The Stata Journal (*yyyy*) *vv*, Number *ii*

General to Specific Modelling in Stata

Damian C. Clarke
Department of Economics, The University of Oxford
Oxford, United Kingdom
damian.clarke@economics.ox.ac.uk

Abstract. This article describes the application of a well known model fitting algorithm to Stata: general-to-specific (GETS) modelling. It discusses a number of empirical issues in GETS modelling, specifically how such an algorithm can be applied to estimations based upon cross-sectional, time-series and panel data. A command is then presented, written in Stata and Mata, which implements this algorithm for various data types in a flexible way. This command, `gets`, is illustrated using real empirical data from applied studies of GETS modelling with Monte Carlo simulation. It is shown to perform as empirically predicted, and to have good size and power (or gauge and potency) properties under simulation.

Keywords: notag27, gets, model selection, specification tests, general to specific, encompassing

1 Introduction

General-to-specific modelling (hereafter GETS) is an important class of model selection in econometrics. In comparison to econometrics based on *a priori* economic models which are only loosened if found to fail important econometric tests, a GETS-based approach allows the econometrician to be reasonably un-restrictive with their theory, letting the data 'speak for itself' when identifying the final model. This is of course, not to suggest that this is a-theoretical econometrics. The econometrician must still base inference on an economic theory which allows them to propose some general unrestricted model (or GUM). However, in the general-to-specific approach, this GUM is challenged, with the goal being the identification of some final model which is both true to the information contained in the GUM, but also offering a more parsimonious representation of the world.

This can in some sense be though of as data-based model selection. While this may seem concerning to economists typically concerned with 'data-mining', the LSE-school of modelling, led perhaps by Hendry and co-authors, suggests that this is not necessarily the case. Rather than assuming that the econometrician perfectly understands the true data generating process (DGP)[1], a GETS approach espouses the formation of some general theory which may be appropriate, and encompass, a wide range of true models. Then, by the theory of reduction, the true model—simply a more compact form of the GUM—is estimated, which is void of redundant variables.

This paper does *not* aim to examine the philosophical validity of the LSE class of

1. Which Campos et al. (2005) suggest is an "arduos" task given the complex nature of the world...

econometrics compared to other methods of model selection. Simply, we aim to introduce an alternative tool for model selection to econometricians using Stata. This tool, `gets`, is similar to that which already exists in certain other languages like OxMetrix, and is a useful extension to Stata's functionality. We show that `gets` performs as empirically expected, and does a good job in recovering the true underlying model in a range of simulations.

The `gets` program is defined in a flexible way to make it functional in a range of econometric situations. It can be used with cross-sectional, panel, and time-series data, and works with Stata functions which are appropriate in models of these types. It places no limitations on arbitrary misspecification of models, allowing such features as clustered standard errors, robust standard errors, and bootstrap and jackknife based estimation.

In order to define an algorithm which is appropriate for a range of very different underlying econometric models, a number of specific definitions must be made. The nature of general-to-specific modelling requires that the GUM is subject to a range of pre-specification tests in order to ensure that it complies with the modelling assumptions upon which estimation is based. These assumptions, and indeed the resulting tests, vary by model type. In the following section we define and discuss the appropriate tests to run in a range of situations, and also provide discussion of how to select between various competing final models in different circumstances.

This paper then takes a pre-existing benchmark in GETS modelling (Hoover and Perez 1999), and shows that similar performance can be achieved in Stata. These results suggest that general-to-specific modelling, and the user-written `gets` algorithm, may be useful to Stata users interested in defining appropriate, flexible and data-driven economic models.

2 Algorithm Description

The search algorithm undertaken by `gets` depends upon the model type and GUM specified by the user. Whether the underlying model is based upon cross-sectional, time-series, or panel data determines the set of initial tests ("the battery") and the set of subsequent tests run at each stage of the search path. In what follows we discuss the general search algorithm followed for every model, delaying discussion of specific tests until the corresponding subsections for cross-sectional, time-series, and panel models respectively.

