had the least missing values to form time series. Second, we took time series with 3 652 samples from 1990-1999. There were 767 global positions with no missing values across this decade, which are all included as time series in the database. Similarly, we included the 929 time series with no missing values from 1980–1989. Longer series were taken across 50 year periods: fifty series had no missing values across the period 1950–2000 and were included, as were various subsegments of these time series, of length ranging between 500–15 000 samples.

- *Hadley Data*. Data from the Met Office, Hadley Centre passed on by Max A. Little, and includes both rainfall and temperature data at both monthly and daily sampling rates.

- *Teleconnections*. Data provided by Max A. Little. Teleconnections are global climate anomalies related to each other at large distances. We have the East Atlantic Pattern (EA), the East Atlantic/West Russia Pattern (EA/WR), the explained variance (%) of leading 10 modes, the North Atlantic Oscillation (NAO), Pacific/North American Pattern (PNA), Polar/Eurasia Pattern (POL), Scandinavia Pattern (SCA), West Pacific Pattern (WP), and the East Pacific/ North Pacific Pattern (EP/NP). For the EP/NP data, missing values were linearly interpolated.

- *Japanese Meteorology* data from `http://www.cru.uea.ac.uk/cru/data/japan/`, as used in Zaiki et al. [145]. Data is monthly temperature and pressure.

- *UK Meteorology*, including river flow and rainfall data obtained from `http://www.cru.uea.ac.uk/cru/data/riverflow/` with rainfall during approx. 1865 (depending on the station, the starting year is 1850–1865) through to 2002, and reconstructed river flows during approx. 1865 – 2002.

- *Methane Emissions*: annual estimates of global anthropogenic methane emissions from 1860–1994 from the carbon dioxide information analysis centre (CDIAC): `http://cdiac.ornl.gov/`. Methane emissions are at `http://cdiac.ornl.gov/trends/meth/ch4.htm`. Other emissions data were monthly series from 1981.

- *Surface temperature deviations* derived from radiosonde records, by J. K. Angell: `http://cdiac.ornl.gov/trends/temp/angell/angell.html`. All series are short, containing approximately 180 samples each.

- *Historical Isotopic Temperature Record* from the Vostok Ice Core 740 000-year deuterium record in an ice core from Dome C, Antarctica: `http://cdiac.ornl.gov/trends/temp/domec/domec.html`.

- *NDP-048* (2007): three- and six-hourly meteorological observations from 223 former U.S.S.R. Stations, available at `http://cdiac.ornl.gov/ftp/ndp048/`. Data includes air temperature, the height above the ground of the base of lowest cloud (in 100s of meters), humidity deficit (saturation deficiency) – the difference between saturation vapor pressure and the actual water vapor pressure at a given temperature and pressure (measured in 10ths of hPa), low-cloud amount estimated visually by an observer, precipitation amount (to the nearest 10th of a millimeter), pressure tendency value (i.e., the absolute difference between the current air pressure at the station and that observed 3 h in the past), relative humidity (as a percentage, measured using psychrometer), seal level pressure of the air (10ths of hPa, measured by a barometer), soil surface temperature (measured in degrees C), air pressure at station level (using barometer, expressed as 10ths of hPa), total cloud amount estimated visually by an observer using a 10-point system, dew point temperature (degrees C), vapour pressure of the air

(measured in 10ths of hPa determined by a psychrometer), horizontal visibility is a distance in km, wind speed is measured 10-12 m above ground level as a value in m/s, and wind direction is a coded symbolic series.

- *Ozone data* from `http://toms.gsfc.nasa.gov/ozone/ozoneother.html`. Zonal mean: the range of zones over which there was good data was used; missing values were minimal, and were interpolated linearly.

- Seismology We included a dataset consisting of 8 known explosion time series, 8 known earthquake time series, and an unknown event, which took place near the Russian nuclear test facility in Novaya Zemlya [154]. All time series are 2 048 samples long. The dataset, which contains regional (i.e., 100-2 000 km) recordings of typical Scandinavian earthquakes and mining explosions as measured by stations in Scandinavia, was originally constructed by Blandford [155] and is publicly-available at `http://lib.stat.cmu.edu/general/stoffer/tsa2/`. We included all 17 time series, and also divided them into the P and S components of their waveforms, which were the first 1 024 and second 1 024 samples, respectively. The dataset is studied in detail in Sec. S5.2.2 of the supplementary online document

- Geoscience Geoscience data of well depth versus gamma ray intensity were provided by Max A. Little from `http://pubs.usgs.gov/of/2007/1142/`.

E.2.3 Financial

- *World Crude Oil Prices*. Provided as an excel file by Max A. Little. Data are daily spot prices from 1983–2008.

- *Yahoo Finance*: `http://uk.finance.yahoo.com/`. Time series were obtained of high-low (the different between daily highs and daily lows), volume (traded), and opening prices. Missing values were linearly interpolated, in which case a label is added to the time series filename specifying the number of interpolated points. Data include daily share prices and various finance sectors. Log return transformations of the data were also performed and included in the database

in addition to the original series.

- *Refinery Stocks*. Short, monthly stocks at U.S. Oil refineries were obtained at `http://tonto.eia.doe.gov/dnav/pet/pet_stoc_ref_dc_nus_mbbl_m.htm`. Also weekly retail gasoline and diesel prices, world crude oil prices: `http://tonto.eia.doe.gov/dnav/pet/pet_pri_wco_k_w.htm`, and NYMEX futures prices.

- *Daily Exchange Rates* obtained from `http://www-psych.stanford.edu/~andreas/Time-Series/Data/`. Missing values from these series were removed by decimation.

E.2.4 Sound

E.2.4.1 Speech

- Some data was provided by Max A. Little, including a basic vowel sound, an American subject talking: "I see a female tiger with her cub and they look like they are at risk in the jungle somewhere...". Another set of speech data is sampled at 16 kHz from healthy controls, and patients with a range of voice disorders [144].

- *Speech Snippets*: data obtained from `http://alt-usage-english.org/audio_archive.shtml`. All audio files were converted to .wav files using Apple Computer's *iTunes* software, sampling at 24 kHz, 8-bit mono. Time series were then down-sampled to create approximately 2-second snippets of audio containing approximately 10 000 samples. Speech was of a set of five different text passages, spoken with a variety of different accents.

- *Podcast snippets* from Ben Fulcher's personal library: including *Australian Broadcasting Corporation*'s *Radio National* podcast titled 'All in the mind'. Also, snippets from 'Hamish and Andy's' podcast, and from the *PodFather*'s Halloween special. Snippets of a 2–3 s were taken at a sampling frequency of either 3 kHz or 4 kHz, producing time series containing approximately 10 000 samples. Some full podcasts were also outputted, at a very low sampling rate (approx. 0.01 kHz).

- *Parkinsonian Speech* data was obtained from Max A. Little [91]. There are 195 recordings in one of two categories: 'healthy control' or 'Parkinson's disease' measured from 31 subjects, of which eight are age-matched healthy controls (the other 23 are patients with Parkinson's disease). There are six recordings per subject on average, which are of sustained phonations of the vowel 'ahhh'. Originally recorded at 44.1 kHz, they have been downsampled to 16 kHz, and a 'vocally stable' section of the middle of the recording has been retained, starting at 1.0 s [91]. Each recording is 2.0 s in duration, and contain 32 000 samples. We analyzed these original recordings and also generated a set of shorter, non-overlapping segments from them, of length chosen randomly between 800–20 000 samples. Note that this dataset is analyzed in detail as a classification task in Sec. S5.2.5 of the supplementary document.

- *Berlin Emotional Database* is available to download at `http://www.expressive-speech.net/emodb`. The dataset contains speech utterances and was developed by the Technical University of Berlin [134]. Ten different German phrases (five short and five long) were spoken by ten different actors (five males and five females) using seven different emotional states: 'neutral', 'happy', 'angry', 'sad', 'boredom', 'fear', and 'disgust'. The 535 utterances were recorded at a sampling frequency of 48 kHz and later down-sampled to 16 kHz. Due to the length of these audio files, we further down-sampled each audio recording by a factor of two using the MATLAB function `resample(x,1,2)`, where x has been imported directly from the audio wave file. These data are analyzed in detail in Sec. S5.2.3 of the supplementary online document.

E.2.4.2 Music

- Time series of music was obtained from Ben Fulcher's personal music collection, and spans a variety of genres, including Baroque choral music, folk, pop, blues, contemporary classical, techno, rock, sludge metal, electronic, orchestral, hard rock, and progressive rock. All were converted to .wav files sampled at 24 kHz, 8 bit mono, and then processed further within MATLAB. First, random length snippets were taken at a given length and downsampling rate. Time series are

all at 8 bits, and for a variety of durations, from approx. 1 s to 6 s, and sampled at 3–12 kHz. Other time series were formed by severely downsampling every full audio signal to approximately 10 000 samples. Since audio files were typically at least 500 s long, this corresponded to a sample rates of approximately 20 –50 Hz, at 8 bits.

E.2.4.3 Sound effects

- Sound effects were obtained from `www.soundjay.com`, which were converted to .wav files at 24 kHz and 8-bit mono. Sounds that were longer than 10 000 samples were further downsampled so that time series were short enough to calculate quickly. Regions of silence (with amplitude $< 1 \times 10^{-6}$) were removed from both ends of the time series. A selection of sounds were obtained in each of the following categories: ambient, buttons, communication, human, household, machine/mechanical, nature, and transportation.

E.2.4.4 Animal sounds

- *Macaulay Library*: `http://macaulaylibrary.org/index.do`. Time series are recordings of various species, and are usually in the form of short snippets (of approx. 2–5 s) at a sampling rate of 2–4kHz. In total, 485 animal sound recordings were used in our database.

- *Bird Song*. A variety of recordings of bird calls were obtained from Dr. Joseph Tobias and Dr. Nathalie Seddon. In total, 1 292 bird call recordings were used in our database.

E.2.5 Space data

- *Geomagnetic indices* from the Space Physics Interactive Data Resource (SPIDR): `http://spidr.cetp.ipsl.fr/spidr/query.do?group=geomInd`. We retrieved the Aa index, Am index, DST index, Ap index, C9 index, Cp index, Kp index, solar radio flux, and sunspot number over the period 1/1/1990–1/1/2000. For time series longer than 20 000 points, random subsegments of length between 5 000–20 000 samples were taken to represent them.

- *Hemispheric Power Index (HPI)* from the Defense Meteorological Satellite Program (DMSP) `http://spidr.ngdc.noaa.gov/spidrvo/viewdata.do?docname=Hpidmsp` (which had fields for 'meas', 'power', 'corr'). The HPI measures high latitude precipitating energy flux carried by ions and electrons. Measurements were also obtained from NOAA: `http://spidr.ngdc.noaa.gov/spidr/query.do?group=Hpinoaa&` (which had fields for 'power', 'powerIndex', 'normFactor').

- *Vostok*. Polar cap index. Data available at `http://spidr.ngdc.noaa.gov/spidr/query.do?group=PCI`.

- *Solar Data* available at `http://spidr.ngdc.noaa.gov/spidr/query.do?group=SSN&`.

- *Cosmic Ray Indices*. These were not processed by any transformations, but have been included in the database precisely as downloaded (in the 4096 format) from `http://spidr.ngdc.noaa.gov/spidr/query.do?group=CRI4096&`. Some dramatic changes happen in some of the signals that almost certainly do not reflect accurate measurements of cosmic ray time series, but there were retained in the database as examples of strange signals.

- *Ionosphere Data* were obtained from `http://spidr.ngdc.noaa.gov/spidr/query.do?group=Iono&`. In total we retrieved 198 such series from a variety of locations.

- *Radio Solar Telescope Network (RSTN)* data was obtained from `http://spidr.ngdc.noaa.gov/spidr/query.do?group=Rstn&`.

- *Interplanetary magnetic field* measurements were taken from stations WIND, ACE, and IMP8: `http://spidr.ngdc.noaa.gov/spidr/query.do?group=IMFMin&`. Magnetic fields were recorded in each direction: *bx*, *by*, and *bz*, as well as the ion density (cm^3), z component of the flow speed (km/s), time delay to magnitosphere, and the position of the satellite (y component). Data is sampled minutely from 1/1/1995–1/6/1995, hourly from 1/1/2000–1/1/2001, and daily from 1980–2000 (although each station has its own time range restrictions).

E.2.6 Transformations of time series

Linearly-detrended time series We applied a linear trend test to all time series, and if the statistic from this test exceeded a given threshold, we applied a basic linear detrending and saved an additional version of the time series. The test involved z-scoring the time series, fitting a line to the time series using least squares, and outputting the mean of a vector containing three elements that summarize the strength of the trend: $|m|/5$, where m is the fitted gradient of the line, MSE, the mean square error between the fitted line and the time series, and the mean δ estimates of error from the MATLAB function **polyval**. When the mean of these three quantities is under a threshold set at 0.666, we remove the best-fit line from the time series and save a new version of the time series to the database with the keyword 'ldt'. In total we detrended 1 158 time series in this way.

Seasonally-detrended time series We applied a seasonality test to all time series in the database, and removed the periodicity/nonstationarity if the output of the test implied a strong seasonal trend in the time series. The statistics used in the test were: (i) the logarithm of the maximum of the power spectral density minus the mean of the log power spectrum, (ii) the difference between the maximum and median, and (iii) the difference between the maximum and the 90th percentile. When then mean of these three numbers exceeded 8, we applied a seasonal detrending routine that filters out peaks in the power spectrum and saved new deseasonalized versions of all such time series. In total, 978 new deseasonalized time series were generated in this way.

Time-series segments Time series longer than 20 000 samples were chosen, and random length subsegments, in the range 1 000 to 15 000 samples, were retrieved and used as time series in the database, alongside the original long versions. In this way, the database contains both the original recordings, as well as smaller portions of dynamics from these processes. Note that operations were typically not calculated on time series longer than 30 000 samples due to the computational expense.

Appendix F

Summary of included operations

This supplementary document is an accompaniment to the article titled 'Highly Comparative Time-Series Analysis: The Empirical Structure of Time Series and Their Methods'. It summarizes the operations included in our database, or 'library' of operations: the code they use, and the types of outputs that they return. For readability, an exhaustive description of over 9 000 operations is omitted, instead they are grouped by the common parts of code that they use. The names of particular pieces of code are distinguished by boldface and all code is available from the authors on request.

Note that, due to the size and interdisciplinary breadth of the time-series analysis literature, we can not hope to single-handedly produce a comprehensive list of operations for time-series analysis. Instead, we have tried to encompass techniques derived from as many of the main areas of research in time-series analysis as we were able to. In the future, as this resource is made publicly-available, we anticipate this list to continue to grow through contributions from the time-series analysis community.

