Novel machine learning for applications in cancer genomics

Ellen Visscher

St Catherine's College

University of Oxford

A thesis submitted for the degree of

DPhil in Health Data Science

October, 2025

Supervised by Christopher Yau

Abstract

Genomics has advanced rapidly in the past decade, with whole-genome sequencing and single-cell RNA sequencing now routine in cancer research. Machine and deep learning have also taken off, but applying them to biology remains challenging due to confounding factors, irregularly distributed data, and a desire for causal insight rather than prediction. In genomics, the lack of ground-truth labels further limits supervised learning. This work develops methods bridging both domains.

The first part of my work revisits copy number alteration calling in cancer, introducing `araCNA`, a deep learning model trained via simulation rather than emulating the outputs of other models. Using novel long-range sequence models like Mamba, `araCNA` predicts copy number profiles on whole-genome sequenced cancer samples. `araCNA` presents a different paradigm for which deep learning models can be applied in genomics - for amortised inference rather than as emulators. The second part of my work focuses on unsupervised discovery in single-cell RNA sequencing (scRNA-seq). I investigate the standard scRNA-seq pipeline assumptions and show how most approaches overlook the sparse, near-binary nature of scRNA-seq data. To address this, I develop `bfact`, a Boolean matrix factorisation method combining combinatorial optimisation with heuristic post-processing. `bfact` outperforms existing BMF methods and, when applied to scRNA-seq, finds biologically relevant gene programs beyond current approaches.

Acknowledgements

Professional

Thank you to my supervisor, Prof. Chris Yau. Chris has a great balance of supporting his students but also giving them the freedom to explore and learn.

Thank you to my now collaborator and ex-lecturer, Dr. Michael Forbes, for his help in developing a Boolean matrix factorisation method, introduced in this thesis.

Thank you to Prof. Jim Davies, our CDT director, who made things happen when they needed to in that first CDT year.

Finally, thank you to the funders of my CDT program, ESPRC, Ref: EP/S02428X/1.

Personal

Thank you.

To mum, for being my biggest supporter. To dad, for telling me not to be stressed when I was stressed. It didn't work, but it made me laugh. You are role models that I am lucky to have.

To the St Catz. gals, for making my first year in Oxford into the dream that it was. I know I have lifelong friends in you.

To Sarah J. Maas, they say all reading helps with writing.

To OUARFC, you taught me what community truly means.

To Lucy and Ellie, there are few things more special than getting to come home to laughter and joy. But also a shoulder to cry on occasionally.

To Isob, Emski, Haze and Anna, c'est la vie - thanks for getting me through it, especially when there was an adjective involved!

To Edgers et. al. for all the fun we have, still aspiring to be as edgy as you all.

To Guy, for the support, rants, cryptics, laughs and adventures.

Contents

List of Figures

Acronyms

AMI	Adjusted mutual information
ATAC-seq	Assay for transposase-accessible chromatin using sequencing
BAF	B-allele frequency
BMF	Boolean matrix factorization
BRCA	Breast cancer
CN	Copy number
CNA	Copy number alteration
cNMF	Consensus non-negative matrix factorization
CNN	Convolutional neural network
CRC	Colorectal cancer
DEG	Differentially expressed gene
GEP	Gene expression programs
HLCA	Human lung cell atlas
HVG	Highly variable gene
IBD	Identity by descent
IP	Integer programming
LLM	Large language model
LP	Linear programming
MAE	Mean absolute error
MAP	Maximum a posteriori
MDL	Minimum description length
MILP	Mixed integer linear programming
MIP	Mixed integer programming

ML	Machine learning
MLP	Multilayer perceptron
MP	Master problem
NLP	Natural language processing
NMF	Non-negative matrix factorization
OV	Ovarian cancer
PALM	Proximal alternating linearized minimization
PBMC	Peripheral blood mononuclear cell
PC	Principal component
PCA	Principal component analysis
PCR	Polymerase chain reaction
PP	Pricing problem
RMP	Restricted master problem
RMSE	Root mean square error
RNN	Recurrent neural network
SBI	Simulation-based inference
scRNA-seq	Single-cell RNA sequencing
SNP	Single nucleotide polymorphism
SNV	Single nucleotide variation
SSM	State space model
TCGA	The Cancer Genome Atlas
UMAP	Uniform manifold approximation and projection
UMI	Unique molecular identifier
WES	Whole exome sequencing
WGS	Whole genome sequencing

1 Introduction

Contents

1.1 Overview

Perhaps like many modern PhDs in method development, mine has spanned several projects, unified by a few major themes rather than one single project. My PhD lies at the intersection of genomic data, its applications to cancer, and modern machine learning and deep learning techniques. The overarching goals were to improve existing methodologies, bring new perspectives and ideas, and learn as much as possible about machine and deep learning when applied to real and messy biological data.

My thesis covers two distinct areas of genomic methods development: first, using deep learning approaches to call copy number alterations (CNAs) from whole genome sequencing (WGS), and second, developing methodologies for discovery from single-cell RNA sequencing analysis (scRNA-seq). Even within the latter,

there are two central pillars: first, the methodology itself, a new Boolean matrix factorisation approach, and second, the application to single-cell analysis.

A unifying theme of my work is how we can develop and evaluate methods when there is no known ground truth, as is common in genomic data. Many deep learning applications skip over this fact and end up training kinds of emulators- models that predict the output of another existing model. Even unsupervised methods are often benchmarked by treating labels as ground truth, meaning these methods, too, are selecting for inductive biases that align with the model that generated said labels. Both of the main projects presented here deal with this in different ways.

The copy number calling project uses what we do know about the data-generating process to simulate ground truth data, on which a deep-learning sequence model is trained. Once trained, it is applied to real WGS datasets to call copy numbers. Hence, instead of treating existing copy number profiles as ground truth to train and bias a model towards existing algorithms, the known generating process of the data is instead used for training. Although not Bayesian, this approach is in a similar vein to so-called 'simulation-based inference', where a deep learning model is used to predict the posterior distribution from training on simulated samples of the joint distribution. It demonstrates a different way in which deep learning models can be applied to biological data without redundantly treating the output of some other model as the ground truth.

The Boolean matrix factorisation work takes a more traditional data analysis approach, interrogating the way that current analysis is done, showing that there are flaws in this approach, and investigating a binary factorisation approach for the discovery of different signals underpinning scRNA-seq. Methods developed in this context often use some other computational output as a ground truth, leading to emulators without much new insight into the task at hand. My work demonstrates how the data distribution leads to misleading summary statistics and metrics, and that new methodologies would benefit from challenging existing assumptions and further consideration of the data at hand.

1.2 Thesis summary

Because of the modular nature of my PhD, I have opted to include relevant background literature within each of the respective chapters, which I believe makes it easier to digest. My thesis consists of four main content chapters, with the first one being standalone, and the last three more tightly linked.

Chapter 2 details the CNA calling project and introduces the method, `araCNA`, that can accurately predict CNAs in real WGS cancer genomes. It introduces the concept of CNAs and highlights existing work in the field, particularly showing that no current deep-learning approaches exist for allelic CNA calling. Using recent advances in deep sequence models, it uses state-space-model (SSM) based methods as the underlying architecture, which allows for genomic-scale calling in a single inference pass. Given the lack of ground truth, benchmarking is difficult; however, `araCNA` performs on par with existing approaches, with much faster inference, and requires only a tumour sample. Further, the approach outlines how a simulation-based approach may be a different paradigm under which deep learning can be successfully applied to biological data, where ground truth is often unknown.

Chapter 3 details the concept of Boolean matrix factorisation (BMF) in a methodological sense, introducing both combinatorial optimisation theory and existing methodological approaches. I present a new approach, `bfact`, based on mixed integer programs (MIPs), drawing on my historical experience with combinatorial optimisation. `bfact` is shown to work well compared to other BMF approaches in simulated settings and on scRNA-seq datasets, for which it is intended. Although the following chapter inspired my personal motivation for developing this method, because `bfact` is methodologically standalone, I introduce it here first.

Chapter 4 critically evaluates the existing scRNA-seq standard pipeline for identifying cell types and cellular abundance. This is particularly relevant for applications in cancer, where rare cell types may exist or where cellular abundance between disease states is used to understand possible disease mechanisms. I show how the existing pipeline makes many assumptions and uses many sequential

steps that all introduce sources of error. This chapter demonstrates how the data distribution of scRNA-seq is extremely sparse, almost binary, which is often overlooked and causes misleading summary statistics and downstream results. This forms the motivation for using a methodological approach that explicitly accounts for the binary nature of the data, BMF.

Chapter 5 evaluates and motivates the use of BMF specifically for scRNA-seq, in a bioinformatics sense. I explain both from a data-driven and methodological perspective why a factorisation approach may be preferable to the standard clustering approach, as well as the binary focus, which may be an advantage. As well as comparing to existing clustering, I also compare to a non-negative matrix factorisation (NMF) approach. I evaluate `bfact` in several ways, both using summary statistics and also looking at conservation of informative genes and enrichment in reference biological databases. `bfact` appears to select different, yet biologically relevant marker gene sets, and should be further investigated for its potential in cancer applications. I also show that NMF appears to be more robust than standard clustering approaches.

Finally, in **Chapter 6**, I conclude my thesis, reiterating some of the main messages across the chapters.

2

Calling copy numbers with araCNA

Contents

2.1 Introduction

In this chapter, I propose `araCNA`, a novel deep learning-based approach to the call somatic copy number alterations (CNAs), prevalent in cancer genomes. Current algorithms that call CNAs from whole-genome sequenced (WGS) data have not exploited deep learning methods owing to computational scaling limitations. `araCNA` uses novel transformer alternatives (e.g Mamba) to handle genomic-scale sequence lengths and learn long-range interactions, and is trained only on simulated data that can accurately predict CNAs in real WGS cancer genomes. Results are highly accurate on simulated data, and this zero-shot approach is comparable to existing methods when applied to 50 WGS samples from the Cancer Genome Atlas. Our approach performs well with only a tumour sample and not a matched normal sample, while most methods require a matched normal [1–3]. Newer versions of both ASCAT and CNV-Kit do have a tumour-only mode, [4, 5], although specify preference for both samples. Further, `araCNA` has fewer markers of overfitting, and performs inference in only a few minutes. The approach demonstrates that simulation-based inference provides a novel way to use modern deep learning techniques when the ground truth of biological samples is unknown. This approach opens up new opportunities for creating CNA callers that learn and can be fine-tuned for specific sub-tasks of interest, or integrated directly with deep learning models for other data modalities.

This chapter is standalone to the other three chapters; however, it links to the main thesis themes of method development in cancer genomics, for which there is often no known ground truth. This chapter also forms the basis of a publication that is now in Nucleic Acids Research, [6].

2.2 Background

Somatic copy number alterations (CNAs) are genomic regions that are amplified or deleted when somatic cells replicate. They are a hallmark of many cancers and a driving factor of tumorigenesis [7], leading to the amplification of oncogenes [8, 9].

CNA abundance is known to be associated with disease stage, prognosis and response to treatment, and particular cancer types can be characterised by specific classes of structural variation, often in specific chromosomal regions [10]. Accurate CNA profiles of cancer samples are important for downstream analysis, such as association studies to identify underlying cancer signatures or as prognostic biomarkers [11].

The copy number landscape of a tumour can be profiled using array and sequencing-based technologies, where increases and decreases in signal intensity or sequence read depth correspond to gains and losses of genomic segments [12, 13]. Allelic information from single-nucleotide polymorphisms (SNPs) can also be exploited to determine copy number, **Figure 2.1**, **Figure 2.3**. Data can then be processed using CNA calling algorithms to derive copy number states.

CNA calling algorithms generally consist of an approach to segment the genome into regions of constant copy number and a copy number calling or classification step (into deletion, duplication, etc). Methods adopt a variety of approaches. This includes calling total copy numbers only [2, 14–16] while others provide allele-specific major and minor copy number calling capabilities [1, 4]. Some adopt a hybrid approach, first calling total copy numbers and then assigning these to major and minor copy numbers [5, 17]. Approaches that account for tumour heterogeneity can also provide sub-clonal copy numbers [18–21], again sometimes calling allele-specific copy numbers [3, 22–24]. When multiple tumour samples are available from the same individual, information from across samples can be used to posit evolutionary trees upon which somatic point mutations and CNAs can be placed [25, 26].

Standard CNA callers adopt the use of segmentation algorithms or hidden Markov models, which are effective at processing one cancer sequencing sample at a time [27]. CNA callers process every sample as though it were the very first sample they have seen, rather than learning from data examples. Therefore, despite large-scale mapping exercises such as The Pan-Cancer Genome Atlas studies [11], few new CNA calling approaches have been developed.

While many areas of 'omics analysis have been heavily influenced by deep learning in recent years, the technology has had relatively little impact on CNA

calling algorithms. For example, the use of transformer-based models has led to the creation of massive foundation models for single-cell and integrative 'omics [28]. However, the quadratic scaling limits of transformers have meant that these same principles are often not applicable to genome-based applications, like CNA calling, where distances between genomic regions of interest might be vast [29]. One of the few examples is ECOLE, [30], which is designed for whole exome sequencing (WES), which uses a transformer architecture to classify exome regions into neutral, deletion, and duplication, etc. Since WES has lower data resolution as it captures only the coding portion of the genome, ECOLE tackles the simpler task of classifying exome regions into these broad categories and does not call the exact copy number or allele-specific copy numbers.

Deep learning models have been developed in the context of calling germline variants, such as single-nucleotide variants (SNVs) and indels, from normal samples [31], as well as some to call SNVs in tumour samples [32, 33]. However, the detection of CNAs from tumour samples is more complex. Unlike germline contexts with a well-defined diploid background, tumour samples have variable purity, with genomes containing many different aneuploid states, obfuscating the true copy numbers. Germline and somatic SNV calling typically focus on local sequence context, while accurate CNA detection requires both global reasoning to infer sample purity and ploidy, and local reasoning to infer changes in copy number states.

2.2.1 SSM-like models

In this chapter, two recent state space model (SSM) deep learning blocks are employed; Hyena and Mamba, [34, 35]. SSM-like models have been developed in an attempt to achieve similar results to transformers but to avoid using the computationally taxing self-attention, where the number of operations scales with the square of the sequence length, L^2. Instead, SSM-like models use discrete convolution, an operation between two sequences, x, h, to produce a third sequence, y, given by:

$$y_t = (h * x)_t = \sum_{k=0}^{M-1} x_{t-k} h_k$$

Where $M \leq L$ is the filter/kernel length (of h), and $t \in 1 \dots L$ is an element in the sequence. For SSM models, take $M = L$.

Although convolution can be applied to any sequence, in the context of SSM-type models, the sequences are assumed to be samples from a continuous function. Here, observations $x_1 \dots x_L$ and outputs $y_1 \dots y_L$ are sampled from a continuous time series, at Δ_t intervals. Under a convolutional model/block, it is assumed that observed y_t are outputs from the convolution of the observed x_t and the filter h_k, to be learnt. Hence, the aim is to learn filters h_k such that the predicted $\hat{y}_t$ approximates the observed y_t. Under a constant Δ_t, convolutions can be efficiently computed using the discrete fast Fourier transform [36].

The choice of how to define h_n separates different classes of models [34]. For example, convolutional neural networks (CNNs) explicitly define the structure of predefined filters, h_k, of size M. The number of parameters scales with filter size, M, and so M is often chosen to be very short (e.g 3x3 kernels in imaging). Instead, these filters may also be defined implicitly as $h_k = \gamma_\theta(k)$, which removes the link between the filter size and number of filter parameters. Hyena defines the filters implicitly by using feed-forward networks to parameterise N learnable filters $h^1(k) \dots h^N(k)$. Despite the motivation of uncoupling sequence length from the filters, Hyena still uses positional encodings as part of its filter architecture; thus, the parameter count scales linearly with the sequence length. However, due to the efficiency of the convolution, Hyena performs accurate inference on much longer sequence lengths (i.e order of 1 million) than previously seen with transformers [34].

Another way to implicitly define h is as the response to a state space model (SSM), which is loosely what Mamba and other similar SSM models do [35, 37]. A state space model comes from a dynamical system of differential equations:

$$\frac{dh}{dt}(t) = Ah(t) + Bx(t)$$
$$y(t) = Ch(t)$$

Here, $h(t)$ is some unknown hidden state, $x(t)$ is a continuous input, and $y(t)$ is some continuous output to be predicted. A, B and C are matrix parameters to be learned. The continuous model is discretised, according to;

$$h_t = \overline{A}h_{t-1} + \overline{B}x_t$$
$$y_t = Ch_t$$

Where, again, $x_1 \dots x_L$ are samples from the continuous time series, at Δ_t intervals. Here $\bar{A} = f_A(\Delta_t, A)$ and $\overline{B} = f_B(\Delta_t, A, B)$, where f_A and f_B are discretisation rules (e.g first order, second order, zero-order hold) [35].

When Δ_t is constant, the recurrence defined above can be recast using convolution, as:

$$\overline{K} = (C\overline{A}, C\overline{AB}, C\overline{AAB}, \dots, C\overline{A^{L-1}B})$$
$$y = x * \overline{K}$$

The Mamba model takes this SSM approach, and further makes $B(x)$ and $C(x)$, $\Delta_t = \phi(x)$, learnable functions of the input. In Mamba-2, the main difference is that these parameters are no longer dependent on the input; however, Δ_t is still learnable [38]. This makes the sequence time-varying, so it loses its equivalence with convolution and must be evaluated sequentially [35]. To maintain efficiency, the authors have designed it to be hardware-aware. They avoid loading from slow GPU memory when information is only temporarily required and ensure no unnecessary intermediate states are stored. It thus maintains efficiency on long sequence lengths, at the expense of being hardware-specific to GPU A100 architectures [35].

Although Hyena does not define filters using an SSM assumption, it has been classed in literature as an SSM-type model, due to its similar approach using implicit convolutions [34, 35]. I also note that despite the theoretical inspiration of such models, their adaptation to machine learning applications often violates initial assumptions. For example, these implicit convolutional approaches are applied to any sequential data [35, 39], without any notion of time, or sampling at regular

intervals, Δ_t, such as language or DNA sequences. In the Mamba model, the initial discretisation of the SSM rule would be different if the functions B, C and Δ_t were explicitly modelled as functions of the input x_t. Further, by assuming Δ_t is parameterised, it treats the sampling interval between the observed data as unknown, which in time-series data is untrue. In other sequence types, it could perhaps be interpreted as learning a kind of pseudo-time interval between each sequence element.

Like many approaches in machine and deep learning, these methods empirically work, and it is that, over their theoretical inspiration, which has brought them to prominence. SSM-type models have been shown to be on par with transformer architectures, able to learn long-range interactions yet also ingest much longer sequence lengths [34, 35, 38, 39]. Recent work has shown that these SSM-type models are similar to traditional recurrent neural networks (RNNs), with a few key changes [40]. Here, both Hyena and Mamba are implemented, to compare the more popular (Mamba), with the less resource-constrained (Hyena).

2.2.2 Mathematical Preliminaries

The goal here is to infer CNAs from measured data, and this can be done using the known mathematical link between true CNAs in a genome and the measured data, first defined in Van Loo et al. [4].

To see this, let $C_{T,i}$ be the total copy number at locus i in the tumour, $C_{P,i}$ the copy number of the paternal chromosome at locus i, $C_{M,i}$ the copy number of the maternal chromosome at locus i and $C_{B,i}$ be the B allele copy number. Further, let ρ be the purity of the tumour sample (the proportion of tumour vs non-tumour) and r_d be the expected number of sequencing reads per copy number. In practice, samples are sequenced up to a given average read depth, which assumes a uniform coverage of the short reads across the whole genome, [41]. When many duplicated regions exist, the actual average read depth per copy number r_d is unknown. Finally, let R_i be the total number of reads at a locus and B_i be the B allele frequency (BAF). The sequence data is therefore a collection $\{R_i, B_i\}_{i=1}^{L}$ for L loci.

For a pure tumour sample, the total copy number is:

$$C_{T,i} = C_{P,i} + C_{M,i}.$$

Considering sample impurity, the sample copy number C_T^s is:

$$C_{T,i}^s = \rho C_{T,i} + 2(1-\rho),$$

Where it is assumed that the contaminating normal cells have copy number 2 at all loci. While normal cells may possess some copy number variants, the size of these regions is typically negligible compared to the cancer-associated alterations we aim to detect, and hence are ignored for simplicity.

The B allele copy number is defined as:

$$C_{B,i} = s_{p,i} C_{P,i} + s_{m,i} C_{M,i},$$

Where $(s_{p,i}, s_{m,i}) \in \{(0,0),(0,1),(1,0),(1,1)\}$ denotes whether the paternal and maternal chromosomes respectively have the specified SNP B allele at locus i.

Adding sample impurity, the sample B allele copy number is:

$$C_{B,i}^s = s_{p,i}((1-\rho) + \rho C_{P,i}) + s_{m,i}((1-\rho) + \rho C_{M,i})$$

The total number of reads at a locus R_i is then:

$$R_i = r_d C_{T,i}^s$$

While the B allele frequency (BAF) B_i at locus i is given by:

$$B_i = \frac{C_{B,i}^s}{C_{T,i}^s}$$

Both R_i and B_i are measured data, obtained from sequencing a tumour sample. **Figure 2.1** illustrates the relationship from $(\rho, r_d, \{C_{P,i}, C_{M,i}\}) \rightarrow \{R_i, B_i\}$. The aim is to do the reverse- to infer $(\rho, r_d, \{C_{P,i}, C_{M,i}\})$ from $\{R_i, B_i\}$. However, in the above formulation, both R_i and B_i remain the same even if the values for $C_{P,i}$ and $C_{M,i}$ are swapped, meaning the parental copy numbers cannot be uniquely inferred - they are non-identifiable. Instead, let $A_{M,i}, A_{m,i} = \max(C_{P,i}, C_{M,i}), \min(C_{P,i}, C_{M,i})$ be the major/minor allele-specific copy numbers, which are identifiable. Hence, the aim is amended to infer $(\rho, r_d, \{A_{M,i}, A_{m,i}\})$ from $\{R_i, B_i\}$.

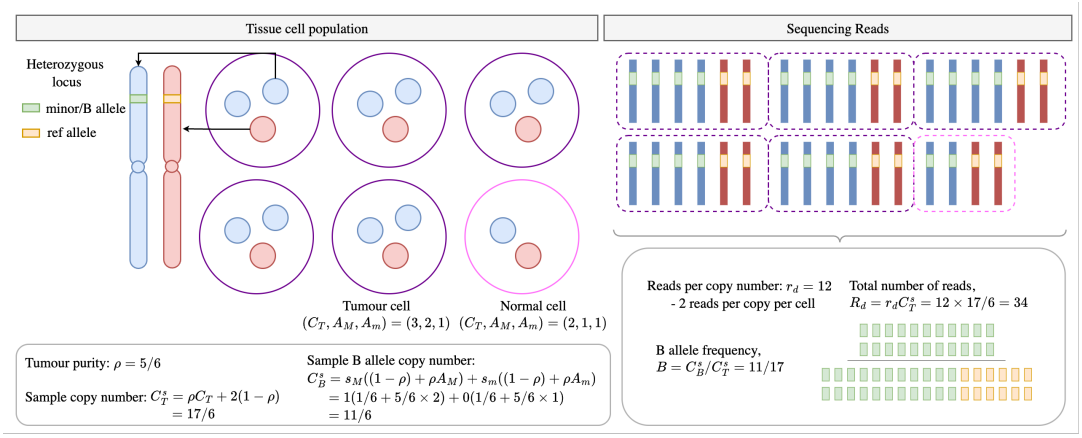


Figure 2.1 Mathematical construction of copy number calling. Illustration of how purity, copy number, read depth per-copy number, and heterozygous loci result in measured read depth and B allele frequency.

2.2.3 Identifiability

Inferring correct copy numbers is difficult and sometimes impossible, as different combinations of copy numbers, sequencing depth and sample purity can result in very similar read depth and BAF values. This concept is called non-identifiability, **Figure 2.2**. Two profiles might be identical unless one profile has a downstream segment that makes correct inference of the previous segments possible. A profile may not become identifiable until many megabases into the sequence. Hence, long-range information is required to correctly identify profiles.

Real tumour copy number profiles often contain many segments with deletions and different duplication combinations, making it more likely that only one unique copy number profile can explain the data (it is identifiable). However, they also contain local variation in both read-depth and BAF measures, making some regions ambiguous and difficult to call. When a profile has few segments and is non-identifiable, it is assumed that the smallest ploidy solution is more likely.

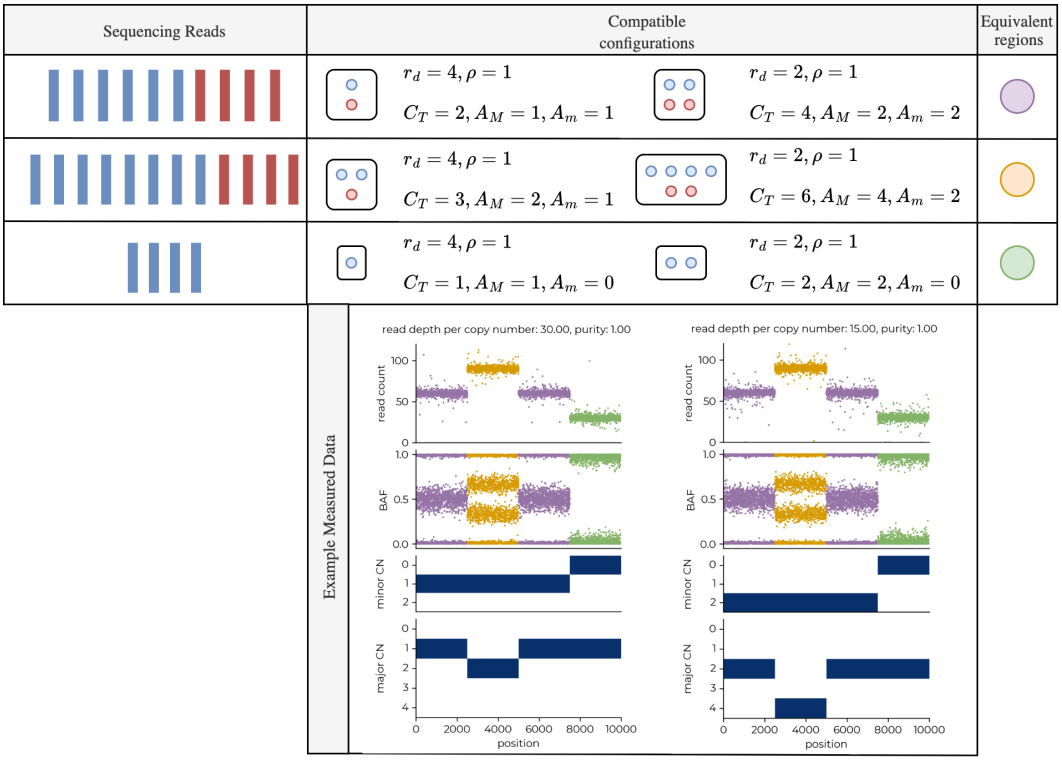


Figure 2.2 Identifiability of copy number profiles. Example illustrating how different copy number profiles can result in the same observed sequencing read depth and B-allele frequency measures. Here, each copy number profile on the right is exactly double that of the left, while the read depth per copy number value is half the left. Blue and red reads are those from the paternal/maternal chromosome, respectively, where the B-allele frequency will reflect their chromosome ratio at heterozygous loci. This combination results in the exact same measured data, as highlighted in the bottom panel, where the colours link to each configuration profile above. In general, many combinations of copy number profile, tumour purity and read depth per copy number parameters may give rise to similar observed data. In the presence of noise, segments that may uniquely determine the copy number profile can be missed.

2.2.4 Simulation-based inference

Simulation-based inference (SBI) is a Bayesian approach, applicable when the $p(\phi|x)$ is intractable but simulations from the joint distribution, $p(\phi, x)$ are available [42]. By generating pairs (ϕ, x) from the joint distribution, a model can be trained to approximate the posterior $p(\phi \mid x)$ directly. This approach has recently been applied in population genetics, [43, 44]. Amortised maximum a posteriori (MAP) estimation is a subset of SBI, where instead of estimating the full posterior, the MAP, $\hat{\phi}$, is predicted from the data instead [45].

The approach used in `araCNA` and introduced in **Section 2.3** uses the mathematical formulation introduced above to simulate data samples x from known copy number parameters, ϕ. `araCNA` is then trained to predict $\hat{\phi}$ from x, which can be viewed as a form of amortised point-estimate inference. This point estimate will align with the MAP in most cases, although weak identifiability and noise mean that this is not theoretically guaranteed. Accordingly, `araCNA` can be regarded as utilising a form of simulation-based inference for estimation.

2.3 Methods

2.3.1 Model overview

The input data for `araCNA` is assumed to be a sequence of allele-specific read counts at genome-wide loci, **Figure 2.3A**, which can be converted into total read depth (R) and B allele frequency (B) values as is commonplace for copy number callers. These are fed into a long-range sequence model, which converts the input sequence into an output sequence consisting of the probabilities of major and minor copy number values at corresponding loci. Global parameters of interest, such as tumour purity or ploidy, can also be provided, **Figure 2.3B**.

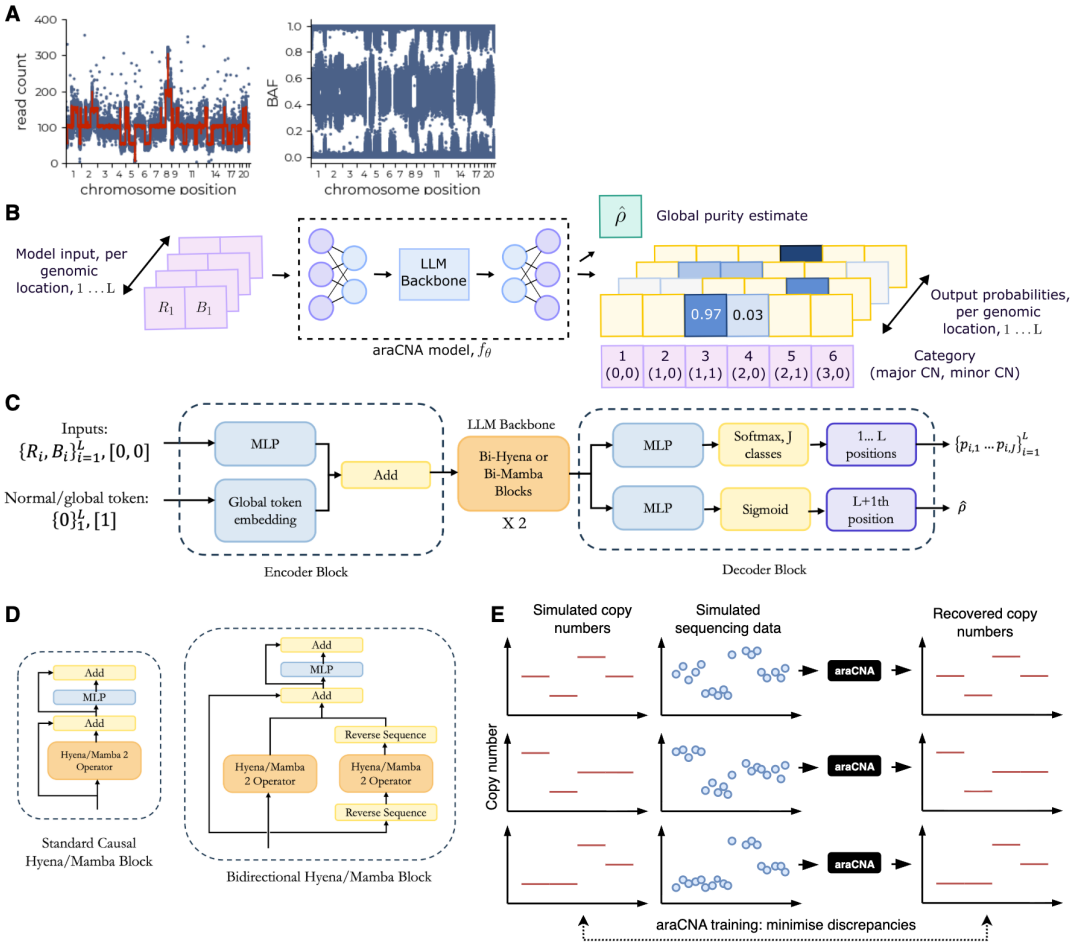


Figure 2.3 Overview of `araCNA` model. The (A) input data (read count and B allele frequency) is (B) converted by `araCNA` into a sequence of probabilistic copy number calls and global parameter estimates. The `araCNA` model architecture (C) contains bi-directional variants (D) of the causal Hyena or Mamba blocks. (E) The model is trained on simulated data.

2.3.2 The araCNA model

The function of `araCNA` can be summarised as $f_\theta(\{R_i, B_i\}) \to (\{p_{i,k}\}, \hat{\rho})$ where f_θ is the long-range sequence model parameterised by network weights θ. $\hat{\rho}$ is `araCNA`'s global purity estimate, while $p_{k,i}$ is the probability that `araCNA` assigns the locus as belonging to copy number profile K_j. The profile categories $K_1, \ldots, K_J$ correspond to major/minor parental copy number combinations, with $K_1 := (A_M = 0, A_m = 0)$, $K_2 := (A_M = 1, A_m = 0)$ etc. Here, $J = \sum_{i=0}^{T_{\max}} \min(i+1, T_{\max} - i + 1)$, where $T_{\max}$ is a hyper-parameter corresponding to the maximum modelled total parental copy

number. From these $\hat{r}_d = \mu_{\text{robust}}(R)/\Phi^s$ can also be estimated, where $\mu_{\text{robust}}(R)$ is the robust or trimmed mean of the read depth vector and Φ^s is the expected value of the overall sample copy number (i.e sample ploidy).

The model is trained using simulated datasets where the ground truth copy numbers and purity are known. The details of this simulation are discussed later in **Section 2.3.4**. The loss function consists of a supervised sequence loss and a supervised global loss. The supervised sequence loss is the cross-entropy:

$$\mathcal{L}_{ss} = -\frac{1}{L}\sum_{i=1}^{L}\sum_{j=1}^{J} I\{c_i = K_j\}\log(p_{k,i})$$

where L is the sequence length, $c_i \in K_1 \dots K_J$ is the known target profile of a genomic locus. The supervised global parameter losses are:

$$\mathcal{L}_{sr} = |r_d - \hat{r}_d|,$$
$$\mathcal{L}_{s\rho} = |\rho - \hat{\rho}|.$$

The total loss is then given by:

$$\mathcal{L} = \mathcal{L}_{ss} + \lambda_r\mathcal{L}_{sr} + \lambda_\rho\mathcal{L}_{s\rho}.$$

Where $\lambda_r = \lambda_\rho = 1$ was found to work well. The model was trained using a curriculum learning approach, iteratively increasing the simulation complexity, detailed in **Section 2.3.5**.

Model Architecture

Sequence inputs are entered into `araCNA` together with global placeholders, and projected through an encoder block, backbone and decoder block to give the copy number and purity outputs, **Figure 2.3B,C**. Elements used as global estimates (i.e. purity) in the output are appended to the end of the normal sequence, with zero values for read depth and minor allele frequency dimensions. To differentiate these elements from normal elements, a global token feature is used, encoded as 0 for normal elements and 1 for global prediction sequence elements.

The normal observational data (read depth and minor allele frequency), is projected into a dimension d using a multi-layer perception (MLP) with ReLU

non-linearities, while the token feature is projected into dimension d using a simple embedding. The two inputs are combined in this projected space to serve as the encoder block for the main backbone of the model. The output from the main backbone diverges into two streams in the decoder- one for the prediction of the copy numbers and one for the prediction of the global parameters. Both streams use MLPs with ReLU activation, with a softmax to project the copy number stream into a probability over the J possible classes, and a sigmoid to project the purity estimate between 0 and 1.

`araCNA` was implemented to use either of the two recently developed long-range sequence models, Hyena [34] or Mamba [35, 38], in its main backbone. As introduced in **Section 2.2.1**, unlike prevalent transformer approaches, these SSM-like models offer training times that scale linearly with sequence length, which is necessary in genomic data. The `araCNA` models with Mamba/Hyena blocks are termed `araCNA`-mamba and `araCNA`-hyena, respectively. Mamba-2 is the more popular of the SSM models, but is constrained to GPUs with Nvidia A100 architecture (for training and inference). Similar to [46], the blocks are adapted to be bidirectional rather than causal (unidirectional) as used in natural language processing, **Figure 2.3D**. Here, an additional Hyena/Mamba-2 operator processes the reversed sequence, and its reverse projection is combined with the output projection from the Hyena/Mamba-2 operator over the original sequence. Hence, each sequence element has access to every other sequence element.

The hyperparameters of `araCNA`-mamba and `araCNA`-hyena are outlined in **Table A.1**, resulting in parameter counts of 70K and 12.09M, respectively. Each hyena block contains two positional encodings of length $L_{\max}$ corresponding to a hyperparameter of the maximum input sequence length. For the pretrained models, $L_{\max}$ was set to 1M, hence the majority of the parameters, 12M, can be attributed to the positional encodings. Notably, 70K parameters for `araCNA`-mamba are small for a modern neural network [47], especially given that the task at hand is not trivial.

2.3.3 Smoothing

Given the noise in real data, many copy number calling methods employ smoothing or thresholding in some way to reduce the number of CNA segments and reduce false positives. For example, ASCAT has a 'penalty' parameter that controls smoothing [4], Battenberg has several 'gamma' hyperparameters [1], HMMCopy has 'e' and 'strength' parameters [2] and CNV Kit uses thresholding to segment and call different copy numbers [5]. `araCNA` also employs a smoothing technique, based on the predicted output probabilities.

Using a similar approach to the Viterbi algorithm in HMMs, the aim is to find the 'best' joint sequence $\mathbf{S} \in \{1, \dots J\}^L$ given the model output probabilities, $p_{k,i}$, that a locus, i, is in copy number state k. The optimal sequence is found by balancing the probabilistic evidence of being in a given state, with the penalty λ_t of transitioning to a different state as:

$$\hat{\mathbf{S}} = \arg\min_{\mathbf{S}} \left[-\sum_{i=1}^{L} \sum_{k=1}^{J} I\{S_i = k\} \log p_{k,i} + \lambda_t \sum_{i=1}^{L-1} I\{S_i \neq S_{i+1}\} \right]$$

For a given λ_t, this is solved using a dynamic programming approach. Here, $\lambda_t = 500$ was found to be suitable.

2.3.4 Synthetic data simulation

For this work, `araCNA` is trained using simulated copy number profiles, **Figure 2.3E**. This approach was taken since (i) there exists no ground-truth, high-resolution copy number profiles upon which `araCNA` could be trained and (ii) to avoid using copy number profiles produced by other methods to perform fair comparisons later on. Synthetic copy number profiles were generated using the following procedure:

1) Sampling the number of segments. This is done by sampling the approximate number of copy number segments, $\hat{N}_s$, using a mixture approach; first, sample a uniform variable, u, such that under a user-defined swap probability, q_s, the number of segments is sampled uniformly between 1 and N, where N is a hyperparameter. When $u > q_s$, a Poisson distribution is used to skew sampling towards smaller total segments. This is to oversample weakly identifiable profiles with fewer

segments, where it is harder to estimate global parameters like read depth per copy number and purity.

2) Sampling the segment breakpoints. This is done by randomly sampling $b_1, \ldots, b_{\hat{N}_s}$ breakpoints from $1 \ldots L$, the unique set of these breakpoints defines the segments, and $N_s = |\{b_1, \ldots, b_{\hat{N}_s}\}|$. Only segments that have a minimum segment length of $L_{\min}$ are kept.

3) Sampling the segment profiles. Sample A_M, A_m of each segment from the possible copy number profiles. Logic is injected here to preferentially sample profiles closer in copy number to 1-1 when there are fewer segments. This is due to the identifiability issue. When there are more segments, profiles are sampled more uniformly but still with a preference for lower copy numbers, to inject an implicit bias towards lower ploidy solutions when the profiles are weakly identifiable.

From a sampled profile, the sequencing read depth and B allele frequency data are simulated. Each of the L loci is considered a commonly varying single-nucleotide polymorphism (SNP). For both parental alleles, A_M, A_m, each SNP is sampled as binomial with a probability of 0.5, the purity, ρ, is sampled uniformly from a range between 0.5 and 1. This gives the sample minor allele copy number, $C^s_{B,i}$ and the sample total copy number, $C^s_{T,i}$. The read depth per copy number, r_d is sampled uniformly between 5 and 70, and together with $C^s_{T,i}$ the overall read depth, R_i is sampled from this mean with additional noise. The BAF, B_i is sampled using total reads sampled based on R_i and the subset of B-allele reads using a binomial probability of $C^s_{B,i}/C^s_{T,i}$, with added noise.

In real data, there exist regions of prolonged homozygosity that can be attributed to identity-by-descent (IBD) regions (i.e identical regions inherited from both parents due to a common ancestor), Figure 2.4. To emulate this, regions of prolonged homozygosity are randomly injected into the model. In these IBD regions, the BAF cannot be used to infer copy-number, and the model must use context from before/after the homozygous region for correct prediction.

Thus this sampling procedure gives targets $(\{A_{M,i}, A_{m,i}\}, \rho, r_d)$ that generate inputs $\{R_i, B_i\}$, which together are used in the training of `araCNA`. Hence, `araCNA`

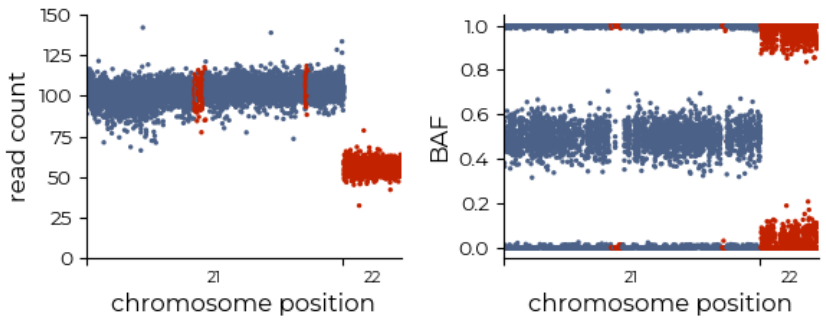


Figure 2.4 Example of homozygous regions. Real sample example illustrating prolonged homozygous regions on chromosome 21, where the BAF noise is very low. In contrast, the region on chromosome 22 indicates a minor copy number of 0, and hence, although the BAF profile is close to 1 and 0, the noise profile is larger, where normal contamination at heterozygous loci brings the BAF closer to 0.5. Red colouring indicates regions of interest.

can be interpreted as performing inference on the statistical formulation introduced in **Section 2.2.2**, when $(\{A_{M,i}, A_{m,i}\}, \rho, r_d)$ are treated as unknowns. Further details of this procedure are included in **Appendix A.2**. Using these simulations, `araCNA` is exposed to different sequence data distributions and learns how to process these into copy number calls. As a consequence, when used at test-time for zero-shot inference, it is not necessary to retrain `araCNA` to process any tumour sample, unlike conventional CNA callers, which are optimised on each individual sample.

2.3.5 Curriculum Learning Procedure

To train `araCNA`, a curriculum learning procedure was taken, gradually increasing the complexity of the problem. This was found to make model training easier, and a similar approach was taken in Nguyen et al. [39], where they gradually increase the sequence length.

The training procedure was:

1. Begin the synthetic data generating procedure with $\rho = 1$, and without sampling the noise parameters. Use only up to a maximum total copy number of 2, that is profiles: $(A_M, A_m) \in \{(0,0), (1,0), (1,1), (2,0)\}$. Sample r_d, and start with sequence length 10000. Train until convergence.

2. Using the previously trained model weights as initialisation, add in purity and noise parameter sampling. Train until convergence.
3. Using the previously trained model weights as initialisation, slowly increase the maximum total copy number to 8. Train until convergence.
4. Using the previously trained model weights as initialisation, slowly increase the maximum sequence length to 650,000. Train until convergence.

The total copy number is capped at 8, as the utility of modelling high-level amplifications (e.g. CN $>$ 8) is limited, often being grouped for downstream analysis, e.g COSMIC, [48], groups $\geq$ 9 CNs together. However, a higher total copy number could be easily incorporated with an additional curriculum learning step, increasing training time. Depending on the model capacity, increasing the total copy number may also require a larger model, either with increased hidden dimensions or more layers.

2.3.6 LogR Input variant

`araCNA` was developed initially to require only a tumour sample. However, existing methods often use a normalised read depth input that corrects for mappability and GC bias and usually require a matched normal sample [1, 4]. To highlight both how `araCNA` can be finetuned with different input data, as well as evaluate `araCNA` on this more standard data, an `araCNA`-logR variant was also implemented.

This variant takes the `araCNA`-mamba pretrained model and finetunes it using a new simulation procedure. Here the matched normal read depth, r_d^n is also sampled, and instead of R_i as input, the log normalised read depth, $l_{r,i}$ is used instead, given by:

$$t_{r,i} = \frac{R_i}{2r_d^n}$$
$$l_{r,i} = \log \frac{t_{r,i}}{\mu_{\text{robust}}(t_r)}$$

Where $\mu_{\text{robust}}(t_r)$ is the robust mean, trimming the top/bottom 5% of data. Further details of the simulation procedure are included in **Appendix A.3**.

This emulates the approach used in ASCAT for real data. `araCNA`-logR is finetuned using samples drawn from this new simulation procedure with initial weights given by `araCNA`-mamba, training until convergence. In the model, the R_i channel is replaced with $l_{r,i}$, and the $\mathcal{L}_{sr}$ loss term with $\mathcal{L}_{s\Phi} = |\Phi^s - \hat{\Phi}^s|$, where Φ^s is the sample ploidy, given by $\mu(C_T^s)$. This term links the purity estimate to the output CNs. Given the replacement of R_i with $l_{r,i}$, no architecture changes are required; however, if there were (e.g additional input information), a different encoder model could be used with the backbone and decoder weights given from the pretrained model.

For evaluating `araCNA`-logR on real TCGA data, the output of the ASCAT preprocessing pipeline is used as input to `araCNA`-logR, [4]. This pipeline corrects for mappability and GC content.

2.3.7 Caller implementations

To evaluate `araCNA`, I compared it to several existing CNA calling tools, two of which are commonly-used algorithms that can call allele-specific copy numbers: ASCAT, [4], and an adaptation of ASCAT called Battenberg, [1], which can also model sub-clonal populations, by optionally assigning a fraction of each CNA segment to another subclone. Results are compared to Battenberg without and with its default sub-clonality modelling.

`araCNA` is also compared to CNV Kit, [5] and HMMCopy, [2]. CNV Kit works with WGS, although it is intended primarily for WES, and gives allele-specific copy numbers after first calling total copy numbers, while HMMCopy only calls total copy number [2]. Both ASCAT and CNV Kit performed well in a recent benchmark [49], while Battenberg and HMMCopy are popular methods included in other method benchmarks [24, 50].

ASCAT

ASCAT uses a piecewise constant fitting algorithm to segment the data based on step changes present in logR, and BAF [4]. It then uses a grid search approach

using values for the purity, ρ and ploidy ψ, to find integer copy numbers which best explain the measured data.

ASCAT is implemented according to the GitHub, following closely the example given by path: `ExampleData/README.md` with heading 'Extracting logR and BAF from HTS data and running ASCAT'. Please see the araCNA codebase for the exact snakemake workflow used.

Battenberg

Battenberg takes a similar approach to ASCAT, however, may also assign a fraction of each segment to a subclone with a different set of copy numbers [1]. This gives an extra set of parameters for each loci/segment; τ_i, the fraction of the sample at that segment attributed to clone 1, and the major/minor copy numbers of clone 1, $A^1_{M,i}, A^1_{m,i}$, as well as $(1-\tau_i)$, $A^2_{M,i}, A^2_{m,i}$ for clone 2 [1]. They also add an additional constraint that $A^2_{M,i} + A^2_{m,i} = A^1_{M,i} + A^1_{m,i} \pm 1$, although multiple copies of a region might occur after the emergence of a subclone. If subclones exist in the data, the distribution τ_i over segments should be centred around a few distinct modes, where each mode has a different set of copy numbers over regions.

Battenberg is implemented according to the GitHub, using R package 'battenberg', following closely the example given by path `inst/example/battenberg_wgs.R`. Please see the araCNA codebase for the exact snakemake workflow used.