In defining the search algorithm, we follow that described in Hoover and Perez (1999), and in appendix A of Hoover and Perez (2004). Hoover and Perez (1999) is generally considered to be an important starting point in the description of a computational general-to-specific modelling process (see for example Campos et al. (2005)). The algorithm takes the following form:

1. The user specifies their proposed GUM, and indicates to Stata the relevant data,

if necessary using *if* and *in* specifiers.

2. 90 percent of the full sample is retained, while the remaining 10 percent is set aside for out-of-sample testing. On this 90 percent sample the battery of tests is run at the nominal size.[2]. If one of these tests is failed, it is eliminated from the battery in the following steps of the search path (for the current replication). If more than one of these tests is failed by the GUM, the user is instructed that the GUM is likely a poor representation of the true model, and an alternative general model is requested.[3]

3. Each variable in the general model is ranked by the size of their t-statistic and the algorithm then follows n (by default 5) search paths. The first search path is initiated by eliminating the lowest (insignificant) variable from the GUM. The second follows the same process, but rather than eliminating the lowest, eliminates the second lowest. This process is followed until reaching the n^{th} search path which eliminates the n^{th} lowest variable. For each search path, the current specification then includes all remaining variables, and this specification is estimated.

4. The current specification is then subjected to the full battery of tests, along with an F-test to determine if the current specification is a valid restriction of the GUM. If any of these tests are failed the current search path is abandoned, and the algorithm jumps to the subsequent search path.

5. If the current specification passes the above tests, the variables in the current specification are once again ordered by the size of their t-statistic, and the next lowest variable is eliminated. This then becomes a potential current specification, which is subjected to the battery of tests. If any of these tests are failed, the model reverts to the previous current specification, and the second lowest (insignificant) variable is eliminated. Such a process is followed until a variable is successfully eliminated, or all insignificant variables have been attempted. If an insignificant variable is eliminated, stage 5 is then restarted with the current specification. This process is then followed iteratively, until either all insignificant variables have been eliminated, or no more variables can be successfully removed.

6. Once no further variables can be eliminated, a potential terminal specification is reached. This specification is estimated on the full sample of data. If all variables are significant it is accepted as the terminal specification. If any insignificant variables remain, these are eliminated as a group and the new terminal specification is subjected to the battery of tests. If it passes these tests it is the terminal specification, and if not, the previous terminal specification is accepted.

7. Each of the n terminal specifications is compared, and if these are different, the final specification is determined using an information criterion or encompassing (see the related discussion in sections 2.1 -2.3).

2. In subsections 2.1–2.3 we discuss the specific nature of these tests, and the determination of the in-sample and out-of-sample observations
3. As in all terminal decisions, the user is able to override this decision and continue with their proposed GUM if so desired.

2.1 Cross-sectional Models

Cross sectional models are subjected to an initial battery of five tests. These are a Doornik-Hansen test for normality of errors, the Breusch and Pagan (1979) test for homoscedasticity of errors[4], the RESET test for the linearity of coefficients (Ramsey 1969) and an in-sample and out-of-sample stability F-test. These two final tests consist of a comparison of regressions of each subsample to estimation results for the full sample: in the in-sample case the two subsamples are made up of two halves of the full sample, while in the out-of-sample test this is a comparison between the 90 percent and ten percent samples. These tests are analogous to Chow (1960) tests.

The determination of the final model using OLS with cross-sectional data is via information criteria. For each of the n potential terminal specifications, a regression is run and the Bayesian Information Criterion (BIC) is calculated. That terminal specification which has the lowest BIC is determined to be the final specification.

2.2 Time-Series Models

In time series models, an additional test in included in the battery discussed above. Along with Doornik-Hansen, Breusch-Pagan, RESET, and in- and out-of-sample Chow tests, a test is run for autocorrelated conditional heteroscedasticity (or ARCH) up to the second order (Engle 1982). In order to partition the sample into 'in-sample' and 'out-of-sample', the final 10% of observations are initially discarded to be used in out of sample tests. These are (as in all cases) returned to the sample in the calculation of the final model, and in choosing between various terminal specifications, a BIC is once again used.