Some basic notation is used throughout this document: μ is used to represent the mean of the input time series, σ represents the standard deviation of the time series, and the word 'raw' refers to the input time series without applying a z-score transformation.

F.1 External Time-Series Analysis Packages

In this section, we list the main external packages of code that are used in our library of time-series analysis operations. They will be referred to throughout this document.

The source of other pieces of code are described in the individual operations that they are used in.

- *TISEAN* is a popular package for nonlinear time-series analysis [92]. We use version 3.0.1 of the code, which is publicly-available at `http://www.mpipks-dresden.mpg.de/~tisean/Tisean_3.0.1/index.html`.
- The *TSTOOL* MATLAB toolbox for nonlinear time-series analysis is freely available at `http://www.physik3.gwdg.de/tstool/index.html`.
- The *ARfit* package [257, 258] is freely-available at `http://www.gps.caltech.edu/~tapio/arfit/`.
- Michael Small's time-series analysis code [130] is available at `http://small.eie.polyu.edu.hk/matlab/`.
- The *GPML* MATLAB code for simulating Gaussian Processes was written by Carl Edward Rasmussen and Hannes Nickisch [10]. We use version 2.0 of the code. This, and newer versions of the code are available at `http://www.gaussianprocess.org/gpml/code/matlab/doc/index.html`.
- Zoubin Gharamani's implementation of HMMs for real-valued Gaussian observations can be downloaded from `http://www.gatsby.ucl.ac.uk/~zoubin/software/hmm.tar.gz` or `http://mlg.eng.cam.ac.uk/zoubin/software.html`, and `http://mlg.eng.cam.ac.uk/zoubin/software/hmm.tar.gz`.
- Max A. Little's step detection toolkit for MATLAB [259, 260] is available for download from `http://www.maxlittle.net/software/`.

F.2 Basic Statistics

- **ST_length**. Returns the length of the time series.
- **ST_burstiness**. Returns the 'burstiness' statistic: $(\sigma - \mu)/(\sigma + \mu)$ [261].
- **ST_mm**. Returns the maximum and minimum values of the z-scored time series.

- **ST_bs**. Returns some basic statistics about the time series including the proportion of zero-crossings of the z-scored time series, the proportion of local maxima and minima in the time series, and the ratio of the number of times that the z-scored time-series crosses +1 to the number of times that it crosses -1.

- **ST_propsimp**. Returns statistics on the values of the raw time series: the proportion of zeros in the raw time series, the proportion of positive values, and the proportion of values greater than or equal to zero.

F.2.1 Location

A number of measures of location are implemented in the database that operate on the raw time series. These include the arithmetic mean, harmonic mean, the mode using a variety of histogram representations, and trimmed means, where outliers are first removed.

- **ST_means**. Measures the mean of the time series, as arithmetic, harmonic, root-mean-square, interquartile mean, or midhinge.

- **ST_mode**. Measures the mode of the time series using histograms with different numbers of bins.

- **ST_median**. Outputs the median of the time series.

- **ST_trmean**. Outputs the mean of the trimmed time series using the MATLAB function **trimmean**.

F.2.2 Spread

- **ST_spreads**. Returns the spread of the raw time series, as the standard deviation, interquartile range, mean absolute deviation, or median absolute deviation. We also implement this code to evaluate the standard deviation of the differenced time series.

F.2.3 Hypothesis tests

- **HT_hyptests**. Outputs the p-value from a statistical hypothesis test applied to the time series. Tests are implemented as functions in MATLAB's *Statistics Toolbox*, and include the sign test (**signtest**), runs test (**runstest**), variance test (**vartest**), Z-test (**ztest**), Wilcoxon signed rank test for a zero median (**signrank**), and the Jarque-Bera test of composite normality (**jbtest**). We also apply the Ljung-Box Q-test for residual autocorrelation, implemented via **lbqtest** from MATLAB's *Econometrics Toolbox*.

- **EC_vratiotest**. This code performs a variance ratio test on the time series, implemented via the **vratiotest** code in MATLAB's *Econometrics Toolbox*. It assesses the null hypothesis of a random walk in the time series. Inputs are a vector (or scalar) of period(s) and a vector (or scalar), representing boolean values indicating whether to assume independent and identically distributed (IID) innovations for each period. We implement this code for single periods: for (period,IID) = (2,0), (2,1), (4,0), and (4,1). And we also compare a range of periods simultaneously using periods = (2,4,6,8,2,4,6,8) and IIDs = (0,0,0,0,1,1,1,1).

- **EC_pptest**. Applies the Phillips-Peron unit root test for a time series via the code **pptest** from MATLAB's *Econometrics Toolbox*. Inputs are: (1) a vector of lags, (2) a specified model, either *ar*: autoregressive, *ard*: autoregressive with drift, or *ts*: trend stationary, and (3) the test statistic: either a standard t-statistic, or a lag-adjusted, 'unStudentized' t statistic. Outputs are statistics on the p-values and lags obtained from the set of tests, as well as measures of the regression statistics. We implement the code using lags, τ ranging from $\tau = 0, 1, 2, ..., 5$, and using the standard t-statistic. The code is run with these parameters using each of the three model choices.

- **EC_kpsstest**. Performs the KPSS test, of Kwiatkowski, Phillips, Schmidt, and Shin [124], for stationarity using the code **kpsstest** from MATLAB's *Econometrics Toolbox*. The null hypothesis is that a univariate time series is trend stationary, the alternative hypothesis is that it is a non-stationary unit-root

process. We implement the code for a variety of individual lags: $\tau = 0$, 1, and 2. We also evaluate changes in p-values and test statistics obtained by applying the test across a range of lags $\tau = 0, 1, ..., 10$.

F.2.3.1 Miscellaneous

- **SY_stdnthder**. Based on an idea by Vladimir Vassilevsky, a DSP and Mixed Signal Design Consultant in a MATLAB forum, who stated that "You can measure the standard deviation of the nth derivative, if you like"[1]. We estimate the derivative very simply by simply taking successive increments of the time series; the process is repeated to obtain higher order derivatives. The metric is calculated for a number of differences $n = 1, 2, ..., 10$.

- **SY_dynstdder**. This operation returns statistics on how the output of **SY_stdnthder** changes with the order of the derivative of the signal. An exponential function, $f(x) = A\exp(bx)$, is fitted to the variation across derivatives n from $n = 1, 2, ..., 10$.

- **KP_crinkle**. Calculates Theiler's 'crinkle statistic' on the time series using code by D. Kaplan obtained at `http://www.macalester.edu/~kaplan/software/`.

- **KP_theilerQ**. Calculates a Q statistic coded by D. Kaplan, obtained at `http://www.macalester.edu/~kaplan/software/`.

F.3 Measures of the distribution

F.3.1 Basic summaries

- **DN_moments**. Outputs measure moments, m of the distribution, for $m = 3, 4, .., 11$. This operations is applied to both raw and z-scored time series.

- **DN_skewy**. Estimates custom skewness measures. The 'Pearson' skewness (for the raw time series) and 'Bowley' skewness (for z-scored time series) are calculated.

[1] `http://www.mathworks.de/matlabcentral/newsreader/view_thread/136539`

- **DN_quanp**. Measures the proportion of the time series that lies within a proportion p standard deviations of its mean. For $p = 0.5, 1, 1.5, 2, 2.5, 3$.

- **DN_pleft**. Measures the maximum distance from the mean at which a given fixed proportion, h, of the data points are further. This quantity is evaluated for $h = 0.05, 0.1, 0.2, 0.3, 0.4, 0.5$.

- **DN_quant**. Calculates the quantile value at a specified proportion p using MATLAB code **quantile**. For $p = 0.01, 0.02, ..., 0.05$ and $p = 0.10, 0.20, ..., 0.90$ and $p = 0.91, 0.92, ..., 0.99$.

- **ST_cv**. Calculates the coefficient of variation, σ^k/μ^k , of order k, for $k = 1, ..., 6$.

- **ST_bendist**. Calculates a statistic related to the mean of the time series data that is above the (global) time-series mean compared to the mean of the data that is below the global time-series mean.

- **DN_kssimp**. Fits a kernel-smoothed distribution to the data using the **ksdensity** function from MATLAB's *Statistics Toolbox* and outputs a set of statistics summarizing the obtained distribution, including the number of peaks, the distributional entropy, the number of times the curve crosses fixed probability thresholds, the area under the curve for fixed probability thresholds, the arc length, and the symmetry of probability density above and below the mean.

F.3.2 Distributional fits

- **DF_mlefits**. Fits either a Gaussian, Uniform, or Geometric distribution to the data using maximum likelihood estimation via the MATLAB function **mle** from the Statistics Toolbox. Outputs are parameters from the fit.

- **MF_M_mtlbfit**. Fits different distributions to the time series using the MATLAB routine **fit** from the Curve Fitting Toolbox. The distribution of time-series values is estimated using either a kernel-smoothed density via the MATLAB function **ksdensity** with the default width parameter, or by a histogram with $n_{\text{bins}} = 10, 30$ bins. We fit single Gaussians (using the 'gauss1' input to **fit**), a

sum of two Gaussians (using 'gauss2'), an Exponential distribution (using the 'exp1'), and a power law (using 'power1'). Outputs are the goodness of fit, R^2, root mean square error, the autocorrelation of the residuals, and a runs test on the residuals.

- **DF_nlogL_norm**. Fits a Gaussian distribution to the data using the **normfit** function in MATLAB's Statistics Toolbox and returns the negative log likelihood of the data coming from a Gaussian distribution.

- **DN_M_kscomp**. Returns simple statistics on the discrepancy between the kernel-smoothed distribution of the time-series data, and the distribution fitted to it by some model: Gaussian (using **normfit** from MATLAB's Statistics Toolbox), Extreme Value (**evfit**), Uniform (**unifit**), Beta (**betafit**), Rayleigh (**raylfit**), Exponential (**expfit**), Gamma (**gamfit**), Log-Normal (**lognfit**), and Weibull (**wblfit**). Outputs include the absolute area between the two distributions, the peak separation, overlap integral, and relative entropy.

- **HT_disttests**. Fits a distribution to the data and then performs an appropriate hypothesis test to quantify the difference between the two distributions. We fit Gaussian, Extreme Value, Uniform, Beta, Rayleigh, Exponential, Gamma, Log-Normal, and Weibull distributions, using code described for **DN_M_kscomp** above. Several hypothesis tests are performed: the χ^2 goodness of fit test using $n_{\rm bins} = 5$, 10, 25, 50, and 100 (via the function **chi2gof**), a Kolmogorov-Smirnov test (using **kstest**), and a Lilliefors test (using **lillietest**). All of these codes for hypothesis testing are implemented in MATLAB's *Statistics Toolbox*.

F.3.3 Extreme values and outliers

- **CO_acfremove**. A proportion p of points are removed from the time series according to some rule, and a set of statistics are computed before and after the change. Points are removed according to the following rules: (i) those that are the closest to the mean, (ii) those that are the furthest from the mean, (iii) the lowest values, and (iv) the highest values. Output statistics include the

change in autocorrelation, time scales, mean, spread, and skewness. We use $p = 0.1, 0.5$, and 0.8 for each of the four rules described above.

- **ST_cumrange**. Measures of the range of the time series as a function of time, i.e., range$(x_{1:i})$ for $i = 1, 2, ..., N$, where N is the length of the time series. Outputs are based on the dynamics of how new extreme events occur with time.

- **DN_olmi**. Measures a range of different statistics about the time series as more and more outliers are included in the calculation according to a specified rule: (i) *abs*: outliers are furthest from the mean, (ii) *p*: outliers are the greatest positive deviations from the mean, or (iii) *n* outliers are the greatest negative deviations from the mean. The threshold for including time-series data points in the analysis increases from zero to the maximum deviation in increments of 0.01σ, where σ is the standard deviation of the time series. At each threshold, the mean, standard error, proportion of time series points included, median, and standard deviation are calculated, and outputs from the algorithm measure how these statistical quantities change as more extreme points are included in the calculation. Most of the outputs measure either exponential, $f(x) = A\exp(Bx) + C$, or linear, $f(x) = Ax + B$, fits to the sequence of statistics obtained in this way.

- **EX_altmannben**. A measure based on a moving threshold model for extreme events [262]. This algorithm is based on this idea: it uses the occurrence of extreme events to modify a moving 'barrier' that classes new points as 'extreme' or not. The barrier begins at 1σ, and if the absolute value of the next data point is greater than the barrier, the barrier is increased by $a\%$, otherwise the position of the barrier is decreased by $b\%$. Outputs are the mean, spread, maximum, and minimum of the barrier's time course, the mean of the difference between the barrier and the time series values, and statistics on the occurrence of 'kicks' (times at which the threshold is modified), and by how much the threshold changes on average.

- **OL_bentest**. Removes the p% of highest and lowest values in the time series and returns the ratio of either the mean or the standard deviation of the time series, before and after this transformation. We implement this operation using parameters $p = 2$, 5, and 10.

F.4 Correlation

F.4.1 Autocorrelation

We implement linear and nonlinear measures of autocorrelation in the time series.

- **CO_autocorr**. Measures the linear autocorrelation in the time series at lag τ, for $\tau = 1, 2, ..., 40$.

- **CO_fzcac**. Returns the first zero-crossing of the autocorrelation function, up to a maximum time-delay $\tau = 400$.

- **CO_f1ecac**. Finds where autocorrelation function first crosses $1/e$. Uses **information** code by R. Moddemeijer downloaded from `http://www.cs.rug.nl/~rudy/matlab/`.

- **CO_fmac**. Returns the time at which the first minimum in the autocorrelation function occurs. Note that this is an unusual operation: standard metrics are the first zero-crossing of the autocorrelation as in **CO_fzcac** above, or the first minimum of the mutual information function, as in **CO_fmmi**.

- **CO_autocorrs_nl**. Nonlinear autocorrelations of the form $\langle x_i x_{i-\tau_1} x_{i-\tau_2} ... \rangle$. Note that the usual two-point autocorrelation has simply a single τ, τ_1; using multiple τ is a nonlinear generalization of this technique. We calculate 55 different combinations of τs.

- **CO_tc3**. Computes the *tc3* function, a normalized nonlinear autocorrelation, at a given time-delay τ, for $\tau = 1, 2, 3$, for τ equal to the first zero-crossing of the autocorrelation function, and for τ equal to the first minimum of the automutual information. Outputs from the function are the raw tc3 expression,

its magnitude, the numerator and its magnitude, and the denominator. See *TSTOOL* documentation for details of the function.