HMMCopy

HMMCopy uses only the read depth ratio with normal reads of binned genomic regions, correcting for GC content and mappability. It uses an HMM to segment the genome into regions of constant copy numbers and call these total copy numbers [2].

HMM Copy is implemented according to the manual, which is available with its installation through Bioconductor. It also requires using HMM Copy Utils GitHub for preprocessing the BAM files. The main script follows the Ontario Institute for Cancer Research GitHub script `run_HMMcopy.R`. The same parameter tuning as the script above is adopted.

It should be noted that the HMM parameters sometimes need individual tuning post-inference to obtain fewer total segments and correct the copy number calls [2]. HMMCopy results could perhaps be improved with further parameter tuning on individual datasets; however, results obtained without manual tuning capture the reduced utility of this tool compared to other methods that do not require such tuning.

Please see the araCNA codebase for the exact snakemake workflow used, including some preprocessing required as described in the HMM Copy Utils GitHub at `README.md#Example#Onbinwidths`.

CNV Kit

CNV Kit is primarily designed for hybrid capture to infer copy number states, even at regions with very low coverage; however, it also works for WGS data [5]. It uses circular binary segmentation [51] to segment the genome into areas of constant copy number, then assigns total copy numbers using thresholds on the log read ratio between normal and tumour. After, it uses BAF to estimate the proportion of each total copy number that can be assigned to each allele [52]. Hence, it only uses the BAF to infer allelic copy numbers after initial total copy number calling.

CNV Kit is implemented according to the documentation. The docker image installation was used with the 'batch' function pipeline, specifying the method as whole genome sequencing. CNV Kit can also do allele-specific calling if provided with VCFs, so I used VarScan, [53], to obtain VCFs of the tumour files at SNP loci, followed by calling with CNV Kit. CNV Kit does not directly account for purity/ploidy but can take both as an input- mainly affecting thresholds for calling copy numbers [52]. Hence, I further implemented CNV Kit with ASCAT purity/ploidy as a final comparison point. As this performed better, **Figure A.1**, it has been used as the default, denoted CNVkit*.

`araCNA`

The implementation of `araCNA` is available on GitHub, araCNA codebase.

`araCNA`-mamba and `araCNA`-hyena perform copy number inference using a tumour BAM file, without requiring a matched normal, and a SNP file with relevant, non-problematic SNP locations. The `araCNA`-logR variant performs inference using logR/BAF values derived in standard ways with a matched normal, and correcting for mappability and GC content, as output by the ASCAT preprocessing pipeline. `araCNA` outputs a CN segments file, and a summary information file including the purity, ploidy and whole genome duplication estimates. In the real data, a non-problematic SNP list was derived from a blacklisted region reference in literature and SNP locations, having a length of ∼650k. Hence, the pretrained models were trained up to this length, and tolerate some sequence length variation, but may struggle if the input SNP list differs substantially. However, they can easily be fine-tuned up to 1 million SNPs if a higher resolution is required. Current `araCNA` pretrained models do not explicitly account for sex chromosomes and are limited to WGS-derived data, though they could be fine-tuned to other data modalities as outlined in the discussion. `araCNA`-mamba and `araCNA`-logR pretrained models require an A100 GPU for inference.

2.3.8 Metrics

Given the ground-truth is unknown on real data samples, a proxy for the accuracy of each model is the reconstruction error of the input data, with the caveat that more expressive models can overfit to data (and hence have a lower reconstruction error). Reconstruction error for both the BAF and the read depth is used.

The BAF root mean square error (RMSE) is calculated as:

$$\sqrt{\frac{1}{L}\sum_{i=1}^{L}(B_i - \hat{B}_i)^2}$$

$$\hat{B}_i = \min_{s_1,s_2} \frac{\hat{C}^s_{B,i}(s_1,s_2)}{\hat{C}^s_{T,i}}$$

$$= \min_{s_1,s_2} \frac{s_1((1-\hat{\rho}) + \hat{\rho}\hat{A}_{M,i}) + s_2((1-\hat{\rho}) + \hat{\rho}\hat{A}_{m,i})}{2(1-\hat{\rho}) + \hat{\rho}(\hat{A}_{M,i} + \hat{A}_{m,i})}$$

where the reconstructed BAF, $\hat{B}_i$ is calculated from the most probable haplotype $(s_1, s_2) \in \{(0,0),(0,1),\ (1,0),(1,1)\}$ at each locus.

The read depth mean absolute error (MAE) is calculated as:

$$\frac{1}{L}\sum_{i=1}^{L}|R_i - \hat{R}_i|,$$

$$\hat{R}_i = \hat{r}_d \hat{C}^s_{T,i},$$

$$\hat{r}_d = \frac{\mu_{\text{robust}}(R)}{\frac{1}{L}\sum_{i=1}^{L}\hat{C}^s_{T,i}}.$$

Here, $\mu_{\text{robust}}(R)$ is the robust or trimmed mean of the data, excluding data at the highest/lowest 5% of values. Absolute error, rather than MSE, is more appropriate here due to mapping errors that can lead to read depth and hence MSE inflation.

Battenberg may assign a fraction of each segment to a subclone with a different set of copy numbers [1]. Hence, for the Battenberg multiclonal reconstruction, the fraction of each segment attributed to some sub-clone is also accounted for. This gives an extra set of parameters for each loci/segment; τ_i, the fraction of the sample at that segment attributed to clone 1, $A^1_{M,i}, A^1_{m,i}$, as well as $(1-\tau_i)$, $A^2_{M,i}, A^2_{m,i}$ for clone B. Hence, the reconstruction is adapted to follow the Battenberg formulation [1], and in their implemented code (https://github.com/Wedge-lab/battenberg/), where the sample copy numbers for each locus is given by:

$$C^s_{B,i} = s_1((1-\rho) + \rho(\tau_i A^1_{M,i} + (1-\tau_i)A^2_{M,i}) + s_2((1-\rho) + (\tau_i A^1_{m,i} + (1-\tau_i)A^2_{m,i}),$$

$$C^s_{T,i} = 2(1-\rho) + \rho(\tau_i(A^1_{M,i} + A^1_{m,i}) + (1-\tau_i)(A^2_{M,i} + A^2_{m,i})),$$

The analysis for the non-multiclonal Battenberg reconstruction uses the copy number predictions of the clone estimated as having the highest proportion (i.e $\max(\tau_i, 1-\tau_i)$ at each locus, with the normal formulation.

The concordance, CC, between two methods, 1 and 2, is calculated according to:

$$
\begin{aligned}
CC_{\text{major}} &= \frac{1}{L}\sum_{i}^{L} I(A^1_{M,i} = A^2_{M,i}) \\
CC_{\text{minor}} &= \frac{1}{L}\sum_{i}^{L} I(A^1_{m,i} = A^2_{m,i}) \\
CC_{\text{both}} &= \frac{1}{L}\sum_{i}^{L} I(A^1_{M,i} = A^2_{M,i} \text{ and } A^1_{m,i} = A^2_{m,i}) \\
CC_{\text{total}} &= \frac{1}{L}\sum_{i}^{L} I(A^1_{M,i} + A^1_{m,i} = A^2_{M,i} + A^2_{m,i}) \\
&= \frac{1}{L}\sum_{i}^{L} I(C^1_{T,i} = C^2_{T,i})
\end{aligned}
$$

For methods like HMMCopy, only C_T is output, so only CC_{total} can be measured. When method 1 is the ground truth, like in simulations, this can be interpreted as an accuracy.

2.4 Results - Simulation study

Figure 2.5 compares results from the two `araCNA` variants (`araCNA`-mamba and `araCNA`-hyena) using simulated data, sampled from the same generating procedure from which the training data was also sampled. Results are across 100 simulated test genomes, with a maximum sampled sequence length of 650k, to emulate the data size of the non-problematic SNP dataset. **Figure 2.5A** shows that both models achieve high copy number classification accuracy for the task, though `araCNA`-mamba slightly outperforms `araCNA`-hyena. Both models perform well at predicting simulated purity and ploidy, **Figure 2.5B**. Here, accuracy can be directly measured since the ground truth is known. However, in real data, there is no known ground truth, so reconstruction error is also included **Figure 2.5C**, where better methods are likely to have a lower BAF and read depth reconstruction. The differences are small, though `araCNA`-mamba achieves slightly better metrics than `araCNA`-hyena. **Figure 2.5D,E** show the reconstructed read depth, BAF and predicted copy numbers from each model for one of the example simulated genomes.

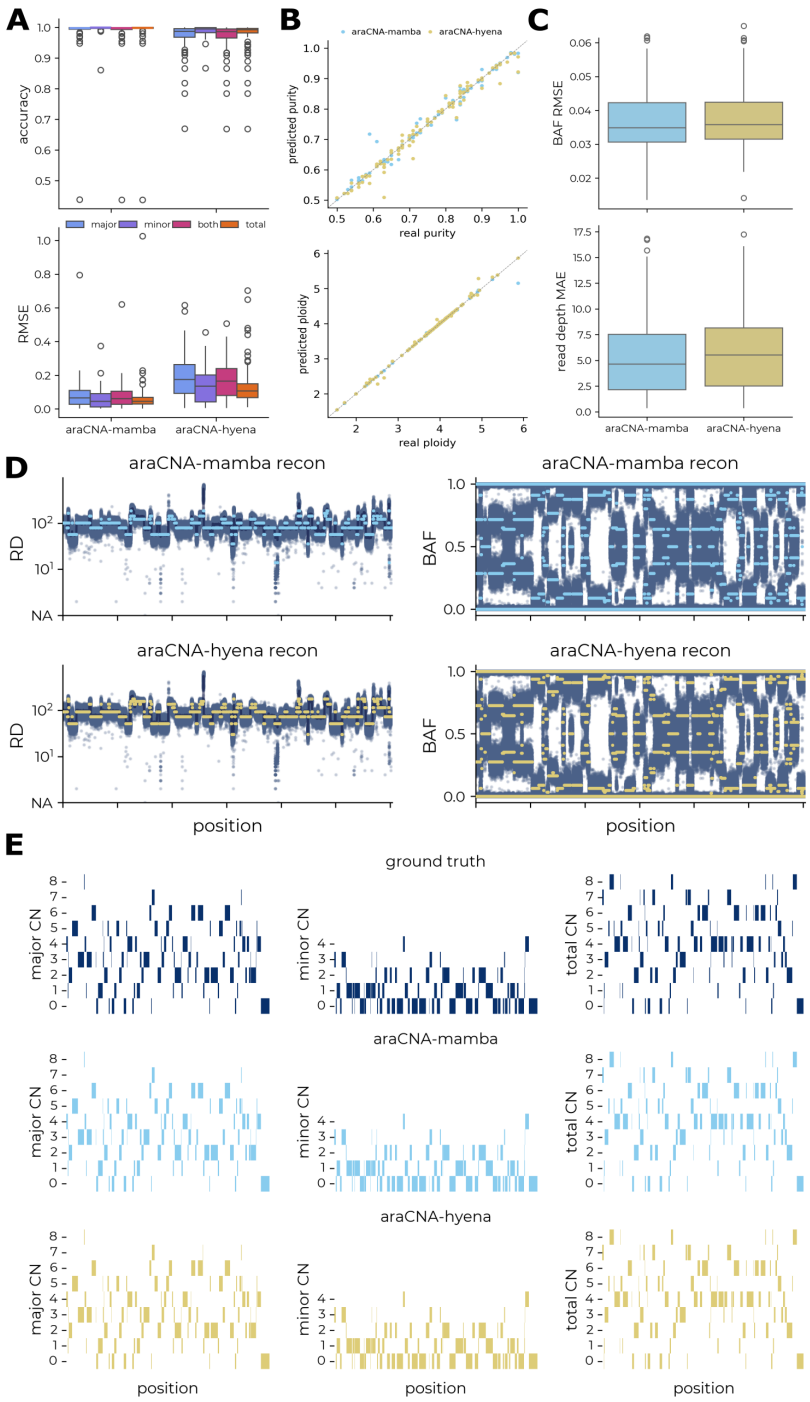


Figure 2.5 Results for `araCNA` model variants on simulated data. A-C show aggregated results across 100 simulated test genomes. (A) The distribution of concordance and RMSE between the predicted and true copy numbers. (B) The predicted purity and ploidy against the true purity and ploidy. (C) The distribution of mean reconstruction error: BAF root mean squared error (RMSE) and read-depth mean absolute error (MAE). (D) The reconstructed read depth and BAF, and E) the underlying predicted copy number segments for `araCNA`-mamba and `araCNA`-hyena on an example simulated test set, the maximum total CN is 8, hence the maximum minor CN is 4 (4, 4).

The Hyena and Mamba models were trained using the same underlying simulation and training procedure. Differences in results can therefore be traced to differences in the model architecture or sensitivity to hyperparameters. Hyena often took longer to converge during training, becoming stuck in local minima, possibly due to the size of the model (12M parameters). In contrast, the Mamba model (70K parameters) achieved better performance despite its significantly smaller size.

2.5 Results - The Cancer Genome Atlas

To evaluate `araCNA` on whole genome sequencing data, 50 tumour samples were chosen from the colorectal (CRC), breast (BRCA) and ovarian (OV) cancer cohorts of The Cancer Genome Atlas (TCGA). These three cancer types were selected due to the extensive and well-characterised aneuploidy present in tumours of these types.

2.5.1 Performance on normal sample

As an initial sanity check, **Figure 2.6** demonstrates that both `araCNA`-mamba and `araCNA`-hyena recover diploid states from a normal sample, using a randomly selected matched normal from the 50 TCGA sample superset.

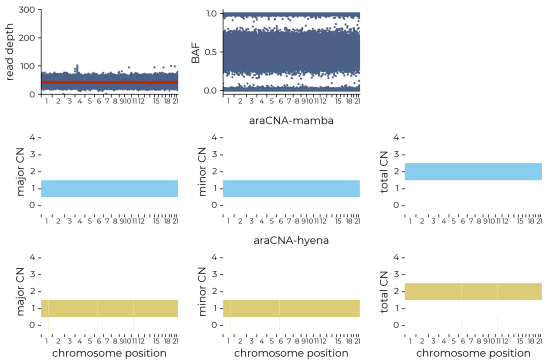


Figure 2.6 Model output on real normal WGS inputs

2.5.2 Performance across tumour samples

Figure 2.7 highlights the performance of `araCNA` compared to other implemented methods across the 50 tumour samples, while **Figure 2.8A,B** details the results on a representative TCGA ovarian cancer sample.

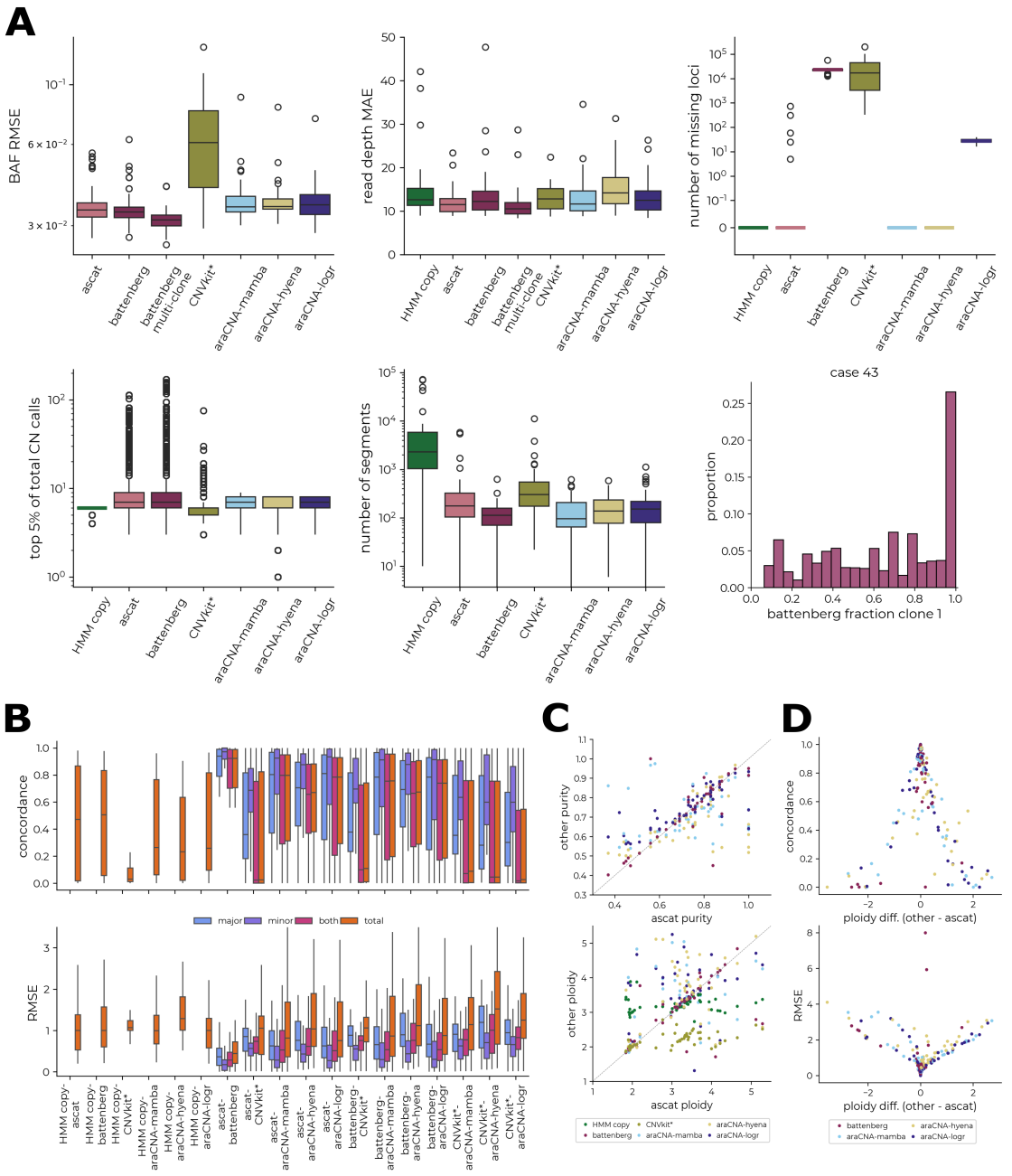


Figure 2.7 Comparison between CNA callers across 50 TCGA cancer samples (A) Boxplots showing the B-allele-frequency root-mean-squared reconstruction error (RMSE), read-depth mean absolute error (MAE), number of missing loci after calling, copy number distribution of top 5% of copy number calls and the number of different copy number segments identified across the tumour samples. Example distribution of Battenberg clonal fraction for a particular tumour. (B) The distribution of concordance and RMSE between the copy number predictions of each method. (C) The predicted tumour purity and ploidy against the ASCAT-derived purity and ploidy. (D) Concordance and RMSE of predicted copy numbers against the ploidy difference between methods and ASCAT.

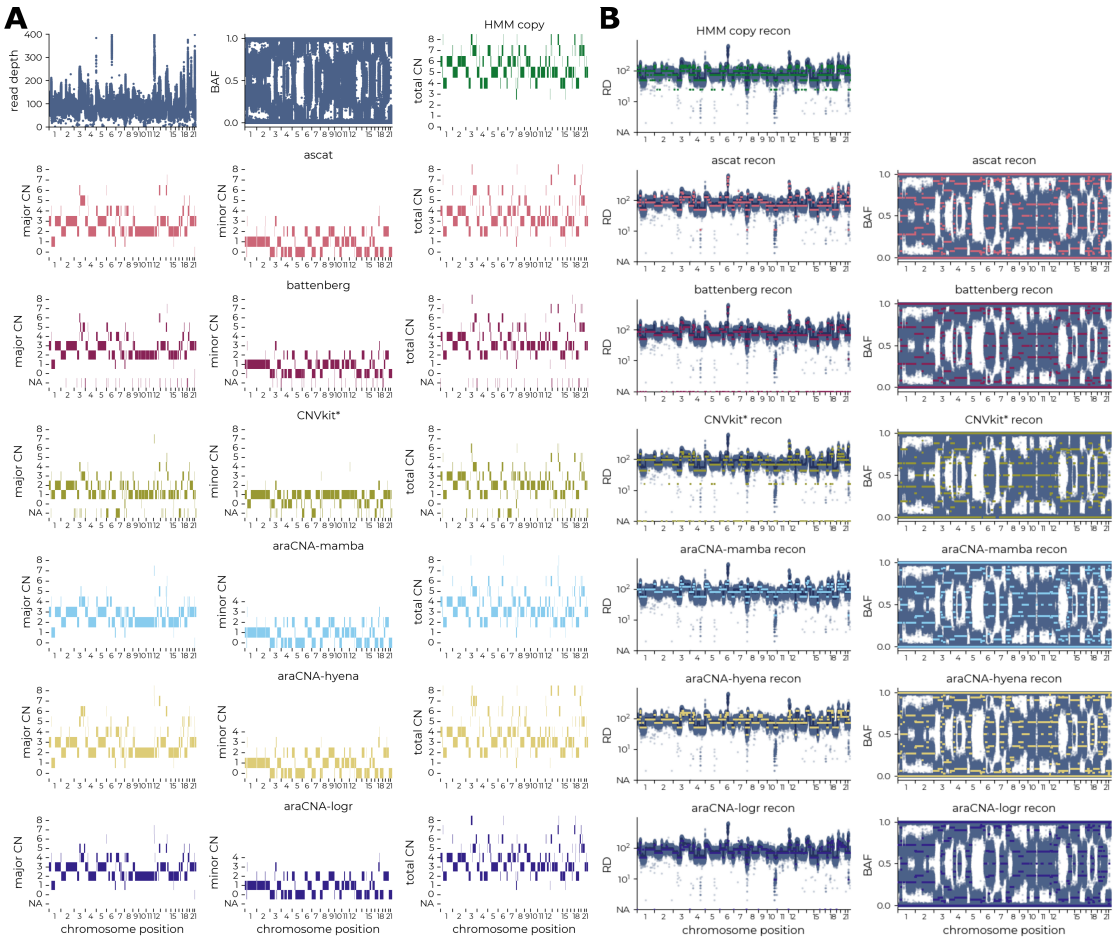


Figure 2.8 Representative TCGA ovarian cancer sample. Each row shows A) the called copy numbers and B) the respective reconstruction of each caller, where `araCNA` has approx. 80% concordance with ASCAT, Battenberg and HMM Copy.

Since there are no ground truth copy number profiles for these tumours, proxy measures such as reconstruction error are used to provide an unsupervised metric for performance, where better methods will be expected to have low reconstruction error. However, it is also important to consider both the number of segments and the range of modelled copy numbers produced by each CNA calling approach. A low reconstruction error accompanied by a large number of highly variable copy number segments may suggest overfitting, while high copy number calls at high read depth regions will have a lower reconstruction error but could be less plausible.

Figure 2.7A shows the distribution of the root mean-squared error (RMSE) and mean absolute deviation (MAE) for the reconstruction of the B allele frequency and read depth over the fifty samples. Using the three `araCNA` variants for zero-shot

inference of the copy number states gives comparable reconstruction performance to the existing CNA calling methods while using similar numbers of segments. Although ASCAT and Battenberg achieve slightly improved reconstruction error performance, they show signs of overfitting to the data, as explored further in **Section 2.5.3**.

`araCNA`-logR, which uses ASCAT-preprocessed matched normal/GC corrected inputs, achieves similar results to the `araCNA`-mamba model, which uses only tumour read depth. This suggests that preprocessing differences are likely negligible, and the observed variation is more likely due to identifiability issues, explored in **Section 2.2.3**, or high-noise tumour regions that are difficult to resolve.

Figure 2.7B details the pairwise copy number classification concordance and the root mean squared difference between copy number calls from different methods. `araCNA`-mamba achieves a median of 80% concordance with ASCAT for all copy number calls across the 50 TCGA cancer samples and a median of 70% concordance with Battenberg, while `araCNA`-hyena was substantially less concordant with ASCAT and Battenberg. However, when the discrepancies are measured using root mean squared difference, the average differences were much less than one, and most discrepancies are due to small numerical differences (e.g. $1 \leftrightarrow 2, 2 \leftrightarrow 3$, etc). Further, differences between ASCAT and `araCNA` appear to be primarily driven by different major and total copy number calls in some samples.

To further investigate call discrepancies, **Figure 2.7C** compares tumour ploidy and purity estimates using ASCAT as a baseline. While tumour purity estimates are well correlated between all methods, tumour ploidy estimates differ between methods for some tumours. Although `araCNA` calls a different copy number profile for some samples compared to ASCAT, it still gives low reconstruction error. Such tumours likely have weak identifiability (see **Figure 2.2**) and `araCNA` identifies a different ploidy state to ASCAT but with a corresponding copy number profile that is still compatible with the observed data. Indeed, the concordance between `araCNA` and Battenberg with ASCAT shows a reduction when there is a significant departure from the ploidy state estimated by ASCAT, **Figure 2.7D**. Interestingly, where there are disagreements, tumours classified as near-triploid by ASCAT will generally

be classified in a higher ploidy state by `araCNA` and vice-versa, which is a hallmark of the identifiability issue. There also exists a subset of tumours where ASCAT and Battenberg disagreed on tumour ploidy despite their algorithmic similarities, which further highlights the sensitivity of methods to the identifiability issue.

2.5.3 Markers of overfitting

While ASCAT and Battenberg are able to achieve slightly improved reconstruction error performance, they sometimes assigned as high as 100 copies to localised genomic regions, **Figure 2.7A**. It is difficult to verify the correctness of such calls given the lack of ground-truth; however, most of these regions contain few or no known cancer or other functional genes, **Figure 2.9**.

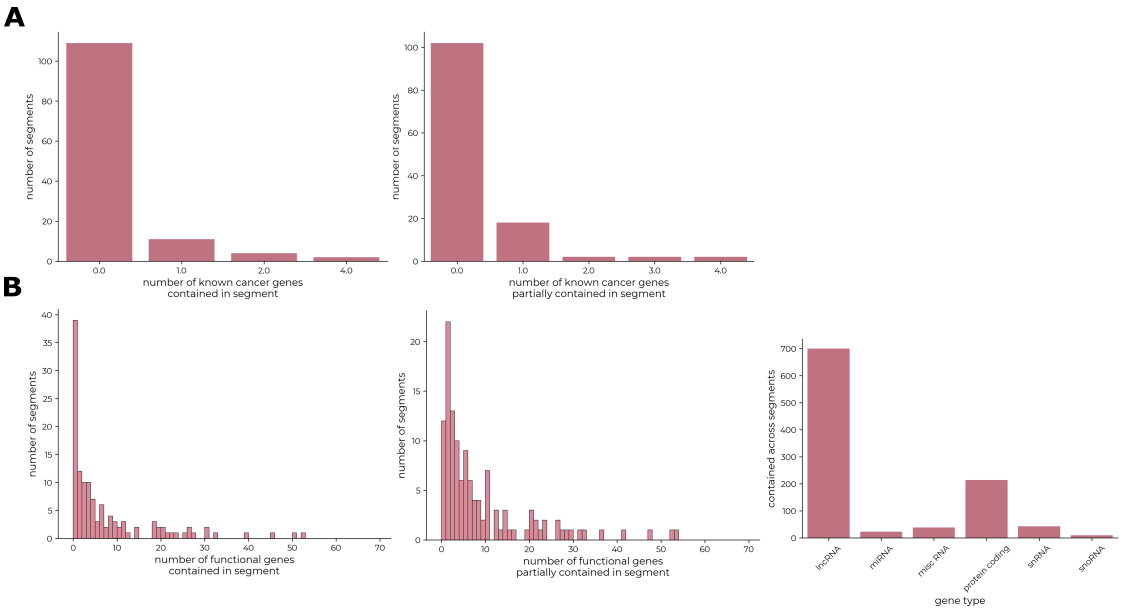


Figure 2.9 Gene coverage of ASCAT focal aberrations. The distribution of (A) cancer and (B) all functional genes wholly or partially overlapped by ASCAT focal aberrations (total copy number ≥ 20). Most ASCAT focal aberrations contain few or no known genes.

Further, Battenberg's multi-clonal formulation uses the same input data with more degrees of freedom to model clonal fractions, **Section 2.3.7**. In the presence of clones, fraction values should concentrate around discrete modes, corresponding to the fraction of the sample originating from a subclone with a diverging copy number pattern across the genome. **Figure 2.10** indicates that in many samples, the

Battenberg multiclone solution is likely overfitting to the input data, as there is often a more uniform distribution over the fraction of clone 1 when clones are detected. Battenberg may correctly identify clonal populations in cases 32 and 40, where it could be argued that there are modes around clonal fractions of 0.5. However, for most other samples, it appears that the Battenberg algorithm is likely overfitting to noisy regions by introducing more modelling parameters than is necessary.

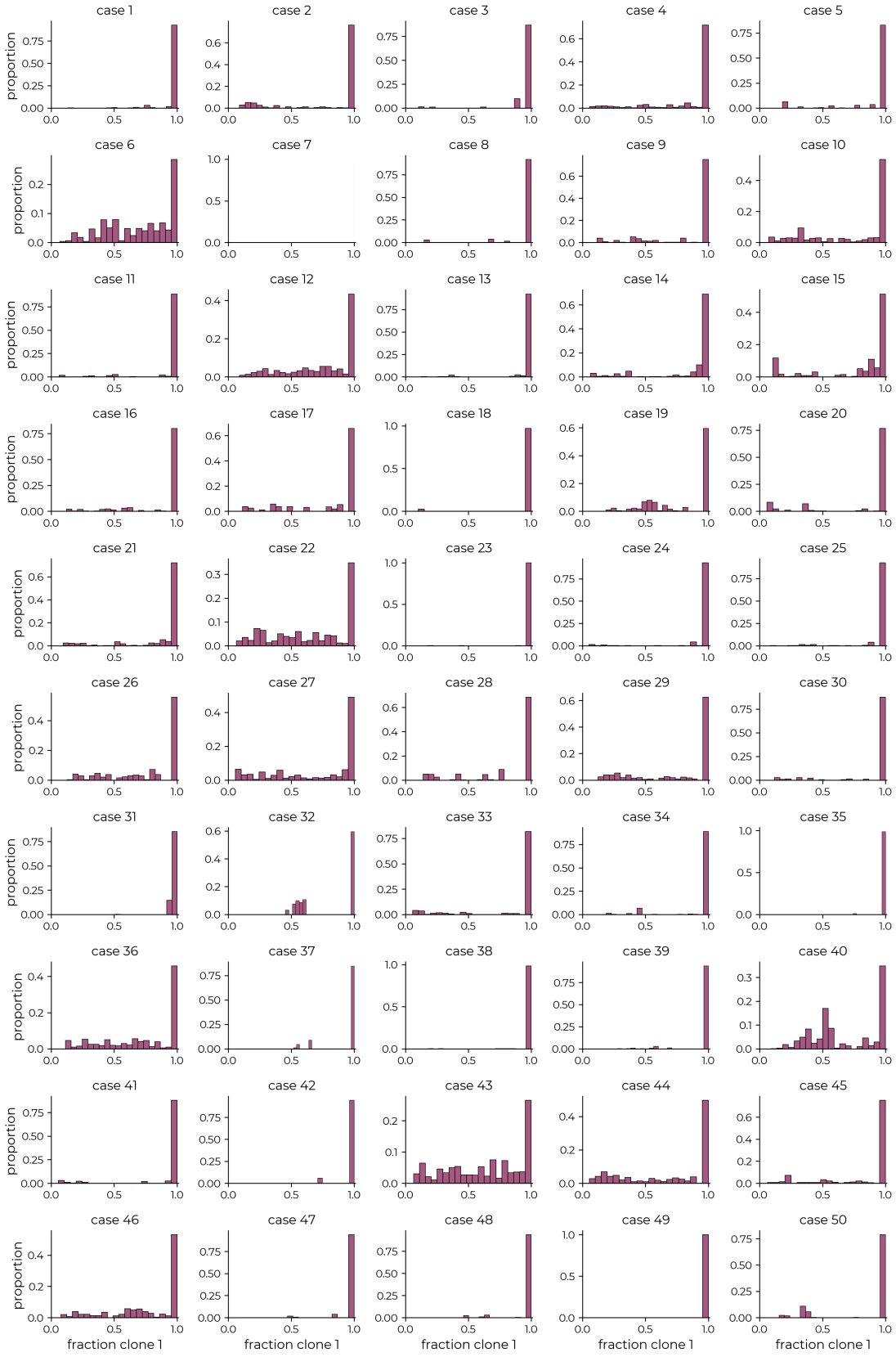


Figure 2.10 Battenberg clonal distribution. Distribution of fraction of sample assigned to clone 1 across loci

2.5.4 Runtime Comparison

Finally, a runtime comparison of different models was performed across a random selection of 5 TCGA samples. Such analysis, however, is not definitive as various factors confound runtime:

- Runtime depends on tumour sample complexity.
- Some methods require user-managed parallelisation (e.g., by chromosome), while others handle this internally.
- Tumour and normal sample processing can be parallelised differently across methods.
- Certain tools (e.g., Battenberg) perform additional steps like phasing, significantly increasing runtime.

The runtime of the workflow implementations is measured following documentation procedures for each method as outlined in **Section 2.3.7**. For ASCAT, Battenberg, CNV Kit, and `araCNA` (preprocessing), 24 CPUs are used. HMM Copy by default does not implement parallelism, so it uses only a single core. Inference for both `araCNA` models was evaluated using an a100 GPU, although `araCNA`-hyena can run on a CPU with a slight runtime impairment.

Figure 2.11 demonstrates that `araCNA` is faster than other methods, and shows that the main bottleneck is the preprocessing (to compute read depth and BAF), which likely could be sped up further, using command line tools rather than pysam (a Python library), as is the default in `araCNA`. Note, this analysis was performed after the primary analysis, and only five samples were used to reduce unnecessary computation.

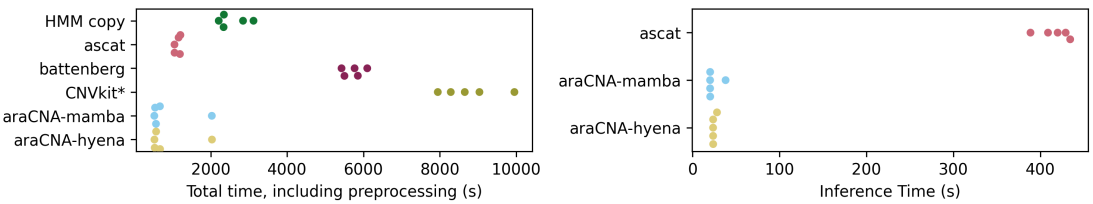


Figure 2.11 Comparison of runtime across models.

2.6 Conclusions

These results show that it is possible to use a deep-learning approach to predict allele-specific copy number alterations from whole-genome sequencing. `araCNA` is trained using a simulation framework that generates supervised data for this biological problem that otherwise would have no known ground truth. The model is applied using zero-shot inference to TCGA cancer samples and found that `araCNA` achieves comparable performance to existing methods despite being trained only on simulated datasets. `araCNA` is also faster than other models, **Figure 2.11**, performing inference in minutes, as it does not have to fit to individual samples. This highlights the utility of having strong mechanistic models that relate copy numbers to sequencing data. Overall, these results demonstrate how domain knowledge can be used to develop simulation training sets for simulation-based inference in biological applications, where computing the posterior may be intractable or computationally intensive [42].

`araCNA` takes advantage of recent advancements in deep learning, such as SSM-like models, that allow for extremely long sequence lengths, on the order of genomic data, and to learn long-range interactions for fast CNA inference. This approach is not limited to CNA calling and could be repurposed for inference in many biological applications where simulations are already used. Differences between Mamba and Hyena highlight that the choice of architecture can be important, and performance can vary between models for a given application.

Importantly, once constructed, it is possible to further refine and retrain a model like `araCNA` with additional data. Unlike classic CNA callers, `araCNA` can continue to learn and improve using standard deep learning techniques such as fine-tuning or transfer learning. While this work was limited to training on simulated data, it is possible to use CNA profiles from existing copy number callers for training or fine-tuning. However, since these would not be ground-truth data, this would result in a model which is an *emulator* of the existing copy number caller. While emulators could not normally exceed the original CNA caller in terms of performance, they can offer usability advantages since, by pre-training the emulator, it can be applied directly at run-time without retraining on each specific sample. There

may also be possibilities for producing ensemble callers by using outputs from multiple existing CNA callers for training.

If the tumour genome is non-identifiable, `araCNA` has been trained to bias towards smaller copy number solutions through sampling bias in the simulated training data. An alternative would be to introduce an explicit penalty term on total copy number in the loss, discouraging high-CN solutions when non-identifiable. However, to do this well, it would be necessary to identify weakly identifiable sampled tumour profiles, which is a complex multivariate optimisation problem. In practice, it was easier to preferentially sample lower-CN profiles for ambiguous settings. However, a selective penalty in the training loss offers an alternative to changing the sampling distribution that may be simpler or more complex, depending on the task.

While deep learning approaches in genomics have previously been limited by a lack of labelled ground truth data and sequence length [54], `araCNA` demonstrates how simulation-based learning in tandem with emerging deep-learning architectures can address both these issues. `araCNA` performs on par with traditional genomic models despite requiring only a tumour sample and not a matched normal sample, contains fewer markers of overfitting, and performs inference in only a few minutes. On top of this, `araCNA` has the ability to learn and adapt using fine-tuning or transfer learning.

2.7 Limitations and future work

The approach here could be extended in several ways. Simulations could explicitly model patterns of sequence variation, due to mappability, amplification or platform sequencing issues. `araCNA` could also be fine-tuned on real genomes, where profiles have been curated and validated using external data sources (e.g flow cytometry for ploidy estimation, or single cell whole genome sequencing). Further, the modelling approach could likely be extended to integrate sub-clonal calling and holistically account for clone-specific mutations and clone-specific copy numbers, or be adapted to include multiple longitudinal samples from the same patient, or for low-quality data or different data modalities (e.g SNP array).

Similarly, `araCNA` could be adapted to estimate copy number profiles in single-cell data by fine-tuning on low-coverage single-cell simulations. An extension of `araCNA` could potentially replace the initial cellular CN calling step in pipelines like CNRein [55]. However, summarising subclonal structure across cellular profiles would require a more complex framework. A similar SSM-based model, taking either raw cellular data or inferred profiles as input to predict representative subclones, could form the basis of future work, particularly given the simulation procedure already introduced in Ivanovic and El-Kebir [55].

A challenge I have highlighted is the issue of identifiability when multiple copy number profiles could explain the same measured sequencing data, and the correct copy number profile might be non-identifiable as a result. Multiple tumour sampling in space and/or time, such as used in TracerX [56], can alleviate the problem by increasing the probability of unique solutions but may also introduce issues since different evolutionary trajectories could also produce similar patterns of CNAs. Further work is required to enable multiple distinct CNA profiles to be produced.

Finally, a potential issue with `araCNA` is the distributional shift between the simulated training data used in training and the actual sequencing data. Given the performance of `araCNA` zero-shot on the real WGS data, this distributional shift is likely small; however, there do exist ways to explicitly account for this. Finetuning on real data using the output of another model could be an option, or else, using something like importance sampling to weight training examples by their similarity to real data samples. This may be a larger issue if `araCNA` was applied in a new context, where there was additional sequencing noise and artefacts (e.g for Formalin-Fixed, Paraffin-Embedded tumour samples).

3

A new method for binary matrix factorisation

Contents

3.1 Introduction

The true motivation for this chapter stemmed from thinking about better approaches for single-cell RNA-sequencing (scRNA-seq) analysis. In particular, scRNA-seq is extremely sparse and near binary, which many existing approaches tend to ignore. Furthermore, a factorisation approach might capture the underlying data in a more interpretable and consistent way than traditional clustering approaches.

This hypothesis led to the development of a novel method for Boolean matrix factorisation using constrained combinatorial optimisation, an approach that draws on my previous background in operations research. Although such problems are often NP-hard, there are still tricks and heuristics that can make them more tractable. The method introduced in this chapter, `bfact`, performs remarkably well, especially at identifying lower rank factorisations, particularly in a scRNA-seq context.

This work runs into both **Chapter 4** and **Chapter 5**, which thoroughly interrogates the underlying assumptions that motivated this chapter, and then investigates Boolean matrix factorisation in a bioinformatics context. However, this chapter is somewhat standalone, since the new Boolean matrix factorisation method applies to any binary context, not just scRNA-seq. Hence, it is introduced here, before the more bioinformatic motivation is outlined.

Parts of this chapter have formed the basis of a publication, currently under review. Further, the initial formulation for the disjoint-BMF and associated pricing problem were devised in collaboration with Dr Michael Forbes- my undegraduate operations research lecturer. The initial idea of a combinatorial optimisation approach, implementation of all approaches and formulation of all other approaches and algorithms is my own.

3.2 Background

3.2.1 Problem definition

The aim of Boolean matrix factorisation (BMF) is to decompose a binary matrix $X \in \{0,1\}^{M\times N}$ into two other low-rank binary matrices, $L \in \{0,1\}^{M\times K}$, $R \in \{0,1\}^{K\times N}$, according to $X_{ij} = \vee_{k=1}^{K} L_{ik}R_{kj}$. Where $\vee$ is the logical or operator, meaning that $X = \min(1, LR)$. Here, (M, N) corresponds to the number of observations and features, respectively and typically $K \ll N$. In practice, observations are corrupted by noise, and it is common to assume that the observations $Y \in \{0,1\}^{M\times N} = X + \epsilon \circ (1 - 2X)$ where $\epsilon \in \{0,1\}^{M\times N}$ is an additive noise matrix, and $\circ$ denotes the hadamard product. The logic $\epsilon \circ (1 - 2X)$ ensures that if $X_{ij} = 1$ that ϵ_{ij} is substracted from X_{ij}, while if $X_{ij} = 0$ then ϵ_{ij} is added to X_{ij}. Hence, in practice with noisy data Y, the aim is to find the lowest rank matrices $\hat{L}$ and $\hat{R}$ that uncover the underlying signal X according to some objective, for example, the reconstruction error on Y.

The exact problem of Boolean matrix factorisation is NP-complete, and its optimisation variant (i.e. minimising reconstruction error for a given rank) is NP-hard, necessitating heuristic or approximate methods in practice [57].

BMF is also closely related to tiling, where "over-covering" is not allowed, that is, the reconstructed matrix cannot be one unless the observed matrix is, i.e $\theta(\hat{L}\hat{R})_{ij} \leq Y_{ij}$. A recent review paper Miettinen and Neumann [57] highlights that there are three equivalent ways to consider BMF. First, as a boolean matrix factorisation, highlighted above; second, using a graph formulation; and third, using a set formulation. The latter becomes relevant in **Section 3.3**, hence is introduced here.

Consider now the matrix Y as a composition of sets of features. Let $\mathcal{N} = \{N_f | N_f = \{j | j \in 1 \dots N\}, f = 1 \dots 2^N\}$, the set of all possible feature sets. For a given observation, there is a collection of observed features, $S_i = \{j | Y_{ij} = 1\} \in \mathcal{N}$. The aim is to select a subset of K sets $\mathcal{R} \subseteq \mathcal{N} = \{Q_k | k = 1 \dots K\}$ to provide an

optimal set basis for Y, given by $\{S_i | i = 1 \dots M\}$. For every observed feature set S_i, there exists a subcollection $\mathcal{L}_i \subseteq \mathcal{R}$ with the following objective:

$$\min \sum_{i=1}^{M} |S_i \oplus \bigcup_{Q_k \in \mathcal{L}_i} Q_k|$$

Here $\oplus$ denotes the symmetric difference between sets. This is equivalent to the standard matrix factorisation, by considering that $L_{ik} = 1$ iff $Q_k \in \mathcal{L}_i$, and that $R_{kj} = 1$ iff $j \in Q_k$.

3.2.2 Existing approaches

There are many existing approaches to Boolean matrix factorisation, developed in the related but distinct contexts of data mining, control, concept analysis and machine learning [57]. However, only a subset of these algorithms also account for finding the best K value, which in practice is necessary and desirable if trying to find underlying factors for the data. I detail the most common and best-performing ones below.

These algorithms tend to fall into two camps: greedy heuristic algorithms that generate candidate patterns often from iterating through columns ordered in some way, or continuous approximations. Exact combinatorial solutions exist that use mixed integer programming (MIP) solvers such as CPLEX and Gurobi; however, these are limited by their scalability [58].

ASSO and MLDBMF

ASSO, [59] was one of the pioneering methods for BMF algorithms. ASSO requires K as an input; however, MDL4BMF, [60] from the same authors, extends ASSO to find the best K that minimises the description length.

ASSO solves the set basis problem highlighted above. It finds possible columns (sets of features) by creating a pairwise correlation matrix, $A : N \times N$, between features and thresholding this correlation. From this matrix, K rows are greedily selected to maximise the cover of the matrix Y, defined as:

$$\begin{aligned} \text{cover}(Y, \hat{L}(k), \hat{R}(k)) =& w^{+} |\{(i,j) | Y_{ij} = 1, (\hat{L}(k)\hat{R}(k))_{ij} = 1\}| \\ & - w^{-} |\{(i,j) | Y_{ij} = 0, (\hat{L}(k)\hat{R}(k))_{ij} = 1\}| \end{aligned}$$

When $w^+ = w^- = 1$, as is default in ASSO, this is equivalent to finding factors that greedily minimise the reconstruction error.

The greedy selection starts with empty matrices $\hat{L}$ and $\hat{R}$. From a given row a_j of A, the corresponding optimal cover membership l_j can be found, and from iterating over all rows, the optimal factor and membership $a_{j'}, l_{j'}$ is selected that maximises the cover. These are then appended to the $\hat{L}$ and $\hat{R}$ matrices until K factors have been chosen. It is greedy because the optimal grouping is considered iteratively and not holistically. It is also heuristic as the model depends on the correlation matrix providing a good basis for candidate columns.

MDL4BMF extends ASSO by running it for multiple different input K values and finding the factorisation with the minimum description length (MDL). Description length hails from information theory, and is a principle for model selection, selecting a model that balances complexity and fit. Complexity here is a measure of the number of bits required to encode the model, and can be done in different ways, as explored in [60]. It implicitly accounts for the number of factors K.

Panda$^+$

Panda$^+$, [61] is a generalisation of the Panda [62] algorithm, by the same authors. It is similar to MDL4BMF in that it generates candidate columns but directly optimises a kind of complexity cost, instead of doing this post-hoc. The internal greedy algorithm also follows a different strategy to ASSO, but also greedily selects patterns with the largest impact on cost. Here, a pattern refers to a pair $\hat{L}_{\cdot k}, \hat{R}_{k\cdot}$. It does this by finding 'core' patterns, that is, patterns with only true positives in the data matrix. It finds the core pattern, if any, that minimises the cost. Once the core pattern is found, it extends this pattern to allow for false positives, again such that this pattern minimises the cost. This process is repeated until a user-specified number of patterns is found or until the MDL cost cannot be improved using this strategy.

Panda finds candidate patterns by first ordering the features in the residual dataset, $Y_r = \min(0, Y - (\hat{L}(k)\hat{R}(k)))$, where $\hat{L}(k)$, $\hat{R}(k)$ is the factorisation at step k in the algorithm (initially empty). Columns can be ordered using various strategies, such as frequency in the data or mutual correlation. Once ordered as $\sigma(1), \ldots, \sigma(N)$, the candidate pattern is constructed by setting each feature entry to 1 in turn, starting with $\hat{R}_{k,\sigma(1)} = 1$ and assigning $\hat{L}_{i,k} = 1$ for all i where $Y_{r;i,\sigma(1)} = 1$. This process continues iteratively for $j \in \sigma(2), \ldots, \sigma(N)$, updating $\hat{L}(k)$ and $\hat{R}(k)$ until no further decrease in cost is observed, before repeating the process for the new residual dataset for $k + 1$. The algorithm stops at $k = K_{\max}$ or if the cost does not improve with the addition of k.

Panda$^+$ allows for different complexity costs alongside the reconstruction error, including an MDL encoding and pattern complexity (number of bits in each profile). The default is adding $|\hat{L}| + |\hat{R}|$ to the reconstruction objective, which encourages sparse $\hat{L}, \hat{R}$.

PRIMP

To move away from heuristic approaches Hess, Morik and Piatkowski [63] take a continuous relaxation of the problem, which can be directly optimised using gradient-based approaches. To do this, they adopt a two-part objective: a term for the reconstruction error (using standard algebra, not Boolean algebra), and a regularisation term on the matrices to encourage binary values and lower rank approximations. Their objective meets the criteria to use proximal alternating linearised optimisation (PALM), a generalisation of Gauss-Seidel for nonconvex, nonsmooth problems [64]. They investigate two regularisation terms, but their better-performing 'PAL-Tiling' approach is detailed here and is referred to as PRIMP throughout this section.