2.3 Panel-data Models

Given the nature of panel data, the inital battery of tests here potentially includes two tests not included in cross-sectional or time-series models. The first of these is a test for serial correlation of the idiosyncratic portion of the error term (discussed by Wooldridge (2002), and implemented for Stata by Drukker (2003)). The second of these is a Lagrange multiplier test for random effects (in the case that a random-effects model is specified), and tests the validity of said model (Breusch and Pagan 1980). Along with these tests a Dornik-Hansen type test for normality of the idiosyncratic portion of the error term is estimated, and in-sample and out-of-sample Chow tests (as discussed in the previous two subsections).

In order to determine the final specification from the n potential resulting terminal specifications, an encompassing procedure is used. Each variable included in at least *one* terminal specification is included in the potential terminal model. This model is then tested according to step 6 of algorithm listed in section 2.

4. This test is not run if the the model estimated is robust to this type of misspecification.

3 The gets command

3.1 Syntax

The syntax of the `gets` command is as follows.

`gets` *yvar xvars* [*if*] [*in*] [*weight*] [`, vce(`*vcetype*`) xt(re|fe|be) ts nodiagnostic tlimit(#) numsearch(#) nopartition noserial verbose`]

Here *yvar* refers to the dependent variable in the general model, and *xvars* to the full set of independent variables to be tested for inclusion in the final model.

3.2 Options

`vce(`*vcetype*`)` determines the type of standard error reported in the estimated regression model, and allows standard errors that are robust to certain types of misspecification. *vcetype* may be robust, cluster clustvar, bootstrap, or jackknife.

`xt(re|fe|be)` specifies that the model is based on panel data. The user must specify whether they wish to estimate a random-effects (RE), fixed-effects (FE), or between-effects (BE) model. By default `re` is assumed. `xtset` must be specified prior to using this option.

`ts` specifies that the model is based on time-series data. `tsset` must be specified prior to using this option, and if specified, time-series operators may be used.

`nodiagnostic` turns off the initial diagnostic tests for model misspecification. This should be used with caution.

`tlimit(#)` sets the critical t-value above which variables are considered as important in the terminal specification. By default this is a value of 1.96.

`numsearch` defines the number of search paths to follow in the model. If not specified, 5 search paths are followed. If a large dataset is used, fewer search paths may be preferred.

`nopartition` uses the full sample of data in all search paths, and does not engage in out of sample testing.

`noserial` requests that no serial correlation test is performed if panel data is being used. This option should only be specified with the `xt` option.

`verbose` requests full program output of each search path explored.

3.3 Returned Objects

`gets` is an eclass program, and returns a number of elements in the e list. It returns the BIC of the preferred model, along with the list of variables preferred by the model, and—

in the case that the final model is not encompassing—a full list of variables which are determined as important in *any* terminal specification. These are available as `e(fit)`, `e(gets)`, and `e(gets_all)` respectively. The full ereturn list which includes regression results for the terminal specification is available as `ereturn list`.

4 Performance

4.1 An example with Empirical Data

In order to illustrate the performance of `gets`, we use empirical data from a well-known applied study of general-to-specific modelling. Hoover and Perez (1999), themselves using data from Lovell (1983), illustrate that general-to-specific modelling can work well in recovering the true data in empirical applictaions, even when prospective variables are quite multicolinear. We use the Hoover-Perez dataset in the below example. A brief description of the source and nature of the data is provided in data appendix 1.