- **CO_trev**. Calculates the *trev* function, a normalized nonlinear autocorrelation, which was found in the *TSTOOL* documentation, for $\tau = 1, 2, 3$, the first zero-crossing of the autocorrelation function, and the first minimum of the automutual information. The quantity is often used as a nonlinearity statistic in surrogate data analysis [224]. Outputs are the same as for **CO_tc3** above.

- **CO_glscf**. The generalized linear self-correlation function. This function was obtained from Duarte Queirós and Moyano [186] and implemented by us. The function takes inputs α, β, and τ and is applied to the z-scored time series. We calculate this function for $(\alpha, \beta) = (1, 1), (1, 2), (1, 5), (1, 10), (2, 2)$, and $(2, 5)$, and for each combination across time-delays $\tau = 1, 2, ..., 5$, and for τ equal to the first zero-crossing of the autocorrelation function.

- **CO_fzcglscf** returns the first zero-crossing of the generalized self-correlation function introduced in Duarte Queirós and Moyano [186], up to a maximum $\tau_{\max} = 400$. The function is evaluated for $\alpha = 1$, $\beta = 1, ..., 10$, and for the following additional pairs: $(\alpha, \beta) = (2, 2), (2, 5), (2, 10), (5, 5), (5, 10)$, and $(10, 10)$.

- **CO_acfshape**. Outputs a set of statistics summarizing how the autocorrelation function changes with the time lag, τ. Outputs include the number of peaks, and autocorrelation in the autocorrelation function itself.

- **TSTL_acf**. Estimates the autocorrelation function using a fast Fourier Transform method implemented in *TSTOOL*. This operation returns the mean square discrepancy between the autocorrelation coefficients obtained from **CO_autocorr** above, and those obtained from *TSTOOL* in this way.

F.4.2 Two-dimensional embeddings

- **CO_basicrecurf**. Obtains a set of measures of point density in a plot of x_i against $x_{i-\tau}$. These statistics are calculated for $\tau = 1$ and for τ equal to the first zero-crossing of the autocorrelation function. Outputs include the number

of points near the diagonal, and similarly, the number of points that are close to certain geometric shapes in the $x_{i-\tau}, x_\tau$ plot, including parabolas, rings, and circles.

- **CO_embed2**. Embeds the z-scored time series in a two-dimensional time-delay embedding space with a given time-delay, τ, and outputs a set of statistics about the structure in this space. Outputs include the distribution of angles between successive points in the space, stationarity of this angular distribution, euclidean distances from the origin, and statistics on outliers. This is evaluated for τ equal to the first zero-crossing of the autocorrelation function.

- **CO_embed2_angle_varytau**. Investigates how the autocorrelation of angles between successive points in the two-dimensional time-series embedding change as τ varies from $\tau = 1, 2, ..., \tau_{\max}$. We use $\tau_{\max} = 50$.

- **CO_embed2_dist**. Returns statistics on the sequence of successive Euclidean distances between points in a two-dimensional time-delay embedding space with a given time-delay, τ. Outputs include the autocorrelation of distances, the mean distance, the spread of distances, and statistics from an exponential fit to the distribution of distances.

- **CO_embed2_shapestats**. Takes a shape and places it on each point in the two-dimensional time-delay embedding space sequentially. The function counts the points inside this shape as a function of time, and returns statistics on this series. We implement this routine with τ as the first zero-crossing of the autocorrelation function, and using a circle of radius 0.1, and of radius 1. Outputs are of the constructed time series of the number of nearby points, and include the autocorrelation, maximum, median, mode, a Poisson fit to the distribution, histogram entropy, and stationarity over fifths of the time series.

F.4.3 Automutual information measures

Many estimates of automutual information are implemented across different time-delays, τ, and different statistics including the automutual information function are

returned.

- **CO_information**. Calculates the automutual information in a signal using **information** code by R. Moddemeijer from `http://www.cs.rug.nl/~rudy/matlab/`. Evaluated for time delays $\tau = 0, 1, ...10$.

- **CO_fmmi**. Calculates the first minimum of the automutual information function using the **CO_information** code above.

- **TSTL_amutual**. Uses **amutual** code from *TSTOOL*, which uses a histogram method with n bins to estimate the mutual information of a (z-scored) time series across a range of time-delays τ. The is calculated for n equal to the square root of one tenth the length of the time series for $\tau = 1, ..., 20$, and for $n = 10$ and $\tau = 1, ..., 20$. In addition, a number of statistics of the function over the range of τ are returned, including the mean mutual information, its standard deviation, first minimum, proportion of extrema, and measures of periodicity in the positions of local maxima.

- **TSTL_amutual2**. Uses **amutual2** code from *TSTOOL* up to a maximum time-delay $\tau_{\max}$. Statistics are returned on the output of **amutual2** over this range, as for **TSTL_amutual** above. We apply this algorithm to time series with $\tau_{\max} = 50$ for the z-scored time series, and also for incremental differences of the z-scored time series.

- **CO_ami_benhist**. Looks at automutual information using histograms, using different approaches to binning the data. Uses **hist2** function by Nedialko Krouchev, obtained from *MATLAB Central*[2]. The automutual information is calculated using (i) evenly-spaced bins through the range of the time series, (ii) bins that extend only up to a multiple of the standard deviation from the mean of the time series to exclude outliers, or (iii) equiprobable bins chosen using quantiles. For each method, the automutual information is calculated for a number of bins $n_{\text{bins}} = 2, 4, 6, 8, 10, 20$.

[2] `http://www.mathworks.com/matlabcentral/fileexchange/12346-hist2-for-the-people`

- **CO_ami_fmin**. Finds the first minimum of the automutual information by various different estimation methods, and sees how this varies over different coarse-grainings of the time series. The function returns a set of statistics on the set of first minimums of the automutual information function obtained over a range of the number of bins used in the histogram estimation, in this case $n_{\text{bins}} = 2, 3, ..., 80$. Outputs include the minimum, maximum, range, number of unique values, and the position and periodicity of peaks in the set of automutual information minimums. We calculate these statistics for four different methods of estimating the mutual information, via the function **CO_ami_benhist**.

- **CO_ami_noise**. Adds Gaussian-distributed noise to the time series with increasing standard deviation, η, across the range $\eta = 0, 0.1, ..., 2$, and measures the mutual information at each point using histograms with n_{bins} bins (implemented using **CO_ami_benhist**). The output is a set of statistics on the resulting set of automutual information estimates, including a fit to an exponential decay, since the automutual information decreases with the added white noise. We calculate these statistics for time delays $\tau = 1, 2$, and for $n_{\text{bins}} = 10$ or 20 bins. This algorithm was implemented by us, based on the ideas of 'noise titration' presented in Poon and Barahona [5].

F.4.4 Transition matrices

- **ST_transition**. Calculates the transition probabilities between different states of the time series given a method to symbolize the time series. We turn each time series into a symbolic string using an equiprobable alphabet of m letters. The transition probabilities are calculated at a lag τ. Outputs include the transition probabilities themselves, as well as the trace of the transition matrix, measures of asymmetry, and eigenvalues of the transition matrix. These statistics are calculated for $m = 2, 3, ..., 5$, and for each of $\tau = 1$ and τ equal to the first minimum of the autocorrelation function.

- **ST_dyntransition**. Calculates the transition probabilities and measures how they change as the size of the alphabet increases. Outputs include the decay

rate of the sum, mean, and maximum of diagonal elements of the transition matrices, changes in symmetry, and the eigenvalues of the transition matrix.

F.4.5 Periodicity

- **PD_wang07**. Implements the periodicity extraction measure proposed in Wang et al. [94]. The function detrends the time series using a single-knot cubic regression spline, and then computes autocorrelations up to one third of the length of the time series. The frequency is the first peak in the autocorrelation function satisfying a set of conditions. We calculate this periodicity measure for a set of thresholds, rather than just the single threshold of 0.01 considered in the original paper. We use $0, 0.01, 0.1, 0.2, 1/\sqrt{N}, 5/\sqrt{N}$, and $10/\sqrt{N}$, where N is the length of the time series.

- **DT_isseas**. A simple test of seasonality by fitting a *sin1* model to the time series using the MATLAB **fit** function. The output is binary: 1 if the goodness of fit R^2 exceeds 0.3 and the amplitude of the fitted periodic component exceeds 0.5; and 0 otherwise.

F.5 Basis function representations

F.5.1 Power spectral density

- **SP_basicmeasures**. Returns a set of measures summarizing an estimate of the Fourier transform of the signal. For this work, the estimation is done using either a periodogram, using the **periodogram** code in MATLAB's *Signal Processing Toolbox*, or a fast fourier transform, implemented using MATLAB's **fft** code. Statistics are returned on either the amplitudes of the Fourier transform, or the square amplitudes (i.e., the power spectral density). Cepstrum spectrums are obtained as logarithms of absolute deviations of the signal from its mean. Outputs are statistics summarizing various properties of the spectrum, including its maximum, minimum, spread, correlation, centroid, area in certain (normalized) frequency bands, moments of the spectrum, Shannon spectral entropy, a spectral flatness measure, power-law fits, and the number of crossings

of the spectrum at various amplitude thresholds.

F.5.2 Wavelets

All algorithms described in this section use code from MATLAB's *Wavelet Toolbox*.

- **WL_cwt**. Applies a continuous wavelet transform to the time series using the function **cwt** from MATLAB's *Wavelet Toolbox*. We implement this code using the Daubechies wavelet, 'db3', and the Symlet, 'sym2', wavelets using scales from 1, 2, ..., 32. Outputs from this function are statistics on the coefficients, entropy, and results of coefficients summed across scales.

- **WL_scal2frq**. Estimates frequency components in a periodic time series using functions from MATLAB's *Wavelet Toolbox*, including the **scal2frq** function. We implement this using the Daubechies wavelet, 'db3' and the maximum scale for this wavelet given the length of the time series (computed using **wmaxlev** from MATLAB's *Wavelet Toolbox*). Outputs are the level with the highest energy coefficients, the dominant period, and the dominant pseudo-frequency.

- **WL_powerlevels**. Compares the detail coefficients obtained at each level of the wavelet decomposition from 1 to the maximum possible level for the wavelet given the length of the input time series (computed using **wmaxlev** from MATLAB's *Wavelet Toolbox*). We implement this code using the Daubechies wavelet, 'db3'. Outputs are a set of statistics on the detail coefficients.

- **WL_coeffs**. Performs a wavelet decomposition of the time series using a given wavelet at a given level and outputs a set of statistics on the coefficients obtained. This code is implemented using the Daubechies wavelet, 'db3', and for levels 1, 2, ..., 5.

- **WL_dwtcoeff**. Decomposes the time series using a given wavelet and outputs statistics on the coefficients obtained up to a maximum level $l_{\max}$. We implement this code using the Daubechies wavelet, 'db3', and the Symlet, 'sym2'.

F.6 Stationarity

F.6.1 Splits

Operations described in this section split the time series into subsegments, compute local measures of certain quantities in these segments, and then compares them to global measures obtained across the full time series.

- **SY_StatAv**. The *StatAv* measure is a simple mean-stationarity metric that divides the z-scored time series into non-overlapping subsegments, calculates the mean in each of these segments and returns the standard deviation of this set of means [123]. We implement this simple metric with different numbers of segments: $n_{\text{seg}} = 2, 3, ..., 10$ and with by specifying different fixed-length segments to divide the time series into: $l_{\text{seg}} = 25, 20, 100, 150, 200, 250, 500, 1\,000$, and $2\,000$.

- **SY_slidwin**. This family of operations is based on sliding a window along the time series, measuring some quantity in each window, and outputting some measure of this set of local estimates of that quantity. The first input to this function specify the measure to calculate in each window: mean, standard deviation, distribution entropy, skewness, kurtosis, the fifth moment of the distribution, the p-value for a Lilliefors Gaussianity test, and the autocorrelation at lag 1. The second input controls how the obtained sequence of local estimates is compared (as a ratio to the full time series): standard deviation, Approximate Entropy [29], or histogram entropy. The final inputs control the number of subsegments, n_{seg}, to divide the time series up into (thereby controlling the window length) for which we use $n_{\text{seg}} = 2, 5, 10$, and the increment, f_{move}, at each step, as a fraction of the window length which we use $f_{\text{move}} = 0.1, 0.5, 1$.

- **SY_dynwin**. Measures how stationarity estimates depend on the number of segments used to split the time series up. Specifically, variation in a range of local measures are implemented: mean, standard deviation, skewness, kurtosis, ApEn(1,0.2), SampEn(1,0.2), AC(1), AC(2), and the first zero-crossing of the autocorrelation function. The standard deviation of local estimates of these

quantities across the time series are calculated as an estimate of the stationarity in this quantity as a function of the number of splits, n_{seg}, of the time series, from $n_{\text{seg}} = 2, 3, ..., 10$. Outputs of the operation are the standard deviation of this set of 'stationarity' estimates across these window sizes.

- **SY_dynpick**. Implements a bootstrap-based measure of stationarity: 100 time-series segments of a specified length are selected at random from the time series and in each segment a local quantity is calculated: mean, standard deviation, skewness, kurtosis, ApEn(1,0.2), SampEn(1,0.2), AC(1), AC(2), and the first zero-crossing of the autocorrelation function. Outputs are the mean and also the standard deviation of this set of 100 local estimates. We calculate these outputs for different fixed lengths of subsegments, $l = 50, 100, 200$, and for $l = 2\tau_0, 5\tau_0$, where τ_0 represents the first zero-crossing of the autocorrelation function.

- **SY_detrendq**. Linearly detrends the time series using the MATLAB algorithm **detrend**, and returns the ratio of standard deviations before and after the linear detrending. If a strong linear trend is present in the time series, this metric will output low values.

- **SY_drifting**. Implements an idea from a MATLAB forum[3]. Splits the time series into segments, computes the mean and variance in each segment and compares the maximum and minimum mean to the mean variance. This is done by splitting the time series into a number of segments $n_{\text{seg}} = 5, 10$, or by specifying a fixed segment length $l = 20, 50$, and 100 samples.

- **SY_compdtn**. Compares the distribution in n consecutive partitions of the signal, returning the sum of differences between each kernel-smoothed distributions (using the MATLAB function **ksdensity**). The operation behaves in one of two modes: (i) *each* compares the distribution in each segment to that in every other segment, and (ii) *par* compares each distribution to the so-called 'parent' distribution, that of the full signal. We compute the statistic for both modes and for a number of segments $n_{\text{seg}} = 2, 3, 4, 5$. Outputs from the opera-

[3] `http://www.mathworks.de/matlabcentral/newsreader/view_thread/136539`

tion are measures of the sum of absolute deviations between distributions across the different pairwise comparisons.