The cost measure in PRIMP is given by the code-table cost:

$$
\begin{aligned}
f_{CT}(\hat{L}, \hat{R}, Y) =& f_{CT}^{\mathcal{D}}(\hat{L}, \hat{R}, Y) + f_{CT}^{\mathcal{M}}(\hat{L}, \hat{R}, Y) \\
f_{CT}^{\mathcal{D}}(\hat{L}, \hat{R}, Y) =& -\sum_{k=1}^{K} |\hat{L}_{\cdot k}| \log(p_k) - \sum_{j=1}^{N} |\epsilon_{\cdot j}| \log(p_{K+j}) \\
f_{CT}^{\mathcal{M}}(\hat{L}, \hat{R}, Y) =& \sum_{k:|\hat{L}_{\cdot k}|>0} (\hat{R}_{k\cdot} c - \log(p_k)) + \sum_{j:|\epsilon_{\cdot j}|>0} (c_j - \log(p_{K+j}))
\end{aligned}
\tag{3.1}
$$

Where ϵ is the error matrix from the reconstruction difference, and the probabilities p_k and p_{K+j} refer to the usage of non-singleton profiles $\hat{R}_{k\cdot}$ and singleton profiles $\{j\}$ (i.e profiles containing only a single feature). These are given by:

$$
p_k = \frac{|\hat{L}_{\cdot k}|}{|\hat{L}| + |\epsilon|}, \quad p_{K+j} = \frac{\epsilon_{\cdot j}}{|\hat{L}| + |\epsilon|}
$$

Further, $c : N \times 1$ is the vector of code lengths for each feature, given by $c_j = -\log(|Y_j|/|Y|)$.

f_{CT} above is an example of a minimum description length (MDL) encoding of the model defined by $(\hat{L}, \hat{R}, Y)$ using code tables. Here, different factor profiles are mapped to binary codes, where shorter codes correspond to profiles that are frequently present in observations [65]. $f_{CT}^{\mathcal{D}}$ refers to the description length of the data given by the model, and $f_{CT}^{\mathcal{M}}$ is the description length of the model itself.

f_{CT} is not continuous, hence PRIMP employs a relaxation of this, which they show is an upper bound on f_{CT}. Further, the algorithm used in PRIMP does not optimise f_{CT} over all K but optimises f_{CT} at incrementally increasing (with Δ_k) values of K up to some user-specified maximum. The factorisation is returned if the binarised optimal matrix factorisation results in a rank less than the input K specified, or if the maximum K has been reached. The optimisation algorithm that solves for each K value is run for some maximum number of iterations I for the PALM solver.

3.2.3 Constrained optimisation

An alternative to gradient-based machine learning methods that explicitly handles discrete variables is combinatorial-based optimisation. Although discrete problems

are often NP-hard, as in this case [57], their difficulty depends on the problem size. Modern discrete optimisation solvers can still exploit parallelism and advanced mathematical techniques to find high-quality, and often guaranteed optimal, solutions within reasonable timeframes [66]. Additionally, problem-specific methods like delayed column generation or Benders decomposition can enable tractable solutions to otherwise intractable formulations [67, 68].

Linear programs (LPs) concern convex optimisation problems with constraints, where the set of constraints forms a feasible region of solutions [69]. In LPs, the optimal solution(s) to the problem lie at extreme points, the vertices of the feasible region. Integer programs (IPs) and mixed integer linear programs (MILPs) are problems where variables can be integer-valued, meaning such problems are no longer convex; however, their linear relaxation is. This special class of problems is relevant in many real-world applications, as they can use decision variables to encode conditionals.

In IPs and MILPs, feasible solutions require integer variables, and the resulting problem is no longer convex. To solve this, a tree-like procedure called Branch-and-Bound is used [70]. If the LP relaxation yields a fractional solution, branching constraints are added to force integrality: one branch enforces the variable to be at most the floor, and the other at least the ceiling of the fractional value [70]. The LP relaxation is then resolved in each subproblem. For example, if the LP relaxation returns a solution with $x = 4.3$, two new subproblems are created, one with the constraint $x \leq 4$, the other with the constraint $x \geq 5$. These are then solved recursively to find the best bound and hence optimal integer solution.

Another important concept in such problems is the idea of the dual problem, which is a reframing of the original 'primal' problem [69]. In the primal problem, an LP aims to find the optimal values of decision variables that minimise the cost, subject to constraints. The dual problem finds optimal values of variables associated with each constraint (dual variables), representing the cost each constraint adds to the primal objective — that is, how much the objective would increase if the

constraint were relaxed. If such a problem is written as a system of equations in matrix form, the primal formulation uses the rows of the matrix, while the dual uses the columns of the matrix (with rhs/objective coefficients as appended column/row) [69].

Delayed column generation

Delayed-column generation (or Dantzig-Wolfe decomposition) is a reformulation technique that can improve performance on MILP problems [71]. The basic premise is that a problem is decomposed into a 'master problem' (MP) and a sub-problem. The MP is formulated in terms of candidate columns (feasible variable solutions)- if all known variable solutions are enumerated, then the MP simply becomes selecting the best solutions for the objective.

Of course, enumerating all possible columns/feasible solutions is usually computationally intractable. Hence, a restricted master problem (RMP) is solved using a smaller set of candidate columns [71]. The subproblem involves finding new columns to add to the RMP that are guaranteed to improve the objective. These candidate columns are injected into the RMP, and the process is repeated until optimality, when no more columns can be added to improve the objective. Such a reformulation into RMP and subproblems can make such problems much more efficient.

To demonstrate this, consider a problem of the form:

$$\min_{y} \; c^T y$$
$$\text{s.t}: \quad Ay \geq a \qquad By \geq b \qquad y \in \mathbb{N}^p$$

Where $c \in \mathbb{R}^p$, $A \in \mathbb{R}^{k \times p}$, $a \in \mathbb{R}^k$, $B \in \mathbb{R}^{l \times p}$ and $b \in \mathbb{R}^l$.

This problem can be decomposed if the second constraint is viewed as a complicating factor. First, consider enumerating all sets of solutions such that $Q = \{y | By \geq b\}$. Let $\lambda_q = 1$ if solution $q \in Q$ is chosen, 0 otherwise. The MP

associated with the original problem is now:

$$\min_{q \in Q} \; c^T q \lambda_q$$
$$\text{s.t}: \quad \sum_{q \in Q} Aq\lambda_q \geq a \qquad \sum_{q \in Q} \lambda_q = 1 \qquad \lambda_q \in \{0, 1\}$$

Because enumerating all possible q initially is intractable, the LP relaxation of the master (i.e $\lambda_q \in \mathbb{R}^+$) is solved using column generation. Hence, an RMP with a subset of columns is first solved. New columns are added by solving the subproblem, also called pricing problem:

$$\min \; (c^T - \pi A)y$$
$$\text{s.t}: \quad By \geq b \qquad y \in \mathbb{N}^p$$

Here, $\pi \in \mathbb{R}^k_+$ represent dual variables of the constraint $\sum_q Aq\lambda_q \geq a$.

While solutions $\bar{y}$ to the pricing problems are negative, they indicate that adding that variable, $q_{\bar{y}}$, to the RMP will reduce the objective value. If no negative solutions exist, then the solution to the LP relaxation of the MP is optimal [71]. To solve the integer MP to optimality, column generation can be used in tandem with branch-and-bound, a method known as branch-and-price.

In cases where a good feasible solution is required (rather than an optimal one), the IP of the RMP can be solved with the final set of columns. If the objective of this IP is the same as the LP relaxation, then the solution is guaranteed to be optimal.

Existing approach for BMF

Kovacs, Gunluk and Hauser [58] use a delayed column generation approach for the BMF problem, leveraging the insight that a rank-K matrix factorisation can be decomposed as the sum of K rank-1 matrix factorisations. They construct a restricted master problem that iteratively selects the K best rank-1 matrices from a set of candidate matrices. They dynamically add advantageous rank-1 matrices to the master problem during optimisation using the associated pricing problem of their RMP. The authors show that this approach achieves the lowest reconstruction error on a number of small datasets (i.e max M = 226, N 94). It is limited to small

datasets as both the number of variables in the master and pricing problems scale with $M \times N$. Further, they do not attempt to find the value K, which is given as a hyperparameter, although it could likely be incorporated into the master objective with some penalty term associated with the number of chosen factors.

3.3 Method

3.3.1 Motivation

Surveying the literature, several points emerge. Scalable methods for binary matrix factorisation typically rely on heuristics or continuous relaxations, but even these approaches have not been tested at the scale of realistic scRNA-seq data (∼100k × 15k). Heuristics often greedily select feature sets using simple metrics such as correlation or frequency, while both heuristic and continuous methods are prone to getting stuck in local minima. Constrained optimisation and exact solutions, meanwhile, are only feasible for smaller matrices.

To address these limitations, I developed an approach, `bfact`, that approximates BMF using a hybrid combinatorial and heuristic approach. `bfact` performs well in simulated data, standard benchmarks, and 14 scRNA-seq datasets from the Human Cell Lung Atlas.

3.3.2 Overview

The developed approach, illustrated in **Figure 3.1**, begins by generating candidate factors through clustering on features. It then solves a warm-started restricted master problem (RMP-w) to approximate the Boolean matrix factorisation using up to K_c of these factors (K_c initialised to some $K_{\min}$). Depending on the selected metric, the method either heuristically reassigns features and prunes factors (`bfact`-recon or `bfact`-MDL) or performs a second combinatorial approach to refine the factorisation (`bfact`-MIP). The process iteratively increases the maximum number of factors, K_c, stopping to give the best factorisation solution if the error metric does not improve within s_i steps. This two-stage framework—starting from disjoint candidate factors and refining via heuristic or optimisation—echoes the ASSO

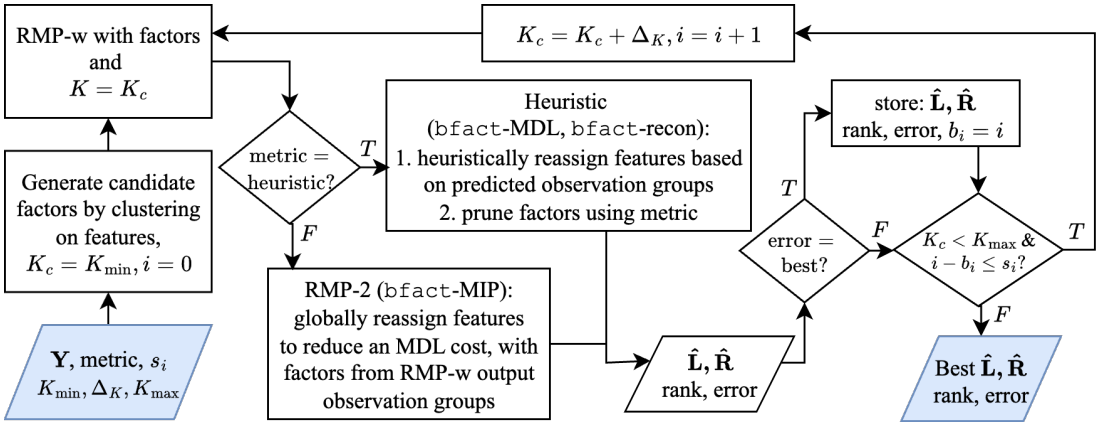


Figure 3.1 Overview of `bfact`

methodology while incorporating modern optimisation techniques. Below, each of these steps is explained in detail.

3.3.3 Master problem for approximate BMF

Here, an MP is defined that approximates the BMF by finding sets of (mostly) disjoint features that best explain the observations. For the observed binary matrix Y, consider the set $\mathcal{A}$ of all possible non-zero feature-sets, or factors, α, where $\delta_\alpha \in \{0,1\}^N$ and $|\mathcal{A}| = 2^N - 1$. Define also $c_{i,\alpha}$, the cost for observation i of choosing factor α, as:

$$C_{i,\alpha} = \left(\sum_{j|(i,j)\in E} \delta_{\alpha,j}, \sum_{j|(i,j)\notin E} \delta_{\alpha,j} \right) \tag{3.2}$$

$$c_{i,\alpha} = \min C_{i,\alpha} \tag{3.3}$$

$$l_{i,\alpha} = \arg\min C_{i,\alpha} \tag{3.4}$$

where $E = \{(i,j)|Y_{i,j} = 1\}$. The left hand side of $C_{i,\alpha}$ is the intersection of features in an observation and in a factor. The right-hand side can be thought of as the complement of an observation - that is, the features included in a factor that are not present in the observation.

The initial MP is thus defined as:

$$\min \sum_j u_j \sum_i Y_{ij} + \sum_\alpha z_\alpha \sum_i c_{i,\alpha} \tag{3.5}$$

$$\sum_{\alpha \in \mathcal{A}} z_\alpha \leq K \tag{3.6}$$

$$\sum_{\alpha \in A} \delta_{\alpha,j} z_\alpha + u_j \geq 1 \tag{3.7}$$

Where $u_j \in \mathbb{R}^+, \forall j = 1, \ldots, N$, and $z_\alpha \in \{0, 1\}\ \forall \alpha \in \mathcal{A}$.

Constraint 3.6 allows at most K profiles to be selected. **Constraint 3.7** ensures every feature is accounted for, either in at least one of the selected factors or by the variable u_j otherwise. If this constraint were an equality, then the above formulation gives the optimal disjoint factorisation (i.e no factors can share overlapping features), and the associated cost is the reconstruction error. The inequality relaxes this assumption, but note that the cost still implicitly favours disjointness, encouraging $\sum_{j=1}^{N} \delta_{\alpha_1,j}\delta_{\alpha_2,j} = 0$ if $z_{\alpha_1} = z_{\alpha_2} = 1$, explored further in **Section 3.3.4**.

Hence, the solution to the MP gives an approximate, (mostly) disjoint, BMF with $\hat{R}_d = \{\delta_{\alpha,j} | z_\alpha = 1\}$, and $\hat{L}_d = \{l_{i,\alpha} | z_\alpha = 1\}$. Note, this is not the same as a clustering approach on the features, as u_j allows any number of features to be excluded from selected factors.

3.3.4 Implicit preference for disjoint factors

Figure 3.2 illustrates why the cost in the master problem penalises chosen factors with shared features, favouring an orthogonal or disjoint selection of factors.

In **Figure 3.2**, the lowest-cost solution contains three candidate factors despite the two true patterns existing in the candidate factors. This illustrates that the disjoint set-partitioning formulation and cost do not favour the lowest rank reconstruction.

However, for the optimal feature set, s_1, only the first and third columns of the membership matrix differentiate the three groups, and these have the same pattern of membership (L) as all the other sets, where $K = 2$. After identification of the best disjoint factors, if observations containing the same factor are grouped, then

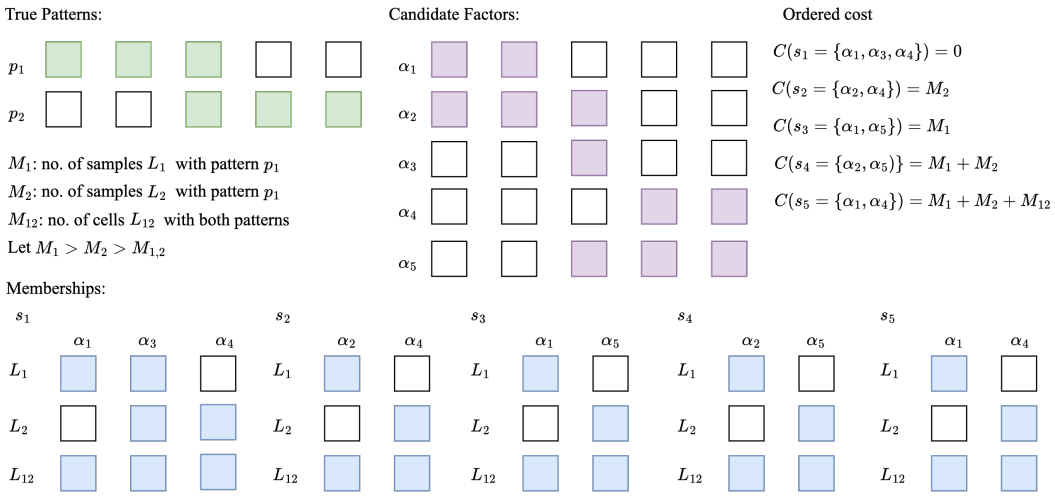


Figure 3.2 Cost favours disjoint factors. Illustration of why cost implicitly favours non-orthogonality- the lowest-cost solution contains three candidate factors despite the two true patterns existing in the candidate factors.

the true features corresponding to this grouping capture the underlying patterns p_1, p_2, and the unnecessary factor α_3 can be dropped. This is an illustrative example, but **Section 3.3.7** introduces formal algorithms for these post-processing steps.

3.3.5 Warmstarted restricted master problem (RMP-w)

Solving the MP above would require enumerating all possible factors $\mathcal{A}$, exponential in the number of variables and intractable. However, instead, a delayed column generation can be used to solve the restricted master problem (RMP) [67]. The RMP is the master problem 'restricted' to a subset of possible factors, $\mathcal{A}' \subseteq \mathcal{A}$, where typically $|\mathcal{A}'| \ll |\mathcal{A}|$, making it tractable to solve. Factors that will reduce the objective are iteratively added to the RMP by solving the pricing problem (based on dual variables from the RMP linear relaxation) [71]. Eventually, no more factors (i.e columns) can be added that will reduce the objective, and the global optima of the linear relaxation is found, without having to realise the full MP linear relaxation. To solve the integer MP to optimality, a branch-and-price approach can be taken, again without having to realise the full MP.

Such an approach can work even when the RMP starts with an empty factor set. However, it is often faster to warmstart the RMP with *a priori* candidate factors.

A benefit of the disjoint formulation is that clustering on the features provides a simple way to generate good candidate factors. To generate candidate factors for this RMP (RMP-1), hierarchical clustering is performed across features, based on their pairwise hamming distance, and the hierarchical tree is cut at several different levels. Each resultant cluster from each level is taken as a candidate factor. Leiden community detection is also performed at different resolutions (a hyperparameter), which initially constructs a K-nearest-neighbour graph, again based on hamming distance. The union of the clustered features is taken as the set of candidate factors, hence $\mathcal{A}' = \{\text{candidate columns}\}$. Together with RMP-1, this is termed RMP-w.

Note, RMP-1 scales (variables, constraints) with $(|\mathcal{A}'| + N,\ N)$. This is more computationally tractable than the exact approach of Kovacs, Gunluk and Hauser [58], which scales with $(\rho MN + |\mathcal{D}'|,\ \rho MN)$, where $\mathcal{D}'$ is the restricted set of all rank-1 $M \times N$ matrices, and ρ is the density of ones in the data matrix.

3.3.6 Pricing Problem

To solve the (linear relaxation of the) disjoint-BMF to optimality, the pricing problem can be used to iteratively add informative feature sets to the RMP. The pricing problem (PP) for the restricted master problem is given by:

$$\min \sum_{i=1}^{M} t_i - \sum_{j=1}^{N} \pi_j^* x_j - \gamma^* \tag{3.8}$$

$$\text{s.t} \quad t_i = \min \left(\sum_{j|(i,j)\in E} x_j,\ \sum_{j|(i,j)\notin E} x_j \right) \tag{3.9}$$

$$x_j \in \{0,1\}, \qquad \forall j \in \{1 \dots N\}$$

$$t_i \in \mathbb{R}^+, \qquad \forall i \in \{1 \dots M\}$$

In practice, to model linearly, **Constraint 3.9** requires four constraints to implement, using a switch binary variable s_i for each M. Here π_j^* is the value of the dual variable of **Constraint 3.7** at the optimal solution of the restricted master problem and γ^* the value of the dual of **Constraint 3.6**. The PP scales (variables, constraints) with $(M + N, M)$.

I first attempted to solve the RMP-w using a delayed column generation approach, like Kovacs, Gunluk and Hauser [58]. Here, the pricing problem struggled to reduce the linear relaxation objective in a reasonable time frame (see **Appendix B.1**), suggesting that the solution to RMP-w already produces good-quality disjoint factorisations. Hence, using RMP-w alone was taken as a first step in `bfact`.

3.3.7 Two-step BMF

RMP-w does not directly solve for a BMF, however, using the intuition from **Section 3.3.4**, it may provide a good basis for a BMF after adding some post-processing. Hence, a two-step approach is adopted. Intuitively, the first step, RMP-w, selects (mostly) disjoint pregenerated feature sets to find likely observation sets, and the second step uses these observation sets to update the feature sets, allowing them to share features. The second step also controls the model complexity to select lower-rank approximations, where appropriate.

Based on this, two approaches were implemented; the first using a second MIP and the second using a heuristic algorithmic approach.

Two-step MIP-BMF:

Consider now the formulation of a second restricted master problem (RMP-2), similar to the above, but instead for (nearly) exact BMF. Let $\mathcal{B}$ be the set of all factors, now with the set of observations, $\delta_\beta \in \{0,1\}^M$ associated with each factor, $\beta \in \mathcal{B}$. RMP-2 is defined to be over a restricted factor set $\mathcal{B}' \subseteq \mathcal{B}$, given by:

$$\begin{aligned}\min \sum_{(i,j)\in E}(1-e_{i,j}) + \sum_{(i,j)\notin E}\sum_{\beta\in\mathcal{B}'}\delta_{\beta,i}r_{\beta,j} \\ + \sum_{\beta\in\mathcal{B}'}\sum_{i=1}^{M} z_\beta\delta_{\beta,i} + \sum_{\beta\in\mathcal{B}'}\sum_{j=1}^{N} r_{\beta,j}\end{aligned} \tag{3.10}$$

subject to

$$\sum_\beta \delta_{\beta,i}r_{\beta,j} \geq e_{i,j}, \quad \forall\, i,j \in E \tag{3.11}$$

$$\sum_j P_{\beta,j} \leq N z_\beta, \quad \forall\, \beta \in \mathcal{B}' \tag{3.12}$$

$$\sum_\beta z_\beta \leq K \tag{3.13}$$

where $z_\beta \in \{0,1\}, \forall\ \beta \in \mathcal{B}'$, $r_{\beta,j} \in \{0,1\}, \forall\ \beta \in \mathcal{B}'$, $j = 1 \dots N$ and $e_{i,j} \in \{0,1\}, \forall\ i,j \in E$. Here again $E = \{(i,j)|Y_{ij} = 1\}$.

The first two terms in **Objective 3.10** capture false negatives and positives, respectively. The third and fourth terms are proxies for the complexity of the model and correspond to $|\hat{L}|$ and $|\hat{R}|$, these regularise the model to select fewer factors, and sparser $\hat{L}$ and $\hat{R}$ (derived from chosen factors). This is similar to the regularisation taken in Panda$^+$ [61].

Constraint 3.11 uses the term $e_{i,j}$ to allow for boolean logic, restricted to non-zero elements of Y to reduce the number of variables in the model. Hence, Boolean logic is encoded only for true positives, and overlaps at false positives are penalised more than in exact BMF. This could easily be remedied by using $e_{i,j}$ for all elements in the matrix, at the expense of scalability.

RMP-2 scales (variables, constraints) with $(N(\rho M + |\mathcal{B}'|),\ \rho MN + |\mathcal{B}'|)$ where ρ is the density of the matrix. A column generation approach could be used for the above, but is likely to be intractable for larger problems given both its scaling and that of the associated pricing problem. It could also be solved similarly to RMP-w, using candidate factors. However, using candidate factors generated by a clustering approach would likely not capitalise on the added expressiveness of this formulation, given that such candidates are disjointly derived.

Instead, RMP-2 can be used as a second step after finding a solution to RMP-w, to refine the feature sets of factors (i.e $\hat{R}_d$) and chosen factors. Here, candidate factors for $\mathcal{B}'$ are taken as the unique set of observation memberships across the selected factors of the solved RMP-w, with $\delta_\beta,\ \beta \in \mathcal{B}'$ given by the unique columns of $\hat{L}_d$. Hence, RMP-2 selects known observation factor groups found with RMP-w, and reassigns features to them based on these groups, removing any redundant factors through the regularisation terms.

Given $|\mathcal{B}'| \leq K$ in the RMP-2 approach, it was found to be computationally tractable for larger matrices ($\sim$100k x 20k, $\rho \approx 0.1$), but with much heavier memory requirements (approx. 400GB). Adapting the above formulation to be exact (given the factors) would require $e_{i,j}$ to cover all locations, intractable given

Algorithm 1 Reassign Features

function REASSIGN($Y, \hat{L}, \hat{R}$, metric, Δt)
  $I_{kj} = \sum_i L_{ik} Y_{ij}, \quad I = \{I_{kj}\}_{K \times N}$
  $N_k = \sum_i L_{ik}, \quad N = \{N_k\}_K$
  $\hat{R}_b \leftarrow \hat{R}$
  $E \leftarrow$ ERROR($Y, \hat{L}, \hat{R}$, metric)
  for $t = 0$ **to** 1 **step** Δt **do**
    $\hat{R}_t = (I > \frac{1}{2} N(1+t)) \mid \hat{R}$
    $E_t =$ ERROR($Y, \hat{L}, \hat{R}_t$, metric)
    if $E_t < E$ **then**
      $\hat{R}_b \leftarrow \hat{R}_t$
      $E \leftarrow E_t$
    end if
  end for
  return $\hat{R}_b$
end function

$\sim 10\times$ the number of variables. Given this, I explored a less memory-intensive, heuristic approach.

Two-step Heuristic-BMF

By grouping observations containing the same factor(s), additional features present in the grouping can be determined and added to that factor's feature set. This is what RMP-2 does, globally based on the RMP-w derived observation sets (and using a basic complexity cost).

To allow features to be allocated to multiple factors, the heuristic approach performs a coarse grid search to identify at which global proportion of representation in an observation group a feature should be added to a factor. This optionally uses either the reconstruction error or MDL loss, as defined in [63] and introduced in **Section 3.2.2**, which accounts for the sparsity, hence implicitly the rank, of the factorisation. The approach is formalised in Algorithm 1.

Following the reassignment of features, redundant factors, or factors that result in marginal improvement of the loss, are greedily removed. To remove a factor with reconstruction error, the error without the factor must be some minimum percentage f of the reconstruction error with the factor. The MDL loss already

Algorithm 2 Iteratively remove factors

function PRUNE($Y, \hat{L}, \hat{R}$, metric, $f = 1$)
  $M, K \leftarrow \dim(\hat{L})$
  $E \leftarrow$ ERROR$(Y, \hat{L}, \hat{R})$
  while $K > 0$ **do**
    $E_{\text{ls}} = []$
    for $r = 1 \dots K$ **do**
      $\hat{L}_c \leftarrow \hat{L}, \hat{R}_c \leftarrow \hat{R}$
      $\hat{L}_c[:, r] \leftarrow 0, \hat{R}_c[r, :] \leftarrow 0$
      E_{ls}.append(ERROR$(Y, \hat{L}, \hat{R}$, metric$)$)
    end for
    $b_c \leftarrow \arg\min E_{\text{ls}}$
    if not $E_{\text{ls}}[b_c] \leq fE$ **then**
      break // no better sln
    end if
    $\hat{L} \leftarrow \hat{L}$.delete(col $= b_c$)
    $\hat{R} \leftarrow \hat{R}$.delete(row $= b_c$)
    $K \leftarrow K - 1$
    $E \leftarrow E_{ls}[b_c]$
  end while
  return $\hat{L}, \hat{R}$
end function

accounts for removing a factor through reduced complexity (so $f = 1$). This is formalised in Algorithm 2.

3.3.8 Finding optimal rank

Under the disjoint factorisation master problem, RMP-w will achieve a lower overall objective when K is overspecified, as demonstrated in **Figure 3.2**. Although the postprocessing should combine or remove redundant features, the initial rank specification will affect downstream reconstruction, reassignment and feature pruning. Hence, the process is performed over multiple initial ranks and the best result is selected (noting the RMP-w can be initialised once, and easily updated with different K values, for which it is very fast to solve). If increasing the rank does not result in a better solution for s_i iterations, the algorithm is stopped early. This is formalised in Algorithm 3.

Hence, `bfact` comprises the sequential pipeline of w-RMP followed by the second

step (RMP-2 or heuristic) followed by algorithm 3. `bfact`-MIP is this pipeline where the second step is RMP-2 (as w-RMP and RMP-2 are both MIPs). `bfact`-recon is the pipeline where the second step is heuristic (algorithms 1 and 2) with reconstruction error, while `bfact`-MDL is the same using the MDL cost.

Algorithm 3 Select best rank over multiple K

function PIPELINE($Y, K_{\min}, K_{\max}, \Delta K, \text{metric}, f = 1, s_i = 2$)
  $A = \text{GENCOLS}(Y)$
  $V_b = \text{null},\ E_b = \text{null}$
  $b_i = 0, i = 0$
  for $K_c = K_{\min}$ **to** $K_{\max}$ **step** ΔK **do**
    $\hat{L}, \hat{R} = \text{RMP}_1(Y, A, K_c)$
    if metric **is** mip **then**
      $\hat{L}, \hat{R} \leftarrow \text{RMP}_2(Y, \hat{L}, K_c)$
    else
      $\hat{R} \leftarrow \text{REASSIGN}(Y, \hat{L}, \hat{R}, \text{metric})$
      $\hat{L}, \hat{R} \leftarrow \text{PRUNE}(Y, \hat{L}, \hat{R}, \text{metric}, f)$
    end if
    $E_k \leftarrow \text{ERROR}(Y, \hat{L}, \hat{R}, \text{metric})$
    if E_b **is** null **or** $E_k \leq f^{K_c - K(V_b)} E_b$ **then**
      $b_i \leftarrow i, E_b \leftarrow E_k,\ V_b \leftarrow (\hat{L}, \hat{R}, K_c)$
    end if
    if $i - b_i > s_i$ **then**
      break // stop early
    end if
    $i += 1$
  end for
  return V_b
end function

3.3.9 Inverse Problems

Both RMP-1 and RMP-2 can be repurposed to solve inverse problems. That is, solving for the optimal $\hat{L} \in \{0, 1\}$ if $X \approx \theta(\hat{L}R)$ and X, R are known.

In particular, in RMP-1, if **Constraint 3.7** is an equality, selected factors of R are disjoint (no shared genes), this gives $X \approx \theta(LR) = LR$. If the candidate factors supplied to RMP-1 are taken as the rows of R, then the solution to RMP-1 gives $\hat{L}$ that minimises the reconstruction error between X and $\hat{L}R$.

Similarly, for RMP-2, if the objective terms related to $|L|$ and $|R|$ are removed, and $e_{i,j}$ were modelled for each matrix element, and the candidate factors were taken as the rows of the known R, then the solution to RMP-2, gives the optimal inverse Boolean factorisation, $X \approx \theta(\hat{L}R)$ for known X, R for maximum specified K.

For both of these problems, the transpose can be taken to find optimal $\hat{R}$ for known X, L.

3.4 Results

3.4.1 Data

Simulation

Several simulation setups are considered to benchmark `bfact`, including those with different matrix sizes, underlying ranks, noise levels and data density. Some rows and columns were also generated to be 'nuisance' variables, imitating realistic datasets where some features or observations are not relevant to the factorisation.

Formally, the main simulation set-up is as follows:

$$\begin{aligned}
\text{inputs:} \quad & M, N, k, q_l, q_r, v_i, v_j, p^+, p^- \\
L \sim \{\text{Ber}(q_l)\}_{M\times k} \qquad & R \sim \{\text{Ber}(q_r)\}_{k\times N} \\
n_i \sim \{\text{Ber}(v_i)\}_M \qquad & n_j \sim \{\text{Ber}(v_j)\}_N \\
L[n_i > 0, .] = 0 \qquad & R[., n_j > 0] = 0 \\
X = LR \qquad & \epsilon^{\pm} \sim \{\text{Ber}(p^{\pm})\}_{M\times N} \\
Y = X + \epsilon^+[X = 0] - \epsilon^-[X > 0] &
\end{aligned}$$

where $\text{Ber}(\alpha)$ represents the Bernoulli distribution with probability α. For each simulated experiment, five replicates are taken.

Note, it is possible that for extremely sparse sampling, the true rank is lower than specified; however, the effect of this is assumed to be negligible and would likely be captured by measured algorithms.

Real-world datasets

Several real-world datasets are considered. These include binarised versions of the Chess and Mushroom UCI datasets [72] and two versions of the MovieLens 10M dataset, [73], where rows are users and columns are movies, with entries the star rating given by a user to a movie. Following Hess, Morik and Piatkowski [63], set $Y_{ij} = 1$, if a user rates a movie with more than 3 stars. This constitutes the larger dataset; a smaller dataset is also taken by filtering to select users who recommend more than 50 movies and movies that receive at least 5 recommendations. While this follows precedent in previous publications, it should be noted that in some of these datasets, the data has been binarised from discrete, categorical data using one-hot encoding. While such transformations enable the use of Boolean matrix factorisation, it is unclear whether they truly reflect the nature of the data.

The other datasets used are scRNA-seq data from the Human Lung Cell Atlas (HLCA), [74], which consists of 14 separately measured datasets on lung-derived cell types. The cleaned raw counts for each dataset are binarised based on zero/non-zero values in the data. Genes are removed that are not expressed in at least 0.5% of cells, and cells are removed that expressed fewer than 200 genes and more than 10,000. The size and density of each example are included in **Table 3.1**.

Origin	Dataset	M	N	Density
UCI	Chess	3196	75	0.493
UCI	Mushroom	8124	119	0.193
Movie Lens	Movies	29 980	9144	0.018
Movie Lens	Movies Big	69 878	10 677	0.008
HLCA	Banovich Kropski 2020	121894	14495	0.101
HLCA	Barbry Leroy 2020	74484	15047	0.102
HLCA	Jain Misharin 2021 10Xv1	12422	13423	0.124
HLCA	Jain Misharin 2021 10Xv2	33135	13392	0.094
HLCA	Krasnow 2020	60982	15139	0.133
HLCA	Lafyatis Rojas 2019 10Xv1	2921	11943	0.073
HLCA	Lafyatis Rojas 2019 10Xv2	21258	13818	0.117
HLCA	Meyer 2019	35554	14153	0.103
HLCA	Misharin 2021	64842	15938	0.157
HLCA	Misharin Budinger 2018	41219	14057	0.136
HLCA	Nawijn 2021	70395	15579	0.119
HLCA	Seibold 2020 10Xv2	12127	15718	0.215
HLCA	Seibold 2020 10Xv3	21466	17825	0.310
HLCA	Teichmann Meyer 2019	12231	14855	0.150

Table 3.1 Dataset statistics

3.4.2 Evaluation metrics

For simulated data, the predicted rank is compared to the true underlying rank, as is the F_1 score of the true signal matrix $X = LR$ compared to the predicted signal matrix $\hat{X} = \hat{L}\hat{R}$. It is important to evaluate these in tandem, as methods with higher-rank predictions can overfit to noise or find less representative factors that are harder to interpret. For real data, where X is unknown, the F_1 score is computed for the observed matrix Y, while considering the predicted rank, K.

Note, MDL costs are not used for comparison as these favour a sparse decomposition, which does not necessarily align with a BMF, particularly when there are overlapping features per factor. Most MDL costs do not explicitly account for reduced rank, and even when they do, sparsity is preferred. Hence, even though MDL costs are a good proxy for BMF, the compression size of X is not necessarily aligned with the smallest rank approximation of X. This is explored below for the MDL code table cost.

Limitations of MDL Code table cost

The MDL Code table cost, introduced in **Section 3.2.2**, favours disjoint representations, leading to sparsity in the left and right decomposed matrices. To demonstrate, consider two true underlying factors- let each factor have N_1, N_2 unique features, and they share N_{12} features. Also, let the number of observations that contain only one of each factor be M_1, M_2, and the number that includes both be M_{12}. For simplicity, let $M_1 = M_2 = M_{12}$. Consider now two scenarios- the first, where true factors are recapitulated, and the second, where three factors are used, each corresponding to the unique features of true factors 1 and 2, and a third factor for their shared features. Here, $\epsilon = 0$ as both these decompositions exactly reconstruct the input data Y. Then the MDL cost in the two instances are:

$$\begin{aligned} f_{CT_1} =& -(M_1 + M_{12}) \log \frac{M_1 + M_{12}}{M'} - (M_2 + M_{12}) \log \frac{M_2 + M_{12}}{M'} \\ & - N_1 \log \frac{N_1}{N'} - N_2 \log \frac{N_2}{N'} - 2N_{12} \log \frac{N_{12}}{N'} \\ =& -4M_1 \log 2/3 - N_1 \log \frac{N_1}{N'} - N_2 \log \frac{N_2}{N'} - 2N_{12} \log \frac{N_{12}}{N'} \qquad (3.14) \\ f_{CT_2} =& -M_1 \log \frac{M_1}{M'} - M_2 \log \frac{M_2}{M'} - M' \log \frac{M'}{M'} - N_1 \log \frac{N_1}{N'} \\ & - N_2 \log \frac{N_2}{N'} - N_{12} \log \frac{N_{12}}{N'} \\ =& -2M_1 \log 1/3 - N_1 \log \frac{N_1}{N'} - N_2 \log \frac{N_2}{N'} - N_{12} \log \frac{N_{12}}{N'} \qquad (3.15) \end{aligned}$$

Where $N' = N_1 + N_2 + N_{12}$ and $M' = M_1 + M_2 + M_{12} = 3M_1$.

Taking the difference between the higher rank decomposition (i.e with three factors), and the lower rank, gives:

$$\begin{aligned} f_{CT_2} - f_{CT_1} &= -2M_1 \log 1/3 + 4M_1 \log 2/3 + N_{12} \log \frac{N_{12}}{N'} \\ &= -2M_1 \log 1/4 + N_{12} \log \frac{N_{12}}{N'} \end{aligned}$$

Plotting the surface, $M_1 = \frac{N_{12}}{2 \log 1/4} \log \frac{N_{12}}{N'}$, gives the values for M_1 above which the higher rank matrix has a lower MDL cost than the lower rank matrix, **Figure 3.3**.

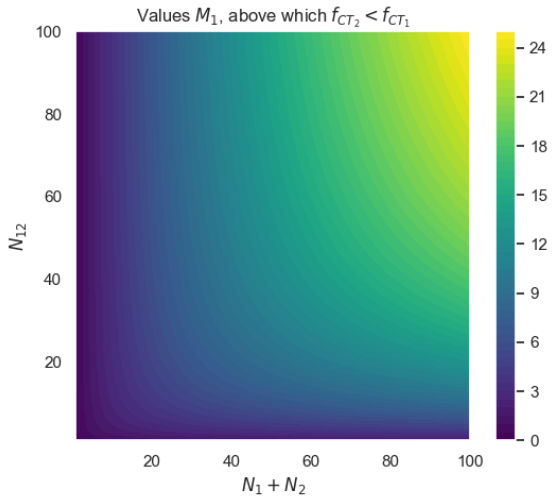


Figure 3.3 MDL cost does not necessarily find optimal rank. Demonstration of how higher rank representation can have a lower MDL cost than the true lower rank representation.

The intuition behind this is that higher-rank representations are sparser, as the repeated features are not included in both factors in the right matrix $\hat{R}$. The slight increase in density in the left matrix (for associating an observation with another factor) $\hat{L}$ is not enough to offset the shared features. The same issue applies to the MDL regularisation given by $|\hat{L}|$ and $|\hat{R}|$.

3.4.3 Method Implementation Details

PRIMP For benchmarks, PRIMP was run for 50000 steps, following what the authors did in [63], and used a $\Delta_k = 5$, from 5 to 100. PRIMP was implemented on simulations with access to 6 CPUs, and 1 NVIDIA GPU (of varying specifications, mostly Quadro RTX 8000 or P100 SXM2).

For the real data, PRIMP was run for 50000 steps, used a $\Delta_k = 10$, from 10 to 100 (the method stops at $\max_K + \Delta_k$). Again, it was run on machines with access to 6 CPUs and 1 NVIDIA GPU.

PANDA$^+$ The PANDA$^+$ documentation provides little guidance on what hyperparameters to use. All were tested, and the best combination was chosen, which was a frequency strategy and a type 1 cost. It was run with a maximum K of 100 for both simulated and real scenarios.

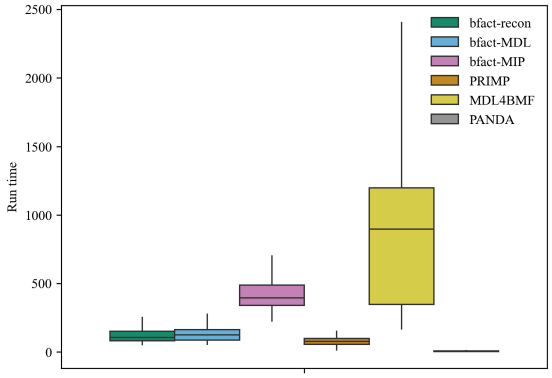


Figure 3.4 Run time across simulations

MDL4BMF MDL4BMF takes longer than the other methods to run, and simulations were implemented with 12 CPUs (double other methods). Hyperparameters were used following the README example given in the MDL4BMF code- with 10 threshold parameters and all error measures. For real data, each dataset had access to 24CPUs, terminating if the model had not completed within 2 days.

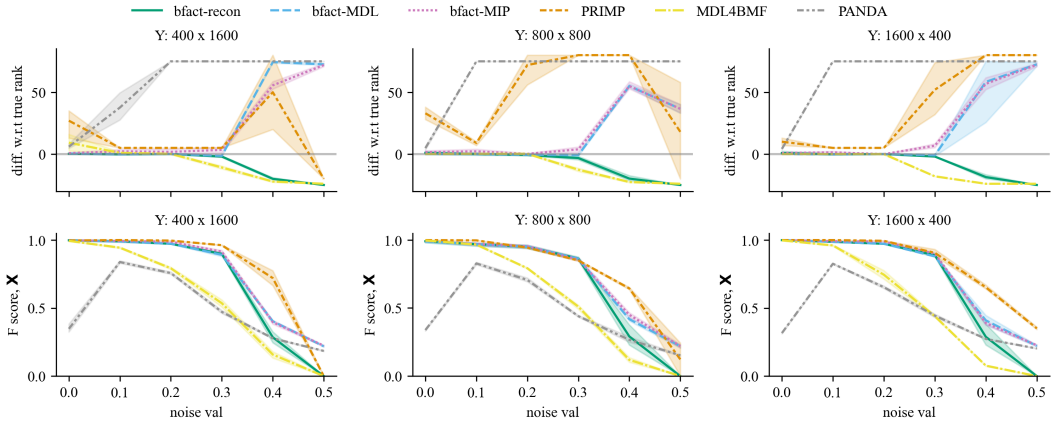
bfact For simulations, `bfact` was implemented on 6 CPUs. For matrices lower than a certain size, $M \times N < 5e6$, the matrix was transposed if $N < M$, while $\Delta_k = 10$, from 10 to 100. For the reconstruction procedure, a constant $f = 0.997$ was used.

For real data, `bfact` was implemented with 12 CPUs. For the real data, the formula $f = \min(1 - 1/\min(M, N), 1 - 1/(\rho \max(M, N)))$ truncated at three decimals was found to work well for reconstruction error.

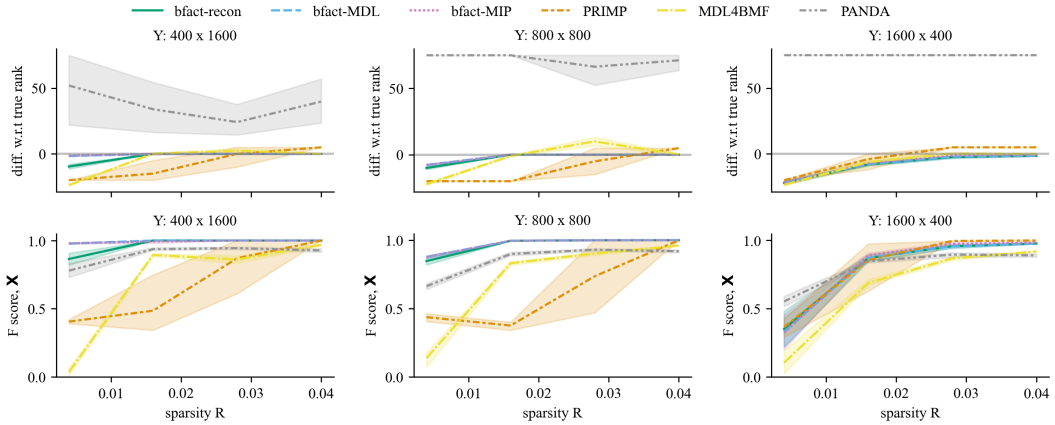
3.4.4 Simulated Results

Figure 3.5, **Figure 3.6**, **Figure 3.7** show that the three variations of `bfact` perform similarly in both F1 score and rank estimation across simulated regimes. Panda consistently overestimates rank with lower F1 scores with respect to signal matrix X, although it requires less computational time than other methods, **Figure 3.4**. PRIMP generally achieves a higher F1 score but has variable rank estimation and does poorly when the sparsity of R is low. MDL4BMF does well at estimating rank but does worse in F1 scores. It is much slower than other methods to run despite being provided double the number of CPUs, **Figure 3.4**. All methods perform worse at higher density. On simulated data, `bfact` has a comparable but

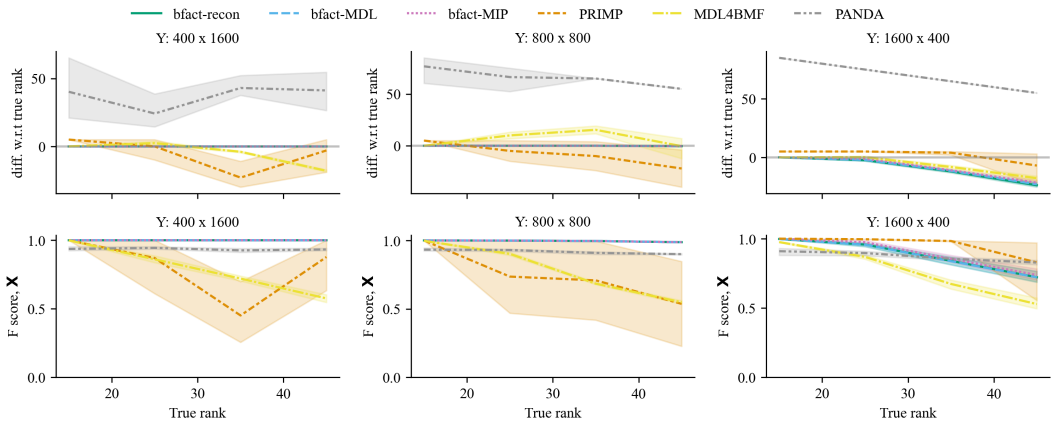
slightly slower run-time to PRIMP, **Figure 3.4**. Interestingly, `bfact`-MIP does slightly worse at recovering the true rank, **Figure 3.6**, **Figure 3.7**. Likely, this is due to the regularisation approach taken, $|\hat{L}| + |\hat{R}|$, which encourages sparse, not necessarily low-rank reconstruction. This is supported by the fact that it has as high F-scores as the other bfact approaches. It could also be due to the BMF approximation of its formulation (where overlapping false positives are penalised more heavily). Given `bfact`-MIP is also slower and more memory-intensive, it is not explored in the real data. For both `bfact`-recon and `bfact`-MDL, the time-limiting factor is the heuristic post-processing, rather than the combinatorial RMP problem.