We use model 5 to provide an example of the functionality of `gets`. As described in table 2, the dependent variable in model 5 is generated according to:

$$y_{5t} = -0.046 \times ggeq_t + 0.11 \times u_t. \tag{1}$$

In the following Stata excerpt, we see that after loading the dataset, defining the full set of candidate variables (first and second lags of all independent variables, and first to fourth lags of y_{5t}), the gets algorithm searches and returns to us a model with only one dependent variable. As desired, this final model *is* actually the true data generating process, with slight sampling variation in the coefficient on *ggeq* due to the relatively small sample size.

It is noted however that `gets` raises one warning: here the GUM does not pass the full battery of defined tests. Specifically, the GUM fails the in-sample Chow test, suggeting that the coefficients estimated over only the first half of the series are statistically different to those estimated over the second half. While this may be evidence of a structural break signalling that the GUM may not be an appropriate model `gets` respects the GUM entered by the user and continues to search for (and find) the true model.

```
. webuse set "http://users.ox.ac.uk/~ball3491/"
(prefix now "http://users.ox.ac.uk/~ball3491/")

. webuse gets_data
(Hoover and Perez (1999) data for use in GETS modelling)

. qui ds y* u* time, not

. local xvars `r(varlist)´

. local lags l.dcoinc l.gd l.ggeq l.ggfeq l.ggfr l.gnpq l.gydq l.gpiq l.fmrra l
> .fmbase l.fm1dq l.fm2dq l.fsdj l.fyaaac l.lhc l.lhur l.mu l.mo

. gets y5 `xvars´ `lags´ l.y6 l2.y6 l3.y6 l4.y6, ts
# of observations is > 10% of sample size.  Will not run out-of-sample tests.
The in-sample Chow test rejects equality of coefficients

The GUM fails 1 of 4 misspecification tests.  Dornik-Hansen test for normality
```

```
> of errors not rejected.  The presence of (1 and 2 order) ARCH components is r
> ejected. Breusch-Pagan test for homoscedasticity of errors not rejected.
```

Source	SS	df	MS		
Model	23.6849853	1	23.6849853	Number of obs	= 143
Residual	1.69848221	141	.012045973	F(1, 141)	= 1966.22
				Prob > F	= 0.0000
Total	25.3834675	142	.178756814	R-squared	= 0.9331
				Adj R-squared	= 0.9326
				Root MSE	= .10975

y5	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
ggeq	-.0463615	.0010455	-44.34	0.000	-.0484284	-.0442945
_cons	-.0042157	.0091781	-0.46	0.647	-.0223602	.0139289

4.2 Monte Carlo Simulation

The example from the preceding section suggests that a general-to-specific algorithm performed well in this particular case, however in reality we are interested in testing whether `gets` performs as empirically expected over a larger range of models. For this reason we run a set of Monte Carlo simulations based upon the empirical data described above and in data appendix 1. The reason we test the performance of `gets` on this data is two-fold. Firstly, the highly multi-colinear nature of many of these variables makes recovering the true DGP a challenge for automated search algorithms. Secondly, and fundamentally, there is already a benchmark performance test of how a general-to-specific algorithm should work on this data available in Hoover and Perez (1999)'s results.

The Monte Carlo simulation is designed as follows. We draw a normally distributed random variable for use as the u term described in table 2. Based upon this draw (u^D), we then generate the corresponding u^*, (u^{*D}), and then, combining u^D, u^{*D}, and the true macroeconomic variables, we simulate each of our nine different outcome variables $y_1^D, \ldots, y_9^D$. Once having one simulation for each dependent variable we run `gets` with the 40 candidate variables, and determine whether the true DGP is recovered. This process makes up one simulation. We then repeat this 1000 times, observing in each case whether `gets` identifies the true model, and if not, how many of the true variables are correctly included, and how many false variables are incorrectly included.

In order to determine the performance of the search algorithm, we compare the performance of `gets` to that of benchmark performance described in table 7 of Hoover and Perez (1999). We focus on two important summary statistics: gauge and potency. Gauge refers to the percent of irrelevant variables in the final model (regardless of whether they are significant or not). The gauge shows the frequency of type I error in the search algorithm, and is analogous to power in typical econometric tests. The potency of our model refers to the percent of relevant variables in our final model. We would hope in most searches that potency is approximately 100%, as the final model should at the very least not discard true variables. Indeed, we would likely prefer to

have a higher gauge (and more irrelevant—and perhaps insignificant—variables), if this implies that the final model includes all true variables.