- **ST_cumul**. Compares statistics measured in a local region of the time series to that measured of the full time series. The local sample is chosen according to one of the following four rules: (i) the first n points in a time series, for which we use $n = 10, 20, 50, 100, 500$ (ii) an initial proportion p of the full time series, for which we use $p = 0.001, 0.005, 0.01, 0.1, 0.5$ (iii) n evenly-spaced points throughout the time series, where we use $n = 10, 20, 50, 100, 500$ (iv) n randomly-chosen points from the time series (chosen with replacement), for which we use $n = 10, 20, 50, 100, 500$. Statistics outputted are the mean, standard deviation, median, interquartile range, skewness, kurtosis, AC(1), and SampEn(1,0.1). This is not the most reliable metric because only a single sample is taken from the time series and compared to the full time series.

- **ST_benlocext**. Finds maximums and minimums within given segments of the time series and analyses the results. The segment length is an input to the function, and can specify either a length, l, in samples, or a number of equal-length segments, n_{seg}, to break the time series up into. Outputs of this operation quantify how these local maximums and minimums vary across the time series. We implement this algorithm for segment lengths $l = 50, 100$, and by specifying a fixed number of segments as $n_{\text{seg}} = 25, 50, 100$.

- **TISEAN_nstat_z**. Uses the `nstatz` routine from the TISEAN package for nonlinear time-series analysis to calculate cross-forecast errors of zeroth-order models for the z-scored time series embedded in a time-delay embedding space. Inputs include the number of subsegments to divide the data into, n_{seg}, and the time-delay embedding parameters, m and τ. The model is fit in each of n_{seg} equally-spaced segments of the time series, and used to predict the other time series segments. We implement this code using $(n_{\text{seg}}, \tau, m) = (4, 1, 3)$, (5,1,3), and (4,'ac',3), where 'ac' chooses τ as the first zero-crossing of the autocorrelation function. Outputs include the trace of the cross-prediction error matrix, the mean, minimum, and maximum cross-prediction error, the mini-

mum off-diagonal cross-prediction error, and eigenvalues of the cross-prediction error matrix.

- **ST_momcorr**. Calculates correlations between simple statistics summarizing the distribution of values in local windows of the signal. Inputs to the function are the sliding window length, l, the overlap between consecutive windows as a fraction of the window length, p, the statistics to investigate correlations between, s_1 and s_2, and the transformation to first apply to the time series, T. We consider three local statistics: (i) *mean*, the local mean of the time series, (ii) *std*, the local standard deviation of the time series (about its local mean), and *iqr*, the local interquartile range of the time series. We consider two transformations of the time series which are applied as a pre-processing: (i) *abs*: takes absolute values of all data points, and (ii) *sqrt*: takes the square root of absolute values of all data points. This code is implemented for for following parameter sets: (l, p, s_1, s_2, T) = (0.02N, 0.2, 'mean', 'std', 'none'), (0.02N, 0.2, 'median', 'iqr', 'none'), (0.02N, 0.2, 'median', 'std', 'abs'), (0.02N, 0.2, 'median', 'iqr', 'abs'), (0.02N, 0.2, 'mean', 'std', 'sqrt'), and (0.02N, 0.2, 'median', 'iqr', 'sqrt'), where N represents the length of the time series.

F.6.2 Change points

- **WL_varchg**. Finds variance change points using functions from MATLAB's *Wavelet Toolbox*, including the primary function **wvarchg**, which estimates the change points in the time series. The output from this algorithm is the optimal number of change points. We implement the code using both the Daubechies wavelet, 'db3', and the Symlet, 'sym2' as mother wavelets. We use a minimum delay equal to 0.01N, where N is the time-series length, and set a maximum number of change points at 10. For each choice of the mother wavelet, we look at levels 2, 3, 4, and 5.

- **ML_step_detection**. Gives information about discrete steps in the signal, using the function **l1pwc** from Max A. Little's step detection toolkit [260]. Note that this code is based on code originally written by Kim et al. [263]. The

input to this operation is the regularization parameter, λ. We return statistics on the output of this function, including the intervals between change points, the proportion of constant segments, the reduction in variance from removing the piece-wise constants, and stationarity in the occurrence of change points. We implement this code using $\lambda = 0.05, 0.2$, and 10.

- **ML_l1pwc_sweep_lambda**. Gives information about discrete steps in the signal across a range of regularization parameters λ, using the function **l1pwc** from Max A. Little's step detection toolkit [260]. At each iteration, the **ML_step_detection** code was run with a given λ, and the number of segments, and reduction in root mean square error from removing the piecewise constants was recorded. Outputs summarize how the these quantities vary with λ. We implement this code using the following range of $\lambda = 0, 0.05, 0.1, ..., 0.95$.

F.7 Scaling

- **fastdfa**. Measures the scaling exponent of the time series using a fast implementation of detrended fluctuation analysis (DFA). Coded by Max A. Little and publicly-available at `http://www.physics.ox.ac.uk/users/littlem/software/index.html`.

- **ST_hurst_exponent**. Code by Bill Davidson that estimates the Hurst Exponent of an input time series. Code was obtained from `http://www.mathworks.com/matlabcentral/fileexchange/9842`.

- **SC_benfa**. Implements fluctuation analysis by a variety of methods. Much of our implementation is based on the well-explained discussion of scaling methods in Talkner and Weber [118]. The main difference between algorithms for estimating scaling exponents amount to differences in how fluctuations, F, are quantified in time-series segments. Many alternatives are implemented in this operation: (i) *endptdiff* calculates the differences in end points, (ii) *range* calculates the range, (iii) *std* takes the standard deviation (similar to Cannon et al. [264]), (iv) *iqr* takes the interquartile range, (v) *dfa* removes a polynomial trend

of order k, (vi) *rsrange* returns the range after removing a straight line fit (as in Caccia et al. [149]), and (vii) *rsrangefit* fits a polynomial of order k and then returns the range. The parameter q controls the order of fluctuations, for which we mostly use the standard choice, $q = 2$, corresponding to root mean square fluctuations. An optional input parameter to this operation is a time lag for computing the cumulative sum (or integrated profile), as suggested by Alvarez-Ramirez et al. [265]. We choose the spacing of time scales τ either linearly or logarithmically through a range from 5 samples to a quarter of the length of the time series. Outputs include statistics of fitting a linear function to a plot of $\log(F)$ as a function of $\log(\tau)$, and for fitting two straight lines to the same data, choosing the split point at $\tau = \tau_{\text{split}}$ as that which minimizes the combined fitting errors. We implement many versions of this algorithm, using each of the methods for measuring fluctuation described above, and for DFA using a range of polynomial orders k. We also apply this algorithm applied to absolute deviations of the time series from its mean, and also for just the sign of deviations from the mean (i.e., converting the time series into a series of +1, when the time series is above its mean, and -1 when the time series is below its mean). All results are obtained with both linearly, and logarithmically-spaced time scales τ.

- **WL_fBM**. Uses the **wfbmesti** function from MATLAB's *Wavelet Toolbox* to estimate the parameters of fractional Gaussian Noise, or fractional Brownian motion in a time series. All three outputs of **wfbmesti** are returned from this algorithm.

F.8 Entropy

F.8.1 Distributional entropies

- **EN_entropies**. Uses the MATLAB **wentropy** code from the Wavelet toolbox to estimate the entropy of the z-scored input time series, using 'shannon', 'logenergy', 'threshold' (with a range of thresholds $r = 0.1, 0.2, 0.5, 1, 1.5, 2$), and 'sure' (with a range of parameters $r = 0.1, 0.2, 0.5, 1, 1.5, 2$).

- **EN_pdfent**. Estimates the Shannon entropy from the probability distribution of the time series.

- **EN_ruden**. Measures the entropy of the time series using an algorithm obtained from `http://www.cs.rug.nl/~rudy/matlab/` by R. Moddemeijer.

- **EN_histent**. Estimates of entropy from the static distribution of the time series. The distribution is estimated either using a histogram with n bins (we use $n = 5, 10, 20, 50$), or as a kernel-smoothed distribution, using MATLAB's **ksdensity** algorithm with width parameter $w = 0.01$, 0.05, 0.1, 0.2, 0.5, 1, and for w equal to the default value (that selects an 'optimal' width parameter from the data). An optional parameter can be used to remove a proportion p_{ext} of the most extreme negative deviations from the mean and a proportion p_{ext} of the most positive deviations from the mean as an initial pre-processing. We use $p_{\text{ext}} = 0, 0.01, 0.02, 0.05, 0.1, 0.2, 0.3$.

- **EN_TSentropy**. Estimates the Tsallis entropy of a signal using a parameter q that measures the non-extensivity of the system; $q = 1$ recovers the Shannon entropy. We evaluate this quantity for $q = 0.2$, 0.4, 0.6, 0.8, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5. Uses code obtained written by D. Tolstonogov and obtained from `http://download.tsresearchgroup.com/all/tsmatlablink/TSentropy.m`.

F.8.2 Embedding entropies

- **EN_ApEn**. Estimates the Approximate Entropy of the time series, ApEn(m, r) [2]. We calculate for $(m, r) = (1, 0.1), (1, 0.2), (2, 0.1)$, and $(2, 0.2)$.

- **EN_sampenc2**. Estimates the Sample Entropy of the time series, SampEn(m, r) [50]. All combinations of $m = 1$, 2, 3, 4, and $r = 0.05, 0.1, 0.15, 0.2, 0.3$ are evaluated. We also calculate the SampEn of successive increments of time series, i.e., we apply an initial incremental differencing pre-processing, as used in the so-called Control Entropy quantity [63]. In this case we evaluate for all combinations of $m = 1, 2, 3, 4$ and $r = 0.1, 0.2$. Our implementation is based on publicly-available code from `http://www.physionet.org/physiotools/sampen/matlab/`

`1.1/sampenc.m`.

F.8.3 Symbolic strings

- **IN_surprise**. Coarse-grains the time series, turning it into a sequence of symbols of a given alphabet size, n_g, and quantifies measures of surprise of a process with local memory of the past m values of the symbolic string. The symbolization was done according to three different methods: (i) *quantile*: an equiprobable alphabet by the value of each time-series datapoint, (ii) *updown*: an equiprobable alphabet by the value of incremental changes in the time-series values, and (iii) *embed2quadrants* by the quadrant each data point resides in in a two-dimensional embedding space. We then consider a memory length, m, of the time series, and use the data in the proceeding m samples to inform our expectations of the following sample. This expectation is in the form of probabilities based on (i) *dist*: the values of the time series in the previous m samples, (ii) *T1*: the one-point transition probabilities in the previous m samples, and (iii) *T2*: the two-point transition probabilities in the previous m samples. The 'information gained', $\log(1/p)$, at each sample using expectations calculated from the previous m samples, is estimated. Outputs of this operation are summaries of this series of information gains, including the minimum, maximum, mean, median, lower and upper quartiles, and standard deviation. This operation is calculated for symbolizations *quantile* and *updown* using alphabet sizes $n_g = 2$, 3, 4, and 5, and memories $m = 5$, 20, 50, and 100 samples for each of the three expectation methods described above. We also calculate this operation for *embed2quadrants* using the following parameter sets: (n_g, τ, *expectation process*) = (5, 1, 'dist'), (5, 'ac', 'dist'), (10, 'ac', 'dist'), (20, 'ac', 'dist'), (10,'ac','T1'), (20,'ac','T1'), (50,'ac','T1'), (50,'ac','T2'), (100,'ac','T2').

- **ST_benbin**. Measures the longest stretch of consecutive zeros or ones in a symbolized time series as a proportion of the time-series length. The time series is symbolized to a binary string by whether it's above (1) or below (0) its mean.

- **MS_complexity**. Calculates the Lempel-Ziv complexity of a n-bit encoding of the time series using Michael Small's code **complexity** [130], available at `http://small.eie.polyu.edu.hk/matlab/`. The code uses the associated mex file compiled from **complexitybs.c**. The output is the normalized complexity, the number of distinct symbol sequences in the time series divided by the expected number of distinct symbols for a noise sequence. We implement this for $n = 2, 3, ..., 10$ for z-scored time series, and for the incremental differences of time series (i.e., using an initial pre-processing of incremental time-series differences).

- **MS_shannon**. Calculates the approximate Shannon entropy of a time series using a n-bin encoding and d symbol sequences. Uniform population binning is used, and the implementation uses Michael Small's code **shannon** [130], available at `http://small.eie.polyu.edu.hk/matlab/`. Our code varies over d and n to return statistics on how the obtained entropies change. For example, we consider variation in entropy across the range $n = 2, 3, ..., 10$ for $d = 2$, $d = 3$, and $d = 4$. We also calculate statistics on the entropies obtained across a range of $d = 1, 2, ..., 10$ for $n = 2$, $n = 3$, and $n = 4$. We also calculate single entropies for each of the pairs $(n, d) = (2, 2)$, (2,3), and (3,2).

- **MD_polvar**. Calculates the *Plvar* measure for a time series. The measure was originally applied to sequences of RR intervals [177] and the code is based on that provided by Max A. Little. Inputs are the symbolic coding (amplitude) difference, d, and the word length, D. The output from this function is the probability of obtaining a sequence of consecutive ones or zeros. Although the original measure used raw thresholds on RR intervals (in milliseconds), we apply the code to z-scored time series. We implement this algorithm using parameters $(d, D) = (1, 3)$, $(0.5, 3)$, $(0.1, 3)$, $(1, 5)$, $(0.5, 5)$, $(0.1, 5)$, $(1, 4)$, $(0.5, 4)$, $(0.1, 4)$, $(1, 6)$, $(0.5, 6)$, and $(0.1, 6)$.

- **ST_benincdec**. Returns statistics on a binary symbolization of the time series (to a symbolic string of 0s and 1s), which is done according to some rule, either: (i) *diff*: by whether incremental differences of the time series are positive (1), or negative (0), (ii) *mean*: by whether each point is above (1) or below the

mean (0), or (iii) by whether the time series is within the interquartile range (1), or not (0). Outputs include the Shannon entropy of the string, the longest stretches of 0s or 1s, the mean length of consecutive 0s or 1s, and the spread of consecutive strings of 0s or 1s.

- **ST_benmotif_bin**. Looks at local motifs in a binary symbolization of the time series, which is performed by: (i) *diff*: incremental time-series increases are encoded as 1, and decreases as 0, (ii) *mean*: time-series values above its mean are given 1, and those below the mean are 0, (iii) *median*: time-series values above the median are given 1, and those below the median 0. Outputs are probabilities of words in the binary alphabet of lengths 1, 2, 3, and 4, and their entropies.

- **ST_benmotif_tri**. As for **ST_benmotif_bin** described above, but using an alphabet of three letters. Statistics are returned on words of length 1, 2, 3, and 4.