(a) Varying noise, $p^+ = p^-$, $ql = qr = 0.1, r = 25$

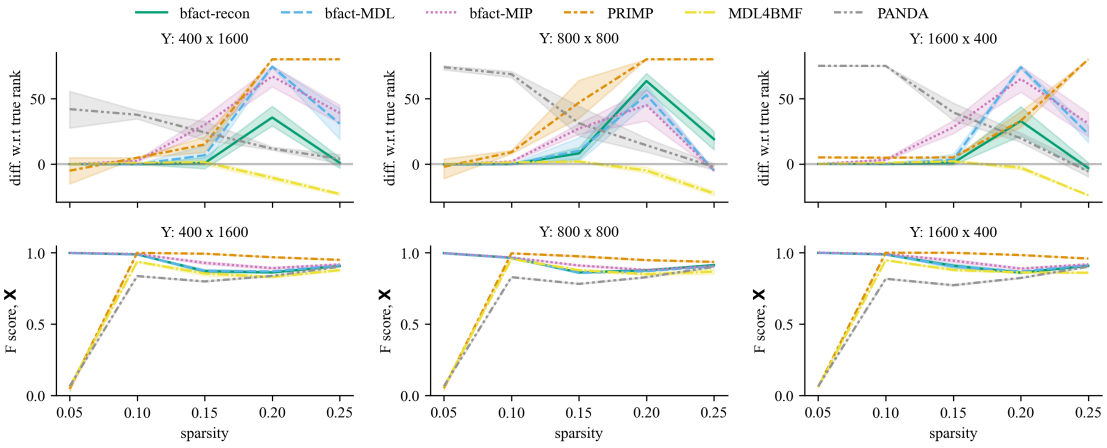


(b) Varying q_r, for low right sparsity, simulating real RNA seq.

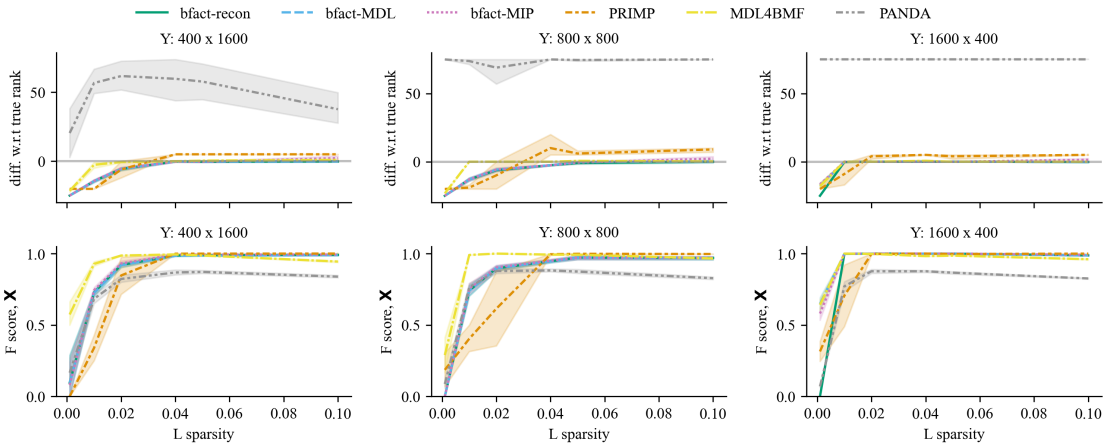


(c) Varying number of factors, controlling for density, $ql = 0.1$, $q_r = 0.1n/r, p^{\pm} = 0.1$

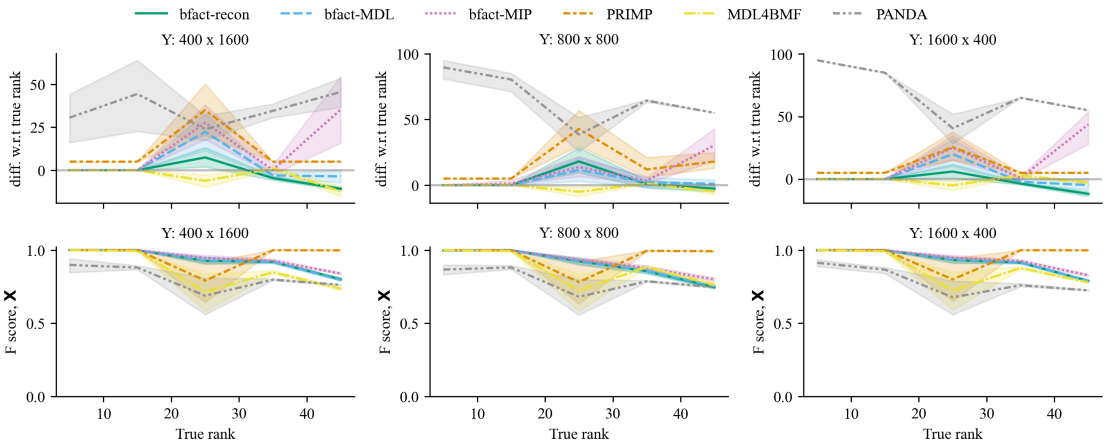
Figure 3.5 Simulation results



(a) Varying total sparsity, $p^{\pm} = 0.1$, $ql = qr, r = 25$

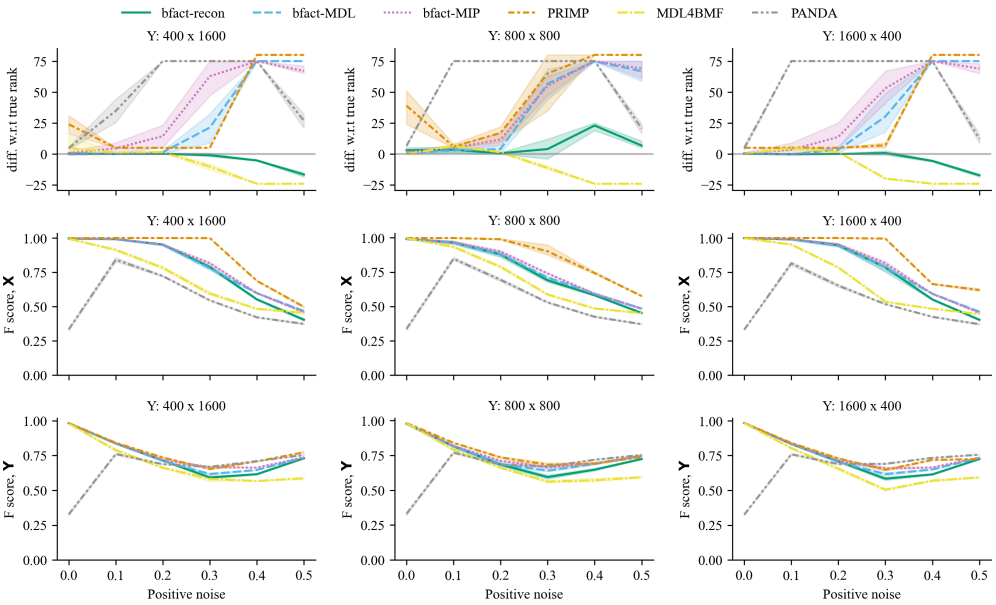


(b) Varying left sparsity, q_l, $p^{\pm} = 0.1$, $qr = 0.1, r = 25$

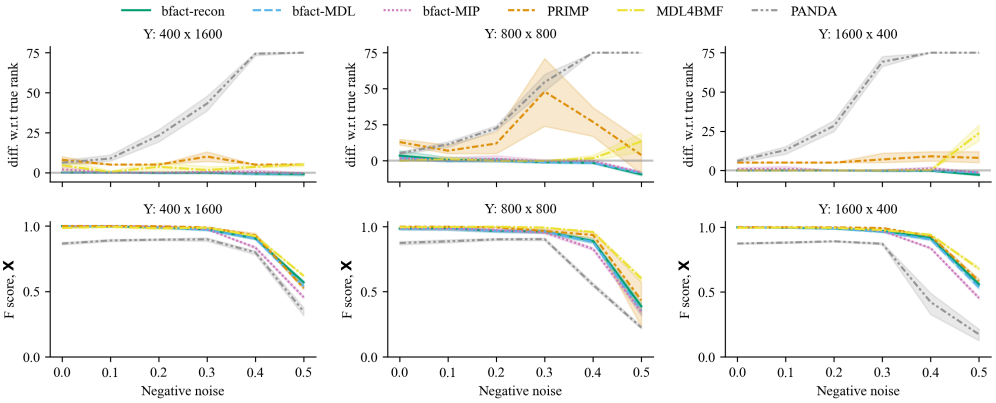


(c) Varying number of factors, $ql = q_r = 0.1, p^{\pm} = 0.1$

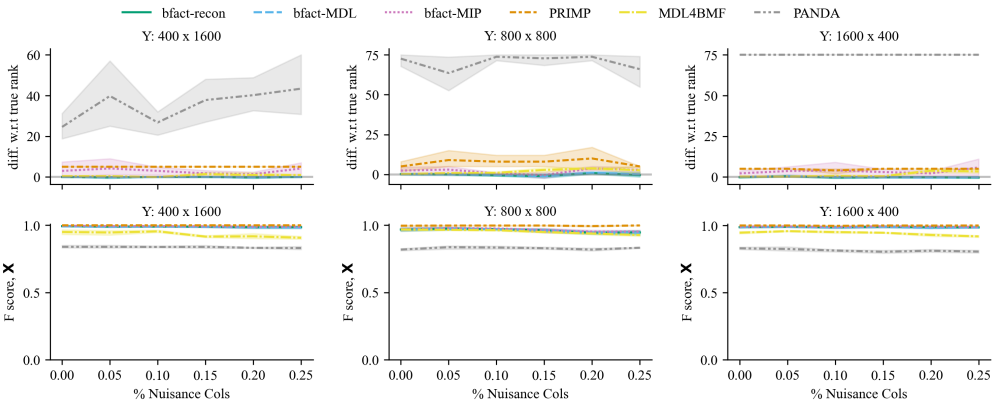
Figure 3.6 Further simulation results, part A



(a) Varying positive noise, p^+, $ql = qr = 0.1, r = 25, p^- = 0.1$



(b) Varying negative noise, p^-, $ql = qr = 0.1, r = 25, p^- = 0.1$



(c) Varying nuisance cols, v_j, $ql = qr = 0.1, r = 25, p^- = 0.1, v_i = 0.1$

Figure 3.7 Further simulation results, part B. For a) I also include the F-score on the data matrix, Y to show that a higher score here does not necessarily translate to a higher score for X, due to overfitting to noise.

3.4.5 Real-world results

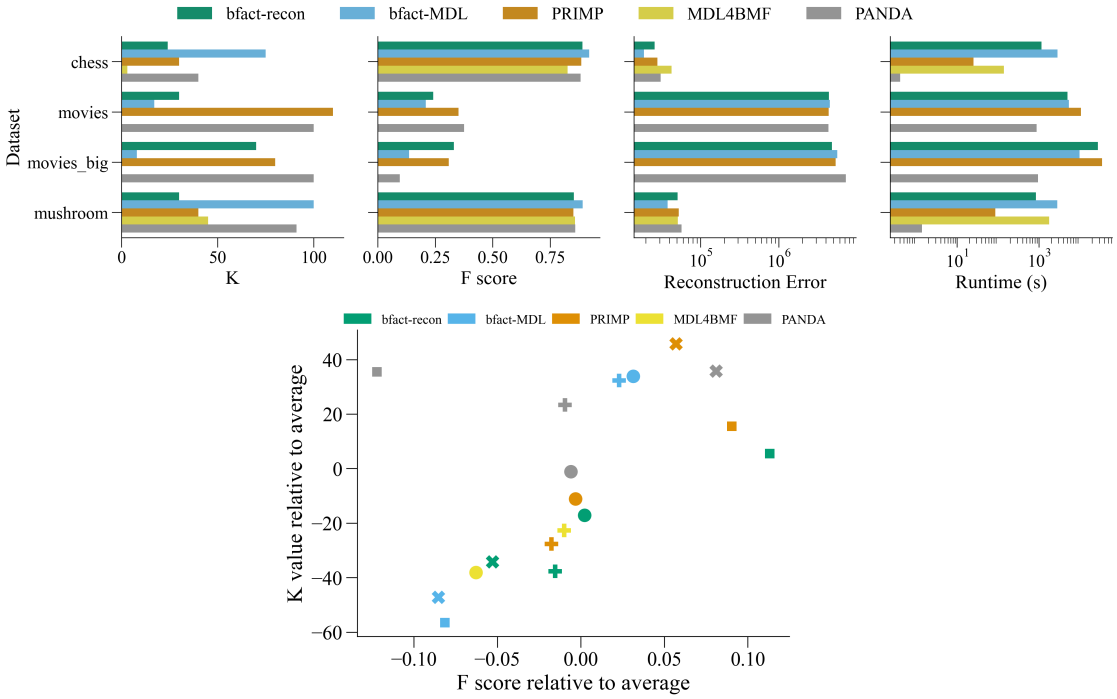


Figure 3.8 Standard Benchmarking results. Shapes in the lower figure correspond to different datasets.

On real-world standard benchmarks, **Figure 3.8**, shows that both `bfact`-recon and `bfact`-MDL variants achieve comparable F-scores and reconstruction errors to other methods, and in particular, `bfact`-recon does this with fewer factors (K). A potential reason for this is that `bfact`-recon is the only method that does not use a complexity-based score for rank approximation, and such scores do not always favour the lowest rank factorisation. Furthermore, some of these datasets have been one-hot encoded from categorical data; hence, it is unclear whether a BMF is the correct model for such data.

On the 14 single-cell RNA sequencing datasets, for which there is more precedent for a BMF, the `bfact` variants achieve both performant F-scores and reconstruction errors but use significantly fewer factors and lower rank matrices, **Figure 3.9**. Fewer factors for similar reconstruction and F1 score is less likely to be fitting to noise in the data, and hence is desirable for downstream analysis, such as identifying marker genes for biological processes.

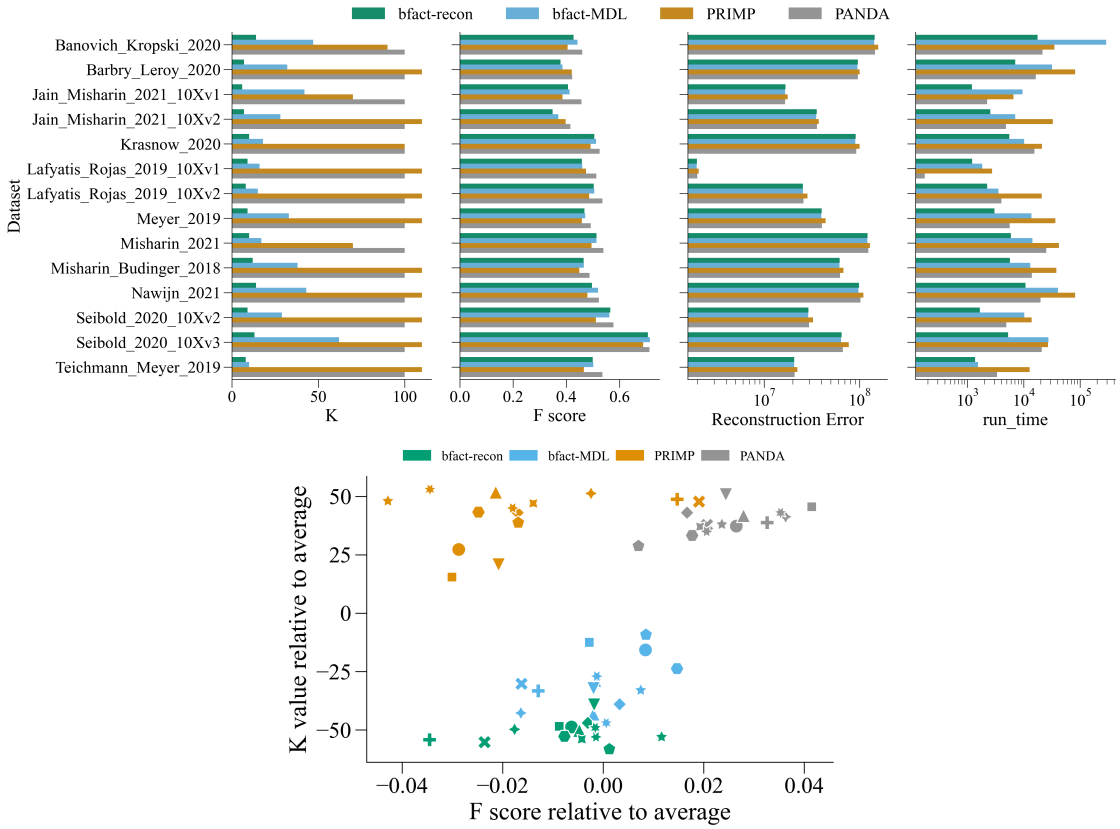


Figure 3.9 HLCA scRNA-seq results. Shapes in the lower figure correspond to different datasets.

PANDA achieves some of the highest F-scores in scRNA-seq data; however, due to its large predicted ranks, this is likely due to overfitting. This was observed in the simulated data, **Figure 3.7**, where PANDA achieved the lowest F score on signal matrix X, despite having the highest F score on observed data matrix Y. Interestingly, PRIMP does worse on some real datasets despite doing generally well in simulations. This might be due to the lower density of the real, particularly RNA-seq datasets. Unfortunately, MDL4BMF often took too long to run (>2days with 24CPUs), so it has not been included for the corresponding datasets.

3.5 Conclusions

This chapter introduces a new binary matrix factorisation approach (including a number of sub-variants) `bfact` which uses a hybrid combinatorial optimisation approach based on *a priori* factors generated from existing clustering algorithms.

`bfact` performs well in simulations and real data, particularly when applied to single-cell RNA sequencing data. `bfact` is available as a pip installable package, with documentation to give guidance on hyperparameters.

As part of the development of `bfact`, I have introduced two novel master problems for recovering either the optimal disjoint BMF or the true BMF. Using both of these, a delayed column generation approach may be used to solve these to optimality for smaller matrices. For larger matrices, using the first master problem warm-started on clustered feature sets works well. This is interesting in itself, as it indicates the link between clustering and factorisation- clusters may be viewed as the intersections and relative complements of factor sets. Further, I explore how an algorithmic approach can uncover the true BMF by using this initial disjoint approach as a basis. Both of these formulations can be directly repurposed to solve inverse problems, i.e $X \approx \theta(\hat{L}R)$, where X and R are known and binary (and where $\hat{L}$ is also binary).

Although combinatorial optimisation techniques are usually used to solve problems to guaranteed optimality, in practice, an approximation is often good enough. `bfact` shows how such an approach can be approximated, and demonstrates how even NP-hard problems can take advantage of combinatorial optimisation techniques and highly optimised solvers like Gurobi. For example, RMP-w in practice is extremely fast to solve, even on scRNA-seq datasets.

3.6 Limitations and Future Work

In practice, `bfact` works better on larger matrices, or when the right-hand matrix R is larger than the left-hand matrix L. Likely, this is due to the candidate factor generation struggling to generate good-quality feature clusters when the number of observations and/or features is small.

Further, for larger matrices, `bfact` does have higher memory requirements, and the computational time increases as K_c increases. This is because of the heuristic algorithm, which greedily removes factors for every increase in K_c. Likely, an improvement to the heuristic algorithm could be implemented- perhaps with

some caching based on previously considered factors. It already uses some parallel processing.

Finally, this chapter introduced how valuing sparsity and disjointness can lead to higher rank factorisations, showing, for example, that the MDL cost can favour sparser solutions when factors have higher overlap. However, there is not necessarily a correct solution for this, which is a consequence of non-identifiable LR. The disjoint/sparse approaches impose additional constraints, which make their solutions more identifiable. Similarly, a lowest rank constraint favours a different kind of factorisation. But it is unclear which is preferable; from an interpretability perspective, both lower rank but higher disjointness is preferable.

4

A critical evaluation of the single cell RNA sequencing pipeline

Contents

4.1 Introduction

This chapter introduces some of the motivating rationales for **Chapter 5**. In particular, I investigate the standard single-cell RNA sequencing pipeline and how it is currently used for identifying cell types and cell type abundances. The aim here is not to be completely comprehensive or to benchmark methodology, but to analyse each step in the context of the data and current recommended best practice. Through this, I aim to highlight current issues and assumptions in scRNA-seq, demonstrating that there is considerable room for improvement and the need for integrated approaches that combine methodological steps.

Single-cell sequencing technologies and analysis methods are constantly evolving. One of the difficulties with scRNA-seq in particular is its origins with bulk RNA-seq and historical scRNA-seq methods, where PCR deduplication was not possible [75, 76]. Methods and assumptions have been biased towards historical technologies, and it is not always clear whether certain approaches have been adopted because of convention or evidence [77–79]. Much recent work challenges the assumptions taken from bulk RNA-seq, demonstrating that new methodology is required, although the literature uptake is slow [78–81].

4.2 Background

4.2.1 scRNA-seq

scRNA-seq measures gene expression at the resolution of individual cells. Unlike bulk RNA-seq, which averages expression across a population of cells, scRNA-seq allows quantification of cell-to-cell variability in gene expression, the measurement of the relative abundance of existing cell types and the discovery of new cell types [82]. Particularly, in cancer, abnormal cell types are thought to represent minimal residual disease after treatment, give insight into disease severity, and potentially serve as a biomarker for disease prognosis [83].

Most current scRNA-seq use a droplet-based system, where single cells are encapsulated into oil droplets, each containing millions of barcoded oligonucleotides [84, 85]. Inside a droplet, the cell is lysed and its mRNA binds to these oligos, which carry a cell-specific barcode and a unique molecular identifier (UMI) to tag individual RNA molecules. Oligos are reverse-transcribed, pooled across cells and amplified to a certain depth using polymerase chain reaction (PCR). After sequencing, barcodes allow transcripts to be traced back to their cell of origin, and UMIs identify unique RNA transcripts from amplified PCR duplicates [84]. The primary output of scRNA-seq measurement is a sparse gene expression matrix, where each entry represents the transcript count, after UMI deduplication, of a given gene in a particular cell.

Several technical artefacts exist in scRNA-seq. The total UMI counts per cell vary between cells, and a proportion of this variation is technical, resulting from differences in capture efficiency, reverse transcription efficiency, or sequencing depth, rather than true biological differences in total mRNA content [76, 85]. So-called 'dropout' events, reflecting the high number of zeros in gene transcripts, are another source of variation [77]. A cell may not express a transcript at the time of measurement, or a transcript may not be captured due to low RNA input or capture inefficiency, resulting in many zero counts. Historically, zeroes were thought to be technical variation, but emerging evidence suggests they are in fact biological, representing cell-type and time-dependent heterogeneity [77,

86]. Additionally, doublets, where two or more cells are captured in a single droplet, can confound cell identity [84]. Ambient RNA, which refers to free-floating RNA in solution, may also be captured and incorrectly assigned to cells [85]. Finally, batch effects, stemming from differences in experimental conditions (time of measurement, equipment/chemistry/operator for measurement), can add other sources of variation [87].

4.2.2 The pipeline

The standard analysis pipeline can be summarised as, [87–89]:

1. Raw processing of RNA transcripts to get UMI counts of each gene per cell.
2. QC steps to filter low-quality cells, remove doublets and optionally correct for ambient contamination.
3. Normalisation of cell counts to correct for cells with different library size (total UMI counts) due to technical variation in the sequencing process.
4. Reduce the dimensionality of the dataset (that can have as many as 30000 expressed genes, sometimes more) to avoid the curse of dimensionality and focus on biologically relevant information.
   (a) Feature selection. Different methods exist; selecting highly variable genes (HVGs) is the prevailing approach. Recent evidence has shown that highly deviant genes are better for cluster performance and other metrics such as variance and deviance explained, [78, 90]. This is recommended in recent best practice [87]. The number of selected genes can be anywhere from 1000 to 5000.
   (b) Dimensionality reduction, usually taking the first 50 PCs of the selected genes.
5. Batch correction (and cell-cycle annotation). There is no consensus on the best batch-correction approach. Here, I opt to use Harmony [91], as two recent benchmarking papers show this is a top-performing approach [92, 93]. Most batch correction approaches are used on a subset of selected features, or

further reduced dimensions, [93], while some occur earlier in the pipeline [94]. Harmony uses PCA-based data.

6. Clustering using the Leiden/Phenograph community detection algorithm.
7. Annotation of clusters. Multiple approaches exist to do this.
   (a) Using known cell type marker genes and finding clusters that have the highest correlation with these.
   (b) Using differentially expressed genes of the detected clusters and comparing them to marker genes or looking for enrichment in known cell identities. 'Known' cell identities are often based on the pipeline above.
   (c) Assigning cells to labels using transcriptional similarity to a reference dataset, then assigning each cluster to the most prevalent label amongst its assigned cells [95, 96]. The complexity of this mapping varies, including deep learning methods for end-to-end integration. Again, 'known' reference labels are often based on the pipeline above.

Although this pipeline has become the prevailing approach in most scRNA-seq analyses, it is unclear which steps have been adopted owing to evidence or convention. Much work already exists benchmarking different approaches in this pipeline, or highlighting issues with certain steps [79, 97–100]. However, evaluation is difficult due to the inherent lack of ground truth. Historical computational annotations are frequently used as 'ground truth' in benchmarking, integration, or supervised learning, biasing methods towards predictions that align with the pipeline used in the original annotations [97–99, 101]. Simulations offer an alternative but may encode assumptions about data structure that favour certain approaches [100]. Some benchmarks use external datasets with known markers or curated cell types as a source of ground truth; however, these often feature easily separable cell types (e.g. PBMCs) or distinct origins (e.g. different cell lines), and may not reflect typical scRNA-seq complexity [90, 97, 102]. Other approaches use proxy measures (i.e variation explained, similarity), an approach adopted here as well [79, 90].

4.2.3 R or python

One would think that methodology should be language agnostic. However, this is not the case. Historical bioinformatics has been R-centric, and packages such as Seurat, Single Cell Experiment, SCRAN and the wider Bioconductor suite exist to perform single-cell analysis, [103–105]. In recent years, Python-centric Scanpy, [106], has become prevalent for several reasons. Python has become dominant in machine and deep learning, and younger generation researchers code more in Python than in other languages [107, 108]. In addition to this, the Theis lab has become a dominating group in single-cell publications, including their best practice guide, analysis of cell atlases and data integration [74, 87, 92, 101, 106]. Scanpy, also originating from this group, offers some integration with methods that have come before (for example, Seurat gene selection).

Most of the analysis in this chapter follows Scanpy methodology, with some direct R implementations and some using external tools through the Scanpy interface. Scanpy version 1.11.5 was used.

4.2.4 Notation

For clarity, this section introduces some recurrent notation used in this chapter. To begin, consider a measured matrix of raw UMI counts $Y : N_C \times N_G$, where N_C is the number of cells (from multiple donors), and N_G is the number of features/genes. Individual entries are y_{ij} for a given cell i and given gene j. Normalised UMI data is denoted $\tilde{Y}$. Matrix X is used to denote a generic matrix, assumed for consistency to also have shape $N_C \times N_G$.

4.3 Data Overview

4.3.1 Origin

The data used throughout this section originates from the Human Lung Cell Atlas (HLCA), [74], which consists of 14 separate datasets that have been individually QCed, processed, annotated and collated. Cells in each dataset have consistent

protocols and sequencing platforms. For this analysis, batch effects from each dataset are assumed to originate solely from donor effects. The data superset also contains integrated annotations across all 14 datasets (integrated in [74] based on HVGs within datasets). The QCed raw counts are provided and are the basis for all subsequent analysis. **Figure 4.1** demonstrates the uniform-manifold approximation and projection (UMAP) of the integrated superset of data, with their curated cell type annotations.

One of the datasets ("Banovich Kropski 2020"), with the largest number of cells (dimension $\sim$ 122k $\times$ 27k), is used as a representative dataset for parts of the analysis. For other parts, results are presented across all datasets, as specified in figure captions.

For parts of the analysis, common housekeeping genes are also used, taken from the housekeeping transcript atlas, [109].

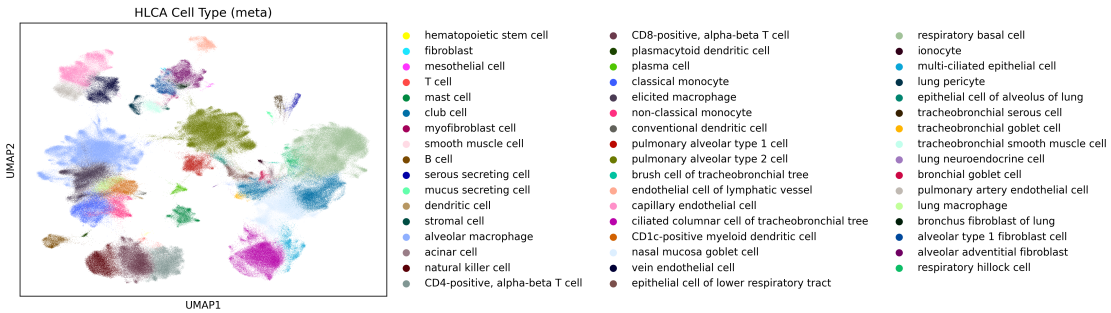
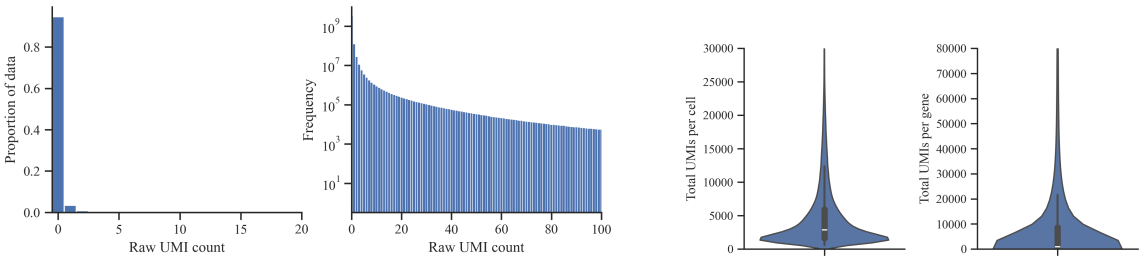


Figure 4.1 UMAP projection of the HLCA final cell types. From HLCA integration of the 14 different datasets. UMAP is a non-linear projection method that has projected the 27k dimensions into two dimensions for visualisation. Each dot represents a cell.

4.3.2 Properties

Figure 4.2a illustrates typical raw counts from an scRNA-seq dataset, highlighting its extreme sparsity- over 95% of entries of the cell-gene matrix are zero, and UMI counts greater than two are proportionally rare. Higher UMI counts are clearer in log space, indicating the long tail for higher counts.

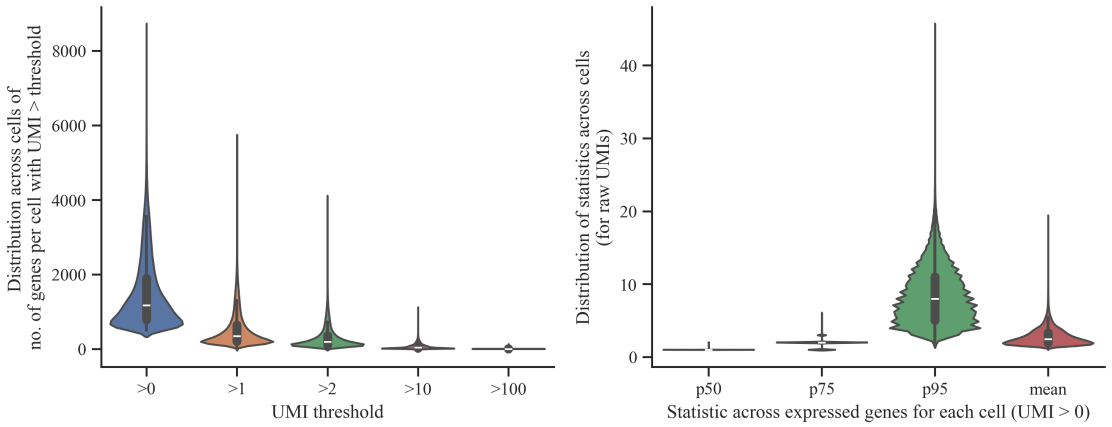
There is also a right skew in cellular UMIs, **Figure 4.2b**, suggesting some cells have more highly expressed genes than others. To investigate whether these



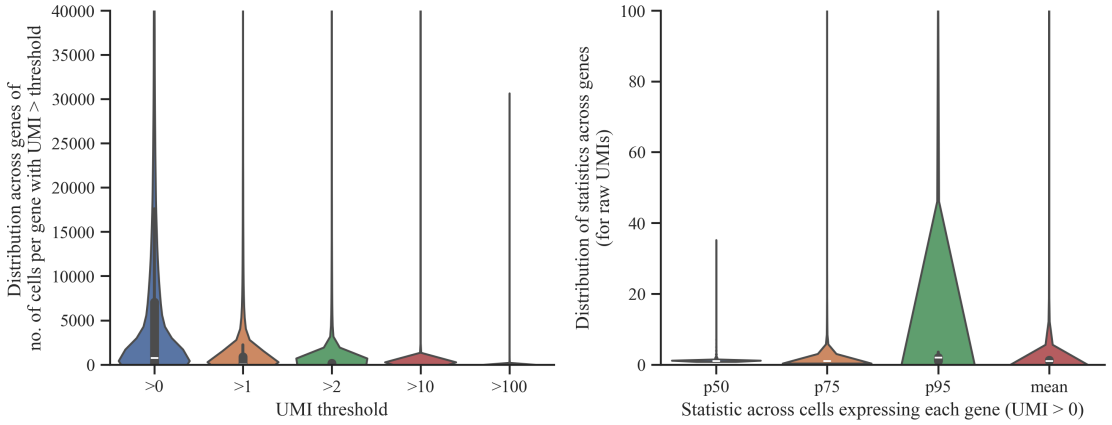
(a) Raw UMI counts of gene-cell pairs as a proportion (left) of all entries and total values (right) on log scale.

(b) Distribution of raw UMI counts across cells (left) and genes (right)

Figure 4.2 Raw UMI properties for scRNAseq



(a) Distribution across cells of gene statistics within a cell. The right plot shows the gene UMI count of different percentiles (and the mean) across all cells.



(b) Distribution across genes of cell statistics within genes. The right plot shows the cell UMI count of different percentiles (and the mean) across all genes.

Figure 4.3 Distribution of count and summary statistics for raw UMIs in scRNA-seq

skews tend to be cell or gene-specific or both, summary statistics are plotted across cells (for genes within each cell), **Figure 4.3a** and across genes (for cells

expressing that gene), **Figure 4.3b**.

The vast majority of genes expressed in cells have UMI counts of 10 or fewer, and 75% of expressed genes in cells have counts of 2 or fewer. Further, the mean of expressed genes is shifted higher than the median (and higher even than the 75th percentile), demonstrating that cell-specific differences are driven by a few highly expressed genes.

There are a few genes with very high expression across many cells, but the vast majority of genes tend to have low UMI counts in most cells in which they are expressed **Figure 4.3b**. Given this, and that for most genes the majority of cells have zero expression, the means and variance of such genes are dominated by the number of zeros versus non-zeros. This is important to consider in downstream analysis.

4.4 Normalisation

Normalisation can refer to many things- harmonising across batches, correcting for technical variation through cellular differences (library size) or transforming data towards a specific distribution (i.e normal distribution) [81]. In the standard pipeline, all are done, but this section focuses on the latter two, as batch correction is done at a later stage [91, 110].

4.4.1 Statistical models for data

In literature, there has been much discussion and evolution on the recommended normalisation process. Various models have been proposed as the statistical model underlying gene UMI counts across cells in scRNA-seq, including the negative binomial, Poisson, multinomial and zero-inflated models [78, 80, 111, 112]. Each assumes certain distributional properties of the data, such that technical variation is accounted for under those assumptions. Zero-inflation models assume that there is a higher-than-expected number of zeros in the data due to technical dropout [112]. Recent evidence has shown that the so-called zero-inflation is likely a consequence of cellular heterogeneity and need not be corrected for [77, 86, 113].

The challenge in assessing such models is that scRNA-seq data are highly sparse, with the majority of gene–cell counts being 0 or 1 ($> 97\%$ in HLCA data). In this near-binary and sparse setting, different count models (Poisson, negative binomial and multinomial across genes with small category probabilities) produce very similar expectations. Consequently, the models are statistically hard to distinguish based on the data alone, likely also why all have been used. To clarify, for each cell-gene entry, if $y_{ij} \sim \text{Ber}(p_{ij})$ when $p_{ij} \ll 1$, this is equivalent to a Poisson with $\lambda_{cj} = p_{ij}$, negative binomial with parameters (r_{ij}, q_{ij}), giving expectation $\mu_{ij} = \frac{r_{ij}q_{ij}}{1-q_{ij}} = p_{ij}$ and multinomial with $c_i q_j = p_{ij}$ for total cell count c_i and gene probability q_j (i.e they all end up with $P(y_{ij} = 0) \approx 1 - p_{ij}$ and $P(y_{ij} = 1) \approx p_{ij}$).

However, models make slightly different assumptions about parameters shared across genes or cells, and some genes violate the sparsity and binary assumption under which the above are equivalent. Yet for the majority of genes, the above holds, which is why these models are similar for such genes, as demonstrated in the distribution comparison in Hafemeister and Satija [80].

One of the common assumptions is that cell-gene specific parameters, λ_{ij}, μ_{ij} and p_{ij} are a composition of a cell-specific size-factor c_i and a gene-specific parameter μ_j (i.e $\lambda_{ij} = c_i \mu_j$). This is explored in more depth below.

4.4.2 Library size correction

As a normalisation step in most scRNA-seq pipelines, a cell-size correction factor is used. Underpinning this is some statistical model that assumes the counts y_{ij} are drawn from some distribution with parameter $\mu_{ij} = c_i p_j$. Hence, to compare p_j across genes, dividing by the cell-size factor c_i is necessary. In the standard and simplest case, c_i is the total number of UMI counts within a cell.

SCRAN size-factor correction is another cell-size correction factor approach, recommended in best practice [87]. SCRAN aims to account for depth variability in the presence of biological heterogeneity that can also affect the depth count. SCRAN estimates stable size factors through pooling and deconvolution: it forms overlapping pools of similar cells, computes pool size factors, and then deconvolves

these to recover per-cell size factors that best explain the total sizes of the groups. Hence, individual cell-sizes, c_i are estimated by pooling on the cell's group rather than over the whole dataset.

The motivation behind cellular correction is that technical effects result in a cellular bias- increasing the number of transcripts in some cells and not others, for example, due to different droplet capturing efficiencies [84]. Sequencing depth in scRNA-seq is highly related to the average number of expressed genes (or number of zeros) ($r^2 = 0.95$) [113]. Intuitively, this means that if a cell has a higher capture efficiency, it is more likely to capture transcripts of any gene, and hence, when estimating the average expression of a gene across cells, capture efficiency should be accounted for.

However, consider the binary extreme, with two cells with capture efficiencies p_1 and p_2. Under cell-size normalisation, the squared Euclidean distance between c_1 and c_2 is scaled by a factor of $1/(p_1 p_2)$ compared with no normalisation, creating a distortion in distance between otherwise similar cell profiles with fewer versus more expressed genes. Normalising by such factors rescales the nonzero (often single UMI) gene counts in a cell, making the data appear continuous at the gene level, **Figure 4.4**.

Under this assumption, if two cells do have lower capture efficiency, we would expect them by chance to have more differences in their raw UMI counts - hence, to upweight this difference (and conclude they are even further away from each other) is counterintuitive. In true statistical approaches, residuals are used, which account for variance in the correction, resulting in more intuitive distance measures. Cell-size normalisation affects downstream analysis, given that pairwise distance underpins both PCA and clustering algorithms. Other works have pointed out that cell-size normalisation approaches induce bias, suggesting probability-based residuals for downstream analysis [78, 80]. However, much literature still uses the cell-size correction as the standard, as is recommended in [87].

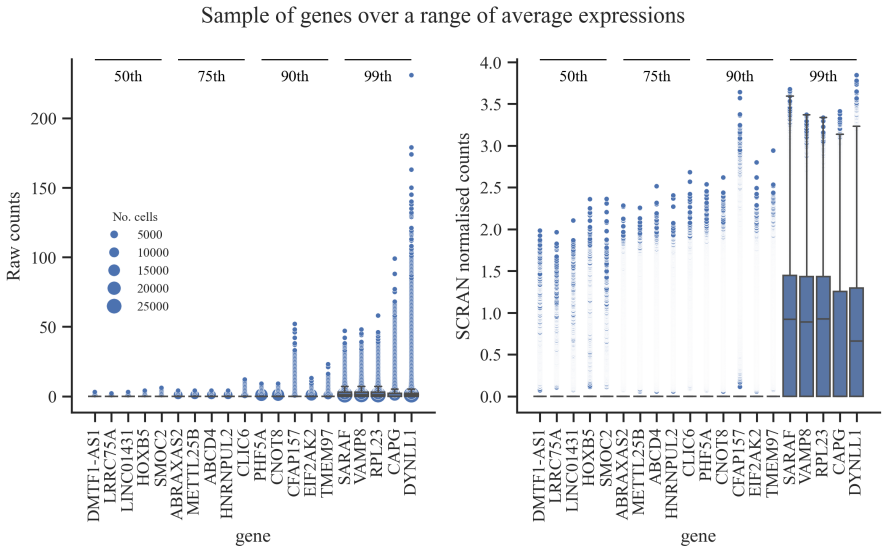


Figure 4.4 Example distribution of genes with means at different percentiles. Left is raw counts, right is normalised data.

4.4.3 Log-normalisation

The second part of the normalisation procedure is to log-normalise the data. This is because UMI counts can be very right-skewed, for technical reasons and biological, including 'transcription bursts' (when transcription occurs in periodic bursts, leading to high UMI counts) [87]. Indeed, the skewedness of the data is clear when the gene distribution is considered, **Figure 4.3b**. Hence, to make data less extreme, and more similar to a normal distribution, a log transform of the data is often taken as $\tilde{y}_{ij} = \log(1 + y_{ij}/c_i)$.

Throughout plots, 'SCRAN-normalisation' refers to using pooled cell correction factors from the SCRAN library, followed by log-normalisation; 'log1p' refers to standard size-factor correction, followed by log-normalisation. **Figure 4.5** (and **Figure 4.4**) demonstrate the effect of normalisation on the data, essentially taking it from discrete to continuous. Both normalisations give similar distributions; however, the effect of each is investigated in downstream steps.

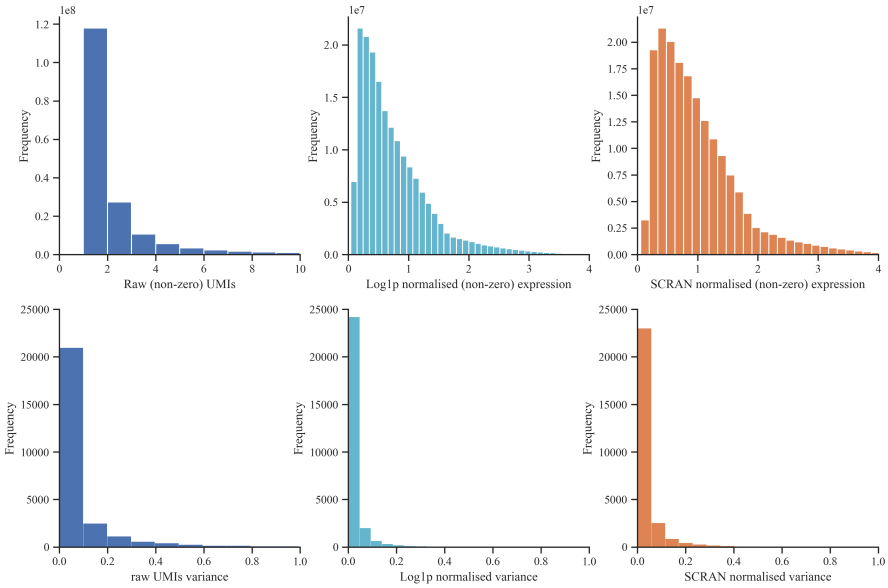


Figure 4.5 Effect of normalisation on data distribution. SCRAN normalisation refers to using pooled cell correction factors from the SCRAN library followed by log-normalisation. Log1p uses standard cell size correction factors.

4.5 Dimensionality reduction with feature selection

In scRNA-seq, feature selection improves computational tractability and reduces the curse of dimensionality. Ideally, the most biologically relevant information is selected, and there exist different approaches in the literature that prioritise different metrics for importance.

4.5.1 Existing Approaches

HVGs, as defined in [114], are the prevailing approach in the literature, as well as the default in Scanpy pipelines and many tutorials. Here, the mean and dispersion (variance/mean) across cells of each gene are used to select genes that have a higher dispersion compared to other genes with a similar average expression, doing so by binning expression. The premise is based on a Poisson model, although the methodology through binning is heuristic. This approach is termed Seurat-HVGs in this chapter, because the authors of [114] are also the same group that maintains the R package Seurat. However, the implementation in this chapter is through

Scanpy (using the 'highly_variable_genes' function with 'flavor' parameter set to 'seurat_v3'). A test was done using the 'FindVariableFeatures' function of the Seurat R package, version 5.4.0, and selected genes showed high overlap with the version 3 implementation in Scanpy, including selection at very low gene mean.

Another variant considered is the selection of HVGs as defined through the SCRAN R package, which is a common alternative in R-based analysis [103]. The mean-variance relationship over all genes is calculated, and genes are selected that vary most from the expected relationship, which assumes a negative binomial (and performs no binning) [103].

Finally, so-called highly deviant genes are considered, [78]. Here, informative genes are selected using a binomial null model- testing across each cell, i, whether a gene, j has a constant relative abundance, $\pi_{i,j}$ across cells, where $\pi_{ij} \approx \frac{y_{ij}}{\sum_j y_{ij}}$. Test statistics select genes that deviate most from this null. Recent evidence shows highly-deviant gene selection yields better results and is recommended in the recent best practice guide [87, 90]. Despite the recommendation of highly deviant genes, HVG remain dominant in the literature.

4.5.2 Mean-Variance relationship in sparse data

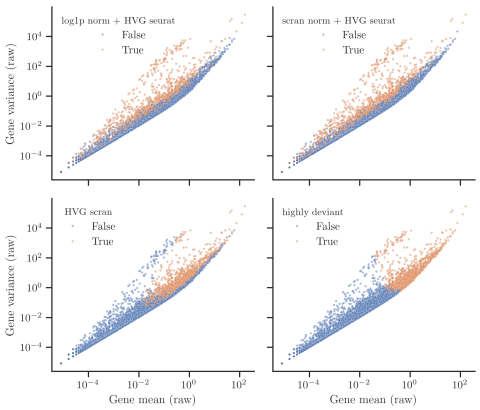
The idea behind selecting highly-variable genes is linked to the assumed data distribution, and is inherited from bulk RNA sequencing [78]. Genes that have a higher-than-expected variance are likely to be a mixture of different distribution parameters. Such genes are found by modelling the mean-variance relationship. For the simplest case, assume an underlying Poisson distribution for the data, and consider a gene that is expressed differently depending on cell type. Under a mixture model, the expectation of measures from such a gene is $\mathrm{E}(X) = \pi\lambda_1 + (1-\pi)\lambda_2$, and variance is $\mathrm{Var}(X) = \pi\lambda_1 + (1-\pi)\lambda_2 + \pi(1-\pi)(\lambda_1 - \lambda_2)^2$. Where $\pi \in [0, 1]$ is the mixing proportion of the cell types and λ_1, λ_2 are the parameters (Poisson mean) of the two different cell types. Thus, the rightmost term corresponds to the additional variability from a mixture of more than one distribution (as $\mathrm{E}[X] = \mathrm{Var}[X]$ for a standard Poisson). Hence, in scRNA-seq, when the variance

of a gene exceeds its expectation, it suggests this gene may be present in different cell types with different proportions.

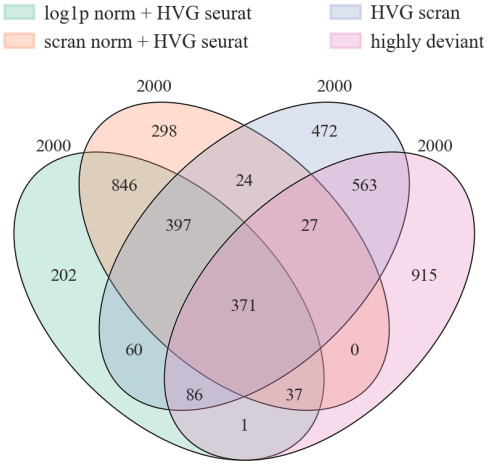
However, it is unclear if this assumption is necessarily true in scRNA-seq, where it is possible $\lambda_1, \lambda_2 \ll 1$ for many genes, given the sparse distribution. Consider the limiting case of all UMI counts being 1 or 0. Then the two cell-types might express a gene according to two different Bernoulli distributions with probabilities $p1$ and $p2$. Under a mixture model for all cells for a given gene, this gives $P(x = 1) = \pi p_1 + (1-\pi)p_2$, which corresponds to $\text{Ber}(\pi p_1 + (1-\pi)p_2) = \text{Ber}(p')$. This means that selecting genes based on the mean-variance relationship may ignore a subset of genes that could be relevant in biological processes (if considered in tandem with other genes), but are not captured under a Poisson/negative binomial model to select HVGs. The mean and variance for a Bernoulli is $(p, p(1-p))$, so particularly for low p, this has the same mean-variance relationship as a Poisson distribution, and for higher p, it will seem that a gene has lower-than-expected variance if a Poisson model were assumed.