Table 1 presents the performance of the `gets` search algorithm, and compares this with the benchmark levels expected. In each case we see that `gets` performs approximately identically to Hoover and Perez's empirical observations. Fundamentally, the potency of `gets` is identical to that expected based upon this data, suggesting that the search algorithm performs as expected in identifying true variables. We do however see that `gets` is slightly more likely to incorrectly include false variables, having a higher gauge than benchmark performance. This is likely due to a slight difference in the nature of the battery of tests in `gets` compared with that of Hoover and Perez's algorithm. In `gets` by default the critical value for the battery of tests is set at 5%, as this increases the likelihood that a specific test is retained for the full search-path. In the below simulations, Hoover and Perez report results for a critical value of 1% in the battery tests, while the `gets` algorithm reports results at 5% (and 1% for the critical t-value when eliminating irrelevant variables).

(*Continued on next page*)

	Models									
	1	2	3	4	5	6	7	8	9	Average
Panel A: Algorithm Performance										
Average rate of inclusion of:										
True Variables	N/A	1.00	1.89	1.00	1.00	1.01	3.00	2.95	2.33	-
False Variables	0.24	1.42	0.64	0.51	0.55	0.62	2.29	0.73	1.39	0.93
Gauge	0.6%	3.6%	1.6%	1.3%	1.4%	1.5%	5.7%	1.8%	3.5%	2.3%
Potency	N/A	100.0%	94.5%	100.0%	100.0%	50.1%	100.0%	98.2%	46.6%	87.7
Panel B: Benchmark Performance										
Average rate of inclusion of:										
True Variables	N/A	1.00	1.89	0.99	1.00	1.01	2.82	3.00	2.86	-
Falsely Variables	0.29	2.31	0.39	0.34	0.32	0.26	1.23	0.38	1.20	0.75
Gauge	0.7%	5.7%	0.9%	0.8%	0.7%	0.6%	3.0%	0.9%	3.2%	1.8%
Potency	N/A	100.0%	94.7%	99.9%	100.0%	50.3%	94.0%	99.9%	57.3%	87.0%

NOTES: Panel A shows the performance of the user-algorithm written for Stata `gets`, while Panel B shows the benchmark algorithm of Hoover and Perez (1999) who simulate using the same data (see table their 7 for original results). Results from each panel are from 1000 simulations with a two-tail critical value of 1%. The DGP for each model is described in the data appendix of this paper, and each model includes a constant which is ignored when calculating the gauge and scope. Full code and simulation results for replication are made available on the author's website.

Table 1: Performance of `gets` in Monte Carlo simulation

5 Conclusion

This paper introduces the `gets` command to Stata, and shows that this command behaves approximately as empirically expected, and is successful in recovering the true model when passed a large set of potential variables to choose from. Such a modelling technique offers important benefits to a range of users who are interested in identifying an underlying economic model whilst being relatively agnostic, or placing few restrictions to their general theory. This so called 'LSE approach' is popular in macroeconometric modelling, however is also amenable to cross-sectional and panel data based microeconometric approaches.

The `gets` command is flexible, allowing the user to choose from a wide array of models, using either time-series, panel, or cross-sectional data. This paper discusses a number of empirical considerations in developing such an algorithm: particularly the nature of the tests desired when examining the proposed *general* model, and how to deal with model selection when choosing between a number of *specific* models.

6 References

Breusch, T. S., and A. R. Pagan. 1979. A simple test for heteroscedasticity and random coefficient variation. *Econometrica* 47: 1287–1294.

———. 1980. The Lagrange multiplier test and its applications to model specification in econometrics. *Review of Economic Studies* 47: 239–253.