F.8.4 Other measures

- **EN_permen**. Computes the permutation entropy [266] of a time series of order m. We use $m = 2, 3, 4, 5$. Code is based on **logisticPE.m** code obtained from `http://www.nbb.cornell.edu/neurobio/land/PROJECTS/Complexity/index.html`.

- **EN_prograz**. Progressively randomizes the input time series according to some randomization scheme, and returns measures of how the properties of the time series change with this process. We implement three different randomization schemes: (i) *statdist*, which substitutes a time series data point with a random point in the original time series at each iteration, (ii) *dyndist*, which overwrites a random element of the time series with another random element at each iteration, and (iii) *permute*, which randomly permutes two elements of the time series at each iteration. The procedure is repeated $2N$ times, where N is the length of the time series. Outputs of the algorithm summarize how the

properties change as one of these randomization procedures is iterated, including the cross correlation with the original time series, the autocorrelation of the randomized time series, its entropy, and stationarity. Statistics are calculated every $N/10$ iterations, and thus 20 times throughout the process in total. Most statistics measure how these properties decay with randomization, by fitting a function $f(x) = A\exp(Bx)$.

F.9 Nonlinear time-series analysis

F.9.1 Dimension estimates

- **TISEAN_d2**. Implements the `d2` routine from the popular *TISEAN* package for nonlinear time series analysis [92]. Our code uses the outputs from this algorithm to return a set of informative features about the results. The inputs include a time-delay, τ, a maximum embedding dimension, $m_{\max}$, and a Theiler window, w [13, 237]. We use $(\tau, m_{\max}, w) = (1, 10, 0)$, and ('ac',10,0.01$N$), where 'ac' chooses τ as the first zero-crossing of the autocorrelation function, and N is the length of the time series in samples. We compute Taken's estimator for the correlation dimension, and related statistics, including other dimension estimates by finding appropriate scaling ranges, and searching for a flat region in $h2$ output, which indicates determinism/deterministic chaos. To find a suitable scaling range, we use a penalized regression procedure to determine an optimal scaling range that simultaneously spans the greatest range of scales and shows the best fit to the data, and return the range, a goodness of fit statistic, and a dimension estimate.

- **TISEAN_c1**. Implements the `c1` and `c2d` routines from *TISEAN*. Outputs optimal scaling ranges and dimension estimates for a time delay, τ, embedding dimensions m ranging from $m_{\min}$ to $m_{\max}$, a time separation, t_{sep}, and a number of reference points, N_{ref}. We use parameter sets $(\tau, m_{\min}, m_{\max}, t_{\text{sep}}, N_{\text{ref}}) = (1, 1, 7, 0.02N, 0.05N)$ and $(1, 2, 6, 25, 0.1)$, where N is the length of the time series. Note that we fix N_{ref} to be no less than 100 and no greater than 2 000 points.

- **TSTL_cao**. Implements *TSTOOL* code **cao**, with maximum embedding dimension $m_{\max}$, time delay τ, number of nearest neighbors n, and number of reference points N_{ref}. This code determines the minimum embedding dimension for a time series using Cao's method [254]. It computes the quantities E and E^* for a range of embedding dimensions $m = 1, ..., m_{\max}$. We implement the code using parameter sets $(m_{\max}, \tau, n, N_{\text{ref}})$ = (10, 'mi', 3, 500), (10, 'mi', 2, 100), and (10, 'mi', 2, $N/5$), where 'mi' chooses τ as the first minimum of the automutual information function and N is the number of samples in the input time series. Outputs are statistics on the result, including when the output quantity first passes a given threshold, and the m at which it levels off.

- **TSTL_corrdim**. Uses *TSTOOL* code **corrdim** to estimate the correlation dimension of a time-delay embedded time series using a box-counting approach. Inputs are the number of bins, n_{bins}, which sets the maximum number of partitions per axis, the time delay τ, and the embedding dimension m. We implement this code for $n_{\text{bins}} = 50, \tau$ chosen as the first zero-crossing of the autocorrelation function, and $m = 5$. Output statistics are simple summaries of the outputs from this algorithm.

- **TSTL_corrsum**. Uses *TSTOOL* code **corrsum** to compute scaling of the correlation sum for a time-delay reconstructed time series by the Grassberger-Proccacia algorithm [267] using fast nearest neighbor search. Inputs are the number of (randomly-chosen) reference points, N_{ref}, the maximum search radius relative to the attractor size, r, the Theiler window, w, number of bins, n_{bins}, and time-delay embedding parameters m and τ. We implement this code with $(N_{\text{ref}}, r, w, n_{\text{bins}}, m, \tau)$ = ('all', 0.5, 40, 20, 'ac', 'cao'), ('all', 0.5, 100, 20, 'ac', 'cao'), ('all', 0.1, 40, 20, 'ac', 'cao'), and ('all', 0.1, 40, 40, 'ac', 'cao'), where 'all' uses all available reference points, 'ac' chooses τ as the first zero-crossing of the autocorrelation function, and 'cao' chooses the embedding dimension by Cao's method via the code **TSTL_cao**. Outputs of the operation are basic statistics on the outputs of **corrsum**, including iteratively re-weighted least squares linear fits to log-log plots using the **robustfit** function in MATLAB's

Statistics Toolbox. All calculations are repeated using **corrsum2** code from *TSTOOL*.

- **TSTL_dimensions**. Computes the box counting, information, and correlation dimension of a time-delay embedded time series using the *TSTOOL* code **dimensions**. We implement the code for a maximum number of partitions per axis, $n_{\text{bins}} = 50$, a time-delay τ chosen as the first minimum of the autocorrelation function, and an embedding dimension m chosen by Cao's method using **TSTL_cao**. We have developed extensive code for estimating the best scaling range to estimate the dimension using a penalized regression procedure, and produce a range of statistics about how each dimension estimate changes with m, the scaling range in r, and the m at which the best fit is obtained.

- **TSTL_fracdims**. Computes the fractal dimension spectrum, $D(q)$, using moments of neighbor distances for time-delay embedded time series using the *TSTOOL* code **fracdims**. The code takes a number of inputs: the minimum number of neighbors for each reference point, k_{min}, the maximum number of neighbors for each reference point, k_{max}, the number of randomly-chosen reference points, N_{ref}, the starting value for moments, g_{start}, the end value for moments, g_{end}, the Theiler window, w, the number of moments to calculate, n_{mom}, and the embedding parameters τ and m. We implement the code using $(k_{\text{min}}, k_{\text{max}}, N_{\text{ref}}, g_{\text{start}}, g_{\text{end}}, w, n_{\text{mom}}, \tau, m)$ = (5, 20, $0.2N$, 1, 10, 0, 32, 'ac', 'cao'), (5, 100, $0.2N$, 1, 10, 0, 32, 'ac', 3), (2, 100, $0.2N$, 1, 5, 10, 32, 'ac', 5), (2, 100, $0.2N$, 1, 5, 10, 32, 1, 5), and (2, 10, 'all', 1, 5, 10, 32, 1, 5), where 'ac' refers to τ chosen as the first zero-crossing of the autocorrelation function, 'cao' refers to the embedding dimension chosen using Cao's method via **TSTL_cao**, 'all' uses all available reference points, and N is the length of the time series in samples. Outputs include basic statistics of $D(q)$ and q, statistics from a linear fit, and an exponential fit of the form $D(q) = A\exp(Bq) + C$.

- **TSTL_takens_estimator**. Implements the Taken's estimator for correlation dimension [232] using the *TSTOOL* code **takens_estimator**. Inputs are the number of reference points, N_{ref}, maximum search radius, r, Theiler window,

w, and the embedding parameters τ and m. We implement this code for $(N_{\text{ref}}, r, w, \tau, m)$ = ('all', $0.05N$, $0.05N$, 1, 8), ('all', $0.05N$, $0.05N$, 1, 3), ('all', $0.05N$, $0.05N$, 'ac', 8), ('all', $0.05N$, $0.05N$, 'ac', 3), ('all', $0.05N$, $0.05N$, 'mi', 8), ('all', $0.05N$, $0.05N$, 'mi', 3), ('all', $0.05N$, $0.05N$, 'mi', 'cao'), ('all', $0.05N$, $0.05N$, 'mi', 'fnnmar'), and ('all', $0.1N$, $0.05N$, 1, 10), where 'all' uses all available reference points, 'ac' sets the time delay as the first zero-crossing of the autocorrelation function, 'mi' sets the time delay as the first minimum of the mutual information function, 'cao' chooses the embedding dimension using Cao's method, 'fnnmar' sets the embedding dimension using a false nearest neighbors approach, and N is the length of the time series in samples. The output of this operation is simply the Taken's estimator of the correlation dimension, $d2$.

F.9.2 Time-delay embedding

- **NL_fnnmar**. This operation uses the *Cross Recurrence Plot Toolbox*, Version 5.13, Release 26, available at `http://www.agnld.uni-potsdam.de/~marwan/toolbox/` [268]. Being a visualization tool, this is relatively slow code and requires manual tweaking of parameters, making it less suitable for our purposes. This code uses the **fnn** code from the Toolbox to estimate the proportion of false nearest neighbors at different embedding dimensions $m = 1, 2, ..., m_{\max}$. We use $m_{\max} = 10$, a radius threshold $r = 2$, and with three time delays: (i) $\tau = 1$, (ii) τ equal to the first zero-crossing of the autocorrelation function, and (iii) τ equal to the first minimum of the automutual information function. Outputs are the proportion of false nearest neighbors at different m, and the point at which the proportion of false nearest neighbors drops below various thresholds.

- **MS_fnn**. Determines the number of false nearest neighbors for the embedded time series using Michael Small's false nearest neighbor code [130], available at `http://small.eie.polyu.edu.hk/matlab/`. The algorithm returns statistics on the proportion of false nearest neighbors as a function of the embedding dimension $m = m_{\min}, m_{\min} + 1, ..., m_{\max}$ for a given time lag τ, and distance

threshold for neighbors, d_{th}. False nearest neighbors are judged using a ratio of the distances between the next k points and the neighboring points of a given datapoint. We implement the code using $m_{\text{min}} = 1$ and $m_{\text{max}} = 10$, a time delay τ chosen as the first minimum of the automutual information function, $d_{\text{th}} = 5$, and $k = 1$. Outputs include the proportion of false nearest neighbors at each m, the mean and spread, and the smallest m at which the proportion of false nearest neighbors drops below each of a set of fixed thresholds.

- **TSTL_delaytime**. Uses the *TSTOOL* code **delaytime**, that computes an optimal delay time using the method of Parlitz and Wichard (this method is specified in the *TSTOOL* documentation but without reference). We implement the code with a maximum delay of $0.1N$, where N is the length of the time series, and a 'past' parameter of 1 and return some basic statistics on the resulting output.

F.9.3 Other measures

- **TSTL_acp**. Implements the *TSTOOL* routine **acp** using a time lag τ, a Theiler window w, maximum delay τ_{max}, maximum embedding dimension m_{max}, and number of reference points N_{ref}. We implement the code using parameter values $(\tau, w, \tau_{\text{max}}, m_{\text{max}}, N_{\text{ref}}) =$ ('mi',1,$N/4, 10, N/10$) and $(1, 0.01, N/4, 10, N/2)$, where 'mi' chooses τ as the first minimum of the mutual information function, and N is the length of the time series in samples. Outputs are statistics summarizing the output of the routine.

- **TSTL_localdensity**. Uses *TSTOOL* code **localdensity**, which is poorly documented in the *TSTOOL* documentation, but we can assume it returns local density estimates in the time-delay embedding space. Parameters are the number of nearest neighbors, n, a Theiler window, w, and the embedding parameters τ and m. We implement this code using parameter values $(n, w, \tau, m) =$ (5,40,'ac','cao'), (5,40,'ac',2), and (5,40,1,3), where 'ac' chooses τ as the first zero-crossing of the autocorrelation function, and 'cao' chooses the embedding dimension using Cao's method via **TSTL_cao**. Outputs are various statistics

on the local density estimates at each point in the time-delay embedding, including the minimum and maximum values, the range, the standard deviation, mean, median, and autocorrelation.

- **TSTL_poincare**. Obtains a Poincaré section of the embedded time series, producing a set of vector points projected orthogonal to the tangential vector at the specified index using *TSTOOL* code **poincare**. Our code then tries to obtain interesting structural measures from this output. Inputs include the reference point, which we choose as the first maximum in the time series, and the embedding parameters, which are chosen as (τ, m) = ('mi',3), ('ac',3), and (1,3), where 'mi' chooses τ as the first minimum in the automutual information function, and 'ac' chooses τ as the first zero-crossing in the autocorrelation function. Since $m = 3$ in each case, the resulting vectors are two dimensional. Outputs include statistics on the x and y components of these vectors on the Poincaré surface, on distances between adjacent points, distances from the mean position, and the entropy of the vector cloud.

- **TSTL_largelyap**. Computes the largest Lyapunov exponent of a time-delay reconstructed time series using the *TSTOOL* code **largelyap**. The algorithm used is very similar to the Wolf algorithm [269]. Inputs are the number of reference points, N_{ref}, the maximum prediction length, $t_{\max}$, Theiler window, w, number of nearest neighbors, n, and the embedding parameters τ and m. We implement the algorithm using $(N_{\text{ref}}, t_{\max}, w, n, \tau, m)$ = ($0.5N$, $0.1N$, $0.01N$, 3, 'mi', 'cao') and for ('all', $0.1N$, $0.01N$, 3, 1, 4), where N represents the length of the time series in samples, 'all' uses all available reference points, 'mi' chooses τ as the first minimum of the automutual information function, and 'cao' chooses m using Cao's method via **TSTL_cao**. Outputs are a range of statistics on the outputs from this function, including a penalized linear regression to the scaling range in an attempt to fit to as much of the range of scales as possible while simultaneously achieving the best possible linear fit.

- **TSTL_return_time**. Computes a histogram of return times, the time taken for the time series to return to a similar location in phase space for a given

reference point using the code **return_time** from *TSTOOL*. Strong peaks in the histogram are indicative of periodicities in the data. Input parameters include the number of nearest neighbors to use in calculations, n, the maximum return time to consider, $t_{\max}$, the Theiler window, w, the number of reference points N_{ref}, and the embedding parameters, τ and m. We implement this code using $(n, t_{\max}, w, N_{\text{ref}}, \tau, m)$ = (10,1,1,'all','ac',8), (5,1,40,'all',1,8), and (0.05N,1,0.05N,'all',1,3), where N represents the length of the time series, 'ac' chooses τ by the first zero-crossing of the autocorrelation function, and 'all' uses all available reference points. Outputs include basic measures from the histogram, including the occurrence of peaks, spread, proportion of zeros, and the distributional entropy.