Seurat HVGs by design prioritise genes with low expression and higher-than-expected variance. In the sparse data combined with normalisation, this amounts to selecting many genes with UMI counts of 1 in very few cells, **Figure 4.6c**. This rationale affects SCRAN HVGs to a lesser extent as it does not explicitly bucket on mean, and hence selects fewer genes with low mean, **Figure 4.6a**.

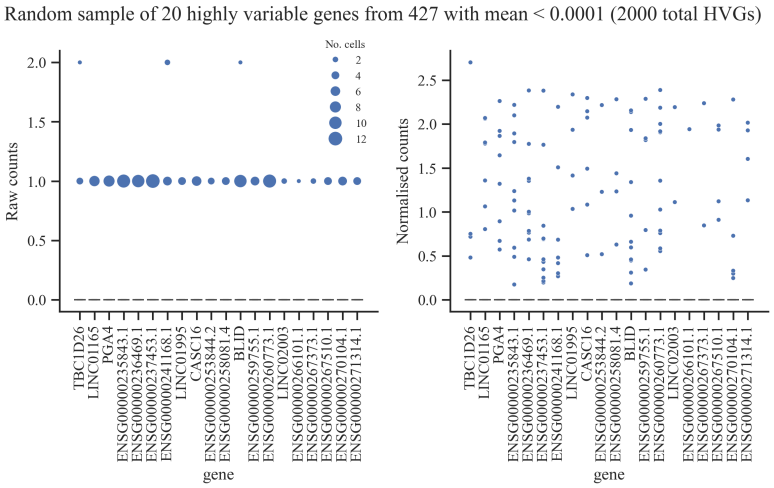
The highly deviant genes method does not control for the mean-variance relationship. However, given the discrete and low-count scRNA-seq data, a statistical test on the uniformity of the relative abundance selects for genes that are either most 'bimodal' or show an abnormal number of high UMI counts across enough genes. Here, 'bimodality' for binary data is defined as $\min(\sum_i I\{y_{ij} = 1\}/N_c, \sum_i I\{y_{ij} = 0\}/N_c)$, where N_c is the number of cells in the dataset. Hence, in either of these cases, highly deviant is a proxy for high mean/high variance, as demonstrated in **Figure 4.6a**, with a mixture of most 'bimodal' as well as higher expressing genes, **Figure 4.8**.



(a) Variance vs mean for different feature selection methods with 2000 genes



(b) Overlap between selected genes using different feature selection methods.



(c) Normalisation in tandem with Seurat HVGs selects low-signal genes. The same genes UMI counts (left) are effectively binary with very few UMI counts of even 1, after normalisation according to cell size factors, these counts are spread to appear to have a higher variance than expected, as Seurat bins on low-mean, these effectively binary, low-signal genes are selected. Boxplot of count distribution across random selection of HVGs with low mean (using SCRAN norm, seurat selection), on raw and transformed counts. Blue dots are all non-zero reads across the 122k cells. Taken on raw and transformed counts.

Figure 4.6 Feature selection properties

4.5.3 Consistency

The consistency of selected genes across the 14 different datasets in HLCA is compared as another performance measure of the feature selection methods, **Figure 4.7**. If genes are important for identifying lung cell types, then they should likely be selected across many datasets. To account for consistent yet uninformative genes,

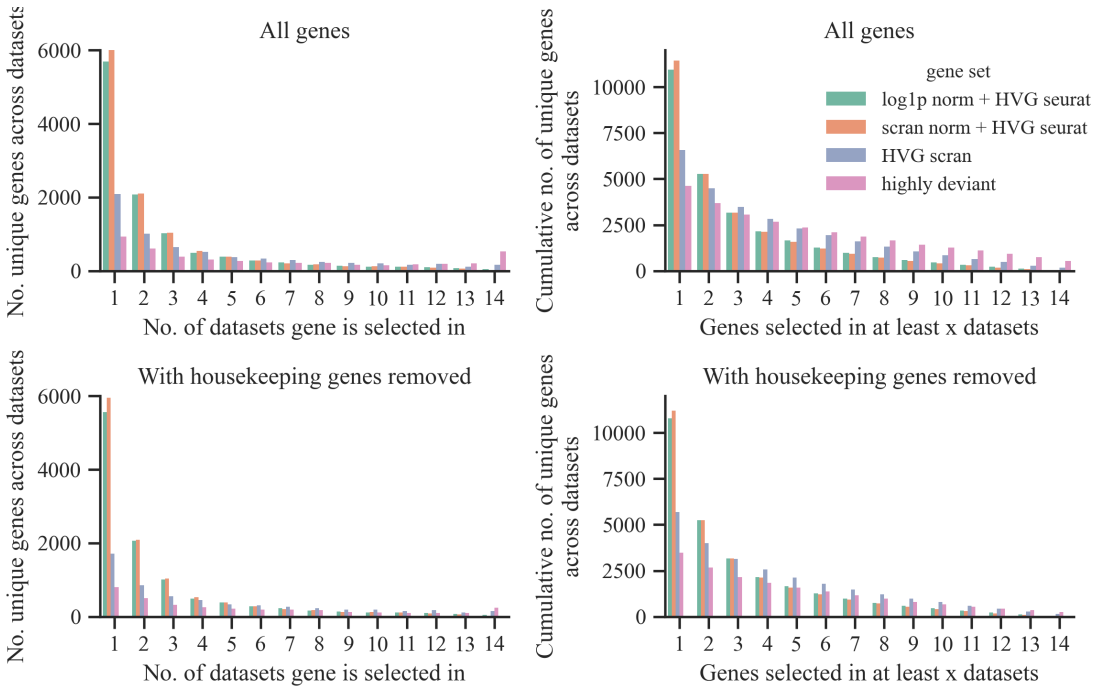


Figure 4.7 Consistency of selected genes across different feature selection methods. 2000 genes selected per dataset.

common housekeeping genes are also removed.

Although highly deviant genes appear more consistent, after the removal of housekeeping genes, much of this effect goes away. This suggests that a proportion of selected, high-mean genes are uninformative housekeeping genes. Such genes correspond to those with abnormally high cell counts for the data distribution, **Figure 4.8**, **Figure 4.4**.

After removal of housekeeping genes, SCRAN HVGs seem to be the most biologically informative (in terms of consistency). Seurat HVGs are highly inconsistent, evidence that much of their selection is stochastic, likely due to the inclusion of spurious genes with very low expression that are sensitive to sampling.

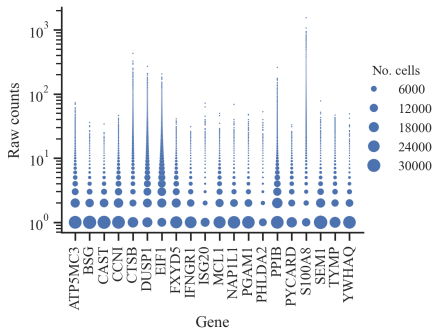


Figure 4.8 Counts of consistently deviant genes. Raw counts in the representative dataset of 20 randomly selected genes that are highly deviant across all 14 datasets, some of which correspond to common housekeeping or stress response genes.

4.6 Dimensionality reduction with principal components analysis

After selecting informative genes, most scRNA-seq pipelines reduce the data dimensionality using principal components analysis (PCA). This step projects the high-dimensional expression data into a lower-dimensional space that captures the major patterns of variation. PCA is typically used to mitigate against the curse of dimensionality, reduce noise, and enable downstream analysis like clustering and visualisation.

This section evaluates the impact of different feature selection methods and scaling approaches on PCA-based dimensionality reduction, focusing on how much biologically relevant variation is captured.

4.6.1 Application of PCA for scRNA analysis

When PCA is applied to an scRNA-seq gene expression matrix, it transforms the data into a set of orthogonal axes, ordered by the amount of variance they explain. This is typically done on normalised counts from a selected subset of genes. A limited number of PCs (usually 50) are then retained to represent the data.

Formally, PCA finds the eigendecomposition of the centred covariance matrix,

given by:

$$\Sigma = \frac{1}{N_G - 1} X^T X = U \Lambda U^T$$

The centred covariance matrix is Σ, while $U = [u_1, u_2, \ldots, u_{N_G}]$ is the matrix of eigenvectors, and Λ is a diagonal matrix with eigenvalues, λ_j along the diagonal, ordered such that $\lambda_1 \geq \lambda_2 \cdots \geq \lambda_{N_G}$. Entries of matrix X are $x_{ij} = x'_{ij} - \mu_j$ for non-centered entries x'_{ij}, with $\mu_j = \frac{1}{N_C} \sum_i x'_{ij}$.

The eigenvectors, u_j, are ordered by their variance explained, given by eigenvalues, λ_j. The top $N \leq N_G$ principal components hence correspond to the subset of N columns of U.

4.6.2 Scaling

A key user-defined choice is whether to scale the data before applying PCA using Z-score normalisation. Without scaling, genes with higher expression and variance will dominate the PCs, whereas scaling centres genes and standardises variance to give equal weighting to all genes. Formally, such scaling involves performing: $\frac{x'_{ij} - \mu_j}{\sigma_j}$, for standard deviation, σ_j of gene j, for non-centered data entries x'_{ij}.

There is no consensus in the literature about whether scaling should or should not be performed before PCA in scRNA-seq, [115]. The Scanpy tutorials [106], the best practice [87], and the R-based Bioconductor tutorial [104] do not specify or recommend against performing scaling. The argument against scaling is that relative magnitudes of expression are biologically relevant for cellular-identity discrimination [115]. In contrast, the Seurat documentation and other online tutorials highlight that scaling is important [116, 117] before performing PCA.

Without scaling the data, the rationale behind some of the feature selection methods is redundant. For example, both SCRAN and Seurat select genes that are variable with lower means and lower absolute variance, **Figure 4.6a**. Without scaling, PCA is unlikely to consider such genes, as their relative contribution to

the variance is low. Only if such genes are highly correlated will they be favoured over higher variance, uncorrelated genes.

To investigate the assumptions around scaling or not, the following are compared:

- SCRAN-normalised data (cell size correction, log normalisation)
- SCRAN-normalised data (cell size correction, log normalisation) with Z-score scaling across each gene

4.6.3 Variance attribution

For feature selection methods, a measure of the usefulness of each selected gene is how much a gene contributes to explaining the total variance (of all the data) through the top 50 principal components (PCs). The relative gene contributions are calculated according to $v_j = \sum_{j'=1}^{k} u_{jj'}^2 \lambda_{j'}$. The loading $u_{jj'}$ represents the contribution of the original feature j to eigenvector j'. Hence, the variance explained by feature j across the top $k = 50$ eigenvectors (PCs) is given by the variance explained by each eigenvector, which is the eigenvalue $\lambda_{j'}$, weighted by the variance proportion attributable to feature j, given by $u_{jj'}^2$.

For this analysis, a maximal baseline for comparison is included, which performs PCA on the complete data matrix (for each dataset), then selects the top 2000 genes based on their contributions v_j, before repeating the process with PCA performed on the subset of those 2000 genes.

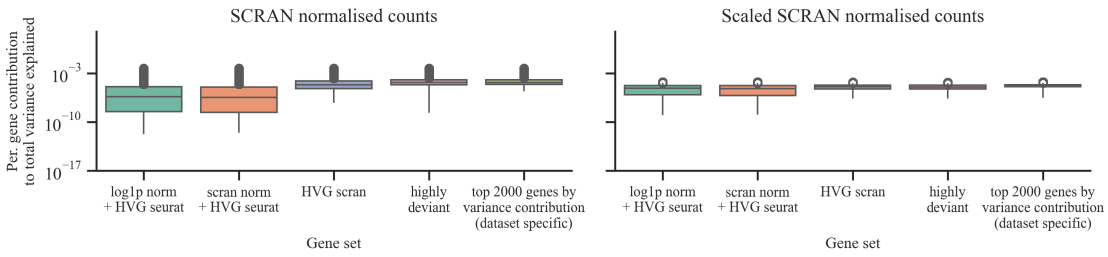


Figure 4.9 Variance attribution. Distribution variance explained by each gene through their contribution to top 50 PCs, across all datasets.

Figure 4.9 shows the variance attribution across features for each feature selection method with both unscaled (left) and scaled (right) SCRAN-normalised data. Outliers on the left plot indicate that scaling indeed impacts the PCs, with a few genes dominating variance contribution. Interestingly, for scaled data, the Seurat selection method still results in lower per-feature contributions to the principal components. This supports the conclusion that the selection method favours uninformative noise genes - if they were informative in explaining variance in other features, there would be a more even contribution to principal components. Despite 2000 genes being selected in the Seurat HVG selection step, in practice, only a subset of these are used in the reduced PC coordinate system. Important information is likely lost as a consequence of selecting uninformative genes. In contrast, SCRAN HVG and highly deviant approaches seem to have comparably informative selected genes, based on their contribution to variance explained. They also perform similarly to the maximal baseline.

4.6.4 Variance explained

As another way of investigating both the effect of scaling and the impact of different feature selection methods, the total variance explained of each approach is compared. Consider:

$$V_T = \mathrm{tr}(\Sigma)$$
$$V_m = \sum_{j=1}^{50} \lambda_j(X_m)$$

where tr denotes the trace of a matrix. V_T is the total variance of the full centred matrix X, while V_m is the variance of the top 50 PCs of the smaller matrix X_m containing only the subset of features given by the feature selection method, m. Hence, the full matrix X of SCRAN-normalised data is considered both with and without scaling, and the total variance explained is calculated using the top 50 PCs on each feature-selected data subset. A baseline is also considered using the top 2000 features by variance explained, as well as a maximal baseline of using the complete data matrix for the top 50 PCs.

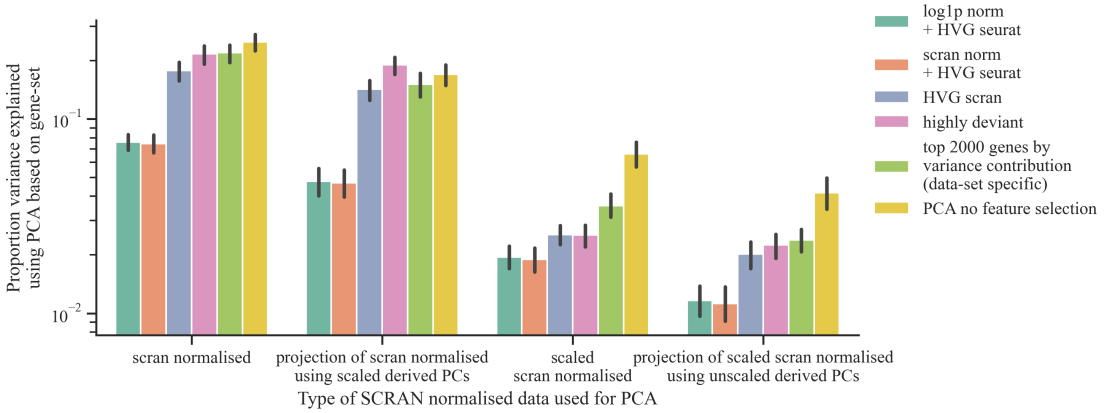


Figure 4.10 Variance explained. Proportion of total data variance explained (with each scaling) using the top 50 PCs derived from each gene set, standard error bars across 14 datasets.

The total variance explained in unscaled and scaled data differs because they have different total variances. Hence, to compare directly, each scaled/unscaled dataset can be reconstructed based on the top PCs derived from the opposite (unscaled/scaled) dataset. This shows the effect of scaling on the PC representation. This is done according to:

$$\hat{X}_{2,m} = X_{2,m} U_{50}(X_{1,m}) U_{50}^{T}(X_{1,m})$$
$$\mathrm{VE}_{m,p} = \frac{\mathrm{tr}(\hat{X}_{2,m}^{T} \hat{X}_{2,m})}{\mathrm{tr}(X_2^{T} X_2)}$$

Where $\mathrm{VE}_{m,p}$ is the variance explained of method m under the projection into the opposite space. Here $X_{1,m}$ denotes the scaled/unscaled and centred dataset on which the PCs were derived (subset to features m) while $X_{2,m}$ denotes the opposite (unscaled/scaled) dataset subset to features m.

Figure 4.10 highlights the variance explained by each method using both unscaled and scaled data (as well as projections based on PCs derived in the opposite). The difference in variance explained based on PCs derived from scaled vs unscaled data is not large, particularly for SCRAN HVGs and highly deviant- the PC representation of these features is not sensitive to scaling. Likely, this is because these features are already of comparable variance, so rescaling makes little difference, also indicated by **Figure 4.6a**. Conversely, there is a larger difference

between Seurat-selected genes, suggesting scaling makes a larger difference for these genes, which is intuitive given the forced selection at lower mean/variance.

Using PCA on the whole data increases the variance explained, particularly in scaled space, but also using unscaled PC loadings. This suggests that the data contains a shared substructure omitted when selecting features. Using the top 2000 genes by variance contribution underperforms compared to using all genes for PCA, indicating that the number of selected genes is likely the limiting factor.

Using PCA on the whole data will always explain more variance, by definition, than on a subset of features. The question then arises as to why even select a subset of features. There are two reasons given in the literature for this: first, that it is computationally faster to perform PCA on a subset of data, and second, that structured noise or batch effects risk getting amplified and propagated [87]. Although the first point is indeed true, the time taken to perform PCA on large datasets is not limiting (i.e it is still only in the order of minutes).

The second point may be relevant; however, it is difficult to find literature that definitively shows this. Circular arguments on feature selection are made using 'ground truth' annotations based on HVG selection. Hence, further analysis considers using PCs derived without any feature selection.

4.6.5 Summary

These experiments further demonstrate that Seurat HVGs are likely a poor feature selection choice. They also show that for other feature selection methods, differences in PC loadings on scaled vs unscaled data are small. Hence, the further analysis continues using unscaled data, which aligns with best practices recommended in Luecken and Theis [115]. Further, given that the results for the log1p and the SCRAN normalisation for Seurat HVG selection are similar, the subsequent analysis continues with Seurat HVGs on SCRAN normalised data only- this is also the recommended normalisation in the best practice [87].

4.7 Batch correction

Batch correction is used to correct for different sequencing technologies and platforms. In datasets where inter-donor variability is not of biological interest (i.e not looking at the association between disease/genomic profile), it is common to correct for donor as part of batch correction as well [92]. Many methods exist that perform batch integration [92, 93]- some as initial preprocessing steps and some are done before clustering and after PCA. The reasoning for performing batch correction after these steps is likely due to computation time and the curse of dimensionality. However, performing feature selection before batch correction may result in biologically uninformative selected genes.

In Luecken et al. [92], they compare methods both before and after feature selection and found that most worked better after feature selection. However, given that their annotations are based on a pipeline using HVG selection for individual batches, there may be some data leakage/bias towards the pipeline for annotation.

Here, the batch-correction method, 'Harmony', [91], is used, as it has performed well in two different independent benchmarks, [92, 93]. Harmony is applied after feature selection and PCA. Clustering results are compared both with and without batch correction.

Internally, Harmony works by iteratively clustering and then correcting PC space, such that the centroids of cells originating from each batch and assigned to each cluster are aligned between batches.

4.8 Clustering

The next step in the pipeline is clustering, where the scRNA-seq pipeline uses Leiden clustering, a community detection algorithm, [118]. Although different clustering procedures exist, this has become the prevailing approach due to automatically detecting the number of clusters (subject to hyperparameters).

4.8.1 Procedure

In scRNA-seq, different clustering approaches and hyperparameters give different cluster results [118, 119]. Slovin et al. [89] showed how different pipelines (e.g Seurat vs Scanpy) often give significantly different cluster results. Here, something similar is done but on a smaller scale, specific to the standard Scanpy pipeline. In particular, I:

- Use different feature selection methods to find a set of N features. Usually, 2000 are used, but different numbers of selected features are also investigated.
- Perform PCA on the SCRAN normalised data subset to each method's selected features. The top 50 PCs are retained. PCA is also performed without any feature selection.
- Optionally perform batch normalisation on each PCA result, giving corrected PCs.
- Perform Leiden clustering on each of these PC projections using a resolution parameter of 1.

4.8.2 Leiden Algorithm

The Leiden clustering algorithm uses an adjacency matrix that defines edges between nodes in a graph to detect communities by maximising modularity. Modularity measures how well a graph is partitioned into communities by comparing the density of edges within groups to the expected density in a random graph with the same degree distribution (the degree of a node is the number of edges to/from that node). The Leiden algorithm, however, does not measure the statistical significance of communities, and often finds communities in random graphs [120]. Leiden also has a resolution parameter that controls the granularity of clusters.

In Scanpy, Leiden uses a connectivity matrix based on a K-nearest neighbours graph from the top PCs of the selected features, [106]. The K-nearest neighbours for a cell i are the closest K other cells i' based on Euclidean distances in the PC

space. The connectivities between neighbours are defined using a UMAP-based approach, [121], according to:

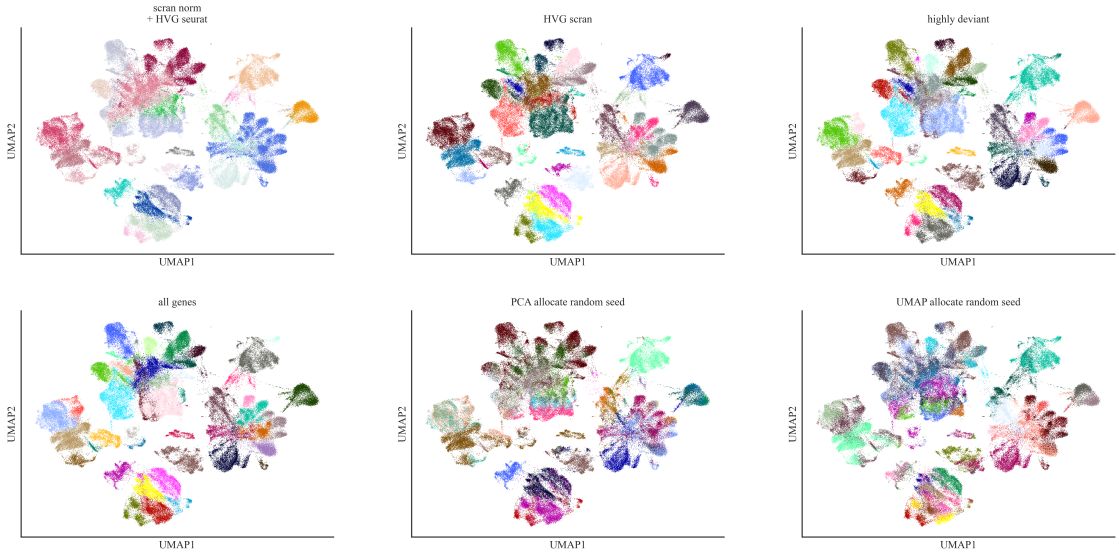
$$a_{i'|i} = \exp\left(\frac{-\max(0, d(i,i') - \rho_i)}{\sigma_i}\right) \quad i' \in N_K(i)$$
$$C_{ii'} = a_{i|i'} + a_{i'|i} - a_{i|i'}a_{i'|i}$$
$$\rho_i = \min_{i' \in N_K(i)} d(i,i')$$

Where here $N_K(i)$ is the set of K nearest neighbours of i, $d(i,i')$ is the Euclidean distance in PC space, between two cells i and i', and ρ_i is the distance to the nearest neighbour. The scaling parameter σ_i is defined by solving $\sum_{i' \in N_K(i)} a_{i'|i} = \log_2 K$ [121]. The scaling essentially serves to downweight neighbours that are further away in dense regions as compared to sparser regions. The connectivity weights defined by $C_{ii'}$ are then used in the Leiden algorithm to detect communities. In practice, this dynamic density adjustment means that in denser regions, where groups of cells are very similar, small changes in PC space are upweighted.

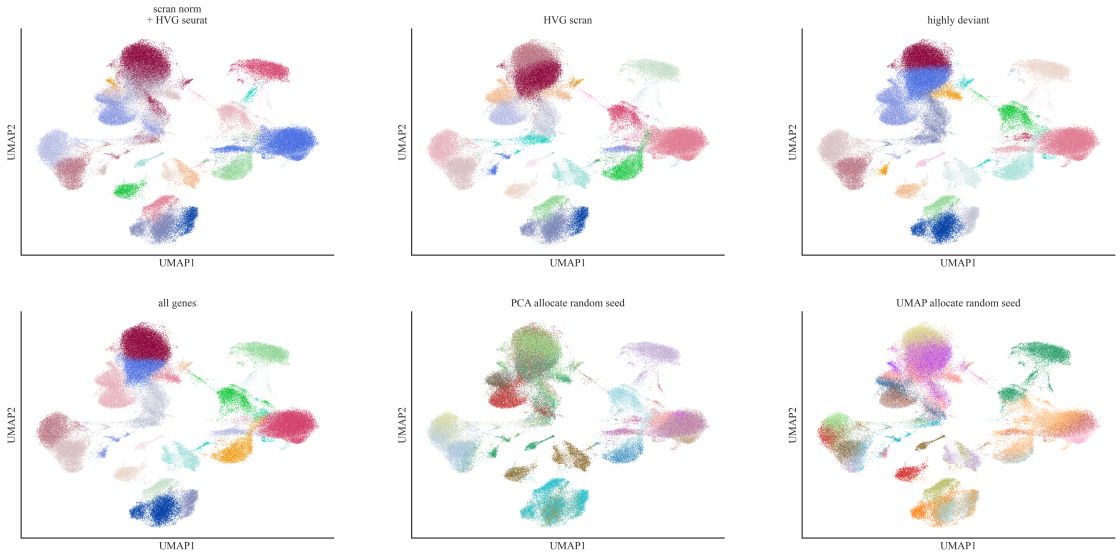
4.8.3 UMAP visualisation

UMAP is a commonly used visualisation technique (very similar to t-SNE). UMAP uses an optimisation algorithm to non-linearly project data into a lower-dimensional (usually 2D) space to preserve KNN-graph connectivities, as defined above. Distances are not literal, but represent projected connectivities. **Figure 4.11** introduces the UMAP projections of the scRNA-seq representative dataset, coloured by different clustering approaches. From this, it is clear that batch correction removes much of the substructure/variance in the data, which is likely to be batch-specific.

It can be tempting when considering UMAP projections to conclude that clustering accurately detects groups of cells, based on the visual groups in the UMAP, **Figure 4.11**. However, alignment between Leiden clustering and UMAP projection is expected, given that Leiden clustering is based on the UMAP connectivities. Hence, the visualisation and clustering are affected by the same pipeline and potential artefacts, so just because a cluster 'appears' in the visualisation, it does



(a) Without batch correction



(b) With batch correction

Figure 4.11 UMAP visualisations of clustering results from each approach. UMAP projections are based on the entire dataset and not just selected genes, while colourings are based on cluster results after selecting 2000 genes.

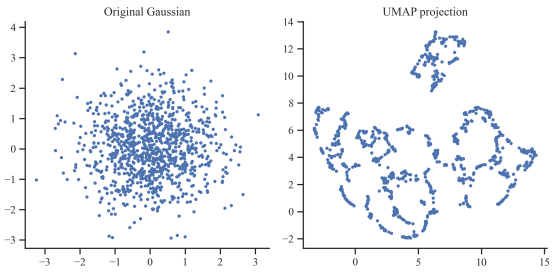


Figure 4.12 UMAP projection can create false positive clusters in data. The left shows the original data, with a single mode, whereas the right shows the UMAP projection of that same data, where a second group is separated away. This is due to local density differences from sampling.

not necessarily mean it exists in the data, **Figure 4.12**. scRNA-seq data is exceptionally high-dimensional, with other nuances due to the discrete distribution, batch and technical effects. A projection into 2D will never capture the nuance of the data, and even in that projection, many cells may go unnoticed due to $\sim$ 100k points on a small plot.

4.8.4 Evaluating clusters

Evaluating the performance of different clustering approaches is difficult when the ground truth is unknown. Various works in literature treat the output of previous annotations as ground truth, but this approach will benefit methods that best align with the annotation pipeline. Here, the similarity of different clustering methods is compared, as well as the similarity of the same method under different conditions.

Clusters are compared using the adjusted mutual information (AMI), which quantifies the agreement between two clusterings by measuring how often pairs of cells are assigned to the same or different clusters in both. It adjusts for chance agreement and is less sensitive to differences in the number of clusters than other metrics like adjusted rand index or raw mutual information [122]. It is defined by:

$$\begin{aligned}
\mathrm{AMI}(U,V) &= \frac{\mathrm{MI}(U,V) - \mathbb{E}[\mathrm{MI}(U,V)]}{\max\{H(U),H(V)\} - \mathbb{E}[\mathrm{MI}(U,V)]} \\
\mathrm{MI}(U,V) &= \sum_{i,j} \frac{n_{ij}}{N} \log\left(\frac{n_{ij}\,N}{n_{i.}\,n_{.j}}\right) \\
H(U) &= -\sum_{i} \frac{\sum_j n_{ij}}{N} \log \frac{\sum_j n_{ij}}{N},
\end{aligned}$$

Here, U and V denote different cluster representations, and n_{ij} denotes the shared cells in class i of U and class j of V. The expectation is taken over random clusters with the identical cluster size distributions. H indicates the entropy, while MI indicates the mutual information- how much knowing about one cluster result, U, is informative of another cluster result V. AMI is between 0 and 1, where 1 indicates perfect agreement and 0 means no more agreement than random chance.

4.8.5 Random baseline

Although an AMI of 0 and 1 are interpretable, to quantify intermediate agreements, it is easier to compare to some baseline. Since cells that are close in distance tend to end up in the same cluster, an informative baseline is to randomly select K seed cells and assign each other cell to its nearest seed. This produces a Voronoi partition - a random distance-based partition of the space. Such Voronoi partitions are generated in two spaces: the PCA space (specific to each feature selection and base data combination) and the UMAP space (also for each combination). The base data considered is the SCRAN-normalised data with and without batch correction. **Figure 4.13** shows an example of such a random partition in UMAP space.

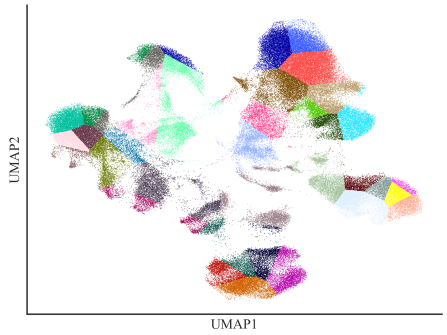


Figure 4.13 Example of the UMAP random seed allocation cluster result. Based on deviant selection, batch corrected.

The two spaces serve different purposes:

- The UMAP-based Voronoi partition aligns with how Leiden clustering constructs KNN graphs, since the Scanpy pipeline applies UMAP-based connectivities for the adjacency matrix used in Leiden.

- The PCA-based version allows us to assess how much structure is preserved or distorted by UMAP relative to the original data space.

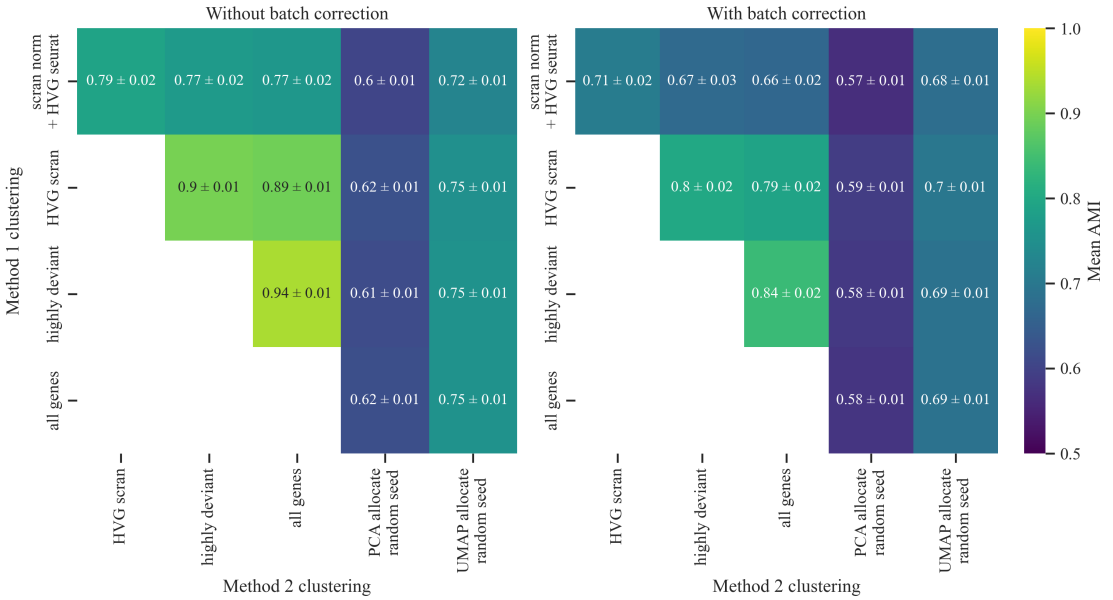
$K = 35$ is used for the Voronoi partition on non-batch corrected data and $K = 20$ for batch corrected- to match the maximum number of clusters output using Leiden with resolution 1 on the data **Figure 4.14b**. Five repeats are taken to get metrics across different random partitions.

4.8.6 Similarity between approaches

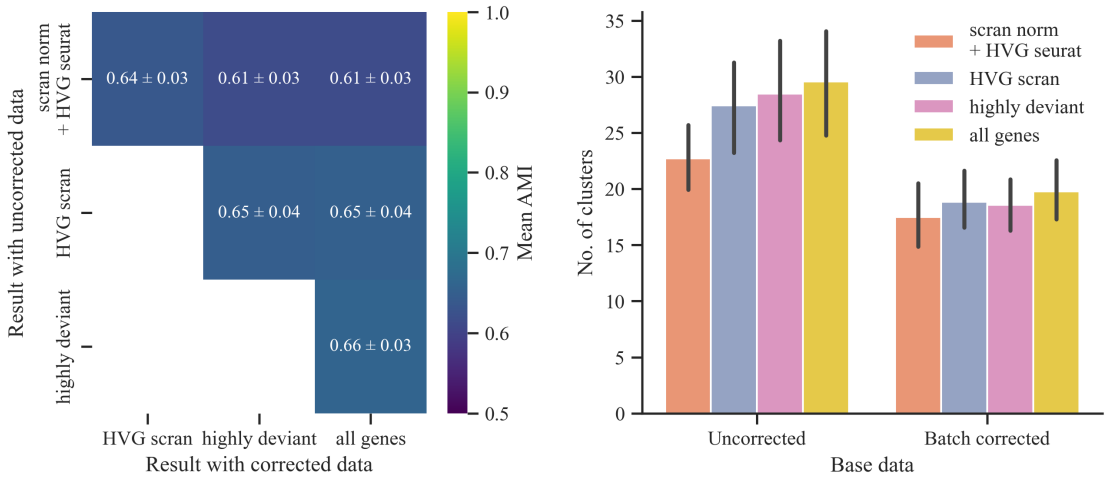
Figure 4.14 compares cluster results across different approaches with a default of 2000 selected genes (unless using all genes). Clusters based on PCA with no gene selection or with highly deviant or SCRAN HVG feature selection are all equally similar, while Seurat-based clusters follow behind. Notably, the Seurat-based clusters are as similar to other cluster results as they are to a random-based partition in UMAP space. Differences between cluster similarity with batch correction are all lower than without batch correction, including random partitions, which may be a consequence of fewer clusters, **Figure 4.14b**.

Differences between UMAP and PCA-based random clusters indicate that connectivity density distortion from UMAP has a moderate effect on cluster results.

The similarity between uncorrected and batch-corrected data for each method is comparable to the similarity between the methods and random partitions in PCA space, **Figure 4.14b** (left). This indicates that batch correction has a significant effect on identified clusters, as confirmed visually with UMAPs, and the fewer number of clusters, **Figure 4.11**.



(a) Similarity between results for method combinations on the same base data



(b) Similarity between results for method combinations using different base data (left), and number of clusters for each method on each base data (right)

Figure 4.14 Cluster similarity between different feature selection methods within (top) or between (bottom, left) uncorrected or batch-corrected base data. Feature selection done according to each method's requirements, then using data → take feature subset → PCA → cluster. Error bars on random baselines correspond to 5 repeat random partitions for each dataset. Standard error bars on other comparisons are taken across the 14 datasets. Cluster similarity is somewhat sensitive to the number of clusters. 2000 selected features for each method.

4.8.7 Similarity across different selected genes

To further understand differences between approaches, the similarities are explored when selecting more or fewer genes for downstream analysis. The cluster results are

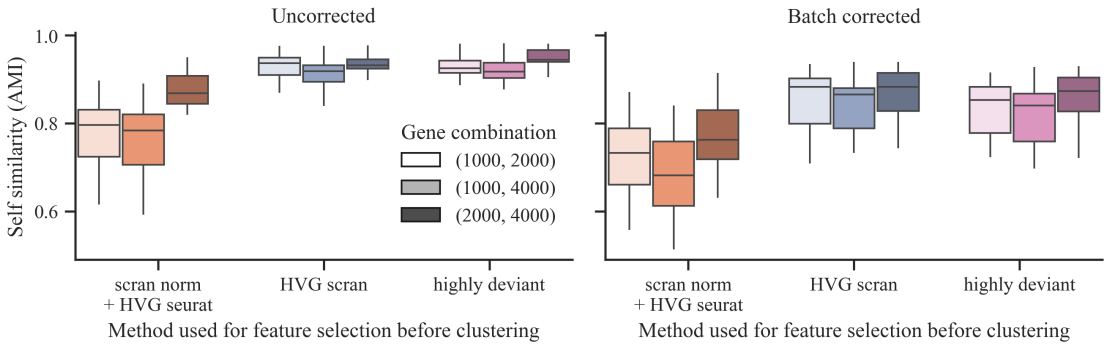


Figure 4.15 Self-similarity between clusters. Using the same feature selection method over different numbers of selected genes.

also considered across different numbers of genes within the same feature selection approach (self-similarity), **Figure 4.15** and across different approaches, **Figure 4.16**.

Notably, the self-similarity and similarity between cluster results are much more sensitive to gene selection in the batch-corrected data than in the uncorrected data. This is likely a consequence of the batch correction step, Harmony, introducing noise by refining PCs based on relative batch contribution. However, this does not necessarily mean it is less correct. For example, the average cluster result could be more accurate than non-corrected data (if ground truth was known), with a higher variance estimate. In contrast, uncorrected measures could have lower variance and higher bias. Results in the literature, [92, 93], indicate this approach performs better in simulated and experimental scenarios with batch effects. An improvement could perhaps be made on Harmony by bootstrapping features to decrease the variance in the corrected data.

Similarity with Seurat HVGs is particularly low for batch-corrected data, **Figure 4.16**, suggesting the method selects genes that are variable due to batch effects. Further, they become more similar to other methods as more genes are selected, again suggesting Seurat HVGs are less robust. Intuitively, as more genes are selected from any method, their similarity approaches that of using all genes. However, similarities between clustering using SCRAN HVG, deviant or all genes are similar even for 1000 selected genes, suggesting that most informative genes are captured even at 1000 genes.

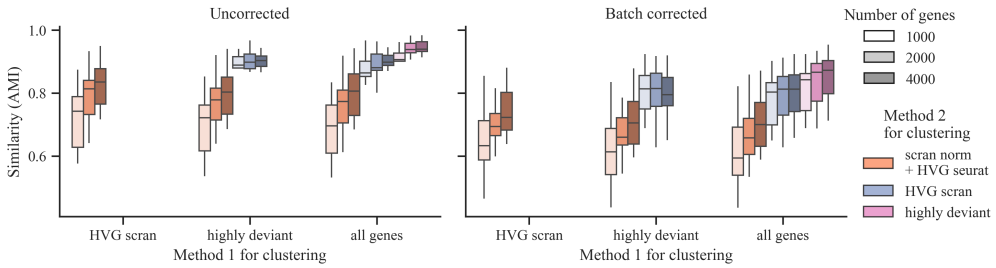


Figure 4.16 Similarity over different numbers of selected genes. Between clusters using different feature selection, Error bars over all datasets.

4.8.8 Similarity with HLCA annotations

As a final metric, results are compared to the annotations in the HLCA for this dataset:

- The original annotation from each respective dataset publication.
- The HLCA final annotation. This came from integrating across all datasets with scANVI (a deep-learning-based integration approach), using HVGs calculated with the cell ranger flavour of Scanpy HVGs. Leiden clustering was performed on the latent dimensions from scANVI [74]. Expert input was used to join some clusters.

Figure 4.17 demonstrates the imprecise nature of the scRNA-seq pipeline. Despite using the best practice in the literature, there is low reproducibility of cluster assignments. The AMI between original annotations and cluster results on the individual datasets is 0.6, approximately the similarity between each method and

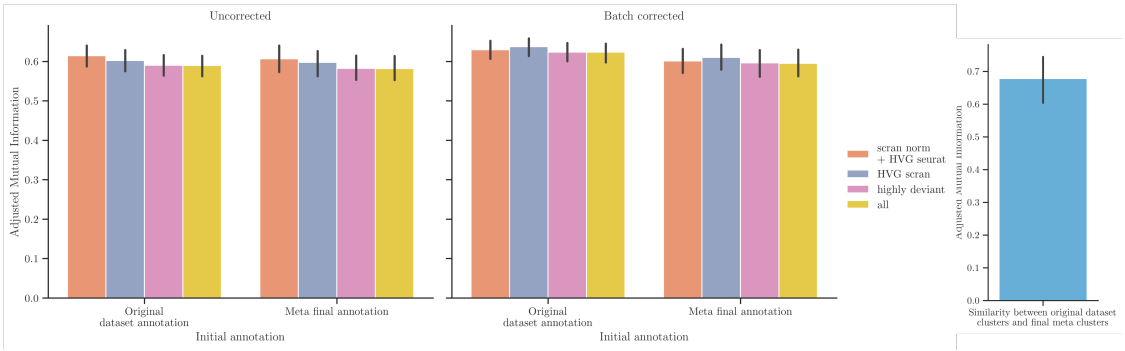


Figure 4.17 Similarity with original annotations. Self-similarity of original annotations (right).

the random PCA partition, **Figure 4.14**. Despite results throughout this chapter indicating that Seurat HVGs are likely a poor choice, slightly higher agreement with the HLCA annotations comes from using uncorrected data with these annotations. This supports an implicit bias towards HVGs used in annotations. The similarity between annotations from the original dataset and HLCA is slightly higher (0.7 rather than 0.6), likely because scANVI integrates data using the known dataset labels (based on those from the original dataset).

4.8.9 Summary

To summarise some of the key findings above:

- Genes should not be selected using the Seurat approach, despite being the default in Scanpy and many other scRNA-seq analyses in the literature. Increasing the number of genes makes it more similar to other approaches.
- SCRAN-based gene selection and highly deviant gene selection result in similar clusters, and appear similar to using no feature selection.
- Batch normalisation using Harmony introduces variance into the clustering results as it appears less robust to selected features.
- Clusters are sometimes as similar as a random partition in UMAP/PCA space, meaning cells that are close together are likely to be similar cell types, but exact boundaries are not robust. This has implications on downstream analysis, particularly when cell-type abundance to quantify disease effects.
- Visual 'validation' on UMAP should be used with caution.
- Cluster results are difficult to reproduce and depend on upstream pipeline choices.

4.9 Annotation

The annotation pipeline for scRNA-seq according to [87] should consist of:

- Manual annotation of identified clusters using a combination of enrichment/overlap with known marker genes and the differentially expressed genes of clusters.

- Automated annotation of clusters using methods such as CellTypist or scArches [123, 124].
- Verification by 'experts'

The term 'expert' verification is somewhat misleading; it typically refers to manual interpretation of cluster results by researchers such as PhD students or postdoctoral scientists. In other domains, such as radiology, the use of expert annotation is more defensible: radiologists are medically trained, follow established diagnostic criteria, and often receive feedback through clinical outcomes. Instead of using such language, manual criteria should be specified such that annotations are reproducible and consistent across analyses.

4.9.1 Manual annotation

For manual annotation, reference marker gene sets can come from various sources. Some reference gene sets are calculated similarly to this pipeline. Specifically, scRNA-seq data are clustered (sometimes sub-clustered), and differentially expressed genes (DEGs) are identified and filtered for those unique to each cluster. Clusters are joined depending on identified DEGs, and clusters- now cell identities- are catalogued with corresponding unique DEGs as marker genes [74]. Many original marker gene sets are derived from well-studied cell types and have been validated in multiple studies or using other data modalities such as immunohistochemistry or fluorescence-activated cell sorting [125, 126].

In the three recent tutorials for scRNA-seq [87–89], none specify how exactly manual annotation should be performed. In the Luecken and Theis [115] tutorial, it specifies an enrichment type test. In associated code, they take the top 100 differentially expressed genes (DEGs) of each identified cluster, and use this to calculate an overlap score (not enrichment) with the reference marker gene sets from literature [106]. Based on this, and manual inspection of some of the marker genes, cell identities are assigned.

However, there are some existing enrichment test tools to perform enrichment analysis using reference marker gene sets, including Enrichr and the Python

implementation thereof [127, 128]. Enrichr hosts multiple reference marker genes and can perform enrichment tests based on the number of overlapping genes between the database reference and the query gene set (usually using top cluster-specifc DEGs). The way marker reference gene sets are constructed varies- some are scRNA-seq-based marker genes, identified in a similar way to this pipeline. In contrast, some use ontology and collation of historical publications to link genes in known biological pathways. Enrichr is not investigated here, but it is explored in the next chapter in the context of binary matrix factorisation.

4.9.2 Differentially expressed genes

DEGs are used for manual annotation or sometimes under the hood for some automated annotation methods. There is a certain circularity in identifying DEGs from clusters. Clustering uses selected features (or projections thereof) to separate underlying data into groups; hence, many DEGs are likely part of the initial feature selection. Although if selected genes are informative, perhaps this is not an issue. This section investigates how different upstream pipeline choices, particularly selected genes, affect identified DEGs.

To calculate differentially expressed genes, a Wilcoxon rank sum test quantifies whether cells assigned a specific cluster are likely from a different distribution than cells not assigned that cluster, for each gene [106, 115]. Multiple correction with Benjamini-Hochberg is used. Further, a filtering approach is used to select genes that are more specific to each cluster (as the above can give the same DEGs for multiple clusters). Here, genes are selected if they are expressed in at least $b_1\%$ inside a group and at most by $b_2\%$ in any other group, giving potential marker genes for a cluster. Results using the filtering approach have $b_1 = b_2 = 20\%$, and use SCRAN normalised data as the data for comparison as recommended in [115].