Campos, J., N. R. Ericsson, and D. F. Hendry. 2005. General-to-specific modeling: an overview and selected bibliography. Technical report.

Chow, G. C. 1960. Tests of equality between sets of coefficients in two linear regressions. *Econometrica* 28: 591–605.

Drukker, D. M. 2003. Testing for serial correlation in linear panel-data models. *Stata Journal* 3(2): 168–177.

Engle, R. F. 1982. Autoregressive conditional heteroscedasticity, with estimates of the variance of United Kingdom inflations. *Econometrica* 50: 987–1007.

Hoover, K. D., and S. J. Perez. 1999. Data mining reconsidered: encompassing the general-to-specific approach to specification search. *Econometrics Journal* 2: 167–191.

———. 2004. Truth and Robustness in Cross-country Growth Regressions. *Oxford Bulletin of Economics and Statistics* 66(5): 765–798.

Lovell, M. C. 1983. Data mining. *Reveiw of Economic Statistics* 65: 1–12.

Ramsey, J. B. 1969. Tests for specification errors in classical linear least-squares regression analysis. *Journal of the Royal Statistical Society* 31(B): 350–371.

Wooldridge, J. M. 2002. *Econometric Analysis of Cross Section and Panel Data*. Cambridge, MA: MIT Press.

Appendices

1 Data Appendix

In order to test the performance of `gets`, we use the benchmark performance of Hoover and Perez (1999). They use data from the CITIBASE, the Citibank economic database, with 18 macroeconomic variables over the period 1959 quarter 1 to 1995 quarter 1. These variables include GNP, M1, M2, Labour force and Unemployment rates, government purchases, and so on. They then difference this data to ensure that each series is stationary.

In this paper we work with the same dataset after performing the same transformations. From these 18 underlying macroeconomic variables (and their first lags), Hoover and Perez then generate artificial variables for consumption. Nine such models are generated based upon two different independent variables and their lags, and lags of the dependent variable. In table 2 we briefly describe these models (as laid out in table 3 of Hoover and Perez).

Model	DGP
Model 1	$y_{1t} = 130.0 \times u_t$
Model 2	$y_{2t} = 130.0 \times u_t^*$
Model 3	$ln(y_3)_t = 0.395 \times ln(y_3)_{t-1} + 0.3995 \times ln(y_3)_{t-2} + 0.00172 \times u_t$
Model 4	$y_{4t} = 1.33 \times fm1dq_t + 9.73 \times u_t$
Model 5	$y_{5t} = -0.046 \times ggeq_t + 0.11 \times u_t$
Model 6	$y_{6t} = 0.67 \times fm1dq_t - 0.023 \times ggeq_t + 4.92 \times u_t$
Model 7	$y_{7t} = 1.33 \times fm1dq_t + 9.73 \times u_t$
Model 8	$y_{8t} = -0.046 \times ggeq_t + 0.11u_t^*$
Model 9	$y_{9t} = 0.67 \times fm1dq_t - 0.023 \times ggeq_t + 4.92u_t$

Notes: The error terms follow $u_t \sim N(0,1)$ and $u_t^* = 0.75u_{t-1}^* + u_t\sqrt{7/4}$. Models involving the AR(1) u_t^* can be rearranged to include only u_t and one lag of the dependent variable and any independent variables included in the model. The independent variable $fm1dq_t$ refers to M1 money supply, and $ggeq_t$ refers to government spending.

Table 2: Models to test the performance of `gets`

Each of these nine models results in one artificial consumption variable denominated y_{nt}. These y_{nt} variables are then used as the dependent variable for a general-to-specific model search, with 40 independent variables included as candidate variables. These 40 variables are each of the 18 macroeconomic variables in the CITIBASE dataset, the first lags of these variables, and the first to fourth lags of the y_{nt} variable in question. The full transformed dataset including a simulated set of u and y_{nt} variables is available on the author's website for download.[5]

5. The un-transformed original data is also available.