F.10 Nonlinearity

- **KP_timerev**. Calculates a time reversal asymmetry statistic using the *timerev* code by D. Kaplan obtained at `http://www.macalester.edu/~kaplan/software/`. The method was implemented for time lags $\tau = 1, 2, 3, 4$.

- **KP_quickde**. A simple characterization of determinism, as coded by D. Kaplan obtained at `http://www.macalester.edu/~kaplan/software/`. The code requires an embedding dimension, m, a time lag τ, and a number of points $n_{\min}$ for $\delta - \epsilon$ fitting. We implement this code for $m = 1, 2, ..., 5$ and $\tau = 1, 2$, and $n_{\min} = 500$.

F.10.1 Surrogate data

- **TSTL_surr_nlac**. Generates surrogate time series and tests them against the original time series according to some test statistics: T_{C3}, using the *TSTOOL* code **tc3** or T_{rev}, using *TSTOOL* code **trev**. Inputs are the time delay τ, the number of surrogate datasets to generate, n_{surr}, the surrogate data method, *surrmeth*, and the surrogate function: either 'tc3' or 'trev'. We implement for both the 'tc3' and 'trev' statistics and for all three surrogate data methods using the following parameter values: (τ, n_{surr},*surrmeth*) = (1, 100), ('mi', 100), where

'mi' chooses τ by the first minimum of the automutual information function. Outputs from the operation include the Gaussianity of the test statistics, a z-test, and various tests based on fitted kernel densities.

- **SD_surrogatetest**. Analyzes the test statistics obtained from surrogate time series compared to those obtained from the given time series. The code, developed by us, relies on information found in Kugiumtzis [270], Guarín López et al. [271]. The first input to this operation is the method for generating surrogate time series: (i) *RP*: random phase surrogates that maintain linear correlations in the data but destroy any nonlinear structure through phase randomization, (ii) *AAFT*: the amplitude-adjusted Fourier transform method maintains linear correlations but destroys nonlinear structure through phase randomization, yet preserves the approximate amplitude distribution, or (iii) *TFT*: preserves low-frequency phases but randomizes high-frequency phases (as a way of dealing with non-stationarity [271]). Another input specifies the test statistic to evaluate on all surrogate time series and the original time series: (i) *ami*: the automutual information at lag 1 [272], (ii) *fmmi*: the first minimum of the automutual information function, (iii) *o3*: a third-order statistic used in Schreiber and Schmitz [224], or (iv) *tc3*: a time-reversal asymmetry measure. Outputs of the function include a z-test between the two distributions, and some comparative rank-based statistics. We implement this code using *RP* and 99 surrogates and for each of the test statistics listed above.

F.11 Time-domain transformations

- **PP_compareba**. Applies a given transformation to the time series, and returns statistics on how various time-series properties change as a result. We implement the following transformations: (i) *poly*: polynomial detrendings, both linear and quadratic, (ii) *sin*: sinusoidal detrending with either one or two frequency components, (iii) *spline* removes a least squares spline using MATLAB's *Spline Toolbox* function **spap2** using 2 knots and 4th order interpolants, 4 knots and 4th order interpolants, and with 6 knots and 4th order interpolants, (iv)

diff takes incremental differences of the time series, for which we implement single and twice-differenced time series, (v) *medianf* implements a running median filter using window lengths 2, 3, 4, and 10, (vi) *rav* running mean filter using windows of length 2, 3, 4, and 10, (vii) *resample* resamples the data by a given ratio, we use 1:2 and 2:1, (viii) *logr* takes log returns of the data, and (ix) *boxcox* makes a Box-Cox transformation of the data. Output statistics include comparisons of stationarity and distributional measures between the original and transformed time series.

- **PP_progpp**. Iteratively applies a transformation to the time series: (i) *spline* removes a spine fit, (ii) *diff* takes incremental differences, (iii) *medianf* applies a median filter, (iv) *rav* applies a running mean filter, (v) *resampleup* progressively upsamples the time series, and (vi) *resampledown* progressively downsamples the time series. Outputs are simple fits to how various properties of the time series change as the transformation is iteratively applied to it.

- **MF_preproc**. Performs a range of transformations to the time series and then fits an autoregressive, AR model to all of them and returns the in-sample root-mean-square (RMS) prediction errors for the model applied to each transformed time series as a ratio of the RMS prediction error of the original time series. Transformations applied include (i) incremental differencing, (ii) filtering of the power spectral density function, (iii) removal of piece-wise polynomial trends, and (iv) rank mapping the values of the time series to a Gaussian distribution. We apply the code using AR(2) models as predictors.

- **ST_ftpoly**. Fits a polynomial of order k to the time series, and returns the mean square error of the fit. We implement this operation for $k = 1, 2, ..., 10$.

F.12 Model fitting and forecasting

F.12.1 Nonlinear models

- **MS_nlpe**. Computes the normalized 'drop-one-out' constant interpolation nonlinear prediction error for a time-delay embedded time series using Michael

Small's code **nlpe** [130], available at `http://small.eie.polyu.edu.hk/matlab/`. Inputs are the embedding dimension, m, and time delay, τ. Outputs include measures of the mean error of the nonlinear predictor, and a set of measures on the correlation, Gaussianity, etc. of the residuals. The code is implemented for $(m, \tau) = (2,$'mi'$)$, $(3,$'ac'$)$, and ('fnn','mi'), where 'mi' chooses τ as the first minimum of the automutual information function, 'ac' chooses τ as the first zero-crossing of the autocorrelation function, and 'fnn' chooses the embedding dimension, m, using the point where the proportion of false nearest neighbors falls below 5% via the code **MS_fnn**.

F.12.2 Simple predictors

When applying time-series models to time series it is important to compare the results of any model to that of very simple predictors, e.g., the previous value, the mean of the past 5 values, etc. In this section we describe operations that quantify the behavior of such simple predictors, that are later used to compare the results of more sophisticated time-series models.

- **FC_primitive**. Does local forecasting using very simple predictors using the past l values of the time series to predict its next value. Three prediction methods are implemented: (i) local mean prediction with $l = 1, 2, 3, 4$, and for l equal to the first zero-crossing of the autocorrelation function, (ii) local median prediction with $l = 3, 5, 7$, and (iii) local linear prediction for $l = 2, 3, 4, 5$, and l equal to the first zero-crossing of the autocorrelation function. The mean error, stationarity of residuals, Gaussianity of residuals, and their autocorrelation structure is outputted from this algorithm.

- **FC_dynprimitive**. Analyzes the outputs of **FC_primitive** for a range of local window lengths, l. We implement this for $l = 1, 2, ...,$ 10 using mean prediction, and return statistics including whether the mean square error increases or decreases, testing for peaks, variability, autocorrelation, stationarity, and a fit of exponential decay $f(x) = A\exp(Bx) + C$ to the variation.

- **MF_M_mtlbfit**. Fits simple time-series models using the function **fit** from

MATLAB's *Curve Fitting Toolbox*. We use models *sin1*, *sin2*, *sin3*, and *fourier1*. Outputs include the goodness of fit statistic, R^2, the root mean square error, and the autocorrelation of residuals from these fits. Note that this same function, **MF_M_mtlbfit** is used to fit distributions, as described in Sec. F.3.2 above.

F.12.3 Linear models

- **MF_steps_ahead**. Given a model, quantifies the variation in goodness of model predictions across a range of prediction lengths, l, which is made to vary from 1-step ahead to M steps-ahead predictions. Models are fit using code from MATLAB's *System Identification Toolbox*: we implement (i) AR models using the **ar** code, (ii) ARMA models using **armax** code, and (iii) state-space models using **n4sid** code. The model is fitted on the full time series and then used to predict the same data. Output errors, for prediction lengths $l = 1, 2, ..., M$, are returned for each model relative to the best performance from basic null predictors, including sliding 1- and 2-sample mean predictors and simply predicting each point as the mean of the full time series. Additional outputs quantify how the errors change as the prediction length increases from $l = 1, ..., M$ (relative to a simple predictor). We implement this code using $M = 6$ with (i) an AR(2) model, (ii) an AR(p) mode, where the order of the AR model, $1 \leq p \leq 10$, is chosen to minimize Schwarz's Bayesian Criterion (SBC) using the *ARfit* package, (iii) the optimal order of state space model (using the 'best' input), and (iv) an ARMA(3,1) model.

- **MF_ss_tt**. Looks at the robustness of fitted parameters from a model applied to different segments of the time series. Inputs control (i) how many segments to take from the time series, m, (ii) the length of each segment, n, (iii) how to choose segments from the time series howtosubset: either uniformly or at random, and (iv) the model to fit in each subsegment. We implement several simple time-series models: (I) *arsbc* fits an AR model using the *ARfit* package using an optimal model order from $1, ..., 10$. Statistics are on how the optimal order p_{opt} and the goodness of fit varies in different parts of the time series. (II)

ar fits an AR model of a specified order using the code **ar** from MATLAB's *System Identification Toolbox*. Outputs are on how Akaike's Final Prediction Error(FPE), and the fitted AR parameters vary across the different segments of time series. (III) *ss* fits a state space model of a given order using the code **n4sid** from MATLAB's System Identification Toolbox. Outputs are on how the FPE varies. (IV) *arma* fits an ARMA model using **armax** code from MATLAB's *System Identification Toolbox*. Outputs are statistics on the FPE, and fitted AR and MA parameters. We implement this code using 25 time-series segments uniformly-spaced throughout the time series, of length equal to 10% of the time series length. All four time-series models described above are used. In addition, we also implement the *arsbc* setting with time-series segments chosen at random (rather than uniformly-spaced) positions through the time series.

- **MF_ss_testset**. Similar to **MF_ss_tt**, except fits the model on the full time series and compares how well it predicts time series in different local time-series segments. We implement the code using (i) an AR(4) model, (ii) an AR(p) model of optimal order chosen using the *ARfit* package, (iii) for a state-space model of optimal order, and (iv) a state-space model of order 2.

- **MF_ss_compare_orders**. Fits state space models using **n4sid** (from MATLAB's *System Identification Toolbox*) of orders 1, 2, ..., $k_{\max}$ and returns statistics on how the goodness of fit changes across this range. We implement the code with $k_{\max} = 8$.

- **MF_sissm**. Fits a state space model of given order using **n4sid** code in MATLAB's *System Identification Toolbox*. Outputs are parameters from the model fitted to the entire time series, and goodness of fit outputs from **n4sid**. In the second portion of this code, the state space model is fitted to the first pN samples of the time series, where p is a given proportion and N is the length of the time series. This model is then used to predict the latter portion of the time series (i.e., the subsequent $(1-p)N$ samples). Outputs are statistics on the residuals obtained from this prediction. We implement this code using $p = 0.5$, for 1-step-ahead prediction and model orders: 1, 2, and 3.

- **SID_Exponential_Smoothing**. Fits an exponential smoothing model [93] to the time series using a training set to fit the optimal smoothing parameter, α, and then applies the result to the try to predict the rest of the time series. Code is loosely based on that provided to the authors by Siddarth Arora. Outputs include the fitted α, and statistics on the residuals from the prediction phase. We use the first half of the data as a training set to optimize α and then apply this model across the full time series.

F.12.3.1 Autoregressive (AR) Models

- **MF_M_ARfits**. Fits an AR model of a given order, p, using **arcov** code from MATLAB's *Signal Processing Toolbox*. Outputs include the parameters of the fitted model, the variance estimate of a white noise input to the AR model, the root-mean-square (RMS) error of a reconstructed time series, and the autocorrelation of residuals. We implement this code using five different model orders $p = 1, 2, 3, 4$, and 5.

- **MF_arfit**. Uses various functions implemented in the *ARfit* package. Autoregressive (AR) models are fitted with orders for $p = 1, 2, ..., 8$. The optimal model order is selected using Schwartz's Bayesian Criterion (SBC). Outputs include the model coefficients obtained, the SBCs at each model order, various tests on residuals, and statistics from an eigendecomposition of the time series using the estimated AR model.

- **MF_linmodelorders**. Compares fits of AR models of various orders to the time series. Uses the **arxstruc** code from MATLAB's *System Identification Toolbox*. Outputs of the code are statistics on the loss at each model order, which are obtained by applying the model trained on the training data to the testing data. We implement this algorithm for model orders $p = 1, 2, ..., 10$. In one implementation, we use a training set equal to the first half of the time series and a testing set equal to the second half, and for another we use all of the data to train and test on the same data.

F.12.3.2 Autoregressive Moving Average (ARMA) models

- **MF_armax**. Fits an ARMA(p, q) model to the time series and returns various statistics on the result. Inputs control the orders p and q for the AR and MA components of the model, respectively, the proportion of data to train the model on, p_{train} (the rest is used for testing), and the number of steps to predict into the future for testing the model, t_{steps}. Outputs include the fitted AR and MA coefficients, the goodness of fit in the training data, and statistics on the residuals from using the fitted model to predict the testing data. This code is implemented using the following parameter sets: $(p, q, p_{\text{train}}, t_{\text{steps}}) = (3, 1, 0.5, 1)$, $(1, 1, 0.5, 1)$, and $(2, 2, 0.5, 1)$.

- **MF_arma_orders**. Given a set of AR orders, **p**, and a set of MA orders, **q**, this operation fits ARMA(p, q) models to the time series and evaluates the goodness of fit from all combinations (p, q). Outputs are statistics on the appropriateness of different types of models, including the goodness of fit from the best model, and the optimal orders of fitted ARMA(p, q) models. We implement this for $\mathbf{p} = \{1, 2, ..., 6\}$ and $\mathbf{q} = \{1, 2, 3, 4\}$.

F.12.4 GARCH modeling

- **MF_GARCH**. Simulates a procedure for fitting Generalized Autoregressive conditional heteroskedasticity (GARCH) models to a time series, namely: (1) *preprocessing* the data to remove strong trends, (2) *pre-estimation* to calculate initial properties of the time series, (3) *fitting* a GARCH model, returning goodness of fit statistics and parameters of the fitted model, and (4) *post-estimation* involves calculating statistics on residuals and standardized residuals. All methods implemented are from MATLAB's *Econometrics Toolbox*, including Engle's ARCH test (**archtest**), the Ljung-Box Q-test (**lbqtest**), estimating the partial autocorrelation function (**parcorr**), as well as specifying (**garchset**) and fitting (**garchfit**) the GARCH model to the time series. As part of this code, we implement a very basic automatic pre-processing routine, **MF_preproc**, that applies a range of pre-processings and returns the preprocessing of the time

series that shows the worst fit to an AR(2) model. In the case that no simple transformations of the time series are significantly more stationary/less trivially correlated than the original time series (by more than 5%), the original time series is simply used without transformation. This automatic preprocessing is implemented for two of the versions of this code that we use: one where we subsequently fit the default GARCH(1,1) model, and another where we fit a model with $r = 2$, $m = 2$, $p = 1$, and $q = 1$, where r and m are the autoregressive and moving average orders of the model, respectively, and p and q control the conditional variance parameters. We also implement a version of this code that does not apply any pre-processing whatever to the time series, but attempts to automatically choose suitable parameters: r, m, p, and q for the GARCH model.