Comparing the Z-scores of selected features, **Figure 4.18**, highly deviant genes then HVG SCRAN genes have shifted higher Z scores. SCRAN HVGs have low Z scores in general, but the tail is shifted higher for these- although batch correction

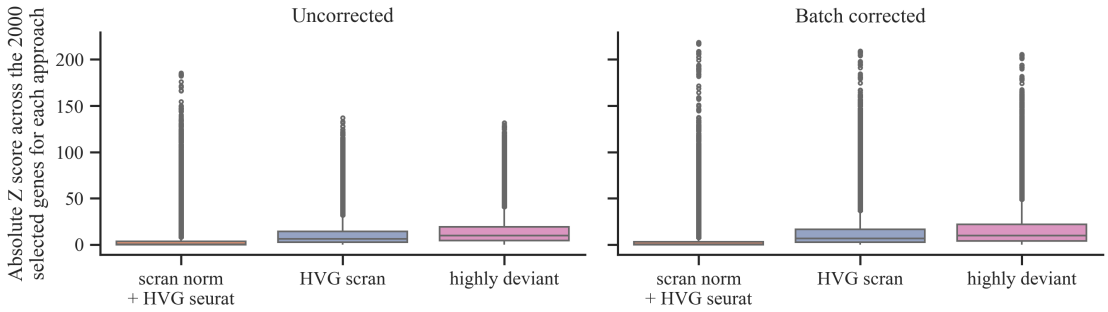
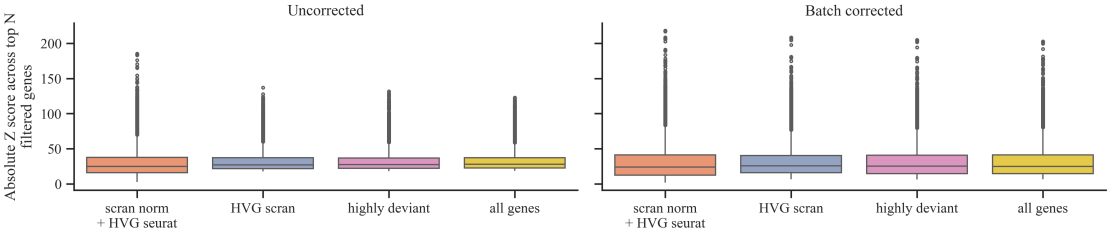
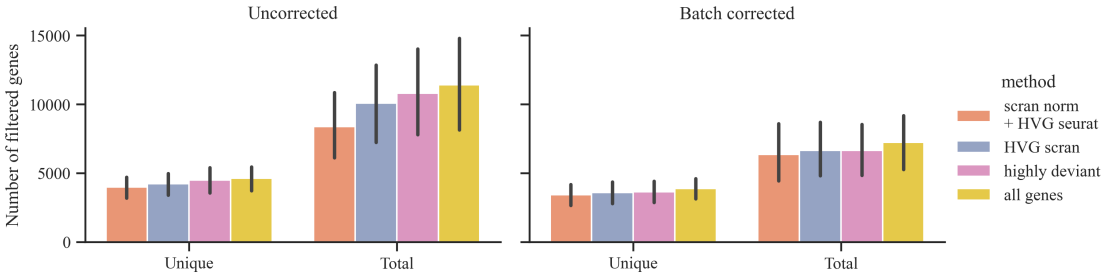


Figure 4.18 Differential Z score distribution of feature-selected genes after clustering. On representative dataset, uses Wilcoxon rank-sum test, across all clusters. For a given cluster, compares cells in cluster to those not in cluster, across all genes.



(a) Distribution of z scores across top filtered genes for each method on each base data normalisation, these have been matched to select the same number of genes for each base data normalisation. Taken over the representative dataset.



(b) Number of filtered genes per method, also selecting unique genes across all datasets.

Figure 4.19 Filtered gene properties

appears to remove some of this signal. This is supported by cluster findings that suggested Seurat HVGs may be batch-specific.

The filtered genes provide a set of potential marker genes for each identified cluster (as they have higher expression in a given cluster and lower expression in other clusters). Despite this, such genes are not unique to clusters with the number of unique genes less than half the total filtered genes, indicating duplicates across clusters **Figure 4.19b**. The total number of filtered genes is likely linked to the number of clusters, **Figure 4.14b**. All methods have a similar number of

unique filtered genes, which may reflect the number of genes with enough signal to pass the $b_1 = 20\%$ filter threshold.

Comparing the same number of top filtered genes across methods, most feature selection approaches have very similar distributions **Figure 4.19a**. Seurat HVGs, however, have shifted lower Z-scores across most top genes with a longer right skew. This suggests that Seurat HVG clusters are based on differentiating a few genes very well, as opposed to other methods where clusters differentiate across more genes less 'cleanly'. Batch correction changes this right skew, likely reducing the impact of batch-related high variance genes on clustering and hence resulting DEGs.

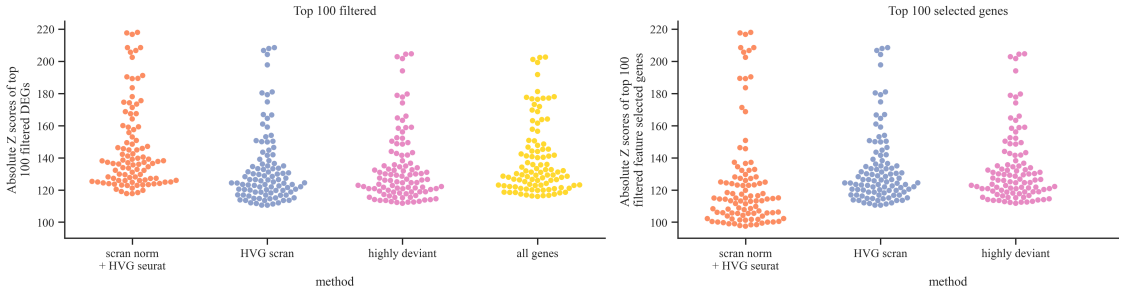


Figure 4.20 Top filtered genes as marker genes. Swarm plot 100 filtered genes (left), and top 100 feature-selected genes (right) on the representative, batch-corrected dataset

Often, a smaller subset of filtered genes is used to define so-called marker genes, particularly if the studied tissue or disease state is novel. Hence, the top 100 filtered genes are used as 'marker genes' **Figure 4.20**. Of interest is that the top DEGs for SCRAN HVGs and deviant are also those used for feature selection, and align with each other and those from the clustering approach without feature selection.

The Seurat HVG top 100 DEGs are somewhat different from those used for feature selection and are slightly higher even than other approaches. However, the highest DEGs for resulting Seurat clusters are those also selected as HVGs. In the context of all other results, it seems unlikely that Seurat HVG clusters are more correct, as it would imply that clustering is more accurate on fewer effective genes (given PCA/variance contribution is low for many selected genes). Likely, this increase is again due to fewer features driving clustering- genes that correlate with these are also DEGs, but the majority of features have lower differentiation

than other approaches **Figure 4.19a**. This also demonstrates that the strength of the top DEGs should not be taken as a proxy for accuracy.

4.9.3 CellTypist

In contrast to using DEGs for annotation, there are many automated alternatives of varying sophistication. CellTypist is one such method, Xu et al. [123], that has seen popularity in literature. CellTypist works by first training prediction models on reference datasets. For a reference dataset, cells are annotated, often using the above manual assignment approach, and this is treated as 'ground-truth'. A model is trained to predict the class of cell (using multiple logistic regression), from a subset of selected features, returning a 'confidence' of the cell belonging to that class. On a novel dataset, a pretrained model can then predict the probability of any cell belonging to each reference class. Each cell is assigned the class with the highest probability. Optionally, a clustering can be superimposed, where cells are reassigned cell types based on the mode of classes in each cluster (i.e majority voting). Manual inspection is also required to ensure no new cell types exist in the data (e.g with a low assignment probability).

A method like CellTypist relies on the reference dataset mapping well to the 'query' dataset, and that both labels and genes generalise to new data. Further, it is unclear how well-calibrated such a model is for distribution shift between the reference and query datasets (i.e how much confidence estimates can be trusted in new data). Given the flaws highlighted in the above pipeline, such issues may propagate into identified marker genes and annotations selected in the pretrained CellTypist models. The downstream effect of such misidentification is difficult to quantify, given the issues related to distribution shift.

Despite these caveats, confidence scores are grouped according to different cluster results across different lung-related pretrained models, with the idea that a higher confidence in cluster labels likely means a more accurate clustering result. **Figure 4.21** shows the confidence scores for each cell in its annotated label, based on its cluster assignment majority vote, on the representative dataset. Given the

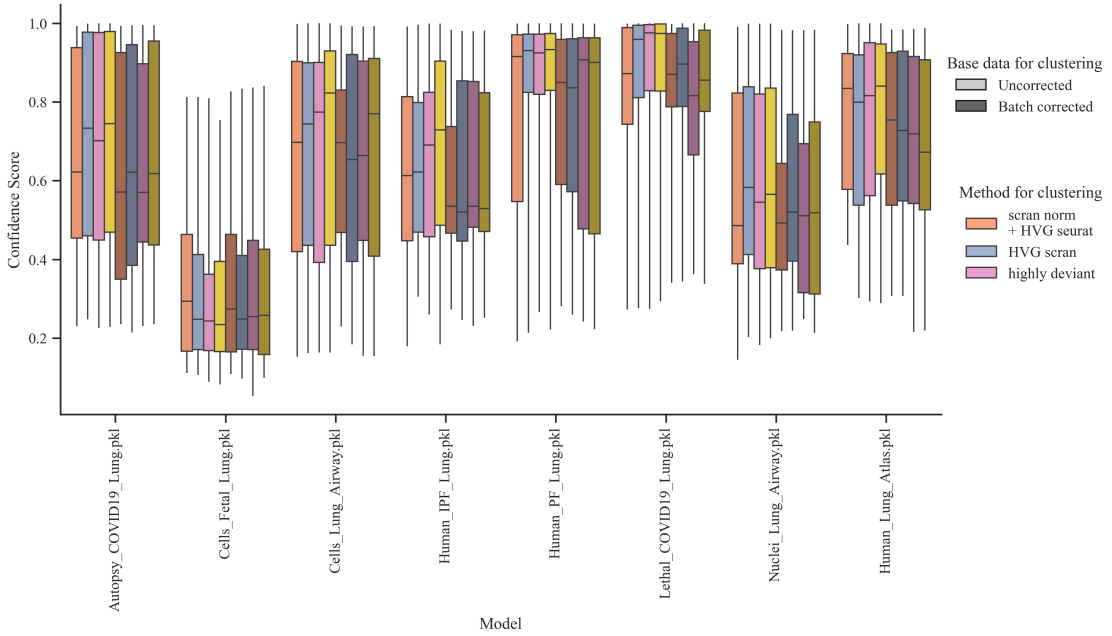


Figure 4.21 Reference dataset across different lung-related CellTypist pre-trained models. Scores are average across cells in each cluster. Error bars indicate standard error across clusters. Each model gives one set of prediction scores, which are then grouped according to each cluster result.

'Lethal_COVID19_Lung' model has the highest confidence scores, the confidence across all datasets are also plotted for this particular model, **Figure 4.22**. There is no clear better or worse base data or feature selection method based on these confidence scores, except perhaps that Seurat HVG has lower confidence scores in most cases.

Of interest is that cluster results from batch correction align less with confidence scores from CellTypist. This suggests that batch-correction clusters miss some biological signal. CellTypist may be less susceptible to batch effects in the query dataset, given it looks to find query signatures based on reference data. However, there could be reproducible confounding batch effects (e.g stress response). It may also be a consequence of the introduced noise that was seen to affect clustering results of batch-corrected data, **Figure 4.15**.

Other differences are difficult to discern, likely due to compounding errors of annotation and clustering misclassification with distribution shift. Although

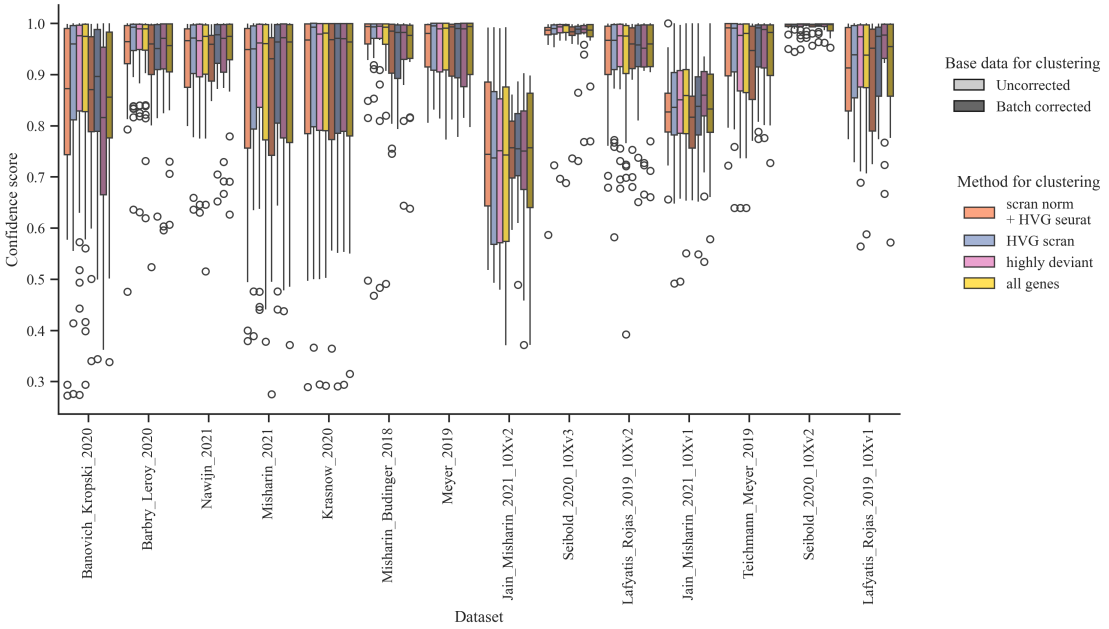


Figure 4.22 CellTypist 'Lethal_COVID19_Lung' model across all HLCA datasets. Scores are average across cells in each cluster. Error bars indicate standard error across clusters. The model gives one set of prediction scores, which are then grouped according to each cluster result.

automated, CellTypist annotations are difficult to interpret when confidence scores are low, hence the need for more manual approaches.

4.9.4 Summary

To summarise some of the key findings above:

- Seurat HVG DEGs are less differentially expressed for the majority of genes, but have a few genes with very high differential expression. This suggests clustering choices are made based on fewer genes for this approach, which is unlikely to be correct.
- Results using PCA with all genes are similar to deviant and SCRAN HVG feature selection, and it is unclear whether feature selection is required.
- Batch correction removes several DEGs.
- CellTypist appears to align better with clusters from uncorrected data, which may be a consequence of batch-correction removing true signal, or introducing noise into cluster results.
- Genes chosen as marker genes from DEGs are influenced by features selected for clustering.

Finally, although I focus here on the use of DEGs for annotating and identifying marker genes for novel cell types, some workflows integrate data across modalities or perform independent experimental validation, increasing the reliability of resulting annotations and marker genes [125, 126]. Annotations in new datasets resulting from high overlap with such marker genes are more likely to be accurate. Moreover, while the results indicate clustering outcomes can be ambiguous and influenced by feature selection, some cell types exhibit such strong expression signals that they are consistently distinguishable [129]. Thus, although the pipeline introduces several potential sources of error, these are more likely to affect rarer or less distinct cell types.

4.10 Conclusions

This chapter provides background to the standard scRNA sequencing pipeline. However, throughout, I have also evaluated the assumptions of each step in the

context of the underlying UMI counts. For the most part, the analysis uses Scanpy, the popular Python library that is the basis for many online tutorials.

The main conclusions from this chapter are:

- The data distribution in scRNA-seq is extremely sparse. Different statistical models make different assumptions, but many genes are likely to fall into a Bernoulli distribution.
- Under a Bernoulli distribution, some genes may be overlooked as having expected variance when they could still be biologically relevant (determinable through patterns of expression with other genes).
- Cell-specific normalisation may be unnecessary as relationships between cellular UMIs and gene UMIs are confounded by zero-proportions.
- SCRAN and log1p normalisation give similar results for highly variable gene selection.
- Genes should not be selected using the Seurat approach, as many uninformative genes are selected. This is the default in Scanpy and most scRNA-seq analyses in the literature. Increasing the number of genes makes it more similar to other approaches.
- SCRAN-based gene selection and highly deviant gene selection are more similar. Highly deviant genes seem to be a proxy for genes with the highest mean, and select more housekeeping genes than other methods.
- If SCRAN-normalised data is used, then scaling the data for PCA does not make that much of a difference.
- Performing PCA on the complete data did not seem to negatively affect results, and maybe should be used to avoid potential biases through feature selection.
- Harmony batch-correction method introduces added noise to clustering results, and is less robust to selected genes.
- Cluster results are difficult to replicate, suggesting that these should be used with caution- particularly for cell-type abundance analysis, which is likely to be most impacted by clustering robustness.

- Annotation that finds marker genes using DEGs can be somewhat circular, as the highest DEGs are often those used in feature selection. This has a larger impact when the feature selection method does not capture all relevant information.
- Reference datasets often treat cell types/marker genes that have been annotated using this pipeline as ground truth. However, these likely contain errors that propagate when validating methods or annotating using these labels.

Overall, there seems to be much room for improvement of the scRNA-seq pipeline. Some procedures seem to have been transported from bulk RNA-seq, and should be reevaluated in the context of the sparse and near-binary single-cell approach. Using summary statistics can be misleading given the data distribution. Further, validating methods through agreement with historical computational-based annotations is flawed and should be avoided where possible.

One key assumption made throughout the scRNA-seq pipeline is that clustering is the best approach to capture cellular identity. In the following chapter, I examine this assumption more closely, specifically by comparing the clustering model to factorisation-based approaches.

4.11 Limitations and Future Work

Although I have tried to be comprehensive in both my analysis and the conclusions that can be drawn from it, there are a few limitations with the results presented. Firstly, the data is all from the same source, both in terms of upstream QC/raw processing as well as the tissue of origin (lung). Sparsity of the data was compared across different datasets, and most had comparable sparsity. However, some statements around distribution should likely be investigated in completely new datasets.

Furthermore, as mentioned in the R vs Python paradigm, there exist two different established software tools- the scverse (Scanpy and other variants, often from the Theis lab), and R-based libraries of Seurat, SCRAN and 'single-cell experiment' (SCE). Although the programming language should not matter, as this chapter has

highlighted, many choices exist at each stage of the pipeline, and different libraries make different choices. Rich et al. [130] showed the low agreement between different pipelines and even versions of the same pipeline. This approach has used Scanpy tools, given that many of the tutorials and best practices also use this approach. However, it would be interesting to look at using a completely R-based pipeline to determine whether some of the same apparent pitfalls exist. Particularly with the Seurat HVGs, as the Scanpy default approach uses the original [114] implementation, and it is unclear to what extent newer Seurat versions may have changed this.

Finally, some results are difficult to interpret and may be affected by nuances in the data. For example, cluster results are confounded by the number of clusters as are the number of DEGs and the extremity of DEGs. Likely, permutation tests or simulation are required to understand how such factors influence interpretation or significance.

5

Binary matrix factorisation for single cell RNA sequencing

Contents

5.1 Introduction

In **Chapter 4**, I introduced the standard RNA sequencing pipeline, where it was clear that many upstream assumptions could impact the downstream identification of cell types and corresponding marker genes. A significant assumption, which I explore here, is whether a clustering approach is the correct model for identifying cellular heterogeneity and gene programs. Another approach, which has had some success in scRNA-seq, yet remains overshadowed by clustering-based approaches, is factorisation. Here, I explore factorisation, and particularly Boolean matrix factorisation (BMF), as an alternative to the standard pipeline.

In **Chapter 3**, I introduced a novel BMF approach, `bfact`, and showed that it worked well in comparison to other existing BMF approaches, particularly on sparse scRNA-seq data. This chapter aims to evaluate `bfact` in a more bioinformatics sense, investigating whether BMF is a viable option for scRNA-seq compared to existing approaches. Hence, I investigate both the rationale and empirical results of whether using BMF or other factorisation might be a better approach than clustering.

5.2 Background

5.2.1 Factorisation or clustering

Clustering and factorisation are both unsupervised learning techniques that model data using latent variables. Here, let X be some numerical observed data matrix with M observations and N features. Both approaches attempt to find some lower-dimensional representation of the data, where each observation i in X is somehow constructed from $K \ll N$ distinct groups.

For clustering, it is assumed that each $i \in \{1 \dots M\}$ has an associated latent variable $0 \leq l_{ik} \leq 1$ denoting the probability that observation i belongs to group $k \in \{1 \dots K\}$. Hence, $\sum_k l_{ik} = 1$. In hard clustering, an additional constraint is imposed: $l_{ik} \in \{0, 1\}$. Each group has a centroid/mean r_k of dimension N, such

that $\mathrm{E}[x_i] = \sum_k l_{ik} r_k$. Hence, clustering can be written as $X \approx LR$, with constraints $0 \leq l_{ik} \leq 1$ and $\sum_k l_{ik} = 1 \ \forall i \in \{1 \dots M\}$, where $L : M \times K$ is the matrix of cluster probabilities and $R : K \times N$, the matrix of cluster means. Depending on the assumed model (e.g hard clustering, Gaussian mixture model) additional constraints are imposed to define each L and R.

In contrast, standard factorisation assumes there are K latent factors, but each observation is a linear combination of those factors. This gives $X = LR$ for matrices $L : M \times K$ and $R : K \times N$. Different factorisation approaches make different assumptions. For example, in non-negative matrix factorisation (NMF), it is assumed that all $x_{ij}, l_{ik}, r_{kj} \geq 0$, i.e., they are non-negative. When clustering and factorisation are written as above, it is clear they share very similar constructions, with differences depending on imposed constraints. Clustering is usually viewed as an observation belonging to a particular group. In contrast, a factorisation can be thought of as multiple groups underpinning the data, with each observation composed of any combination of these groups. Hence, factorisation tends to be more expressive, as it has fewer constraints on the left matrix L. However, the additional expressiveness can come with issues around both interpretation and identifiability (i.e multiple factorisations give the same reconstruction). To see this consider $X = LR = LPP^{-1}R = \hat{L}\hat{R}$, for some invertible matrix P. Hence, multiple left and right matrices exist that decompose such a matrix. The addition of constraints in factorisation increases the identifiability of the factorisation.

In **Chapter 3**, I introduced boolean matrix factorisation (BMF), given by $X_b = \theta(LR)$ This is not a standard factorisation because of the addition of boolean logic, i.e $\theta(1+1) = 1$, and the constraints that both $l_{ik}, r_{kj} \in \{0, 1\}$ (and assumed $x_{b;ij} \in \{0, 1\}$). In the context of scRNA-seq, this makes BMF particularly interpretable. Each latent factor can be seen as containing a set of genes, and each cell is explained by the presence or absence of these factor genes. In contrast, standard or NMF matrix factorisation allows real-valued L and R, enabling more specific reconstructions but often at the cost of interpretability, since reconstructions for each observation depend on some combination of factor usage. This chapter also

introduced the concept of disjoint-BMF, in which Boolean logic is no longer required, as it is assumed that $\sum_k r_{jk} \leq 1, \ \forall j \in 1 \dots N$.

Figure 5.1 gives an overview of the different modelling approaches. Clustering does not illuminate shared processes between clusters. When the true model is a factorisation, then identified clusters are usually some combination of the underlying factors. Hence, if cell-type-specific factors exist, clustering may not uniquely identify the drivers of the factor/cell-type or may identify the intersubsection of different factors as new cell-types.

NMF shows more of the similarities but on a continuous scale, and different L and R combinations could give the same reconstruction, making it harder to interpret. BMF makes it clear both what groups of cells have similar expression profiles, as well as highlighting the common factors between groups and the genes expressed in each of these factors. Disjoint-BMF does not construct the original X^b as well as pure BMF, as all profiles in R are disjoint.

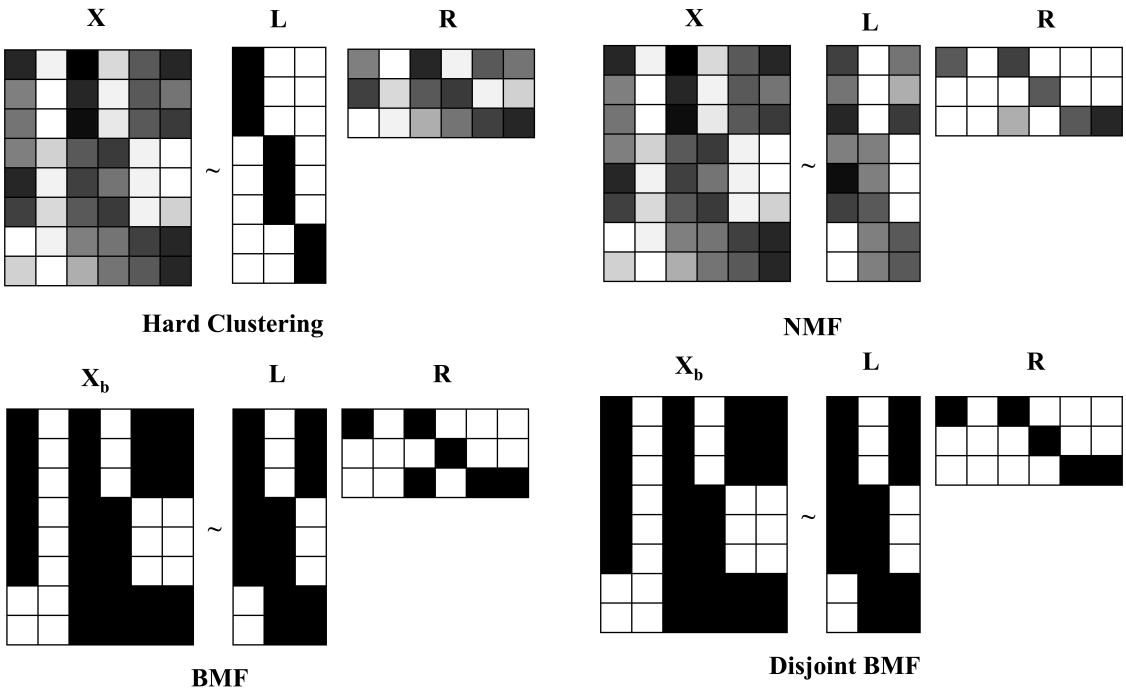


Figure 5.1 Visualisation of differences between hard clustering, NMF and BMF. Disjoint BMF has disjoint R rows, resulting in a slightly worse reconstruction error of the original matrix X^b; for R to be disjoint, $r_{3,3}$ must be set to 0 ($r_{1,3}$ should remain as 1, since reconstruction error will be lower, given three observations contain factor 1 while only two observations contain factor 3).

5.2.2 Evaluating models in scRNA-seq

Various factorisation methods have been applied to scRNA-seq, with NMF being the most common. However, it remains less widely used than clustering, partly due to higher computational cost, sensitivity to initialisation/parameters, and because clustering approaches were adopted early in popular tools such as Seurat and Scanpy, becoming the default by convention [106, 131].

A further challenge is comparing models with different assumptions. Simulations used to benchmark models often presuppose an underlying clustering or NMF approach, inherently favouring models that align with the simulations [132]. The consequences of using misspecified models (e.g. clustering) on real data remain unclear. Moreover, validation practices in scRNA-seq pipelines reinforce clustering biases: cell-type annotations derived via clustering favour factorisations that reproduce cluster-like results, and historical marker genes often identified through

HVG selection bias GSEA towards methods using HVGs [115]. Although HVGs tend to be robust across datasets and thus biologically meaningful, reliance on them risks overlooking other relevant genes with lower dispersion due to sparsity.

5.2.3 Existing Factorisation Applied to scRNAseq

Non-negative matrix factorisation is a method that gained traction in bulk RNA-seq, for cell type deconvolution and metagene discovery. However, it has also seen some popularity in scRNA-seq [133]. In NMF, the left-hand matrix, L is termed the usage matrix and the right-hand matrix, R is referred to as the gene expression programs (GEPs), where each factor has an associated GEP, and each cell has an associated usage of factors. This nomenclature is adopted for the rest of the section.

Many different NMF approaches have been developed in the context of scRNA-seq [134–138]. However, most do not compare with the standard clustering approach, and often use existing approaches as a validation method (i.e using cluster-based cell type labels to validate on real data). Kotliar et al. [136] performs NMF on a set of HVGs of the full matrix and subsamples the matrix to find robust factors that exist across all subsamples. This method has been chosen for further exploration, as it is both used by others in the field [139, 140], has a well-maintained Python library, and is well-documented. It does not automate rank approximation.

Using BMF in the context of scRNA-seq is not an entirely novel concept, [141–144]. However, most of these approaches are published in traditionally methods-focused journals (IEEE, AAAI, ICML), introducing novel methodology as opposed to demonstrating the utility of using BMF over other approaches in scRNAseq. Very few of these show downstream implications for using BMF for discovery, nor have they maintained code libraries.

However, Liang, Zhu and Lu [141] do present and explore BMF for scRNA-seq in a more applied way. They demonstrate superior performance of their method over NMF and traditional clustering, showing factors derived from their model were better at predicting response to immunotherapy in a new cohort (albeit with a very small sample size, N=19, using cross-validation). Their comparisons to both clustering

and NMF on scRNA-seq were limited to a single dataset, using well-characterised immune cells. Their approach does not automate rank approximation.

5.2.4 Linear additive assumption

The default assumption in most scRNA-seq approaches is that a linear additive model is a good underlying model for the data. That is, if a gene is involved in multiple GEPs, then the expected expression in a cell with both programs is the same as the sum of the expected expressions of cells with either one or the other program. However, there is much evidence to suggest that gene expression is not necessarily additive. In this case, relevant genes for processes may be overlooked, and the effects of certain genes might be underestimated or overestimated.

Gene regulation is shaped by complex transcriptional networks, epigenetic modifications, feedback loops, and post-transcriptional mechanisms [145]. For example, transcription factors often exhibit cooperative binding or saturation effects, precluding linear responses to upstream signals [146]. Combinatorial regulation further enables the same gene to be differentially expressed depending on cellular context [147]. Further, systems biology models demonstrate how motifs like feedforward loops and negative feedback yield sub-additive or switch-like outputs [148]. Hence, assuming a linear additive model, such as in NMF and clustering approaches, where relative abundance is the focus, may not be a desirable model for detecting underlying GEPs.

Further, in **Chapter 4**, it was shown that most signal in scRNA-seq data has very low UMI count. Even if transcription was additive in regulation, the sparsity of the data is unlikely to reflect true cellular abundance due to capture efficiency, reverse transcription biases, and transience effects [76, 85]. Hence, an approach that focuses on the presence of expression rather than the level of expression may find more substructure underpinning the data, as well as marker genes that are expressed consistently at low levels.

5.2.5 Zeros in scRNA-seq

In scRNA-seq, "dropouts" refer to the apparent excess of zero counts for many genes in individual cells, historically interpreted as technical failures of capture or amplification. However, recent studies have challenged the notion that scRNA-seq data are intrinsically "zero-inflated" beyond what standard count models predict, suggesting that many zeros result from biological heterogeneity [77, 86]. Rather than treating zeros as noise, they can be viewed as informative features that encode biologically relevant sparsity.

5.2.6 Motivation

Given results in **Chapter 4**, I hypothesised that a BMF or disjoint-BMF approach might be a better approach for scRNA-seq, for the following reasons:

- No preprocessing, naturally taking advantage of the near-binaryness of the data. Only one assumed model, as opposed to a pipeline of many steps with many assumptions.
- Batch/cell-specific abundance effects have less impact due to binarisation. Further, batch-specific gene programs can be modelled as additional factors.
- More natural representation of data as a composition of cellular processes, rather than mutually exclusive ones. For example, cell cycle or stress responses can be present among other cellular signatures.
- More natural interpretation of factors, using GEPs, which are known to underpin the variation in the data, rather than post-hoc selection of marker genes.
- More biologically motivated, no assumed additive effect or assumption that data represents relative abundance.
- Further constrained than NMF, lower likelihood of non-unique solution or data overfitting due to the binary restriction.

However, there are also reasons why such a model may not be the best fit:

- By binarising data, there is some information loss for those cell-gene pairs that have a higher number of UMI counts.
- Incomplete 'penetrance' certain genes may be good markers for cell types despite only being expressed in a subset of cells of that type. Unlike clustering or NMF, standard BMF requires at least 50% of cells with a unique marker gene to express that gene for it to be included in the marker gene set.
- BMF prioritises conserved patterns of expression over more highly expressed gene markers, both an advantage or disadvantage depending on context.
- BMF can capture non-linear additive/saturation but cannot model negative regulation effects (i.e if the presence of another biological process removes the expression of a gene in another)
- Disjoint BMF assumes gene uniqueness to GEPs; however, some genes may naturally belong to multiple GEPs. Hence, the assignment of some genes to specific GEPs may be misleading or non-robust.
- Longer computational time.

Hence, our goal for this chapter is to evaluate whether BMF is a promising direction for future scRNA-seq analysis or whether the choice of clustering, NMF or BMF has little effect on identified marker genes or cell types.

5.3 Methods

5.3.1 Data

Like **Chapter 4**, the same meta dataset from the human lung cell atlas (HLCA) is used, consisting of 14 different single cell datasets, all taken from lung-related tissues [74]. Each dataset is analysed individually, and it is assumed that the datasets should capture similar cell types and biological processes. In some cases, the HLCA annotations performed on the metadataset are also considered [74]. Here, batch correction was performed to control for the dataset of origin, followed by clustering on the whole meta-dataset, with labels then assigned using expert teams.

For housekeeping genes, a reference list is used from the housekeeping transcript atlas [109].

5.3.2 Overview

To evaluate `bfact` in comparison to NMF and traditional Leiden clustering, the following is performed:

- Exploratory analysis of genes identified in NMF/BMF compared with those selected in traditional feature selection. Proportion of housekeeping genes and conserved DEGs across datasets and methods.
- Summary metrics, including reconstruction error and F-score, detailed below.
- Gene set enrichment analysis using EnrichR reference datasets.

5.3.3 Summary metrics

A hybrid approach for summary metrics is used to evaluate the differences between clustering, NMF and BMF. In particular, the reconstruction error is calculated for each model, as well as treating each model-derived GEP matrix, R, as constant and refitting an NMF based on each. This gives a better indication of the accuracy of GEPs, partially controlling for modelling expressiveness (as NMF is the most expressive model). Each result is also adapted to give continuous reconstructions and binary reconstructions. Further, the F-score of each model is considered, based on the binary reconstruction. The derivation of each of these is outlined below.

In addition to using these metrics within each dataset, the GEPs predicted from one dataset are also used as the basis for the GEPs in every other dataset, treating them as signatures to predict the optimal cell usage of each GEP. This serves as a proxy for how well each approach captures the biological signal that is likely to generalise to other datasets, rather than overfitting to dataset-specific variability.

5.3.4 Implemented Methods

Clustering

The clustering approach follows a similar methodology as in **Chapter 4**, according to the same pipeline, but again using different feature selection methods, according to:

1. Select 2000 genes according to each approach: Seurat HVG, SCRAN HVG and highly deviant.
2. Perform SCRAN normalisation.
3. Perform PCA with 50PCs.
4. Perform Harmony batch correction.
5. Use Leiden clustering, with 15 nearest neighbours and resolution 1.
6. These are not annotated; instead, each cluster is used for downstream analysis.

To find continuous R_1 for each clustering result on dataset X_1, the mean expression of cells in each cluster are taken, and L_1 is taken as the one-hot encoded matrix of which cluster cells belong to. This gives $r_{1,kj} = \frac{1}{\sum_i l_{1,ik}} \sum_i l_{1,ik} x_{1,ij}$. Given R_1 should represent the GEP for each cluster, this is modified slightly, such that the above is for genes that are among the top filtered DEGs of all clusters, and any other genes are set to 0.

To obtain a binary estimate, each gene in a cluster is set to 1 if more than half of the cells assigned to that cluster express the gene. This is the same as: $r^b_{1,kj} = \text{round}(\frac{1}{\sum_i l_{1,ik}} \sum_i l_{1,ik} x^b_{1,ij})$, where X^b is binarised based on $x_{ij} > 0$. This gives the best reconstruction error for $X^b_1 = L_1 R^b_1$ given that the L_1 are disjoint (i.e each cell belongs to only one cluster).

To find L_1^{NMF} for hybrid NMF, non-negative least squares is used, where $X_1 \approx L_1^{\text{NMF}} R_1$ and R_1 is derived as above (R_1 is non-negative given it is based on scran-normalised data).

To calculate the reconstruction error (and associated metrics) in a new dataset, X_2 based on identified GEPs, the best L_2 is found that yields $X_2 \approx L_2 R_1$. For a hard clustering approach, L_2 is the row in R_1 closest to each observation in X_2.

Hence, $l_{2,ik} = 1$ if $k = \arg\min_{k'} \sum_j (r_{1,k'j} - x_{2,ij})^2$, and otherwise $l_{2,ik} = 0$. The hybrid-NMF also finds L_2^{NMF} according to non-negative least squares that gives the best approximation to $X_2 \approx L_2^{\text{NMF}} R_1$.

NMF

For the NMF approach, consensus NMF (cNMF) was implemented as introduced in Kotliar et al. [136]. Implementation follows documentation instructions and uses an input $K = 20$ for all datasets, with 2000 highly variable genes, and 20 consensus replicates. Under the hood, cNMF implements its own highly variable gene routine, which is similar to SCRAN HVGs. It fits a Poisson model, then finds genes that have a higher than expected variance under this model, selecting those that deviate most from expectation.

The method works by running NMF for 20 factors, 20 times independently with different gene NMF seed initialisations (replicates), using the Python library scikit-learn NMF implementation. cNMF then clusters the factors across all iterations to group similar factors. It averages the gene programs for factors in the same cluster, as well as the usage matrices for cells in a factor, to get robust factorisations. By default, it uses counts per million (normalised by total cell count). It also normalised the cell usage matrix to sum to 1, so can also be thought of as a soft clustering appraoch.

For a dataset X_1, the cNMF result is $X_1 \approx L_1 R_1'$, where R_1' is the rescaled output of cNMF, to be equivalent to a single UMI count. $X_1^b \approx (L_1 R_1') > 0.5$ is used to obtain a binary estimate on the reconstructed NMF. To calculate the reconstruction error (and associated metrics) in a new dataset, non-negative least squares is used to obtain $X_2 \approx L_2 R_1$, where R_1 is the GEP profile derived from dataset 1 (using NMF to get $X_1 \approx L_1 R_1$).

`bfact`

Raw count data are binarised based on whether a gene-cell entry has a UMI count greater than 0. Genes are removed that are expressed in fewer than 0.5% of cells, or in more than 95.5% of cells and cells are removed that express fewer than 10

genes or more than 10000 genes. `bfact` is performed on this subset; however, all metrics are considered on the whole dataset, binarised if UMI is greater than 0.

Instead of using a pure BMF, a variant of `bfact` is used (using only w-RMP, introduced in **Chapter 3**) to get (nearly) disjoint GEPs, as the reconstruction error and F-score using a normal BMF were comparable, **Figure 5.8**. The implications of this are explored in **Section 5.4.5**, namely that any non-linear non-additivity benefits from BMF are lost. This suggests that even if non-additivity is a suitable model, other assumptions of BMF, combined with data sparsity, do not result in significant improvements. Note, under disjoint factors, $L_1^b R_1^b = \theta(L_1^b R_1^b)$.

Having disjoint gene sets is favourable as it ensures that GEPs expressed in all cells are considered as individual factors rather than being absorbed into different factors. Furthermore, using the disjoint BMF from `bfact`, as discussed in **Chapter 3**, the w-RMP can be used to obtain a kind of binary inverse, for $X_2^b \approx L_2^b R_1^b$ when R_1^b and X_2^b are known. Hence, to obtain the reconstruction error for a new dataset X_2^b the usage L_2^b is found using this binary inverse, assuming known binary GEPs, R_1^b.

To obtain a hybrid NMF using the BMF-derived GEPs, $X_1 \approx L_1^{\text{NMF}} R_1$, where the binary R_1 is derived from the BMF, but L_1^{NMF} is found by fitting non-negative least squares. To get a continuous approximation for X_1 using $L_1^b R_1^b$, the mean of X is found across all cells assigned a factor, according to $\hat{X}_1 = L_1^b R_1^b \frac{1}{M} \sum_i (L_1^b R_1^b)_i X_i$.

5.3.5 GSEA

The Python implementation of Enrichr is used to perform gene-set enrichment analysis [128]. Enrichr hosts several reference marker databases, for example, gene sets known to be associated with cell types, biological processes or GWAS results [127]. Enrichr works by comparing a candidate gene set against a reference database to see if it is enriched in any process more than expected by chance. Reference databases comprise multiple reference sets, and FDR correction is employed to adjust p-values across these sets.

Consider a reference set S and a user-provided marker gene set L. Let $n = |L|$ and $m = |S|$, with the background number of genes N. Then, a hypergeometric test is used to test whether the intersection, I, of two sets is more than expected by chance, given by:

$$P(I \geq i) = \sum_{l=i}^{\min(m,n)} \frac{\binom{m}{l}\binom{N-m}{n-l}}{\binom{N}{n}} \tag{5.1}$$

This can be used to obtain a Z score using the overlap relative to the expectation under a random null.

There is considerable debate over what the background reference genes, N, should be [149]. In general, it should be restricted to the set of background genes expressed in the sample, or genes relevant to the tissue, depending on context. If too many background genes are used, this can also inflate test statistics. The background set of genes used here are those that pass the binary threshold (i.e., they must be expressed in at least 0.5% of cells).

For the GSEA analysis, multiple Enrichr databases are used to examine the enrichment of each factor in each database. The databases used are:

- Azimuth 2023 - this is a database based mostly on scRNA-seq (with some ATAC-seq). References to HLCA are dropped to avoid leakage, as are references to Lung V1 (based on Krasnow 2020) for Krasnow 2020 enrichment analysis.
- CellMarker 2024- another database that collates publicly available cell markers. There is leakage here (i.e some datasets from HLCA are used in the reference for CellMarker, which are difficult to filter out, unlike in Azimuth). Many of the marker gene sets are derived from scRNA-seq.
- ChEA 2022- This is a database for ChIP-seq data related to genes that are likely bound by given transcription factors. It is not cell-type specific. Curated from publications/publicly available data.
- ENCODE TF ChIP-seq, the same as above, using ENCODE data.

- GO Biological/Cellular/Molecular function- gene ontology database, another collation of publication results/databases. Not usually scRNA-seq data sources, more focused on molecular biology assays and sequence similarities.
- GWAS Catalogue- groups of genes mapped from SNPs associated with diseases, measured using GWAS.

Note that the p-value is corrected for multiple testing for a given factor marker set in a reference database, but has not been corrected for multiple testing across different factors and different databases. Hence, it should not be interpreted as a literal measure of significance. However, it is still helpful to compare across different marker sets between methods based on these results.

Finally, because scRNA-seq-derived reference marker gene sets initially select for HVGs, there may be a form of confirmation bias/confounding present. Historically identified marker gene sets are likely to come from HVGs used in the computational pipeline for annotation. Hence, if some HVGs are consistent across datasets, then there is a higher likelihood of this being significant in a query dataset, regardless of its biological relevance (although there is an argument for biological relevance and reproducibility across datasets).

5.4 Results

5.4.1 Factor usage/GEP distribution

This section compares the number of genes per factor, **Figure 5.2**, and the cell-based factor assignment, **Figure 5.3**, for both factorisation methods, cNMF and `bfact`-disjoint. cNMF is closer to a clustering approach, where each factor consists of many genes, and each cell consists of a few factors (majority 1 or 2), **Figure 5.3**. In contrast, BMF cells contain a larger range of factors, each of which has only a few genes. The continuous nature of cNMF can fit very low signals to cells, whereas BMF sometimes assigns no factors to cells with presumably low or low-penetrance signals. The high number of genes for cNMF also indicates that the GEPs are less process-specific and have to be filtered to remove standard processes from each

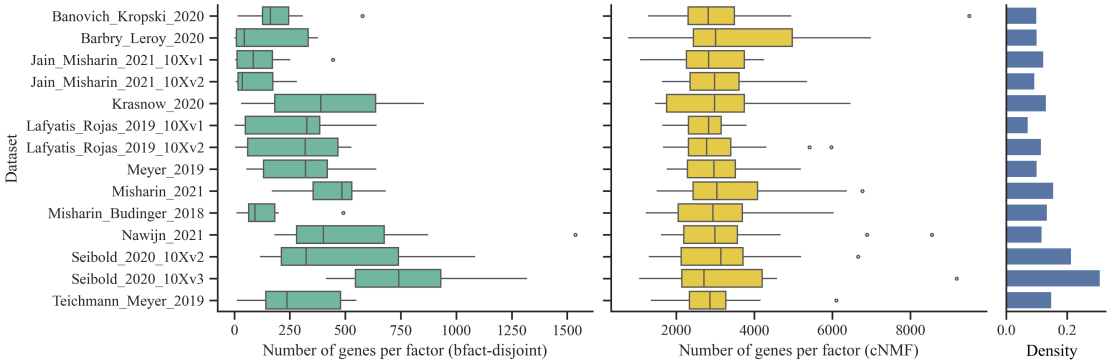


Figure 5.2 Number of genes per factor. For cNMF, which is continuous, a gene is assigned to a factor if the gene's proportion of assigned reads is greater than 5% (proportion taken across factors, in terms of transcripts per million). Error bars taken across factors.

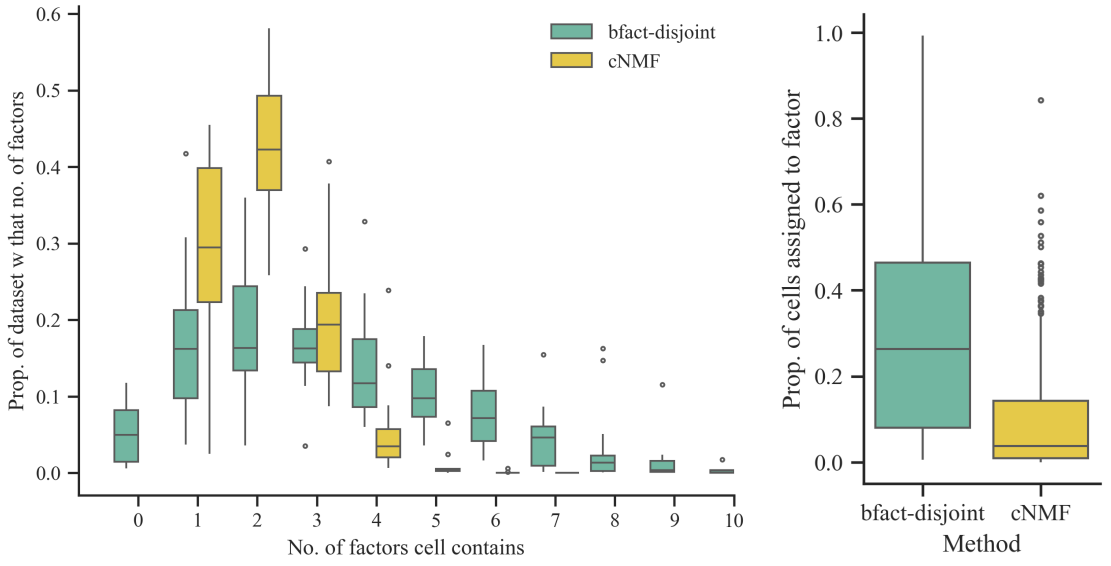


Figure 5.3 Cell factor statistics; number of factors per cell (left) and proportion of cells per factor (right). For cNMF, which is continuous, a factor is assigned to a cell if the proportion of usage is greater than 10% (as this resulted in a high alignment with cluster results in **Figure 5.10**). Error bars taken across datasets. Density measures the proportion of entries in the matrix that are non-zero.

factor. Likely, the fuzzy-clustering approach taken by cNMF forces more disjoint cell representations than disjoint gene representations.

Some factors for BMF contain a high number of genes, which may correspond to background housekeeping genes. The percentage of cells assigned to each factor in `bfact` is also higher, with some very high, which again may correspond to background housekeeping genes. However, some factors have a lower percentage of cells and lower number of genes, which could indicate biologically relevant processes affecting a subset of cells. The number of genes per factor for `bfact` correlates to the density, likely indicating that as capture efficiency increases, so too does the number of genes assigned to factors/biological processes.

5.4.2 Batch effects

This section considers the usage matrices of each cluster approach, L, and in particular how these relate to the sample donor, which is likely to encapsulate batch effects. Leiden clustering approaches have been corrected for batch effects using Harmony. cNMF can optionally use Harmony, but has not been used here to determine if it can distinguish between batch-specific factors and more biological factors.

