Automatic Econometric Model Selection using PcGets

In December 2005, VSAE of the University of Amsterdam visited the University of Oxford for a morning of lectures on econometric modelling. Perhaps expecting a standard textbook address, we hope the students were pleasantly surprised to be given a flavour of a path-breaking piece of software designed to select econometric models: *PcGets*. It is a delight to write an article summarising the lecture, and hopefully tempting more students to test out PcGets – in my view you cannot fail to be impressed by the results the program delivers.

Jennifer Castle

is 26 years old. She is studying for a PhD in Economics. She undertook an undergraduate degree in Economics at Durham University. Then she moved to Oxford for her M.Phil, in which she wrote a thesis on 'Measuring Excess Demand'.

What is PcGets?

PcGets is a computer-automated approach to econometric model selection, developed by Hendry and Krolzig (2001). The program embodies a fast, powerful and effective algorithm that automatically selects empirical models from the observed data, and is based on a general-to-specific methodology. As a fully interactive, menu driven program that forms part of the popular OxMetrics suite of econometrics packages, *PcGets* is user friendly and accessible. Furthermore, the program can be applied to a wide range of problems in numerous disciplines and is not just applicable to econometric analysis.

How does the algorithm work?

The algorithm is based on a general-to-specific approach. As the world is so complex, there are many variables that may explain the phenomenon of interest. Some of the variables will be relevant and some will be irrelevant. To establish which variables are relevant, *PcGets* explores all feasible reduction paths from a very general starting point, eliminating insignificant variables until only the relevant variables are retained.

The first stage of the algorithm requires the user to specify the general unrestricted model (GUM). This just requires the user to identify potentially relevant variables in the database and specify the lag length. The empirical success of *PcGets* depends of course on the creativity of the researcher in specifying a feasible, congruent general model for the available data, but the program implements many tests to check for model mis-specification. Many factors will influence the GUM, including previous theoretical and empirical findings, likely functional form transformations, known anomalies such as breaks and measurement changes, and data availability.

Having specified the initial model, the second stage tests the GUM for mis-specification. There must be sufficient tests to check the GUM for congruency (which implies that the model matches the evidence in all measured aspects) but not too many to induce a large type-1 error. If the mis-specification tests are rejected, the GUM needs to be reformulated, rethinking how to best tackle the problem. However, if the mis-specification tests are passed, a specific model is then selected.

The first stage in selecting a congruent, parsimonious model involves a pre-search simplification of the GUM. This powerful pre-selection screening eliminates variables that are 'highly' irrelevant. The reductions are undertaken based on F-tests and t-tests of the variables ranked in order of their absolute t-values, using loose significance levels. Both 'top-down' and 'bottom-up' searches are undertaken, along with a lag-order pre-selection test to establish the lag length. The diagnostic tests are confirmed at every reduction stage to ensure congruency and the failure of a diagnostic test will terminate the reduction at that point.

Having eliminated highly insignificant variables, the program then implements a multi-path search, commencing from all feasible deletion points. The searches repeatedly filter for

relevant variables using both t-tests and block F-tests. Again, diagnostic tests are checked at every reduction stage. The path is terminated when all the variables that remain are significant, or a diagnostic test fails. Hence, the algorithm will deliver a set of non-null terminal models that comprise the set of distinct, minimal, congruent reductions found along all search paths. When more than one such model is found, encompassing tests are used to select between the candidate congruent models. Each model is tested against the union of the models and selection is again undertaken until a unique model is selected. If the encompassing tests are inconclusive, the final model is selected using information criteria.

The final stage in the algorithm undertakes a post-search evaluation of the selected model to establish sub-sample reliability. *PcGets* reports the reliability of the retained regressors based on their significance in the final model and overlapping sub-samples, capturing recursive estimation characteristics that aim to establish parameter constancy.

"PcGets is an immensely powerful, laboursaving tool"

What is the secret of its success?

In 1997, the computer chess program, Deep Blue, beat the reigning world champion, Garry Kasparov, in a six game match under standard tournament time controls. This was the culmination of a 22 year engineering effort. The reason for Deep Blue's success was that it was able to search many paths ahead in order to determine the optimal move. *PcGets* seeks to mirror the achievements of Deep Blue by commencing from every feasible deletion point and searching all possible reduction paths. This enables the program to check for hidden relations in the data, highlighting relevant variables and ensuring that all reductions are acceptable with diagnostic tests. Making use of the huge advances in computing power enables large data sets in which there are many potentially relevant variables to be explored in this way.

What strategy should I use?