- **MF_compare_garch**. This code fits a set of GARCH(p, q) models to the time series and returns statistics on the goodness of fits across a range of p and q parameters. Outputs include log-likelihoods, Bayesian Information Criteria (BIC), Akaike's Information Criteria (AIC), outputs from Engle's ARCH test and the Ljung-Box Q-test, and estimates of optimal model orders. We implement the code by first applying an automatic pre-whitening routine, **MF_preproc**, and then comparing these statistics across all (nine) combinations of the following parameters: $p = \{1, 2, 3\}$ and $q = \{1, 2, 3\}$.

F.12.5 Hidden Markov Models (HMMs)

All operations described in this section use Zoubin Gharamani's implementation of Hidden Markov Models (HMMs) for real-valued Gaussian observations. See Sec. F.1 for more information.

- **ZG_hmm_fit**. Fits a Hidden Markov Model (HMM) to the time series using Zoubin Gharamani's implementation of HMMs for real-valued Gaussian observations using the code **hmm**. Inputs are the proportion of the time series to train the model on, and the number of states in the HMM. The **hmm** code is used with 30 cycles of expectation maximization, or until convergence is achieved. Outputs are the components of the mean vector, the covariance,

statistics on the transition matrix, and log-likelihood. The HMM trained on the training portion of the time series is then used to predict the test portion via the code **hmm_cl**, and we output the log likelihood and the mean error. We implement this code using $(p_{\text{train}}, n_{\text{states}}) = (0.7, 3)$ and $(0.8, 2)$.

- **ZG_hmm_comparenstates**. Fits HMMs with different numbers of states, and compares the resulting test-set likelihoods. The code relies on Zoubin Gharamani's implementation of HMMs for real-valued Gassian-distributed observations, including the **hmm** and **hmm_cl** routines. Inputs are the proportion of the time series used for training, p_{train}, and the vector setting the numbers of states $\mathbf{n}_{\text{states}}$. Outputs are statistics on how the log likelihood of the test data changes with the number of states n_{states}. We implement the code for $p_{\text{train}} = 0.6$ as n_{states} varies across the range $n_{\text{states}} = 2, 3, 4$.

F.12.6 Gaussian Process models

Algorithms described in this section use GPML MATLAB code for Gaussian Processes written by Carl Edward Rasmussen and Hannes Nickisch. See Sec. F.1 for more information.

- **GP_hps**. Fits a Gaussian Process model to a portion of the time series and returns the hyperparameters, and statistics on the goodness of fit of the model. We consider two different covariance functions: (i) a sum of squared exponential and noise terms, and (ii) a sum of squared exponential, periodic, and noise terms. The model is fitted to 200 samples from the time series, which are chosen by (i) resampling the time series down to this many data points, (ii) by taking the first 200 samples from the time series, or (iii) by taking random samples from the time series.

- **GP_predict**. Fits a given Gaussian Process model to a section of the time series and uses it to predict to the subsequent datapoint. We use a covariance function containing the sum of squared exponential and noise terms. Prediction is done in one of three modes: (i) *beforeafter* predicts the preceding time series values by training on the following values, (ii) *frombefore* predicts the following

values of the time series by training on preceding values, and (iii) *randomgap* tries to predict random values within a segment of time series by training on the other values in that segment. We implement this algorithm for $n_{\text{train}} = 5$ or 10, and $n_{\text{test}} = 3$, and the process is repeated $n_{\text{preds}} = 20$ times. Outputs are summaries of the quality of predictions made, the mean and spread of obtained hyperparameter values, and marginal likelihoods.

- **GP_givealittle**. Trains a Gaussian Process model on equally-spaced points throughout the time series and uses the model to predict its intermediate values. Outputs summarize the error and fitted hyperparameters. We implement this using a sum of squared exponential and noise terms in the covariance function, and use 20 evenly-spaced points throughout the time series to train the Gaussian Process.

F.13 Domain-specific operations

F.13.1 Heart rate variability

- **MD_hrv_classic**. Packages up a bunch of classic heart rate variability (HRV) statistics, including PNNx [174], power spectral density ratios in different frequency ranges [88], a triangular histogram index, and Poincaré plot measures [185]. Code is based on that provided by Max A. Little.

- **MD_pNN**. Applies pNNx measures [174] to time series assumed to represent sequences of consecutive RR intervals measured in milliseconds, with thresholds $x = 5$, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 ms.

- **MD_rawHRVmeas**. Evaluates the triangular histogram index and Poincaré plot measures [185] to a time series assumed to measure sequences of consecutive RR intervals measured in milliseconds.

F.14 Fanciful operations

- **CO_t_shape_translate**. Returns statistics on the number of data points that reside inside a given geometric shape that is moved around the time series.

Inputs specify a shape and its size, and a method for moving this shape through the time domain. We implement this using circles of radius, $r = 1.5, 2.5, ..., 5.5$, and move the shape by placing it on every time series datapoint in the z-scored series.

- **CO_stickangles**. Looks at line-of-sight angles between time-series points where each time-series value is treated as a stick protruding from an opaque baseline level. Statistics are returned on the raw time series, where sticks protrude from the zero-level, and the z-scored time series, where sticks protrude from the mean level of the time series. Output statistics are returned on the obtained sequence of angles, θ, reflecting the maximum deviation a stick can rotate before hitting a stick representing another time point. Statistics include the mean and spread of θ, the different between positive and negative angles, measures of symmetry of the angles, stationarity, autocorrelation, and measures of the distribution of these stick angles.

- **RN_gauss**. Returns a random number drawn from a normal distribution. Sometimes this operation is used as a control, as a kind of 'null operation' with which to compare to the performance of other operations that use properties of the time-series data, but is otherwise filtered out of all analyses used in this work.

F.14.1 Networks

- **NW_visgraphben**. Constructs a visibility graph of the time series. Due to computational constraints, we consider only horizontal visibility graph [28], rather than the conventional visibility graph [6]. Due to memory constraints, we only consider the first 6 000 points of time series. Longer time series are reduced to their first 6 000 samples. Outputs are statistics on the degree distribution, including the mode, mean, spread, histogram entropy, and fits to gaussian, exponential, and powerlaw distributions.

F.14.2 Dimensionality reduction

- **NL_embed_PCA**. Reconstructs the time series as a time-delay embedding, and performs Principal Components Analysis on the result using **princomp** code from MATLAB's *Bioinformatics Toolbox*. This technique is known as *singular spectrum analysis* [131]. Outputs are various statistics summarizing the obtained eigenvalue distribution. We implement this code for $(\tau, m) =$ ('mi', 10) and (1,10), where 'mi' chooses τ as the first minimum of the automutual information function. The suggestion to implement this idea was provided by Siddarth Arora.

F.14.3 Dynamical simulations

- **DV_dynsys_dblwell** and **DV_dynsys_sine**. Couples the values of the time series to a given dynamical system. The time series contributes to a forcing term on a simulated particle in a quartic double-well potential with potential energy $V(x) = x^4/4 - \alpha^2 x^2/2$, or a force $F(x) = -x^3 + \alpha^2 x$, or a sinusoidal potential: $V(x) = -\cos(x/\alpha)$, or $F(x) = \sin(x/\alpha)/\alpha$. The double-well potential has three parameters: α controls the positions of the wells, κ controls the coefficient of friction, and the step size of the simulation is Δt. The sinusoidal potential has parameter α that controls the period of oscillations in the potential, κ as the coefficient of friction, time-step Δt. Outputs of the function are statistics summarizing the trajectory of the simulated particle, including its mean, the range, proportion positive, proportion of times it crosses zero, its autocorrelation, final position, and standard deviation. We simulate particles in the double-well potential, **DV_dynsys_dblwell**, using $(\alpha, \kappa, \Delta t) = (1, 0.2, 0.1), (1, 0.5, 0.2), (2, 0.05, 0.2)$, and $(3, 0.01, 0.1)$. We also simulate particles in the sinusoidal potential, **DV_dynsys_sine**, using $(\alpha, \kappa, \Delta t) = (3, 0.5, 1), (1, 1, 1)$, and $(10, 0.04, 10)$.

- **TR_walker**. This operation simulates a particle (or 'walker'), that moves in the time domain in response to values of the time series at each point. Outputs from this operation are summaries of the walker's motion, and comparisons of

it to the original time series. Four different modes are implemented: (i) *prop*: the walker narrows the gap between its value and that of the time series by a given proportion w, (ii) *biasprop*: the walker is biased to move more in one direction; when it is being pushed up by the time series, it narrows the gap by a proportion w_{up}, and when it is being pushed down by the time series, it narrows the gap by a (potentially different) proportion w_{down}, (iii) *momentum*: the walker moves as if it has mass m and inertia from the previous time step and the time series acts as a force altering its motion in a classical Newtonian dynamics framework, (iv) *runningvar*: the walker moves with inertia as above, but its values are also adjusted so as to match the local variance of time series by a multiplicative factor. Outputs include the mean, spread, maximum, minimum, and autocorrelation of the walker's trajectory, the number of crossings between the walker and the original time series, the ratio or difference of some basic summary statistics between the original time series and the walker, an Ansari-Bradley test comparing the distributions of the walker and original time series, and various statistics summarizing properties of the residuals between the walker's trajectory and the original time series.

Appendix G

List of 200 Operations

In this appendix, we list the reduced set of 200 operations, formed as result of a k-medoids clustering of 8 651 operations on a diverse interdisciplinary set of 875 time series described in Sec. 4.1.1. This set is then used to organize time series in Sec. sec:4.2.

Operations are grouped by broad descriptors, and are each described very briefly. The code is provided for each operation, and corresponds to an operation listed in the full list of Appendix F, where more details can be found about each operation. In cases where the output of the listed operation is a structure, the "→" sign points to the output of that structure used by the target operation.

Location

Trimmed mean: **ST_trmean(x,2)**

Spread

Spread of Gaussian fit: **DF_mlefits(x,1,2)**

Distribution

Gaussian fit to distribution: **MF_M_mtlbfit(y,'gauss1',10)** → **resruns**

Gaussian fit to distribution: **MF_M_mtlbfit(y,'gauss1',30)** → **resAC1**

Exponential fit to distribution: **MF_M_mtlbfit(y,'exp1',0)** → **adjr2**

Outliers: **DN_olmi(y,'abs')** → **mrm**

Outliers: **DN_olmi(y,'abs')** → **mrstd**

Outliers: **DN_olmi(y,'abs')** → **stdrfla**

Outliers: **DN_olmi(y,'abs') → stdrfladjr2**

Autocorrelation change on removing outliers: **CO_acfremove(y,'absfar',0.8) → ac3diff**

Autocorrelation change on removing outliers: **CO_acfremove(y,'min',0.1) → mean**

Spacing of extreme values through time: **EX_altmannben(y,0.1,0.1) → stdq**

Spacing of extreme values through time: **EX_altmannben(y,0.1,0.02) → minq**

Spacing of extreme values through time: **EX_altmannben(y,0.1,0.02) → meanqover**

Pre-processing statistics

Change on sinusoidal detrending: **PP_compareba(x,'sin2') → swms5_2**

Change on spline detrending: **PP_compareba(x,'spline24') → gauss1_h10_adjr2**

Change on spline detrending: **PP_compareba(x,'spline24') → kscn_adiff**

Change on spline detrending: **PP_compareba(x,'spline44') → swms2_2**

Change on spline detrending: **PP_compareba(x,'spline44') → swss2_2**

Change on spline detrending: **PP_compareba(x,'spline44') → olbt_s5**

Change on incremental differencing: **PP_compareba(x,'diff1') → swms5_2**

Change on two incremental differencings: **PP_compareba(x,'diff2') → swss5_2**

Change on two incremental differencings: **PP_compareba(x,'diff2') → gauss1_kd_adjr2**

Change on two incremental differencings: **PP_compareba(x,'diff2') → gauss1_kd_resruns**

Change on median filtering: **PP_compareba(x,'medianf2') → swss5_2**

Change on median filtering: **PP_compareba(x,'medianf2') → olbt_m5**

Change on median filtering: **PP_compareba(x,'medianf4') → swms5_2**

Change on median filtering: **PP_compareba(x,'medianf4') → swss10_2**

Change on running mean filtering: **PP_compareba(x,'rav2') → statav2**

Change on running mean filtering: **PP_compareba(x,'rav3') → swms2_2**

Change on running mean filtering: **PP_compareba(x,'rav3') → kscn_relent**

Change on running mean filtering: **PP_compareba(x,'rav3') → olbt_s2**

Change on running mean filtering: **PP_compareba(x,'rav4') → kscn_peaksepy**

Change on a set of successive spline detrendings:

PP_progpp(x,'spline') → normdiff_trend

Autocorrelation

Linear autocorrelation at lag 1: **CO_autocorr(y,1)**

Shape of autocorrelation function: **CO_acfshape(y) → nmaxima**

Generalized linear self-correlation function: **CO_glscf(y,1,1,1)**

Generalized linear self-correlation function: **CO_glscf(y,1,1,'tau')**

Generalized linear self-correlation function: **CO_glscf(y,1,2,4)**

Numerator of the tc3 statistic: **CO_tc3(y,'mi') → num**

Automutual Information

Histogram automutual information: **CO_information(y(1:end-6),y(7:end))**

Histogram automutual information: **TSTL_amutual(y,20,10) → mami**

Histogram automutual information: **TSTL_amutual2(y,50) → mami**

Histogram automutual information: **TSTL_amutual2(y,50) → modeperiodmax**

Histogram automutual information: **TSTL_amutual2(diff(y),50) → mami**

Histogram automutual information: **CO_ami_benhist(y,4,'std1',6)**

Histogram automutual information: **CO_ami_benhist(y,3,'even',20)**

Histogram automutual information: **CO_ami_benhist(y,2,'quantiles',10)**

First minimum of automutual information function: **CO_ami_fmin(y,'std2',[2:80])**
→ nlocmax

Automutual information at limiting noise addition: **CO_ami_noise(y,1,'quantiles',20)**
→ fitlina

Correlation

Simple two-dimensional embedding properties: **CO_basicrecurf(y,1) → downdiag05**

Simple two-dimensional embedding properties: **CO_basicrecurf(y,1) → parab-down05_n1**

Simple two-dimensional embedding properties: **CO_basicrecurf(y,'tau') → ring1_02**