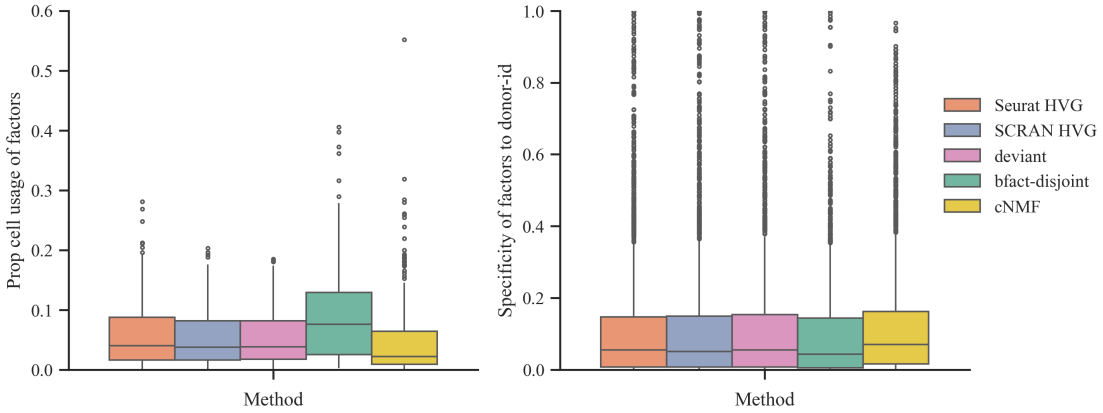


Figure 5.4 Distribution across datasets and factors/clusters of factor-specific statistics. Left: cell usage of factors and right specificity of factor to donor ID. For cNMF, where usage is continuous, the proportion of usage for each factor is used. For left plot, this is given by $\frac{\sum_i l_{ik}}{\sum_{i,k} l_{ik}}$. Right: distribution of specificity of factors to donor IDs, given by $\frac{\sum_{i \in D_d} l_{ik}}{\sum_i l_{ik}}$, where D_d is the set of factors for donor ID d.

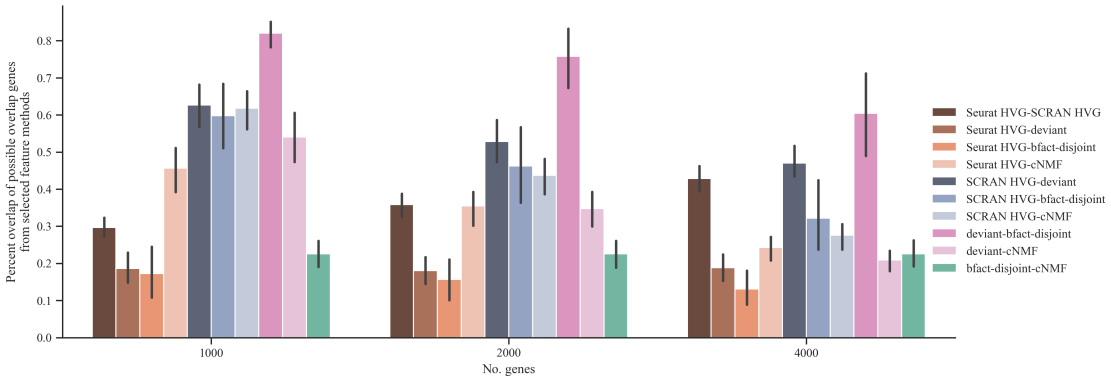
`bfact` has shifted towards higher usage of many factors, **Figure 5.4**, which likely reflects that some factors have a large proportion of cells assigned, probably reflecting uninteresting common biological processes. However, if other methods identified more clusters or factors than exist in the data, then each factor would have a lower proportion of usage, which could also explain some of the lower usage. Conversely, `bfact` factors are shifted lower for specificity to donor ID, implying reduced donor effects. However, some clearer batch-specific factors remain, evidenced by the slight increase in density around 1 for outliers. This pattern may also naturally result from having fewer factors or clusters, as shown in **Figure 5.8**.

5.4.3 Factor genes compared to feature selection methods

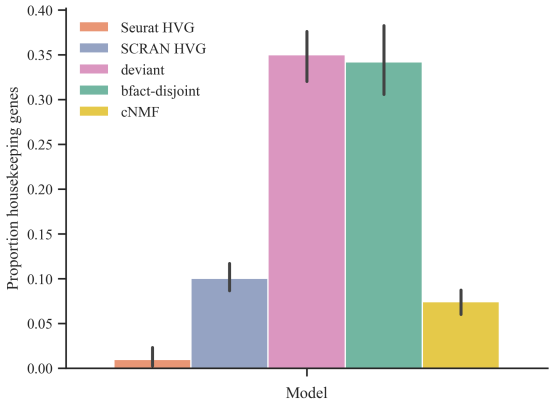
Figure 5.5 compares the genes selected by various approaches. In a clustering approach, there are two sets of relevant genes- those selected by upstream feature selection and those selected as marker genes based on differential expression of identified clusters. Part of the appeal of a BMF is that no feature selection is performed, and relevant genes are an output of the model itself. Hence, I compare `bfact` selected genes to those selected in other approaches. For cNMF, under the hood, it first selects genes in a similar manner to SCRAN-HVG. It also outputs the top 100 DEGs for each factor; hence, I use this to compare the output of cNMF to selected features (all DEGs are considered later as well).

`bfact` and highly deviant genes share the highest percentage of genes, but both also contain a high proportion of housekeeping genes, which may explain their additional shared percentage. In the context of later clustering results (**Figure 5.10**), it is interesting that deviant and `bfact` show low cluster agreement despite sharing around 80% of genes. SCRAN-HVGs share a similar proportion of genes with both `bfact` and cNMF, while `bfact` and cNMF do not share that many genes, despite both being factorisation-based.

Given that cNMF is performed on a set of 2000 HVGs, derived similarly to SCRAN and Seurat, there may be an implicit bias for such DEGs to overlap with these. cNMF exhibits the highest overlap with Seurat HVGs for 1000 selected genes;



(a) Overlap of selected genes for each method, number of selected genes only applies to HVG Seurat, HVG SCRAN and deviant.



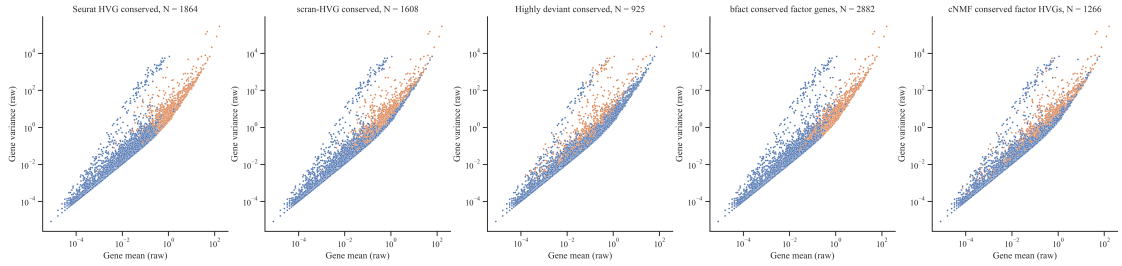
(b) Proportion of housekeeping genes selected in each method across datasets, with 2000 selected genes.

Figure 5.5 Overview of selected genes used to explain data from different methods. For SCRAN HVG, Seurat HVG and deviant, this refers to feature-selected genes. For `bfact`, these are the genes selected in the binary R^b. For cNMF, the genes are the top 100 DEGs of each factor returned from cNMF.

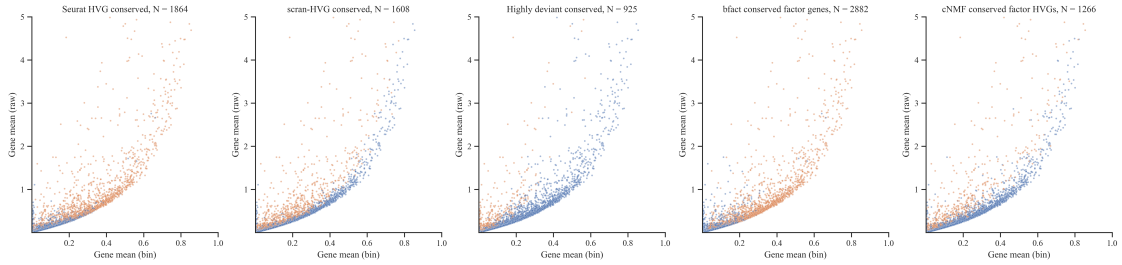
however, it also shows similar overlap with highly deviant and SCRAN. Likely, a set of biologically relevant genes is selected by all methods.

Figure 5.6 explores the distribution, in the representative dataset, of consistently selected genes from each approach. Despite being binarised, `bfact` still selects many genes that have a high non-binarised mean, suggesting that modelling only patterns of expression, rather than abundance levels, still captures relevant information.

For both `bfact` and deviant, the conserved genes are skewed toward a higher mean compared to other methods. Given that both have high housekeeping percentages, a portion of the high-mean genes are likely housekeeping genes. In



(a) Mean vs variance



(b) Binary mean vs actual mean

Figure 5.6 Overview of selected genes used to explain data from different methods. Genes conserved in at least 7 datasets for each method, plotted on representative dataset. The number of conserved genes is contained in the title.

contrast, the conserved genes identified by other methods tend to have higher variance than expected, which is unsurprising given their HVG selection. The consistent high variance of these genes suggests they may be biologically relevant.

In **Chapter 4**, I introduced how the HVG approach will not select potentially biologically relevant genes in the binary extreme because a mixture of Bernoulli distributions is also a Bernoulli distribution. The same applies to Poisson distributions with small λ_1, λ_2. In **Figure 5.6**, `bfact` clearly selects many more genes where the binary and raw mean are similar, which would support that some genes are informative but do not have high dispersions.

Compared to other methods, `bfact` selects fewer high-variance genes. A potential explanation is that abundance effects between cells are missed through binarisation, causing `bfact` to overlook these differences. Another explanation is incomplete penetrance— that is, some markers are not consistently expressed across cells, perhaps reflecting transient expression or transcriptional bursts. In contrast, `bfact` tends to focus on more consistent markers of biological processes, which are expressed in most cells that contain them. This is supported by the observation that other

methods, particularly cNMF, show a lower binary mean for these high-variance genes.

5.4.4 Marker gene differential expression

This section considers the consistency of output marker genes across datasets of the different clustering and factorisation methods, **Figure 5.7**. Marker genes for `bfact` are those predicted in R^b, while for other methods, these are taken as the top 100 differentially expressed genes for each factor/cluster.

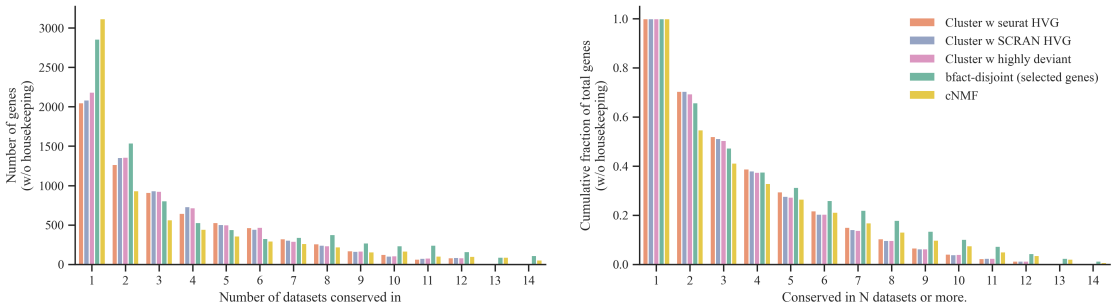


Figure 5.7 Number and cumulative fraction of conserved marker genes across all methods. Using cluster/factorisation results with housekeeping genes removed. Marker genes are taken as the top 100 filtered DEGs for each cluster/factor except `bfact`. For `bfact` marker genes are taken as those selected in R^b. DEGs based on SCRAN-normalised data, and using a Wilcoxon rank-sum test, filtered to have at most 20% presence in other groups and be expressed in minimum of 20% within groups. As methods with more clusters will have more marker genes, the cumulative fraction of total is also included.

Even with the removal of housekeeping genes, `bfact` has the most consistently selected genes across a larger number of datasets, and a similar representation at lower datasets. This suggests that `bfact` may identify biologically relevant genes that other methods do not. However, some such genes may also be considered to be uninteresting, for example, transcriptional activity or ATP production, some of which are present in factors for `bfact` GSEAs in **Section 5.4.6**. cNMF also has more consistently expressed genes than clustering approaches, suggesting a fuzzy clustering approach may improve consistency.

5.4.5 Summary metrics

This section explores some of the summary metrics of each of the approaches in explaining the underlying data signal. If a less expressive approach explains the

data equally well, this supports its suitability by balancing explanatory power with reduced risk of overfitting.

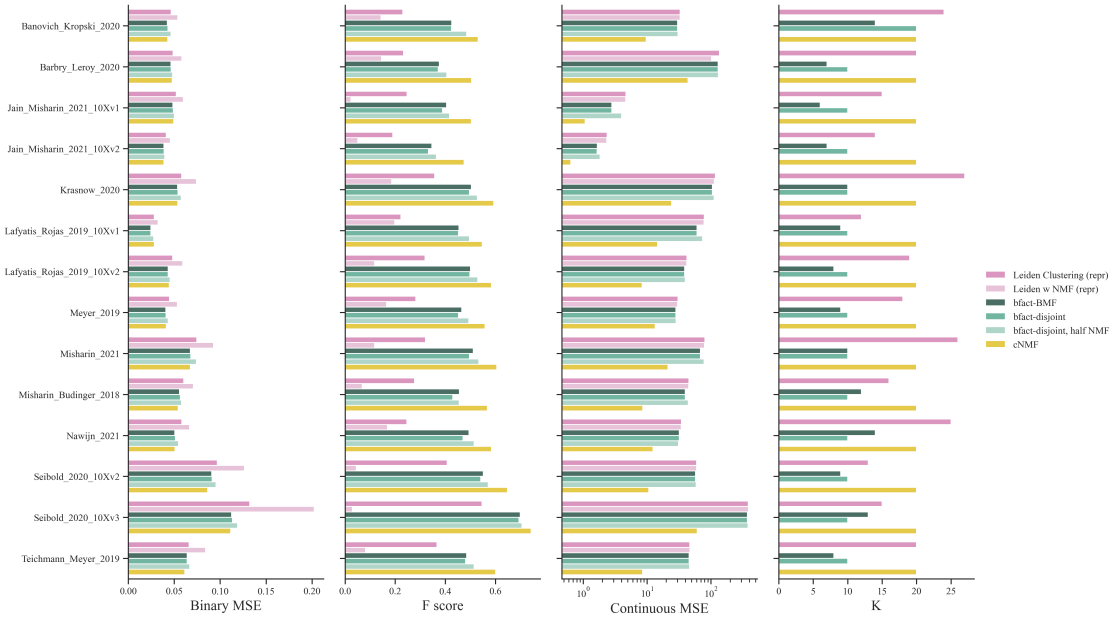


Figure 5.8 Summary metrics for each dataset. As the three clustering methods are so similar, uses the SCRAN HVG-based clustering as a representative to increase plot clarity.

Figure 5.8 includes variants of methods using a half-NMF approach (i.e letting L become continuous based on method-specific R). The methodology for each is introduced in **Section 5.3.4**. The `bfact` standard BMF variant is included as well as the disjoint. These are very similar, and because of the increased identifiability of the disjoint approach, `bfact`-disjoint was chosen for the majority of analysis in this chapter.

Given the strong similarity between `bfact` and `bfact`-disjoint, the advantage of Boolean additivity in BMF does not appear to provide substantial benefits for modelling scRNA-seq data. While any true BMF can be reformulated as a disjoint BMF by introducing new factors to capture overlaps between existing ones, the relatively small number of factors found for disjoint BMF suggests that these non-additive advantages are not being realised in practice. A likely reason is the extreme sparsity and incomplete penetrance of scRNA-seq data, where low counts may cause

genes to be omitted from shared processes. Even if the Boolean aspect of BMF offers no clear advantage, the rationale for using a binary formulation remains valid.

Despite comparing metrics in **Figure 5.8**, it should be noted that direct comparison is difficult due to the different expressivities of the models. For `bfact`, $l_{ik}, r_{kj} \in \{0, 1\}$, for cNMF $l_{ik}, r_{kj} \in \mathbb{R}^+$, with $\sum_k l_{ik} = 1$ and for clustering, $l_{ik} \in \{0, 1\}, \sum_k l_{ik} = 1$, $r_{kj} \in \mathbb{R}^+$. The other form of expressivity comes from the number of factors, which also changes. Both `bfact` and the clustering approaches automate the number of factors, whereas for cNMF this is a fixed input of $K = 20$. The more expressive a model is, the more it can fit (or overfit) to the data. Hence, the interpretation of both reconstruction error and F score should consider this.

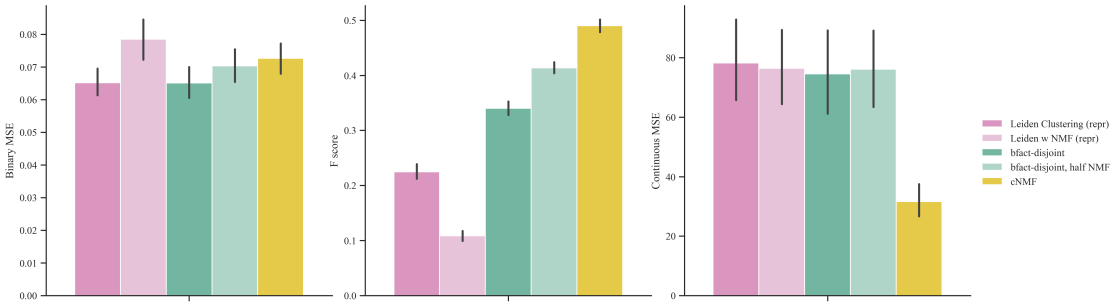
`bfact` and cNMF yield similar binary reconstructions, but in continuous reconstruction, the greater expressivity of cNMF provides a clear advantage. This is also reflected in the F-score, where cNMF achieves higher values overall. A slight performance gain can still be observed for `bfact`-disjoint with half NMF, suggesting that indeed some of the advantage from cNMF may be its expressivity. Conversely, continuous MSE does not decrease when using half-NMF for usage, which suggests that large contributors to MSE are driven by specific genes, not affected in half-NMF when R is fixed.

Applying NMF to cluster-based results tends to hinder performance across most metrics; since clustering assigns each gene set to a single group, reinterpreting these as compositional factors is problematic. The decrease in continuous MSE reflects that it is likely fitting to outliers, with no general improvement, given the worse performance in other metrics.

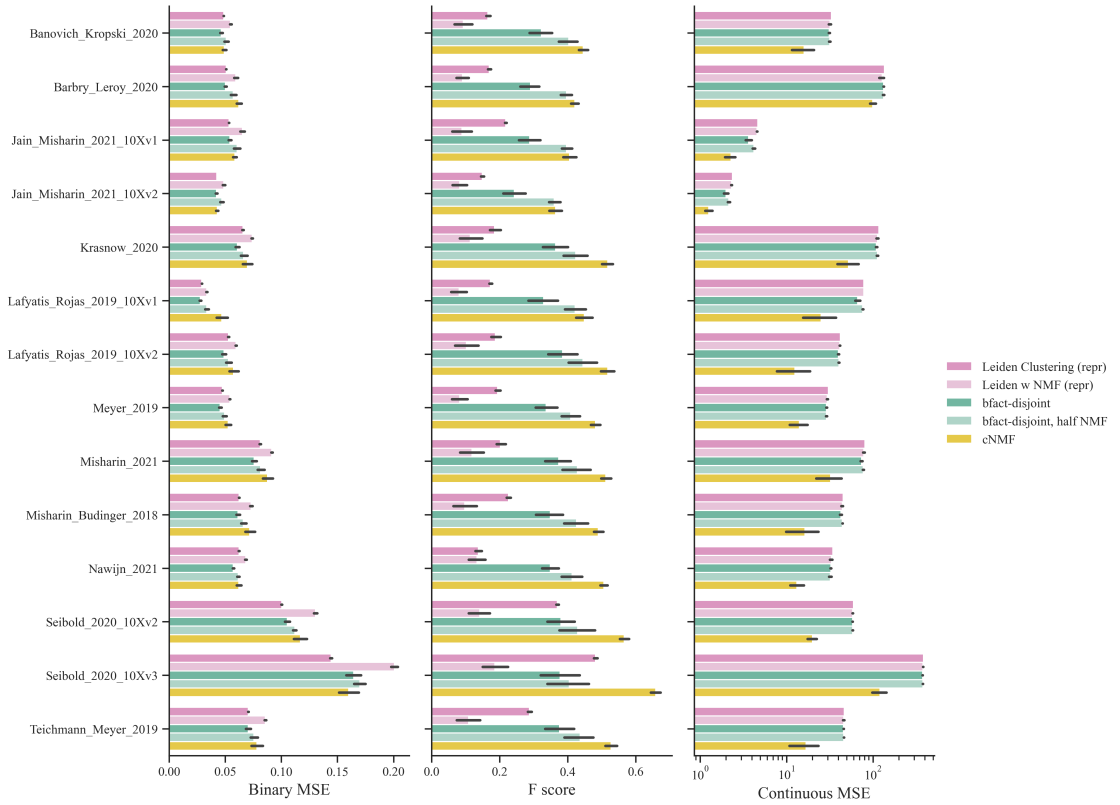
Overall, results for `bfact` are promising given its reduced expressivity and almost half the number of factors as cNNF. Both methods perform better than clustering, which may also partly be due to the expressivity of factorisation compared to clustering.

Predictive performance

Given the potential for models to fit well to the specific dataset, this section considers using predetermined GEPs, R_1, to predict signal in another dataset X_2, details **Section 5.3.4**.



(a) Total across all dataset pairs (both directions).



(b) Distribution of how well gene-sets/spectra found in every other dataset can predict expression in the specified dataset.

Figure 5.9 Factor predictive performance. How well gene-sets/gene spectra found in one dataset predict expression in another dataset. For Leiden clustering, clustering based on SCRAN HVG is used as a representative.

Figure 5.9 shows there is an implicit trade-off between binary MSE and F score- as true positives increase (increased F score), so do true negatives (decreased reconstruction). Whilst `bfact` has lower binary MSE than cNMF, it also has a lower F score. Again, the expressiveness of the model affects predictive performance, supported by the considerably higher F-score of `bfact` with half-NMF. This suggests that `bfact` suffers due to the so-called incomplete penetrance in the data, where genes in certain programs are only expressed a percentage of the time. This explains why a fractional usage matrix L can improve performance using a fixed R^b (for half-NMF).

Given that cNMF's continuous reconstruction error is lower than for other methods, this suggests some high-expression genes are reproducible across datasets. However, cNMF is higher and more similar to other methods in continuous MSE than on the fitted data, suggesting in **Figure 5.8**, cNMF does fit to some dataset-specific extremes.

Although cNMF has better results, given `bfact`-disjoint is more constrained and has fewer factors, this suggests that `bfact` may be a reasonable model for the data. Conversely, clustering-based signals DEGs appear to be poor at describing the bulk of the data signal, given by their low F-scores.

Cluster comparison

For a final summary metric, this section compares the adjusted mutual information (AMI) of the clustering and factor results, **Figure 5.10**. For the factor methods, clusters are taken to be unique rows of the membership matrix L_i and set any clusters with fewer than 20 cells to be a joint 'minor cluster'.

Unsurprisingly, the highest cluster agreement is between clustering approaches, as they share a similar construction, only with the selected genes differing slightly. cNMF is equally similar to all clustering methods, despite being based on an HVG selection process similar to SCRAN. `bfact`-disjoint has a low similarity to other methods, suggesting it represents processes that the other approaches do not (and vice versa). It could also be a consequence of representing cluster membership as

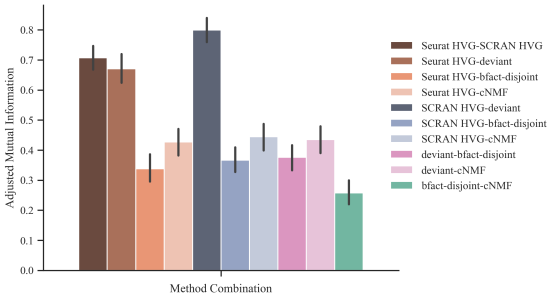


Figure 5.10 Similarity between clusters and factorisations using adjusted mutual information. Here, factor clusters are taken as the unique rows of L_i, i.e the unique factor assignments for each cell. For cNMF, discrete membership to a factor is defined for a usage of 10% or greater of that factor.

the intersection of factors, which may not work well when cells are often assigned many factors **Figure 5.3**. **Chapter 4**, showed that a random partition of PCA space (preserving local similarity) resulted in an AMI of approximately 0.6. That `bfact` has such low similarity likely suggests that `bfact` processes are more globally defined, unifying cells that aren't that close in Euclidean space.

Interestingly, despite `bfact` marker genes having a high alignment with highly deviant genes, their cluster similarity is still very low. As `bfact` uses reconstruction error internally, if specific genes are expressed in only a percentage of cells (termed here as 'incomplete penetrance'), these will not be captured using `bfact`. Conversely, a pairwise approach (clustering) or a continuous approach (cNMF) can capture this. Hence, `bfact` likely picks up on core similarities of groups of cells, whereas clustering approaches pick up on pairwise differences that are more robust to penetrance differences.

5.4.6 GSEA

Figure 5.11 gives an overview of the p values of the top 5 hits for each factor across different factors and datasets in different Enrichr reference databases, as well as the uniqueness of terms across factors as an indicator of factor independence based on hits (very crude, as can see in **Figure 5.12**- **Figure 5.14** many terms are related). Interestingly, for non-SCRNA-seq-derived data, some `bfact`-disjoint factors are more highly enriched with more independent factors than other approaches, particularly

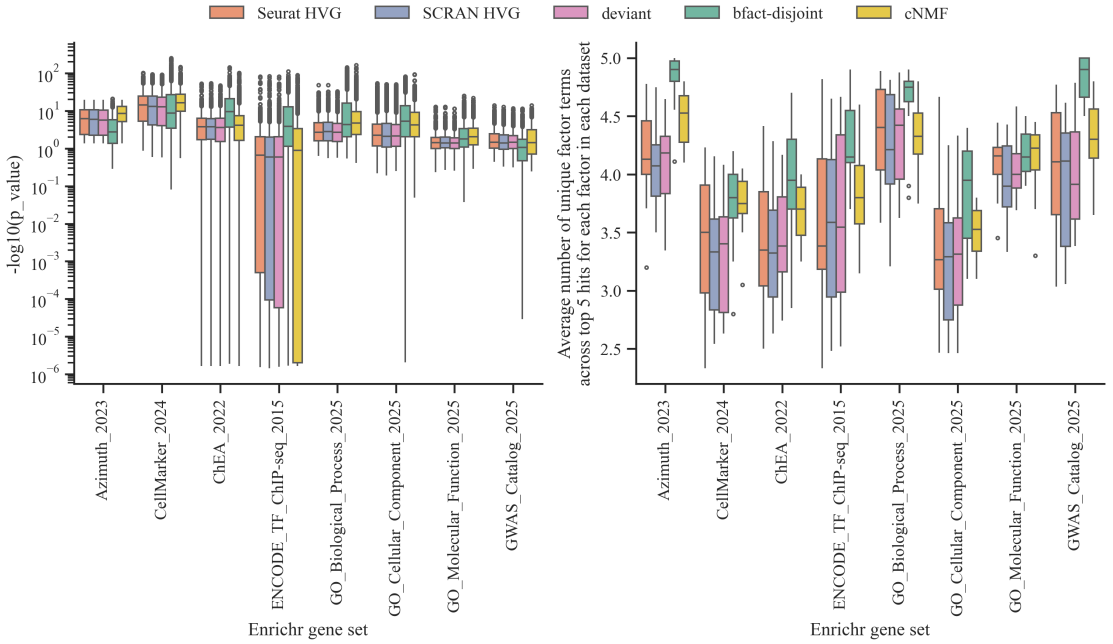


Figure 5.11 Enrichment properties of factor genesets across methods and reference databases. Adjusted p-values over top 5 hits per factor for each method, across all factors and datasets (left). Distribution across datasets of the average number of unique terms per dataset across factors (right), proxy for the independence of identified factors (right).

for biological and cellular processes and CHIP-seq databases. Conversely, for both Azimuth and GWAS databases, `bfact` performs worse than other approaches, although it has among some of the highest extremes, suggesting it identifies some cell types well. CHIP-seq databases capture genes that are likely modified by the same transcription factor, so perhaps `bfact` picks up on these patterns of co-expression, whilst ignoring patterns that are more abundance-driven and appear to be cell-specific.

cNMF is shifted higher than clustering for almost every database, and has notably higher shifted z-scores for cell-specific markers. Given that both Azimuth and CellMarker have gene sets based on the scRNA-seq pipeline, there could be a form of confirmation bias- given these pipelines are usually conditioned on HVGs, then mostly HVGs are identified as marker genes, when other genes may exist that could also be marker genes. However, particularly given its performance with Azimuth, this suggests that cNMF is a superior method to clustering for future analysis.

Overall, these results are promising for `bfact`, and suggest that although it may not be better at detecting cell-types, it can detect other biologically relevant information from scRNAseq that is commonly ignored.

Representative GSEA results

Figure 5.12- Figure 5.14 present representative GSEA results for the representative dataset across all reference databases. The top 5 hits per factor with adjusted p-value less than 0.05 are included in the visualisation. Given the similarity of the three clustering approaches, only results for HVG Scran feature selection followed by clustering are presented. All figures are on the same p-value colour scale.

`bfact` contains some factors with very high usage that tend to correspond to possibly uninteresting processes like gene expression or mitochondrial activity. It also has many hits for CHIP-seq-related reference sets. All methods have a high association with lung-derived database cells, although unfortunately, CellMarker does include marker sets derived from the HLCA base data, which may skew these results. cNMF seems to have the most independent factor sets that appear to correspond to different cell types, based on enrichment results.

Also of note is that some seemingly unrelated reference terms are associated with the same factor gene set. This perhaps suggests error or pleiotropy in reference terms. Further, errors can propagate; if one analysis identifies a geneset as one process but it is confounded by another process, then hits for that geneset may be mislabelled. Hence, all enrichment results should be taken with a pinch of salt.

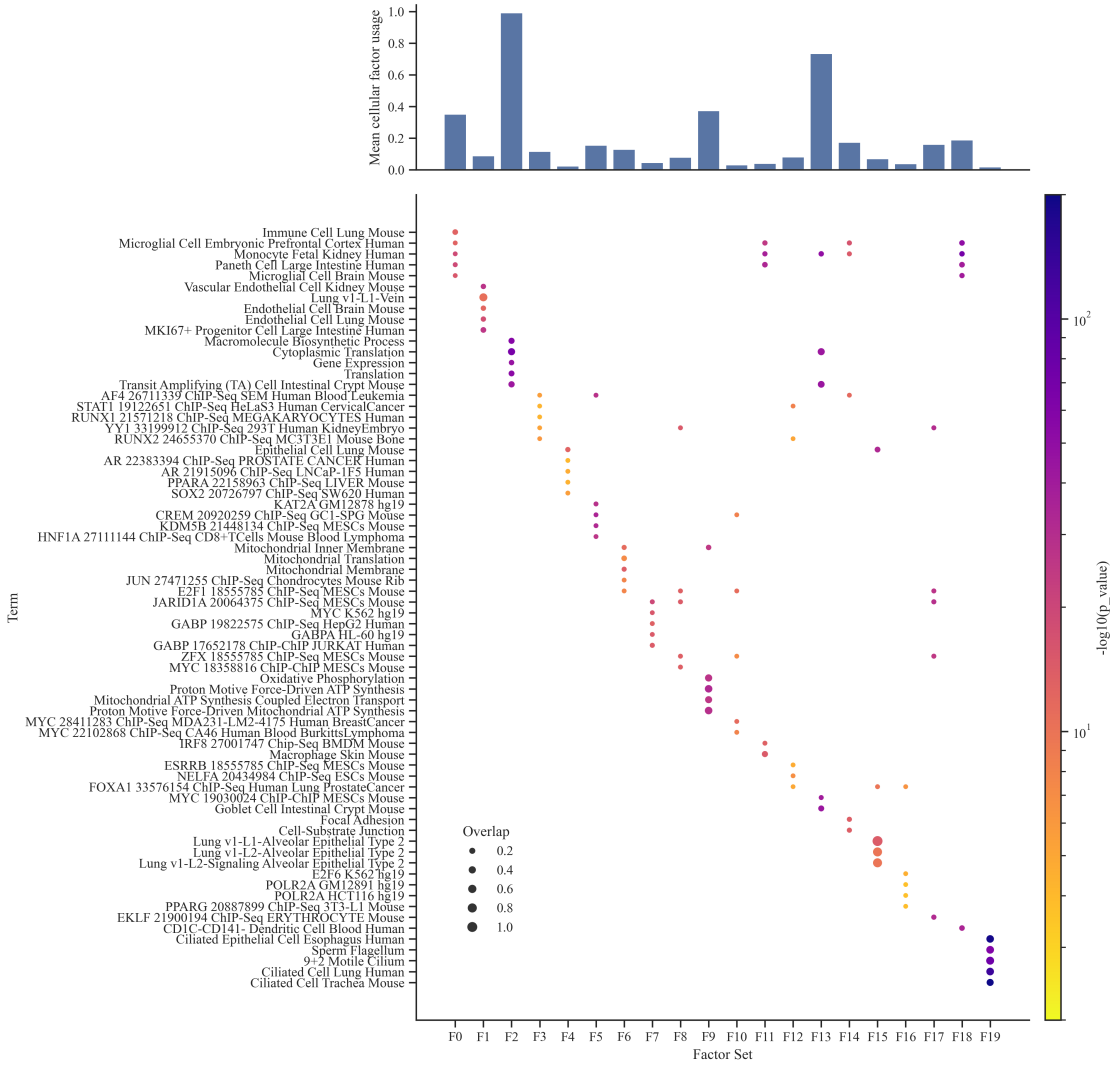


Figure 5.12 GSEA for `bfact-disjoint`. Mean usage is calculated as $\frac{1}{N_c}\sum_i l_{ik}$

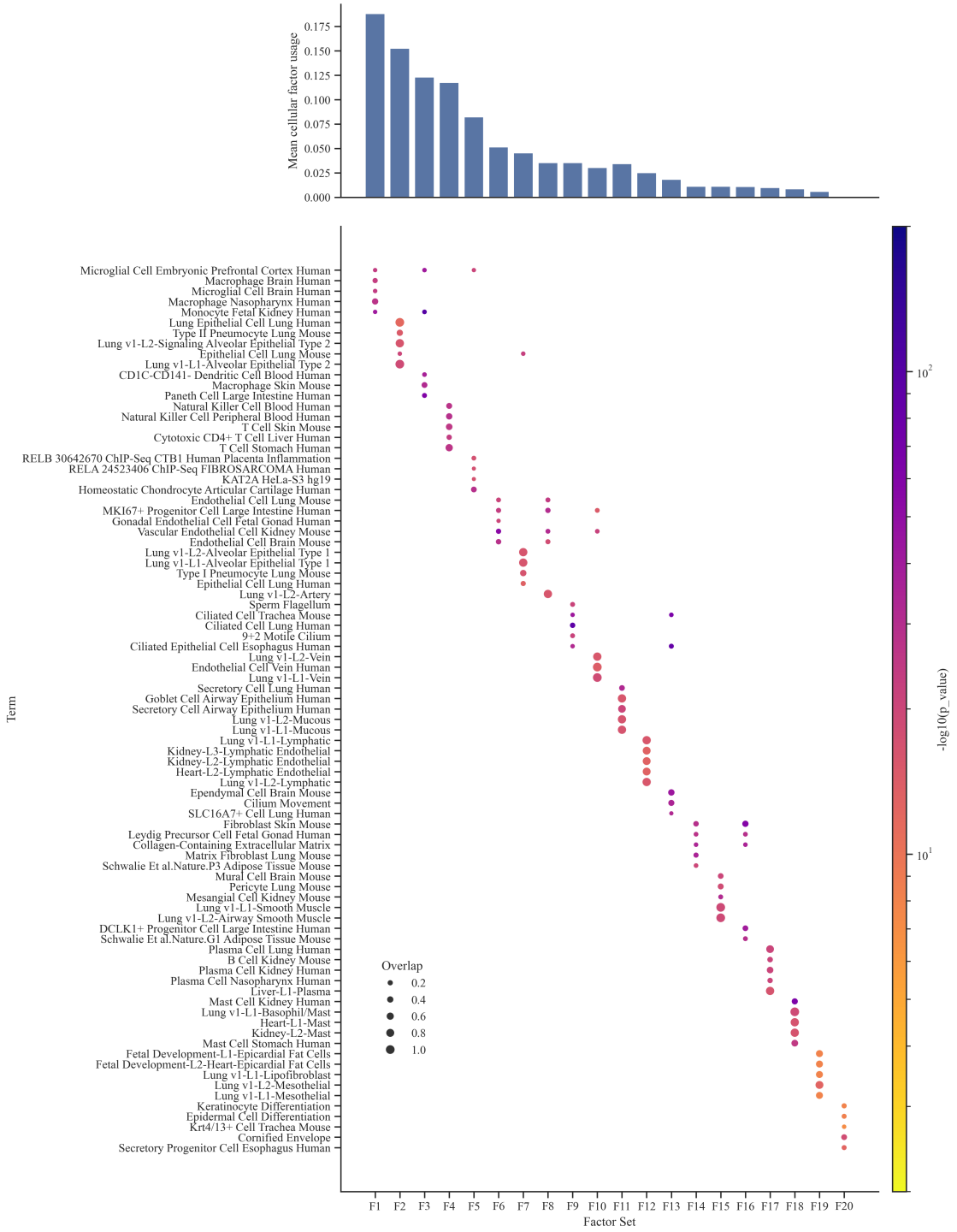


Figure 5.13 GSEA for cNMF. Using top 100 DEGs for each factor, mean usage is calculated as $\frac{1}{N_c}\sum_i l_{ik}$

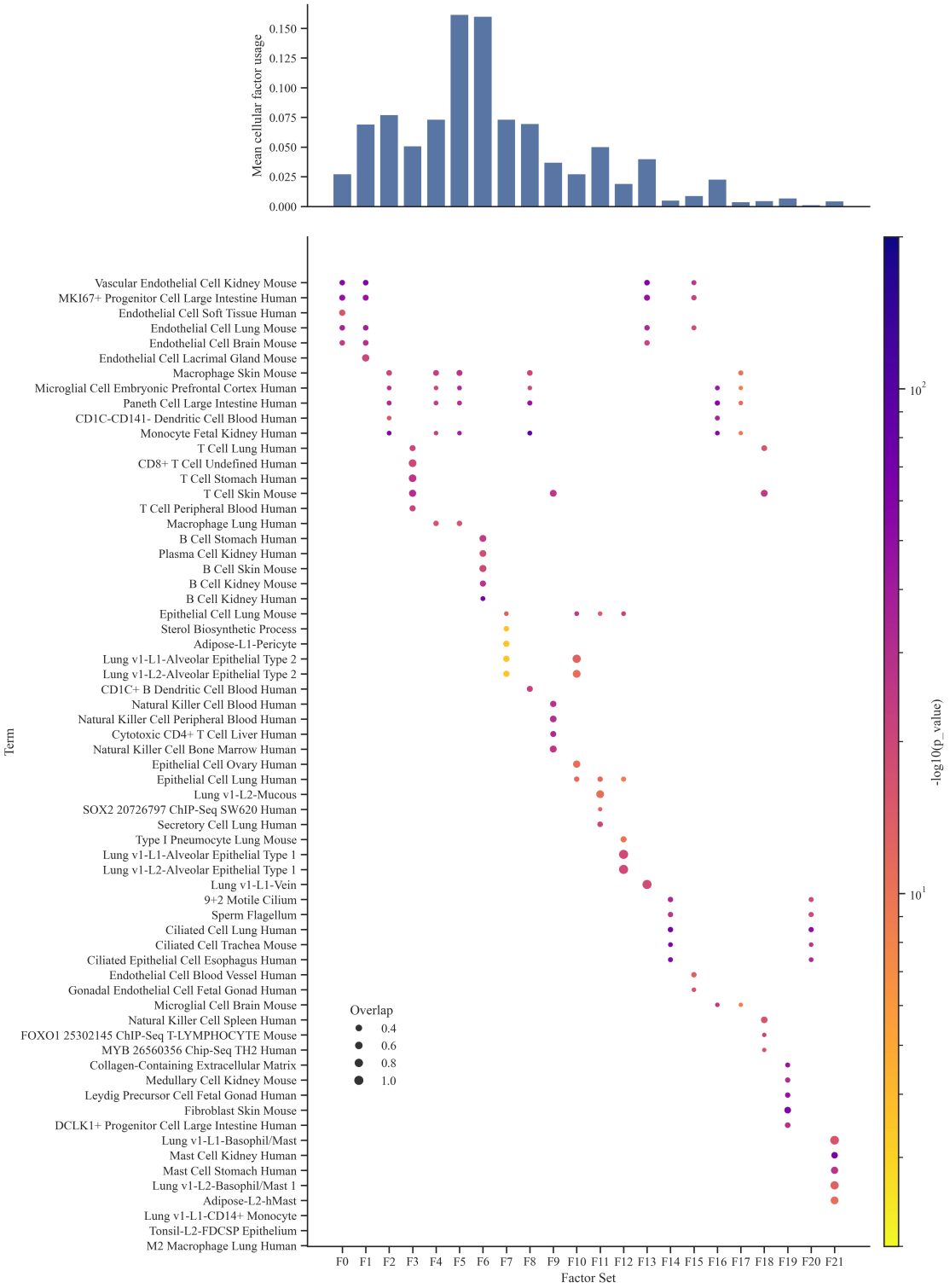


Figure 5.14 GSEA for representative Leiden clustering. Based on SCRAN HVGs (using top filtered DEGs for each factor), mean usage is calculated as $\frac{1}{N_c}\sum_i l_{ik}$

5.5 Conclusions

The motivation for this chapter was threefold. Firstly, **Chapter 4** demonstrated that scRNA-seq data is extremely sparse, yet is not always treated as such, hence a binary approach that capitalised on the data distribution seemed like an interesting idea. Secondly, much of the scRNA-seq pipeline appears circular, so a BMF approach that makes a single assumption about how to binarise the data (thresholded here on non-zero UMI count) avoids many of the upstream assumptions in the scRNA-seq pipeline. And thirdly, it was unclear that a clustering modelling assumption is the correct assumption for the data, and that results should be compared between clustering, NMF (or in this case, fuzzy-clustering) and BMF.

Results indicated that a more constrained disjoint `bfact` approach (where gene-sets for factors are distinct), performed similarly to a Boolean `bfact` approach. However, given that a disjoint approach is more identifiable and interpretable, `bfact`-disjoint was chosen for all analyses. Consequently, any non-additivity benefits from a Boolean approach appear not to benefit modelling in the scRNA-seq data.

`bfact` seems to identify different yet biologically relevant marker gene sets from other approaches. These are less cell-type specific than other methods. However, they may warrant further investigation. `bfact` also selects more consistent genes across datasets; however, some of these may be uninteresting measures of biological processes. Considering the binary constraints on `bfact` decomposition, it performs well at reconstruction and detecting signal in the data.

Conversely, cNMF seems to perform well across the board; however, it is also the most expressive model, more likely to overfit or produce non-unique solutions. However, in gene set enrichment, it has shifted to higher enrichment, particularly for Azimuth and CellMarker- both cell-type marker reference databases. This suggests it may be a better approach than clustering for cell-type identification and identification of marker gene sets.

There is a possible confirmation bias at play based on selecting scRNA-seq HVGs in clustering pipelines and using these as cellular marker genes. Such genes are likely biologically relevant if they reproduce together across datasets; however,

there may be other gene sets that are biologically relevant that can not be captured through higher-than-expected variance arguments due to sparsity. `bfact`, on the other hand, appears to be able to pick up on different patterns of coexpression for such genes, and seems to be more enriched in biological and transcription factor processes than other methods.

As `bfact` uses reconstruction error internally, if specific genes are expressed in only a percentage of cells (incomplete penetrance), these will not be captured using `bfact`. Conversely, a pairwise approach (clustering) or a continuous approach (cNMF) can capture this. Hence, `bfact` likely picks up on core similarities of groups of cells, whereas clustering approaches pick up on pairwise differences that are more robust to penetrance differences.

Overall, results from using a binary factorisation approach are promising, and suggest that scRNA-seq data contains valuable information, not currently exploited by existing methods. In contrast, an NMF factorisation approach should be considered as an alternative to the traditional clustering pipeline, given its better identification of marker genes.

5.6 Limitations and Future Work

One of the main limitations of the BMF approach is its focus on reconstruction error, which is also linked to the incomplete penetrance issue. In general, disjoint-BMF will only assign a gene to a factor if at least half of the cells containing that factor express the gene. This is limiting; there may be many cases where cells of particular types only express genes a certain proportion of the time. Hence, a way to address this limitation whilst still keeping a binary model would be to move away from optimising reconstruction error, and perhaps optimising another metric like positive predicted value. There could be a way of integrating this into the cost measure for `bfact`.

Alternatively, it would be desirable to have a probabilistic model where R is no longer binary but some value between 0 and 1 indicating the probability of expression of the gene in a given factor. However, this would likely be computationally challenging to implement, as one of the benefits of the `bfact` MIP approach was that

it only scaled with the number of genes and candidate factors. However, there may be a continuous method that can be used to approximate this half-binary approach.

Although these results for `bfact` are promising, it would be interesting to further investigate `bfact` results across different datasets. Here, only lung tissue was considered, but it would be interesting to see if `bfact` identified gene-sets differed across different tissues. If not, it would suggest that it is identifying common process pathways rather than common cell type pathways.

Another way to experimentally validate `bfact` in scRNA-seq would be to consider disease vs non-disease settings. If GEPs identified by `bfact` were better for predicting disease status in a new cohort, this would support using `bfact` to supplement current methodologies. Likely, this should also be done in comparison with current NMF approaches.

6 Discussion

Contents

My thesis has combined two large projects, both fitting under the broad umbrella of novel machine learning for applications in cancer genomics. The first project, **Chapter 2**, developed a novel approach for calling copy number alterations from whole-genome sequenced cancer tissue. The second project, **Chapter 3** - **Chapter 5**, concerned developing a new approach to analysing scRNAseq data through assuming a binary factorisation model. The second project is less specific to cancer genomics; however, much scRNA-seq analysis concerns comparing case-control status between diseases, including cancer. Hence, development in the former aids insight into the latter. In each chapter, I have highlighted the main conclusions and future directions specific to the content.

6.1 Copy number calling with `araCNA`

The copy number project developed a model, `araCNA`, using a recent deep-learning architecture, Mamba. Mamba is related to state-space models, similar to recurrent

neural networks but able to work on sequence lengths at the scale of genomic data. Such a model is required to learn long-range dependencies in the data, to find the most likely copy-number state consistent across the whole sequence, rather than local to sequence regions.

Like many problems in genomics, the goal here is to infer latent or hidden variables (the copy numbers) that best explain the data. There is no known ground truth in the measured WGS data. However, the generating process from latent variables to the measured data is well understood. Hence, a simulation procedure can be used to generate training data where both the 'measured' data and the latent variables are sampled and known. This kind of procedure constitutes a simulation-based inference approach.

To intuitively understand this with a simpler example, consider different sets of data, which we assume are all normally distributed. Here, the maximum likelihood estimate (MLE) can be derived to determine the mean and variance for each dataset, which may be considered unknown/hidden variables in this problem. Another way to estimate the mean and variance could be to repeatedly sample different means and variances from some prior and then to sample data vectors from a normal distribution with that mean and variance. A neural network can then be used to learn how to predict the mean and variance from each data vector, having input and output pairs in line with supervised learning, predicting the MAP rather than the MLE (given the prior). This is a form of amortisation, as the model learns to predict the parameters of the statistical model, and once trained, does not require optimisation on a per-dataset level, only requiring inference (prediction). Of course, when estimating the mean and variance is simple, like in a normally distributed case, then developing such an approach is overkill. However, there are many instances where an analytical MLE cannot be derived, or is also computationally expensive, and the trade-off between per-dataset optimisation and an amortised model trained on simulated pairs becomes desirable.

Although training a model on simulated data alone may at first glance seem strange, it is actually similar to traditional approaches where a statistical model is

assumed. The first is an analytic maximum likelihood under the assumed model. The second is an inferred maximum a posteriori learnt through paired samples of distribution data and underlying parameters, also under that assumed model and a prior on the parameters. Hence, the approach used is a special type of so-called 'simulation-based inference', and can only be used in problems where the data-generating procedure is known or assumed. The TabPFN model that has recently seen success in many classification problems is an example of such an approach [150].

`araCNA` is trained only on such simulated data, but applied zero-shot to real WGS-derived data, where it performs well on several metrics. Hence, this serves as a proof-of-concept for other genomics or biological applications, where the data-generating process (from latents to measured data) is known, but inferring such latents using traditional methods is difficult or flawed. The model is available on GitHub.