Imagine you wish to make your fortune by panning for gold. You invest in a pan and a good pair of Wellington boots and head for the river. After a while, your pan will contain a mixture of dirt and sand as well as those valuable nuggets of gold. By carefully sieving through the mud, you should be left with just the gold nuggets. In econometric modelling terms, your GUM will contain a mixture of relevant and irrelevant variables. By carefully throwing out the irrelevant variables, you are just left with the variables that explain the phenomenon of interest.
The nuggets retained may be large nuggets which are easy to see in the mud. In this case, the relevant variables have large non-centralities and a conservative strategy should be used, reducing the non-deletion probability. The conservative strategy is calibrated to deliver a 1% null rejection frequency rate. On the other hand, if the nuggets are small they may be difficult to detect in the mud. In this situation, the relevant variables have low non-centralities and the liberal strategy should be used, reducing the non-selection probability. The liberal strategy is calibrated to deliver a 5% null rejection frequency rate. Of course, if there are small nuggets in a large pile of dirt, the task of sorting is difficult because the null and alternative distributions overlap. In this case, the risks of omitting relevant variables versus the risks of retaining irrelevant variables need to be balanced depending on the problem at hand. Finally, if the pan is too small to contain all the dirt and gold, you can just keep sieving. Split the pile into sub-samples and sieve, storing the retained contents of the pan at each stage. Then, all the retained content is merged and sieved again to deliver the gold nuggets. Thus, the program can even handle more variables than observations.

What if I am no expert at econometric modelling?

There is a mode in *PcGets* for the non-expert user (see Hendry and Krolzig, 2005). The user simply specifies the appropriate functions of the regressand and the basic regressors, and then *PcGets* creates the GUM and selects a model. The strategy, denoted the 'quick modeller' option, only requires the set of variables, leaving the computer to do the hard work for you. The program can select a static regression model, an autoregressive distributed lag model, or an equilibrium-correction model. The program will automatically select the lag length (based on the data frequency and sample size), undertake any outlier correction or seasonality requirements, implement the PcGive unit

root test, transform the variables to differences (based on the largest lag coefficient) and any cointegrating combinations, specify an I(0) representation of the GUM, and re-estimate using the usual procedures for selecting a parsimonious, congruent, undominated model. The quick modeller is designed to open up econometric analysis to non-expert users, and will be particularly useful in less developed countries where knowledge, skill, expertise, and research time is perhaps more limited.

Is there evidence that the program works?

Extensive Monte Carlo simulations and theoretical analyses have demonstrated the remarkable properties of *PcGets* in model selection: see Krolzig and Hendry (2001) and Hendry and Krolzig (2003, 2005). The operational characteristics of *PcGets* are excellent across a wide range of states of nature, though not for all imaginable settings. In particular, false rejection frequencies of null hypotheses can be controlled at the desired level; correct rejections of alternatives are close to the theoretical upper bound of power; model selection is consistent; parameter estimates can be bias corrected for retained variables, downweighting adventitious significance; and reported standard errors can be accurate estimates of those that would be calculated had the correct specification been known. Monte Carlo experiments show that *PcGets* recovers the correct specification from a general model with size and power close to commencing from the data generation process (DGP) itself. Furthermore, numerous empirical econometric studies have demonstrated the excellent properties of the *PcGets* algorithm, proving indispensable in a wide range of both cross-section and time-series studies.

What is the future for PcGets?

Research is ongoing, with many important new developments aimed at further improving the *PcGets* algorithm. Given that the program can now handle more regressors than observations, research into selecting non-linear models is feasible. Furthermore, selecting from a set of perfectly collinear regressors is no longer an intractable problem, and modelling linear simultaneous systems in the absence of a priori identifying restrictions is a further extension.

Why should I use PcGets?

PcGets is an immensely powerful, labour-saving tool. The algorithm represents a switch from a labour intensive activity to a capital intensive activity - freeing invaluable time for the user to think about the model, and interpret the evidence. The user is required to specify the general model nesting alternative economic theories, lag structures etc., and then let *PcGets* – and the computer – do the rest

References

Hendry, D.F., and Krolzig, H-M. (2001). *Automatic Econometric Model Selection.* London: Timberlake Consultants Press.

Hendry, D.F., and Krolzig, H-M. (2003). New developments in automatic general-to-specific modelling. In Stigum, B.P. (ed.). *Econometrics and the Philosophy of Economics*. Princeton: Princeton University Press. 379-419

Hendry, D.F., and Krolzig, H-M. (2005). The properties of automatic Gets modelling. *Economic Journal*, **115**, C32-C61.

Krolzig, H-M., and Hendry, D.F. (2001). Computer automation of general-to-specific model selection procedures. *Journal of Economic Dynamics and Control*, **25**, 831-866.