ACP function: **TSTL_acp(y,'mi',1,[],10,[])** → **ac1_acpf_2**

ACP function: **TSTL_acp(y,'mi',1,[],10,[])** → **sacpf_3**

ACP function: **TSTL_acp(y,'mi',1,[],10,[])** → **ac1_acpf_8**

ACP function: **TSTL_acp(y,1,0.01,0.25,10,0.5)** → **macpf_5**

ACP function: **TSTL_acp(y,1,0.01,0.25,10,0.5)** → **ac1_acpf_8**

Symbolization

Local motifs: **ST_benmotif_bin(y,'mean')** → **ddu**

Local motifs: **ST_benmotif_bin(y,'median')** → **duuu**

Local motifs: **ST_benmotif_tri(y,'quantile')** → **bac**

Local motifs: **ST_benmotif_tri(y,'quantile')** → **bcba**

Local motifs: **ST_benmotif_tri(y,'quantile')** → **cbaa**

Transition matrix: **ST_transition(y,'quantile',3,1)** → **sumdiagcov**

Transition matrix: **ST_transition(y,'quantile',5,1)** → **mineig**

Transition matrix: **ST_transition(y,'quantile',5,'ac')** → **sumdiagcov**

Transition matrix: **ST_dyntransition(y,2:40,1)** → **maxeig_fexpb**

Polvar: **MD_polvar(y,0.5,5)**

Entropy

Histogram entropy: **EN_histent(y,'hist',10,0.05)**

Raw histogram entropy: **EN_histent(x,'ks',[],0.05)**

Lempel-Ziv complexity: **MS_complexity(y,9,[])**

Lempel-Ziv complexity: **MS_complexity(y,9,'diff')**

Shannon entropy: **MS_shannon(y,3,1:10)** → **meanent**

Sample Entropy : **EN_sampenc2(y,4,0.15)** → **meanchsampen**

Control Entropy: **EN_sampenc2(y,4,0.2,'diff1')** → **sampen2**

Symbolic entropy of increments: **IN_surprise(y,'dist',100,4,'updown')** → **uq**

Symbolic entropy of increments: **IN_surprise(y,'dist',100,4,'updown')** → **std**

Symbolic entropy of increments: **IN_surprise(y,'T1',20,2,'updown')** → **median**

Symbolic entropy of increments: **IN_surprise(y,'T1',100,5,'updown')** → **tstat**

Symbolic entropy: **IN_surprise(y,'dist',10,'tau','embed2quadrants')** → **mean**

Standard deviation of eighth derivative: **SY_stdnthder(y,8)**

Progressively randomize the time series: **EN_progranz(y,'statdist')** → **statav5fexpb**

Progressively randomize the time series: **EN_progranz(y,'dyndist')** → **ac1fexpadjr2**

Progressively randomize the time series: **EN_progranz(y,'dyndist')** → **statav5fexpc**

Progressively randomize the time series: **EN_progranz(y,'permute')** → **ac2fexpadjr2**

Scaling

Value scaling: **SC_benfa(y,2,'nothing',25,[],[],1)** → **ssr**

Range scaling: **SC_benfa(y,2,'range',1)** → **r2_se1**

Range scaling: **SC_benfa(y,2,'range',25,[],[],1)** → **r2_linfitint**

Rescaled range analysis: **SC_benfa(y,2,'rsrange',1)** → **linfitint**

Rescaled range analysis: **SC_benfa(y,2,'rsrange',1)** → **r1_linfitint**

Rescaled range analysis: **SC_benfa(y,2,'rsrange',1)** → **r1_stats_coeffcorr**

Rescaled range analysis: **SC_benfa(y,2,'rsrangefit',2,1)** → **r1_se2**

Detrended fluctuation analysis: **SC_benfa(y,2,'dfa',2,1)** → **se2**

Detrended fluctuation analysis: **SC_benfa(y,2,'dfa',2,1)** → **ssr**

Detrended fluctuation analysis: **SC_benfa(y,2,'dfa',25,2,[],1)** → **se1**

Detrended fluctuation analysis: **SC_benfa(y,2,'dfa',25,2,[],1)** → **resac1**

Detrended fluctuation analysis: **SC_benfa(y,2,'dfa',25,2,[],1)** → **r2_se2**

Detrended fluctuation analysis: **SC_benfa(y,2,'dfa',25,1,3,1)** → **stats_coeffcorr**

Detrended fluctuation analysis:

SC_benfa(zscore(abs(y)),2,'dfa',25,2,[],1) → **linfitint**

Stationarity

StatAv of 50-sample segments: **SY_StatAv(y,50,'len')**

Sliding windows: **SY_slidwin(y,'mean','std',10,2)**

Sliding windows: **SY_slidwin(y,'std','ent',5,10)**

Sliding windows: **SY_slidwin(y,'ent','std',5,2)**

Sliding windows: **SY_slidwin(y,'apen','apen',5,1)**

Sliding windows: **SY_slidwin(y,'lillie','apen',10,10)**

Sliding windows: **SY_slidwin(y,'AC1','std',10,10)**

Spread of local standard deviations in random subsegments: **SY_dynpick(y,200)** → **stdstd**

Compare local to global statistics– **ST_cumul(y,'mean','randcg',500)**

Compare local to global statistics: **ST_cumul(y,'median','p',0.01)**

Compare local to global statistics: **ST_cumul(y,'iqr','l',50)**

Compare local to global statistics: **ST_cumul(y,'kurtosis','p',0.1)**

Compare local to global statistics: **ST_cumul(y,'AC1','l',50)**

Cross-forecast errors: **TISEAN_nstat_z(y,4,1,3)** → **stdmedian**

Cross-forecast errors: **TISEAN_nstat_z(y,5,1,3)** → **range**

Cross-forecast errors: **TISEAN_nstat_z(y,4,'ac',3)** → **median**

KPSS test: **EC_kpsstest(y,0:10)** → **minstat**

Models and forecasting

Local mean forecasting: **FC_primitive(y,'mean',2)** → **gofnadjr2**

Local mean forecasting: **FC_primitive(y,'mean',3)** → **stderr**

Local median forecasting: **FC_primitive(y,'median',7)** → **rmserr**

Gaussian Process hyperparameters:

GP_hps(y,'covSum','covSEiso','covPeriodic','covNoise',1,200,'resample') → **h5**

Gaussian Process forecasting:

GP_predict(y,'covSum','covSEiso','covNoise',10,3,20,'randomgap') → **meanerrbar**

Gaussian Process regression: **GP_givealittle(y,'covSum','covSEiso','covNoise',20)** → **logh3**

Sinusoidal time-series model fit: **MF_M_mtlbfit(y,'sin3')** → **rmse**

AR model goodness of fit: **MF_steps_ahead(y,'ar','best',6)** → **rmserr_4**

AR model fit: **MF_ss_tt(y,'arsbc',[],'uniform',[25,0.1]) → orders_mean**

AR model forecasting: **MF_ss_testset(y,'ar','best','uniform',[25,0.1],1) → rmserr_std**

AR model forecasting: **MF_ss_testset(y,'ar','best','uniform',[25,0.1],1) → stdrats_mean**

AR model order comparison: **MF_linmodelorders(y,1:10,'all') → meanv**

AR model order comparison: **MF_linmodelorders(y,1:10,'all') → meandiff**

AR model order comparison: **MF_linmodelorders(y,1:10,'all') → bestmdl**

ARMA model forecasting: **MF_steps_ahead(y,'arma',[3,1],6) → maxdiffrms**

ARMA model fit: **MF_armax(y,[2,2],0.5,1) → acsnd0**

ARMA model order comparison: **MF_arma_orders(y,1:6,1:4) → mean_all_aics**

Exponential smoothing: **SID_Exponential_Smoothing(y,0.5,'best') → p2_5**

Exponential smoothing: **SID_Exponential_Smoothing(y,0.5,'best') → minfpe**

GARCH model fit: **MF_GARCH(y,'nothing','auto') → maxlbqpval_stde2**

GARCH model comparison: **MF_compare_garch(y,'ar',1:3,1:3) → mean_meanarchps**

Measures derived from the Fourier power spectrum

Periodogram (Hamming window):

SP_basicmeasures(y,'periodogram','hamming',[],0,0) → fpoly2csS_p3

Periodogram (Hamming window):

SP_basicmeasures(y,'periodogram','hamming',[],0,0) → logarea_3_2

Periodogram (Hamming window):

SP_basicmeasures(y,'periodogram','hamming',[],0,0) → area_4_2

Periodogram (Hamming window):

SP_basicmeasures(y,'periodogram','hamming',[],0,0) → area_4_3

Periodogram (Hamming window):

SP_basicmeasures(y,'periodogram','hamming',[],0,0) → logarea_5_5

Periodogram (Hamming window):

SP_basicmeasures(y,'periodogram','hamming',[],0,1) → std

Periodogram (Hamming window):

SP_basicmeasures(y,'periodogram','hamming',[],0,1) → q10mel

Periodogram (Hamming window):

SP_basicmeasures(y,'periodogram','hamming',[],0,1) → fpoly2_rmse

Periodogram of absolute deviations from mean (Hamming window):

SP_basicmeasures(y,'periodogram','hamming',[],1,0) → q75log

Periodogram of absolute deviations from mean (Hamming window):

SP_basicmeasures(y,'periodogram','hamming',[],1,0) → area_3_1

Periodogram (Hanning window): very low frequency power:

MD_hrv_classic(y) → vlf

Wavelet analysis

Continuous wavelet transform: **WL_cwt(y,'db3',32) → meanabsC**

Continuous wavelet transform: **WL_cwt(y,'sym2',32) → maxonmedC**

Wavelet decomposition: **WL_powerlevels(y,'db3','max') → wslesr_mean**

Wavelet coefficients: **WL_coeffs(y,'db3',2) → max_coeff**

Wavelet coefficients: **WL_coeffs(y,'db3',5) → wb25m**

Time-series embedding

False nearest neighbors: **NL_fnnmar(y,10,2,'mi') → m02**

False nearest neighbors: **MS_fnn(y,1:10,'mi',5,1) → pfnn_6**

Cao's embedding dimension estimation: **TSTL_cao(y,10,'mi',2,100) → caoo1_5**

Cao's embedding dimension estimation: **TSTL_cao(y,10,'mi',2,0.2) → min2**

Cao's embedding dimension estimation: **TSTL_cao(y,10,'mi',2,0.2) → median2**

Dimension estimates

Dimension estimation: **TISEAN_d2(y,'ac',10,0.01) → benmmind2_logminl**

Dimension estimation: **TISEAN_d2(y,'ac',10,0.01) → benmmind2g_logminl**

Dimension estimation: **TISEAN_d2(y,'ac',10,0.01) → slopesh2_goodness**

Dimension estimation: **TISEAN_c1(y,1,[1,7],0.02,0.5) → mediand**

Dimension estimation: **TSTL_corrdim(y,50,'ac',5) → meanr15**

Dimension estimation: **TSTL_corrsum(y,-1,0.5,100,20,'ac','cao',2) → meanlnCr**

Dimension estimation: **TSTL_dimensions(y,50,'ac','cao') → in_minm1**

Dimension estimation: **TSTL_dimensions(y,50,'ac','cao') → co_lfitbmax**

Dimension estimation: **TSTL_dimensions(y,50,'ac','cao') → scr_in_mopt_pgone**

Dimension estimation: **TSTL_dimensions(y,50,'ac','cao') → scr_co_m3_pgone**

Dimension estimation: **TSTL_dimensions(y,50,'ac','cao') → in_maxscalingexp**

Dimension estimation: **TSTL_fracdims(y,5,100,0.2,1,10,0,32,'ac',3) → meanDq**

Dimension estimation: **TSTL_fracdims(y,5,100,0.2,1,10,0,32,'ac',3) → expfit_adjr2**

Surrogate Data Analysis

100 surrogates: tc3 statistic:

TSTL_surr_nlac(y,1,100,1,'tc3') → normpatponmax

100 surrogates: trev statistic:

TSTL_surr_nlac(y,1,100,2,'trev') → normpatponmax

100 surrogates: trev statistic:

TSTL_surr_nlac(y,'mi',100,2,'trev') → normpatponmax

Other nonlinear time-series analysis methods

Poincar section: **TSTL_poincare(y,'max','mi',3) → ac1y**

Poincar section: **TSTL_poincare(y,'max','mi',3) → meands**

Poincar section: **TSTL_poincare(y,'max','ac',3) → meands**

Largest Lyapunov exponent estimate:

TSTL_largelyap(y,0.5,0.1,0.01,3,'mi','cao') → vse_meanabsres

Largest Lyapunov exponent estimate:

TSTL_largelyap(y,-1,0.1,0.01,3,1,4) → to095max

Miscellaneous measures

Treat time series as drive to particle in double well potential:

DV_dynsys_dblwell(y,[1,0.5,0.2]) → mean

Treat time series as drive to particle in sinusoidal potential:

DV_dynsys_sine(y,[10,0.04,10]) → mean

Treat time series as drive to particle in sinusoidal potential:

DV_dynsys_sine(y,[10,0.04,10]) → std

Analyze distribution of stick angles: **CO_stickangles(y) → mean**

Analyze distribution of stick angles: **CO_stickangles(y) → statav4_p_s**

Analyze distribution of stick angles: **CO_stickangles(x) → statav5_p_m**

Analyze distribution of stick angles: **CO_stickangles(x) → kurtosis_all**

Dynamics of local extrema: **ST_benlocext(y,'l',50) → maxabsext**

Dynamics of local extrema: **ST_benlocext(y,'l',100) → meanext**

Dynamics of local extrema: **ST_benlocext(y,'n',50) → meanmax**

Entropy of visibility graph degree distribution:

NW_visgraphben(y,'horiz') → maxent

Translate a circle through time domain:

CO_t_shape_translate(y,'circle',3.5,'pts') → statav4_s

Simulate a propagating particle trajectory: **TR_walker(y,'prop',0.1) → res_swss5_1**

Simulate a propagating particle trajectory: **TR_walker(y,'prop',0.5) → w_std**

Simulate a propagating particle trajectory: **TR_walker(y,'prop',0.5) → sw_meanabsdiff**

Simulate a propagating particle trajectory: **TR_walker(y,'prop',0.5) → sw_maxrat**

Simulate a propagating particle trajectory: **TR_walker(y,'prop',0.9) → w_mean**

Simulate a propagating particle trajectory: **TR_walker(y,'prop',0.9) → w_median**

Simulate a propagating particle trajectory: **TR_walker(y,'prop',0.9) → w_min**

Change point detection: **WL_varchg(y,'db3',2,10,0.01)**

Phillips-Peron unit root test: **EC_pptest(y,0:5,'ts','t1') → minBIC**