6.2 The scRNA-seq pipeline and `bfact`

The second project introduced here involved analysing scRNA-seq data to better discover underlying cell types or associated biological processes and their gene programs.

As part of this, in **Chapter 4**, I highlight some issues in the scRNA-seq pipeline. In particular, showing that scRNA-seq data are extremely sparse and low count, and that large UMI counts in cells are driven by a few large values rather than consistently higher values in some cells. Hence, normalisation by cell size factors ends up arbitrarily rescaling low-counts, which, if not adequately considered, can lead to the selection of irrelevant HVGs in some feature selection schemes (namely Seurat HVGs). I show that the normalisation here fails to account for the number of zeroes in a given cell, which leads to a misleading relationship between the expression in a given gene and the average expression across all genes, for each cell.

Another part of the pipeline selects a subset of features based on high-variability, which usually assumes a Poisson or negative-binomial model to find genes where expression is likely a mixture of two populations with different expression, having a higher variance than expected. However, I also show that in the binary limit, when parameter values are small (i.e λ in Poisson), it is impossible to tell whether a gene

has different expression rates across different cell types. However, such information is still useful if patterns of expression across genes are considered.

Finally, I show that clustering results are not very robust to feature selection, and that any random partition in UMAP space is as similar as different feature selection approaches, implying that feature selection can conserve local patterns, but boundaries between clusters are less reliable. This has implications for historical results, particularly those which use the abundance of different cell types between disease states, and perhaps also for the robustness of marker genes.

In an attempt to address some of these issues, I take a Boolean matrix factorisation approach, which requires no normalisation, feature selection or clustering, and outputs marker genes related to factors as part of the model. I develop a novel Boolean matrix factorisation approach in **Chapter 3**, called `bfact`, that uses combinatorial optimisation in tandem with a heuristic algorithmic approach. `bfact` works well compared to other approaches in a sparse and high-dimensional setting like scRNA-seq.

In **Chapter 5**, I analyse the results of `bfact` in a scRNA-seq setting and compare it to traditional clustering and NMF approaches, particularly for identifying gene programs. I show that some gene programs are more highly enriched in external data sources, particularly for CHIP-seq and gene-ontology-derived programs. This indicates that there is more to learn from scRNA-seq data, and that binarising the data gains insight into such biological processes.

This project involved understanding the data distribution of scRNA-seq data and existing modelling limitations to develop an approach specifically to exploit that distribution and gain novel insights.

6.3 Concluding statement

Although these projects seem at first glance only broadly related by genomics and developing methodology, they both concern learning and predicting unknown variables - they are both unsupervised approaches. In fact, many biology and 'omics settings are precisely this. This differs from most novel deep-learning contexts

of natural language processing (NLP) and image processing, where humans can easily interpret the data and provide annotated, ground-truth labels for the model to learn. Even though NLP/LLMs are self-supervised, the prediction of language itself is useful as-is and can easily be repurposed into other kinds of prediction tasks, where again annotations exist. However, in health settings, predictions themselves are often second to explainability, or to estimating known but unobserved latent variables. Hence, simply repurposing existing methodology for prediction in genomics is unlikely to yield further insight. For example, in scRNA-seq, there are many examples of research developing deep-learning approaches that predict cell-types, but the 'ground truth' cell-types are defined through previous computational analysis [92, 101, 151, 152]. Hence, accuracy measures only capture how well such approaches emulate the original computational pipeline, and not necessarily that they are more accurate with respect to the unknown ground truth.

Even in a self-supervised context, there is an assumption that the task of predicting data through some bottleneck is enough to uncover the desired latents, and even then, the embedding space is often required to be used in a supervised or regularised way for interpretability. On the other hand, some DNA language models can be exploited to get kinds of per-sequence or per allele effect sizes, through computational mutagenesis, [153]. There is promise for such models to expand their capabilities, but it is unclear the generalisability of these for developing therapeutic targets or for context-specific predictions (e.g tissue expression). Part of the reason for this is that there is not enough diversity across genomes with homologous regions and regulatory elements, even across chromosomes [154]. Hence, there is no way to avoid data leakage. Boer and Taipale [154] argue that the best approach for such DNA language models is to use synthetic, randomly generated sequences with associated measurements instead. Indeed, this is a growing area that holds promise, and ushers in a new paradigm - designing data for models, rather than designing models for data.

In order to leverage modern machine learning techniques with existing data, more thoughtful models and applications of those models are required. Unlike NLP

or imaging, many genomic tasks do not conform neatly to a supervised paradigm. Yet biology offers something those domains do not: a theoretical and mechanistic understanding of the latent processes that generate the data, from the central dogma to regulation, inheritance, and recombination. This knowledge provides priors and constraints for building more meaningful models. For method developers coming from an ML background, progress will require engaging with this biological context and drawing on the existing bioinformatics tools. There remain many unanswered questions and a rapidly growing body of data, making the intersection of machine learning and genomics a rich area for future research.

Appendices

A

Chapter 2 Appendix

A.1 Hyperparameters

Table A.1 describes the hyperparameters used for the model architectures of `araCNA` and the simulation-based training.

Parameter	Description	Value
Hyena Backbone Parameters: unspecified hyperparameters are the same as the defaults provided in Nguyen et al. [39]		
`d_model`	Projection dimension of data after encoder	32
`n_layer`	Number of Hyena blocks	2
`l_max`	Maximum sequence length. Model has capacity for sequences up to this length. Where possible, the model creates arrays of the actual sequence length when less than this to avoid unnecessary memory usage. Some positional encodings of the model cannot be resized and are always set to this value.	1000000
`causal`	Whether to use bidirectional or causal Hyena blocks.	False
`emb_dim`	Dimension of input to inner hyena filter MLP	5
`w`	Frequency of periodic activations of hyena filter	10
Mamba Backbone Parameters: unspecified hyperparameters are defaults of Mamba package at commit `7fb78a5`, paralleling their `create_blocks` function and `Mamba2` class.		
`d_model`	Projection dimension of data after encoder and backbone.	32
`n_layer`	Number of bidirectional Mamba2 blocks	2
`expand`	Dimension of A, B, C parameters in SSM, and the factor by which input dimension expands, before projecting back to model dim.	4
`headdim`	Dimension of broadcast projection of the input data, before passing through main Mamba2 block. Can be thought of as implementing `headdim` separate SSMs.	16
`d_intermediate`	Dimension of hidden layer of MLP used as last step in Mamba2 block. If 0, no MLP used.	0
Encoder/Embedding Parameters:		
`input_dim`	Dimension of input data (read depth and BAF)	2
`embed_dim`	Dimension of embedding, input dimension into backbone	32
`token_dim`	Number of possible tokens. In `araCNA` only 2 tokens are used- global/not-global, but setting it to 24 allows it to be repurposed in another task with more tokens, e.g chromosome token.	24

Decoder Parameters:		
`decoder_dim`	Output data dimension of backbone, same as backbone input dimension.	32
`max_tot_cn`	Maximum total copy number that can be modelled, total copy numbers higher than this are mapped to a surplus category.	10
Learning Parameters:		
`loss_weights`	Array corresponding to λ_r, λ_p for the global reconstruction parameters.	[1, 1]
`avg_rd_trim_ratio`	The proportion of upper/lower quantiles excluded from the average measured read depth in a robust mean calculation.	0.05
Simulation Parameters: these correspond to the final simulation parameters, as a curriculum learning approach iteratively increases the simulation difficulty.		
`read_depth_range`	r_1, r_2	[5, 70]
`purity_range`	ρ_1, ρ_2	[0.5, 1]
`read_depth_scale_range`	$\sigma_{r,1}, \sigma_{r,2}$	[0.01, 0.2]
`baf_scale_range`	$\sigma_{b,1}, \sigma_{b,2}$	[0.02, 0.1]
`max_total`	The maximum total sampled parental copy number, the sampled minor copy number is less than or equal to the sampled major copy number	8
`max_h_segs`	$N_{h,\max}$	100
`h_l_range`	$l_{h,\min}, l_{h,\max}$	[5, 300]
`l_min`	The minimum segment length, $L_{\min}$.	100
`N`	The maximum number of sampled segments, N	max(50, L/1000)

Table A.1 Table of Parameters. Some optional parameters in the code base have not been mentioned as they do not affect the model, and are included for possible later development.

A.2 Additional Simulation Details

To summarise the simulation process mathematically, sets of observation data are simulated based on the sampled parental copy number profiles (A_M, A_m) according to the following scheme:

$$\begin{aligned}
\rho &\sim \mathcal{U}[\rho_1, \rho_2], \\
r_d &\sim \mathcal{U}[r_1, r_2], \\
s_{M,i}, s_{m,i} &\sim \text{Bin}(n = 1, p = 0.5), \\
R_i &= r_d C^s_{T,i} + r_d \sigma_r \epsilon_{i,1}, \quad \sigma_r \sim \mathcal{U}[\sigma_{r,1}, \sigma_{r,2}], \quad \epsilon_{i,1} \sim \mathcal{T}_2,
\end{aligned}$$

where (ρ_1, ρ_2), (r_1, r_2), $(\sigma_{r,1}, \sigma_{r,2})$ are hyperparameters and $\mathcal{T}_2$ is Student's t-distribution with 2 degree of freedom.

The BAF sampling is slightly more involved to reflect real data observations, and can be described by the following scheme:

$$\begin{aligned}
n_{h_s} &\sim \mathcal{U}\{1, N_{h,\max}\}, \\
l_{h,j} &\sim \mathcal{U}\{l_{h,\min}, l_{h,\max}\}, \quad S_{h,j} \sim \mathcal{U}\{0, L - l_{h,j}\}, \quad j = 1 \dots n_{h_s}, \\
N_{r,i} &= \text{Pois}(R_i), \\
N_{B,i} &= \text{Bin}(N_{r,i}, p = \frac{C^s_{B,i}}{C^s_{T,i}}), \\
\nu_i &= \begin{cases} 1, & \text{if } s_{M,i} \neq s_{m,i} \text{ and } i \notin \cup_j \{S_{h,j}, S_{h,j} + l_{h_j}\}, \\ 0.2, & \text{otherwise,} \end{cases} \\
\sigma_b &\sim \mathcal{U}[\sigma_{b,1}, \sigma_{b,2}], \quad \epsilon_{i,2} \sim \mathcal{T}_{150 * \sigma_b}, \\
B'_i &= \begin{cases} \frac{N_{r,i}}{N_{B,i}} + \nu_i \sigma_b \epsilon_{i,2}, & \text{if } 0 \leq \frac{N_{r,i}}{N_{B,i}} + \nu_i \sigma_b \epsilon_2 \leq 1, \\ \frac{N_{r,i}}{N_{B,i}} - \nu_i \sigma_b \epsilon_{i,2}, & \text{otherwise,} \end{cases} \\
B_i &= \max(\min(1, B'_i), 0),
\end{aligned}$$

where $N_{h,\max}$ is the maximum number of homozygous segments, n_{h_s}, to sample. $l_{h,\min}$ and $l_{h,\max}$ are the minimum and maximum lengths, $l_{h,j}$ of the homozygous segments, and are hyperparameters, as is $(\sigma_{b,1}, \sigma_{b,2})$. We introduce N_r and N_B as the number of normal and B allele reads, to ensure the frequency is centered around rational values, as would be in real data. $S_{h,j}$ denotes the start positions of the

homozygous segments, while ν_i is the scaling factor that reduces noise at homozygous loci. The parameter ϵ_2 is sampled from a Student's t-distribution with degrees of freedom, df, that scale with the BAF scale parameter- so that smaller noise samples will have a lower df value to ensure that some extreme values are still sampled.

A.3 LogR Simulation Details

For the LogR, l_r simulation procedure, we use a similar procedure to above, but adding:

$$
\begin{aligned}
r_d^n &\sim \mathcal{U}[r_1, r_2] \\
t_{r,i} &= \frac{\max(1, R_i)}{2r_d^n} \\
l_{r,i} &= \log \frac{t_{r,i}}{\mu_{\text{robust}}(t_r)}
\end{aligned}
$$

The BAF is sampled as above.

A.4 CNVKit performance

CNV Kit does not directly account for purity/ploidy but can take both as an input- mainly affecting thresholds for calling copy numbers [52]. Hence, we further implemented CNV Kit with ASCAT purity/ploidy as a final comparison point, as this performed better (**Figure A.1**), we have used this, denoted CNVkit*, as the default in the main text.

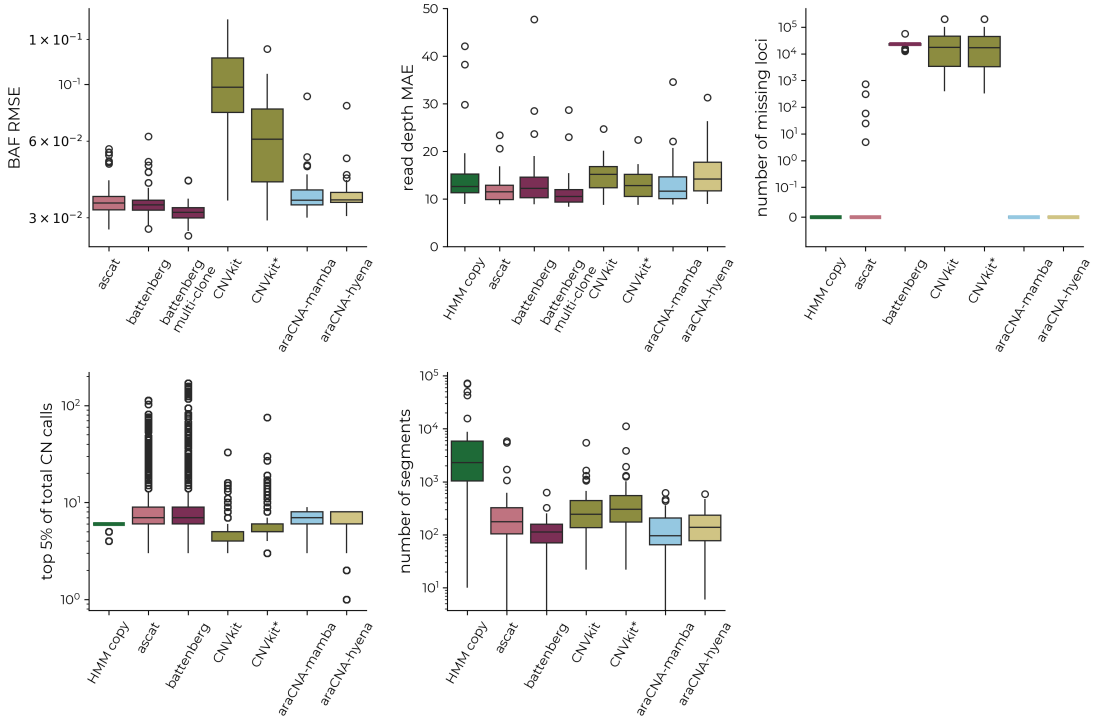


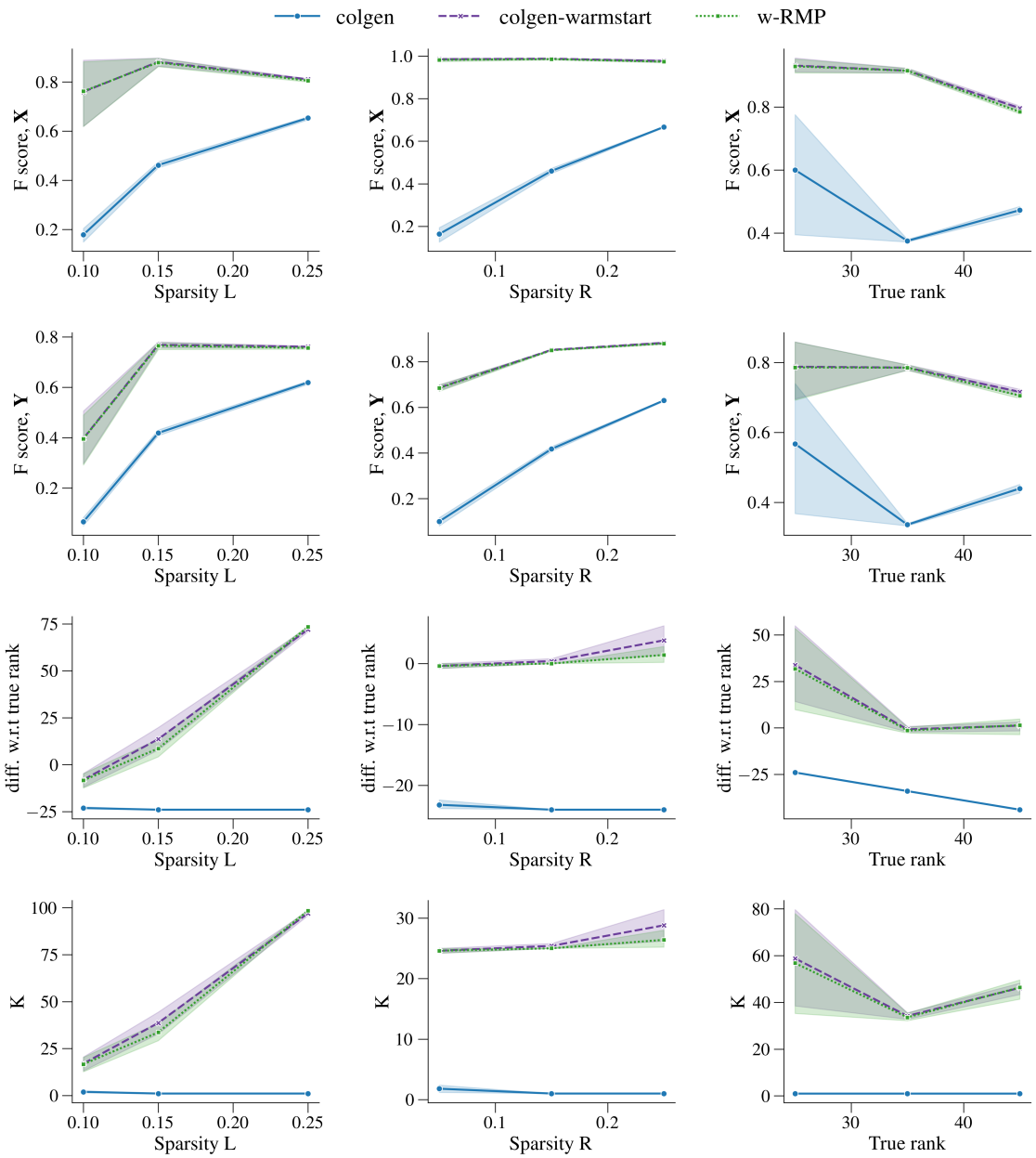
Figure A.1 CNV Kit performs better when provided with the approximate purity. Summary metrics on 50 TCGA samples with purity for CNV Kit* taken from the ASCAT estimates.

B

Chapter 3 Appendix

B.1 Delayed Column Generation

The pricing problem for RMP-1 can solve the disjoint BMF to optimality; however, it is slow- likely due to scaling (variables, constraints) with $(M + N, M)$ where the RMP-1 only scales with N for both. **Figure B.1** shows the results for the PP with the RMP, both warm-started or not, also comparing to RMP-w alone in **Figure B.1**, where the delayed column generation process is capped at 30 minutes. Given that the RMP-w alone takes less than 10 minutes, this demonstrates that the PP does not offer a significant or timely advantage to the RMP when warm-started appropriately. For larger datasets, this computational time is likely to be even slower/intractable



a) Varying left sparsity, $q_l, q_r = 0.1, r = 25$

b) Varying right sparsity, $q_r, q_l = 0.1, r = 25$

c) Varying number of factors $ql = q_r = 0.1$

Figure B.1 Comparison of delayed column generation used in tandem with bfact. simulations with $p^{\pm} = 0.1, M \times N = 1600 \times 400$. Each case is followed by the heuristic postprocessing with MDL cost, run only once for $K_{\min} = K_{\max} = 100$.

References

[1] Serena Nik-Zainal et al. 'The Life History of 21 Breast Cancers'. *Cell* 149.5 (2012), pp. 994–1007.

[2] Gavin Ha et al. 'Integrative analysis of genome-wide loss of heterozygosity and monoallelic expression at nucleotide resolution reveals disrupted pathways in triple-negative breast cancer'. *Genome Research* 22.10 (2012), pp. 1995–2007.

[3] Ronglai Shen and Venkatraman E. Seshan. 'FACETS: allele-specific copy number and clonal heterogeneity analysis tool for high-throughput DNA sequencing'. *Nucleic Acids Research* 44.16 (2016), e131.

[4] Peter Van Loo et al. 'Allele-specific copy number analysis of tumors'. *Proceedings of the National Academy of Sciences* 107.39 (2010), pp. 16910–16915.

[5] Eric Talevich et al. 'CNVkit: Genome-Wide Copy Number Detection and Visualization from Targeted DNA Sequencing'. *PLoS Computational Biology* 12.4 (2016), e1004873.

[6] Ellen Visscher and Christopher Yau. 'araCNA: somatic copy number profiling using long-range sequence models'. *NAR Genomics and Bioinformatics* 7.3 (2025), lqaf124.

[7] Cheng-Zhong Zhang and David Pellman. 'Cancer Genomic Rearrangements and Copy Number Alterations from Errors in Cell Division'. *Annual Review of Cancer Biology* 6 (2022), pp. 245–268.

[8] Hoon Kim et al. 'Extrachromosomal DNA is associated with oncogene amplification and poor outcome across multiple cancers'. *Nature Genetics* 52.9 (2020), pp. 891–897.

[9] Carolina Rosswog et al. 'Chromothripsis followed by circular recombination drives oncogene amplification in human cancer'. *Nature Genetics* 53.12 (2021), pp. 1673–1685.

[10] Kevin Hadi et al. 'Distinct Classes of Complex Structural Variation Uncovered across Thousands of Cancer Genome Graphs'. *Cell* 183.1 (2020), 197–210.e32.

[11] Christopher D. Steele et al. 'Signatures of copy number alterations in human cancer'. *Nature* 606.7916 (2022), pp. 984–991.

[12] H. Nakagawa et al. 'Cancer whole-genome sequencing: present and future'. *Oncogene* 34.49 (2015), pp. 5943–5950.

[13] Daniel Pinkel and Donna G. Albertson. 'Array comparative genomic hybridization and its applications in cancer'. *Nature Genetics* 37.6 (2005), S11–S17.

[14] Derek Y. Chiang et al. 'High-resolution mapping of copy-number alterations with massively parallel sequencing'. *Nature Methods* 6.1 (2009), pp. 99–103.

[15] Sergii Ivakhno et al. 'CNAseg—a novel framework for identification of copy number changes in cancer from second-generation sequencing data'. *Bioinformatics* 26.24 (2010), pp. 3051–3058.

[16] Ruibin Xi et al. 'Copy number analysis of whole-genome data using BIC-seq2 and its application to detection of cancer susceptibility variants'. *Nucleic Acids Research* 44.13 (2016), pp. 6274–6286.

[17] Valentina Boeva et al. 'Control-FREEC: a tool for assessing copy number and allelic content using next-generation sequencing data'. *Bioinformatics* 28.3 (2012), pp. 423–425.

[18] Scott L. Carter et al. 'Absolute quantification of somatic DNA alterations in human cancer'. *Nature Biotechnology* 30.5 (2012), pp. 413–421.

[19] Layla Oesper, Ahmad Mahmoody and Benjamin J. Raphael. 'THetA: inferring intra-tumor heterogeneity from high-throughput DNA sequencing data'. *Genome Biology* 14.7 (2013), R80.

[20] Andrej Fischer et al. 'High-Definition Reconstruction of Clonal Composition in Cancer'. *Cell Reports* 7.5 (2014), pp. 1740–1752.

[21] Gavin Ha et al. 'TITAN: inference of copy number architectures in clonal cell populations from tumor whole-genome sequence data'. *Genome Research* 24.11 (2014), pp. 1881–1893.

[22] Andrew W. McPherson et al. 'ReMixT: clone-specific genomic structure estimation in cancer'. *Genome Biology* 18.1 (2017), p. 140.

[23] Yupeng Cun et al. 'Copy-number analysis and inference of subclonal populations in cancer genomes using Sclust'. *Nature Protocols* 13.6 (2018), pp. 1488–1501.

[24] Simone Zaccaria and Benjamin J. Raphael. 'Accurate quantification of copy-number aberrations and whole-genome duplications in multi-sample tumor sequencing data'. *Nature Communications* 11.1 (2020), p. 4301.

[25] Gryte Satas et al. 'DeCiFering the elusive cancer cell fraction in tumor heterogeneity and evolution'. *Cell Systems* 12.10 (2021), 1004–1018.e10.

[26] Matthew A. Myers et al. 'HATCHet2: clone- and haplotype-specific copy number inference from bulk tumor sequencing data'. *Genome Biology* 25.1 (2024), p. 130.

[27] Christian Janiesch, Patrick Zschech and Kai Heinrich. 'Machine learning and deep learning'. *Electronic Markets* 31.3 (2021), pp. 685–695.

[28] Artur Szałata et al. 'Transformers in single-cell omics: a review and new perspectives'. *Nature Methods* 21.8 (2024), pp. 1430–1443.

[29] Micaela E. Consens et al. 'Transformers and genome language models'. *Nature Machine Intelligence* 7.3 (2025), pp. 346–362.

[30] Berk Mandiracioglu et al. 'ECOLE: Learning to call copy number variants on whole exome sequencing data'. *Nature Communications* 15.1 (2024), p. 132.

[31] Ren Junjun et al. 'A comprehensive review of deep learning-based variant calling methods'. *Briefings in Functional Genomics* 23.4 (2024), pp. 303–313.

[32] Jing Meng et al. 'DeepSSV: detecting somatic small variants in paired tumor and normal sequencing data with convolutional neural network'. *Briefings in Bioinformatics* 22.4 (2021), bbaa272.

[33] Kiran Krishnamachari et al. 'Accurate somatic variant detection using weakly supervised deep learning'. *Nature Communications* 13.1 (2022), p. 4248.

[34] Michael Poli et al. 'Hyena Hierarchy: Towards Larger Convolutional Language Models'. *Proceedings of the 40th International Conference on Machine Learning*. 2023, pp. 28043–28078.

[35] Albert Gu and Tri Dao. 'Mamba: Linear-Time Sequence Modeling with Selective State Spaces'. *arXiv* (2024).

[36] James W. Cooley and John W. Tukey. 'An Algorithm for the Machine Calculation of Complex Fourier Series'. *Mathematics of Computation* 19.90 (1965), pp. 297–301.

[37] Daniel Y. Fu et al. 'Hungry Hungry Hippos: Towards Language Modeling with State Space Models'. *arXiv* (2023).

[38] Tri Dao and Albert Gu. 'Transformers are SSMs: Generalized Models and Efficient Algorithms Through Structured State Space Duality'. *arXiv* (2024).

[39] Eric Nguyen et al. 'HyenaDNA: Long-Range Genomic Sequence Modeling at Single Nucleotide Resolution'. *Advances in Neural Information Processing Systems* 36 (2023), pp. 43177–43201.

[40] Leo Feng et al. 'Were RNNs All We Needed?' *arXiv* (2024).

[41] David Sims et al. 'Sequencing depth and coverage: key considerations in genomic analyses'. *Nature Reviews Genetics* 15.2 (2014), pp. 121–132.

[42] Kyle Cranmer, Johann Brehmer and Gilles Louppe. 'The frontier of simulation-based inference'. *Proceedings of the National Academy of Sciences* 117.48 (2020), pp. 30055–30062.

[43] Daniel R. Schrider and Andrew D. Kern. 'Supervised Machine Learning for Population Genetics: A New Paradigm'. *Trends in Genetics* 34.4 (2018), pp. 301–312.

[44] Grace Avecilla et al. 'Neural networks enable efficient and accurate simulation-based inference of evolutionary parameters from adaptation dynamics'. *PLOS Biology* 20.5 (2022), e3001633.

[45] Andrew Zammit-Mangion, Matthew Sainsbury-Dale and Raphaël Huser. 'Neural Methods for Amortized Inference'. *Annual Review of Statistics and Its Application* 12 (2025), pp. 311–335.

[46] Lianghui Zhu et al. 'Vision Mamba: Efficient Visual Representation Learning with Bidirectional State Space Model'. *arXiv* (2024).

[47] Gousia Habib and Shaima Qureshi. 'Optimization and acceleration of convolutional neural networks: A survey'. *Journal of King Saud University - Computer and Information Sciences* 34.7 (2022), pp. 4244–4268.

[48] John G Tate et al. 'COSMIC: the Catalogue Of Somatic Mutations In Cancer'. *Nucleic Acids Research* 47.D1 (2019), pp. D941–D947.

[49] Daniall Masood et al. 'Evaluation of somatic copy number variation detection by NGS technologies and bioinformatics tools on a hyper-diploid cancer genome'. *Genome Biology* 25.1 (2024), p. 163.

[50] Amjad Alkodsi, Riku Louhimo and Sampsa Hautaniemi. 'Comparative analysis of methods for identifying somatic copy number alterations from deep sequencing data'. *Briefings in Bioinformatics* 16.2 (2015), pp. 242–254.

[51] E. S. Venkatraman and Adam B. Olshen. 'A faster circular binary segmentation algorithm for the analysis of array CGH data'. *Bioinformatics* 23.6 (2007), pp. 657–663.

[52] Eric Talevich. 'CNVkit: Genome-wide copy number from high-throughput sequencing' (2024). URL: https://cnvkit.readthedocs.io/en/stable/index.html.

[53] Daniel C. Koboldt et al. 'VarScan 2: somatic mutation and copy number alteration discovery in cancer by exome sequencing'. *Genome Research* 22.3 (2012), pp. 568–576.

[54] Micaela E. Consens et al. 'To Transformers and Beyond: Large Language Models for the Genome'. *arXiv* (2023).

[55] Stefan Ivanovic and Mohammed El-Kebir. 'CNRein: an evolution-aware deep reinforcement learning algorithm for single-cell DNA copy number calling'. *Genome Biology* 26.1 (2025), p. 87.

[56] Alexander M. Frankell et al. 'The evolution of lung cancer and impact of subclonal selection in TRACERx'. *Nature* 616.7957 (2023), pp. 525–533.

[57] Pauli Miettinen and Stefan Neumann. 'Recent Developments in Boolean Matrix Factorization'. *arXiv* (2020).

[58] Reka A. Kovacs, Oktay Gunluk and Raphael A. Hauser. 'Binary Matrix Factorisation via Column Generation'. *Proceedings of the AAAI Conference on Artificial Intelligence* 35.5 (2021), pp. 3823–3831.

[59] Pauli Miettinen et al. 'The Discrete Basis Problem'. *IEEE Transactions on Knowledge and Data Engineering* 20.10 (2008), pp. 1348–1362.

[60] Pauli Miettinen and Jilles Vreeken. 'MDL4BMF: Minimum Description Length for Boolean Matrix Factorization'. *ACM Trans. Knowl. Discov. Data* 8.4 (2014), 18:1–18:31.

[61] Claudio Lucchese, Salvatore Orlando and Raffaele Perego. 'A Unifying Framework for Mining Approximate Top- k Binary Patterns'. *IEEE Transactions on Knowledge and Data Engineering* 26.12 (2014), pp. 2900–2913.

[62] Claudio Lucchese, Salvatore Orlando and Raffaele Perego. 'Mining Top-K Patterns from Binary Datasets in presence of Noise'. *Proceedings of the 2010 SIAM International Conference on Data Mining (SDM)*. 2010, pp. 165–176.

[63] Sibylle Hess, Katharina Morik and Nico Piatkowski. 'The PRIMPING routine—Tiling through proximal alternating linearized minimization'. *Data Mining and Knowledge Discovery* 31.4 (2017), pp. 1090–1131.

[64] Jérôme Bolte, Shoham Sabach and Marc Teboulle. 'Proximal alternating linearized minimization for nonconvex and nonsmooth problems'. *Mathematical Programming* 146.1 (2014), pp. 459–494.

[65] Arno Siebes, Jiles Vreeken and Matthijs van Leeuwen. 'Item Sets That Compress'. 2006.

[66] Tobias Achterberg and Roland Wunderling. 'Mixed Integer Programming: Analyzing 12 Years of Progress'. *Facets of Combinatorial Optimization: Festschrift for Martin Grötschel*. Ed. by Michael Jünger and Gerhard Reinelt. 2013, pp. 449–481.

[67] George B. Dantzig and Philip Wolfe. 'Decomposition Principle for Linear Programs'. *Operations Research* 8.1 (1960), pp. 101–111.

[68] A. M. Geoffrion. 'Generalized Benders decomposition'. *Journal of Optimization Theory and Applications* 10.4 (1972), pp. 237–260.

[69] Dimitri P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press, 2014.

[70] E. L. Lawler and D. E. Wood. 'Branch-and-Bound Methods: A Survey'. *Operations Research* 14.4 (1966), pp. 699–719.

[71] François Vanderbeck and Martin W. P. Savelsbergh. 'A generic view of Dantzig–Wolfe decomposition in mixed integer programming'. *Operations Research Letters* 34.3 (2006), pp. 296–306.

[72] Markelle Kelly, Rachel Longjohn and Kolby Nottingham. 'The UCI Machine Learning Repository'. *https://archive.ics.uci.edu*.

[73] F. Maxwell Harper and Joseph A. Konstan. 'The MovieLens Datasets: History and Context'. *ACM Trans. Interact. Intell. Syst.* 5.4 (2015), 19:1–19:19.

[74] Lisa Sikkema et al. 'An integrated cell atlas of the lung in health and disease'. *Nature Medicine* 29.6 (2023), pp. 1563–1577.

[75] Simone Picelli et al. 'Smart-seq2 for sensitive full-length transcriptome profiling in single cells'. *Nature Methods* 10.11 (2013), pp. 1096–1098.

[76] Catalina A. Vallejos et al. 'Normalizing single-cell RNA sequencing data: challenges and opportunities'. *Nature Methods* 14.6 (2017), pp. 565–571.

[77] Valentine Svensson. 'Droplet scRNA-seq is not zero-inflated'. *Nature Biotechnology* 38.2 (2020), pp. 147–150.

[78] F. William Townes et al. 'Feature selection and dimension reduction for single-cell RNA-Seq based on a multinomial model'. *Genome Biology* 20.1 (2019), p. 295.

[79] Michael B. Cole et al. 'Performance Assessment and Selection of Normalization Procedures for Single-Cell RNA-Seq'. *Cell Systems* 8.4 (2019), 315–328.e8.

[80] Christoph Hafemeister and Rahul Satija. 'Normalization and variance stabilization of single-cell RNA-seq data using regularized negative binomial regression'. *Genome Biology* 20.1 (2019), p. 296.

[81] Chih-Hsuan Wu, Xiang Zhou and Mengjie Chen. 'Exploring and mitigating shortcomings in single-cell differential expression analysis with a new statistical paradigm'. *Genome Biology* 26.1 (2025), p. 58.

[82] Gunsagar S. Gulati et al. 'Profiling cell identity and tissue architecture with single-cell and spatial transcriptomics'. *Nature Reviews Molecular Cell Biology* 26.1 (2025), pp. 11–31.

[83] Jian Liu et al. 'Progress and Clinical Application of Single-Cell Transcriptional Sequencing Technology in Cancer Research'. *Frontiers in Oncology* 10 (2021).

[84] Evan Z. Macosko et al. 'Highly Parallel Genome-wide Expression Profiling of Individual Cells Using Nanoliter Droplets'. *Cell* 161.5 (2015), pp. 1202–1214.

[85] Grace X. Y. Zheng et al. 'Massively parallel digital transcriptional profiling of single cells'. *Nature Communications* 8.1 (2017), p. 14049.

[86] Tae Hyun Kim, Xiang Zhou and Mengjie Chen. 'Demystifying "drop-outs" in single-cell UMI data'. *Genome Biology* 21.1 (2020), p. 196.

[87] Lukas Heumos et al. 'Best practices for single-cell analysis across modalities'. *Nature Reviews Genetics* 24.8 (2023), pp. 550–572.

[88] Tallulah S. Andrews et al. 'Tutorial: guidelines for the computational analysis of single-cell RNA sequencing data'. *Nature Protocols* 16.1 (2021), pp. 1–9.

[89] Shaked Slovin et al. 'Single-Cell RNA Sequencing Analysis: A Step-by-Step Overview'. *RNA Bioinformatics*. Ed. by Ernesto Picardi. 2021, pp. 343–365.

[90] Pierre-Luc Germain, Anthony Sonrel and Mark D. Robinson. 'pipeComp, a general framework for the evaluation of computational pipelines, reveals performant single cell RNA-seq preprocessing tools'. *Genome Biology* 21.1 (2020), p. 227.

[91] Ilya Korsunsky et al. 'Fast, sensitive and accurate integration of single-cell data with Harmony'. *Nature Methods* 16.12 (2019), pp. 1289–1296.

[92] Malte D. Luecken et al. 'Benchmarking atlas-level data integration in single-cell genomics'. *Nature Methods* 19.1 (2022), pp. 41–50.

[93] Hoa Thi Nhu Tran et al. 'A benchmark of batch-effect correction methods for single-cell RNA sequencing data'. *Genome Biology* 21.1 (2020), p. 12.

[94] Laleh Haghverdi et al. 'Batch effects in single-cell RNA-sequencing data are corrected by matching mutual nearest neighbors'. *Nature Biotechnology* 36.5 (2018), pp. 421–427.

[95] C. Domínguez Conde et al. 'Cross-tissue immune cell analysis reveals tissue-specific features in humans'. *Science* 376.6594 (2022), eabl5197.

[96] Romain Lopez et al. 'Deep generative modeling for single-cell transcriptomics'. *Nature Methods* 15.12 (2018), pp. 1053–1058.

[97] Luyi Tian et al. 'Benchmarking single cell RNA-sequencing analysis pipelines using mixture control experiments'. *Nature Methods* 16.6 (2019), pp. 479–487.

[98] Jeffrey M. Pullin and Davis J. McCarthy. 'A comparison of marker gene selection methods for single-cell RNA sequencing data'. *Genome Biology* 25.1 (2024), p. 56.

[99] Kenong Su, Tianwei Yu and Hao Wu. 'Accurate feature selection improves single-cell RNA-seq cell clustering'. *Briefings in Bioinformatics* 22.5 (2021), bbab034.

[100] Beate Vieth et al. 'A systematic evaluation of single cell RNA-seq analysis pipelines'. *Nature Communications* 10.1 (2019), p. 4667.

[101] Luke Zappia et al. 'Feature selection methods affect the performance of scRNA-seq data integration and querying'. *Nature Methods* 22.4 (2025), pp. 834–844.

[102] Chenling Xu et al. 'Probabilistic harmonization and annotation of single-cell transcriptomics data with deep generative models'. *Molecular Systems Biology* 17.1 (2021), e9620.

[103] Aaron T. L. Lun, Davis J. McCarthy and John C. Marioni. 'A step-by-step workflow for low-level analysis of single-cell RNA-seq data with Bioconductor'. *F1000Research* (2016).

[104] Robert A. Amezquita et al. 'Orchestrating single-cell analysis with Bioconductor'. *Nature Methods* 17.2 (2020), pp. 137–145.

[105] Yuhan Hao et al. 'Dictionary learning for integrative, multimodal and scalable single-cell analysis'. *Nature Biotechnology* 42.2 (2024), pp. 293–304.

[106] F. Alexander Wolf, Philipp Angerer and Fabian J. Theis. 'SCANPY: large-scale single-cell gene expression data analysis'. *Genome Biology* 19.1 (2018), p. 15.

[107] 'Stack Overflow Developer Survey 2023. Stack Overflow.' (2023). URL: https://survey.stackoverflow.co/2023/?utm_source=social-share&utm_medium=social&utm_campaign=dev-survey-2023 (retrieved 7/7/2025).

[108] 'SlashData's Developer Nation Survey Q1 2025. Developer Nation Community.' (2025). URL: https://www.developernation.net/developer-reports/dn29 (retrieved 7/7/2025).

[109] Bidossessi Wilfried Hounkpe et al. 'HRT Atlas v1.0 database: redefining human and mouse housekeeping genes and candidate reference transcripts by mining massive RNA-seq datasets'. *Nucleic Acids Research* 49.D1 (2021), pp. D947–D955.

[110] Jan-Matthis Lueckmann et al. 'Benchmarking Simulation-Based Inference'. *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*. 2021, pp. 343–351.

[111] Dominic Grün, Lennart Kester and Alexander van Oudenaarden. 'Validation of noise models for single-cell transcriptomics'. *Nature Methods* 11.6 (2014), pp. 637–640.

[112] Davide Risso et al. 'A general and flexible method for signal extraction from single-cell RNA-seq data'. *Nature Communications* 9.1 (2018), p. 284.

[113] Kwangbom Choi et al. 'Bayesian model selection reveals biological origins of zero inflation in single-cell transcriptomics'. *Genome Biology* 21.1 (2020), p. 183.

[114] Rahul Satija et al. 'Spatial reconstruction of single-cell gene expression data'. *Nature Biotechnology* 33.5 (2015), pp. 495–502.

[115] Malte D Luecken and Fabian J Theis. 'Current best practices in single-cell RNA-seq analysis: a tutorial'. *Molecular Systems Biology* 15.6 (2019), e8746.

[116] 'Single Cell Transcriptomics with Python – Single-cell transcriptomics with Python. Swiss Institute of Bioinformatics.' URL: https://sib-swiss.github.io/single-cell-python-training/ (retrieved 7/7/2025).

[117] 'Analysis, visualization, and integration of Visium HD spatial datasets with Seurat. Satija Lab.' (2023). URL: https://satijalab.org/seurat/articles/pbmc3k_tutorial.html (retrieved 7/7/2025).

[118] V. A. Traag, L. Waltman and N. J. van Eck. 'From Louvain to Leiden: guaranteeing well-connected communities'. *Scientific Reports* 9.1 (2019), p. 5233.

[119] Angelo Duò, Mark D. Robinson and Charlotte Soneson. 'A systematic performance evaluation of clustering methods for single-cell RNA-seq data'. *F1000Research* (2018).

[120] Tiago P. Peixoto. 'Descriptive vs. Inferential Community Detection in Networks: Pitfalls, Myths and Half-Truths'. *Elements in the Structure and Dynamics of Complex Networks* (2023).

[121] Leland McInnes, John Healy and James Melville. 'UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction'. *arXiv* (2020).

[122] Nguyen Xuan Vinh, Julien Epps and James Bailey. 'Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance'. *Journal of Machine Learning Research* 11.95 (2010), pp. 2837–2854.

[123] Chuan Xu et al. 'Automatic cell-type harmonization and integration across Human Cell Atlas datasets'. *Cell* 186.26 (2023), 5876–5891.e20.

[124] Mohammad Lotfollahi et al. 'Mapping single-cell data to reference atlases by transfer learning'. *Nature Biotechnology* 40.1 (2022), pp. 121–130.

[125] Xiaoping Han et al. 'Construction of a human cell landscape at single-cell level'. *Nature* 581.7808 (2020), pp. 303–309.

[126] Nicholas Schaum et al. 'Single-cell transcriptomics of 20 mouse organs creates a Tabula Muris'. *Nature* 562.7727 (2018), pp. 367–372.

[127] Maxim V. Kuleshov et al. 'Enrichr: a comprehensive gene set enrichment analysis web server 2016 update'. *Nucleic Acids Research* 44.W1 (2016), W90–W97.

[128] Zhuoqing Fang, Xinyuan Liu and Gary Peltz. 'GSEApy: a comprehensive package for performing gene set enrichment analysis in Python'. *Bioinformatics* 39.1 (2023), btac757.

[129] Vladimir Yu Kiselev, Tallulah S. Andrews and Martin Hemberg. 'Challenges in unsupervised clustering of single-cell RNA-seq data'. *Nature Reviews Genetics* 20.5 (2019), pp. 273–282.

[130] Joseph M Rich et al. 'The impact of package selection and versioning on single-cell RNA-seq analysis'. *bioRxiv* (2024), p. 2024.04.04.588111.

[131] Andrew Butler et al. 'Integrating single-cell transcriptomic data across different conditions, technologies, and species'. *Nature Biotechnology* 36.5 (2018), pp. 411–420.

[132] Luke Zappia, Belinda Phipson and Alicia Oshlack. 'Splatter: simulation of single-cell RNA sequencing data'. *Genome Biology* 18.1 (2017), p. 174.

[133] Genevieve L. Stein-O'Brien et al. 'Enter the Matrix: Factorization Uncovers Knowledge from Omics'. *Trends in Genetics* 34.10 (2018), pp. 790–805.

[134] Jeanette A. I. Johnson et al. 'Inferring cellular and molecular processes in single-cell data with non-negative matrix factorization using Python, R and GenePattern Notebook implementations of CoGAPS'. *Nature Protocols* 18.12 (2023), pp. 3690–3731.

[135] Haiyue Wang and Xiaoke Ma. 'Learning discriminative and structural samples for rare cell types with deep generative model'. *Briefings in Bioinformatics* 23.5 (2022).

[136] Dylan Kotliar et al. 'Identifying gene expression programs of cell-type identity and cellular activity with single-cell RNA-Seq'. *eLife* 8 (2019). Ed. by Alfonso Valencia et al., e43803.

[137] Chunxuan Shao and Thomas Höfer. 'Robust classification of single-cell transcriptome data by nonnegative matrix factorization'. *Bioinformatics* 33.2 (2017), pp. 235–242.

[138] Hanjing Jiang et al. 'Graph-Regularized Non-Negative Matrix Factorization for Single-Cell Clustering in scRNA-Seq Data'. *IEEE Journal of Biomedical and Health Informatics* 28.8 (2024), pp. 4986–4994.

[139] Josephine Yates et al. 'Cell states and neighborhoods in distinct clinical stages of primary and metastatic esophageal adenocarcinoma'. *Cell Reports Medicine* 6.6 (2025).

[140] Richard H. Chapple et al. 'An integrated single-cell RNA-seq map of human neuroblastoma tumors and preclinical models uncovers divergent mesenchymal-like gene expression programs'. *Genome Biology* 25.1 (2024), p. 161.

[141] Lifan Liang, Kunju Zhu and Songjian Lu. 'BEM: Mining Coregulation Patterns in Transcriptomics via Boolean Matrix Factorization'. *Bioinformatics* 36.13 (2020), pp. 4030–4037.

[142] Tammo Rukat et al. 'Bayesian Boolean Matrix Factorisation'. *Proceedings of the 34th International Conference on Machine Learning*. 2017, pp. 2969–2978.

[143] Changlin Wan et al. 'Fast and Efficient Boolean Matrix Factorization by Geometric Segmentation'. *Proceedings of the AAAI Conference on Artificial Intelligence* 34.04 (2020), pp. 6086–6093.

[144] Changlin Wan et al. 'Bias aware probabilistic Boolean matrix factorization'. *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*. 2022, pp. 2035–2044.

[145] Jean Hausser et al. 'Central dogma rates and the trade-off between precision and economy in gene expression'. *Nature Communications* 10.1 (2019), p. 68.

[146] Robert C. Brewster et al. 'The Transcription Factor Titration Effect Dictates Level of Gene Expression'. *Cell* 156.6 (2014), pp. 1312–1323.

[147] Michael Levine and Robert Tjian. 'Transcription regulation and animal diversity'. *Nature* 424.6945 (2003), pp. 147–151.

[148] Uri Alon. 'Design principles of biological circuits'. *Biophysical Journal* 96 (2009).

[149] Hannah Tipney and Lawrence Hunter. 'An introduction to effective use of enrichment analysis software'. *Human Genomics* 4.3 (2010), p. 202.

[150] Noah Hollmann et al. 'Accurate predictions on small data with a tabular foundation model'. *Nature* 637.8045 (2025), pp. 319–326.

[151] Yusri Dwi Heryanto, Yao-zhong Zhang and Seiya Imoto. 'Predicting cell types with supervised contrastive learning on cells and their types'. *Scientific Reports* 14.1 (2024), p. 430.

[152] Shangru Jia et al. 'scDeepInsight: a supervised cell-type identification method for scRNA-seq data with deep learning'. *Briefings in Bioinformatics* 24.5 (2023), bbad266.

[153] Hugo Dalla-Torre et al. 'Nucleotide Transformer: building and evaluating robust foundation models for human genomics'. *Nature Methods* 22.2 (2025), pp. 287–297.

[154] Carl G. de Boer and Jussi Taipale. 'Hold out the genome: a roadmap to solving the cis-regulatory code'. *Nature* 625.7993 (2024), pp. 41–50.