Chapter 13: Automatic Regression

Recently, Hoover and Perez[1], with further refinement by Hendry and Krozlig[2], have proposed a computer-based "general to specific" method to recover with great reliability the underlying data-generating process. In GROCER such a method has been programmed, in a function called automatic. Since this method is relatively new, a fist part is devoted to the theoretical background of the method. Unlike most of other econometric methods, this one presents itself as an algorithm, so the use of automatic assumes the understanding of at least the great lines of this algorithm: this is what part 2 of this chapter is devoted to. Part 3 explains more classically how to use automatic and part 4 is devoted, as usual, to a presentation of the specifications of all the functions involved in an automatic regression.

1. Theoretical background of automatic regression

When a data-generating process (DGP) involves a small number of variables among a much greater set of potentially important variables, then a researcher who wants to recover the true model among all the -linear- models than can be built from this set faces the following difficulties:

- the cost of search can be very high: if there are n variables in the initial set and if she allows every variable to be present or absent the regression, then there are 2^n potential models;
- a top-down method based upon the successive elimination, by the mean of a testing procedure (based for instance on the successive elimination of the variable with the lowest Student t), may lead the user to retain erroneous variables, since type I errors are known to accumulate; if the user chooses to protect herself against such a risk and chooses a high significance level, then she will miss some important variables; moreover, if there is some colinearity between variables, then there is a significant risk of eliminating a relevant variable.

The approach advocated by Hoover-Perez and Hendry-Krozlig consists then in exploring a limited number of paths, starting from each initially non significant variable and performing thereafter the successive elimination of the less significant variables, provided that the model passes well chosen specification tests. This approach will lead at the end to a few models, which can then be chosen by encompassing tests, or if this is not sufficient to obtain only one model, by an information criterion such as Akaïke's one.

The exploration of a limited number of paths and the use of specifications tests cover against the risk of eliminating a relevant variable. Hendry and Krozlig Monte Carlo simulations show that this method leads indeed to very satisfactory results: the average inclusion rate of a non relevant variable can be set at a low level, and the average inclusion rate of a relevant variable at a high level.

2. The algorithm

Let the DGP be:

$E\left(y_t / x_t\right) = x_t b$ where x_t is a (1xp) vector of variables $(x_t^1, \ldots, x_t^p)$ and b a (px1) vector of parameters.

[1] K.D Hoover, S.J. Perez (1999) : « Data mining reconsidered: a general to specific approach to specification search », *Econometrics Journal*, n°2, 167-191.

[2] D.F. Hendry and H-M Krozlig (1999): « Improving on 'data mining reconsidered' by K.D. Hoover and S.J. Perez », Econometric Journal, n° 2, 41–58; D.F. Hendry and H-M Krozlig (2000): « Computer Automation of General-to-Specific Model Selection Procedures », *Journal of Economic Dynamics and Control*, 25 (6-7), 831-866.

Let $Z = \{z^1,\ldots,z^n\}$ be a set of variables which are potentially relevant, with $x^1,\ldots x^k$ belonging to Z and let $E(y_t / x_t) = z_t\beta$ be the corresponding postulated *general* model.

Let α, η, φ, γ be 4 real numbers between 0 and 1: these are significance levels; generally, they should be equal to 0.05 or 0.01 (except for φ). Let Ψ be a real number greater than 1.

The algorithm consists in the following potential 5 steps, the algorithm stopping when it has found the *final* model:

First step: Estimation of the general model

i) The *general* model is estimated, which leads to an estimated coefficient $\hat{\beta}$. If all variables are significant at the α level, then the *general* model is the *final* model.

ii) If at least one coefficient is non significant at the pre-specified α level, then perform k significance tests. If these specification tests do not all pass the pre-specified level η[3], then adjust the significance level of the corresponding tests by a factor Ψ (the fact that at least one of the specification test is rejected should lead the user to add other variables to the information set, but this is clearly a task that the computer is not able to do, so the process goes on, but the significance level is adjusted to avoid rejecting too many subsequent models, if not all).

Let then $s^* = [s_1^*;\ldots; s_k^*]$ be the corresponding vector of significance levels:

- $s_i = \eta$ if the corresponding test has been successfully passed
- $s_i = \eta/\Psi$ if the corresponding test has not been successfully passed

Second step: Elimination of globally insignificant variables
(programmed in function <u>auto_stage0</u>)

i) Range all $\hat{\beta}_i$ from the lowest absolute value of Student's t to the greatest. Set (for initialisation sake) j=1; F = 0 and s=s* .
ii) while $F \leq \varphi_u$ and $s \geq s^*$ then:

- estimate the model with the j least significant variables withdrawn;
- calculate the Fisher test, F, of this model against the *general* model;
- calculate the vector s of p-values of the k specification tests.

(*) iv) call the *top-down* model the last model that verify $F \leq \varphi_u$ and $s \geq s^*$
v) take all variables in the general model that are significant at the α level and estimate the corresponding model (*bottom-up* model);
vi) if the *bottom-up* model is not rejected at the φ_b level against the *top-down* one, that define it as the *second step* model; if it is, then define the *top-down* model as the *second step* model

Third step: Multiple reduction paths
(programmed in function <u>auto_stage1</u>)

For all insignificant coefficients at level α of the *second step* model:
i) Drop the corresponding variable from the list of exogenous variables and estimate the corresponding model M1.

[3] In practice, parameter η can also be differentiated with respect to the specification tests; in that case, it is better to interpret η as a vector; in the implementation, η is indeed supposed to be a vector

ii) Calculate the vector s of p-values of the k specification tests.
iii) if $\exists i / \ s_i \leq s_i^*$ then:
stop the process for this variable, do not store the corresponding model and go to the following insignificant variable of the *second step* model, if any.
else:
a) in the estimation of M1, search an insignificant variable at level α, starting from the variable with the lowest Student's t to the one with the greatest Student's t, until the corresponding model verifies $s_i \geq s_i^*$ for all i (that is a model that passes all specification tests).
b) if there is no such variable, then stop the process for this variable, store model M1 in the list of *third step* models and restart all step 3 process with the following insignificant variable of the *third step* model, if any.
c) if there is such a variable and if the corresponding model has already been encountered on a preceding exploration in step 3, then stop the process for this variable and restart all step 3 process with the following insignificant variable of the *second step* model, if any.
d) if there is such a variable and if the corresponding model has not already been encountered on a preceding exploration in step 3, then call M1 this new model and restart the process at step3 ii) a).
(*) iv) for a set of increasing significance levels, take a group of variables whose significance level in the stage 0 model is greater than the significance level and stops whenever the model is not rejected against the stage 0 model or specification tests are rejected; add the corresponding level to the list of *third step* models.

At the end of step 3, the list of *third step* models contains zero, one or a few models.

Fourth step: Encompassing

i) if the list of *third step* models is empty, then the *final* model is the *general* model.
ii) if the list of *third step* models contains only one model, then this model is the *final model*.
iii) if the list of *third step* models contains more than one model, then build the *union* of all these models and tests, at significance level γ, all third step models against this *union* model.
a) if all *third step* models are rejected against the *union* model, then the *union* model is the *final* model.
b) if only one *third step* models is not rejected against the *union* model, then this *third step* model is the *final* model.

Fifth step: redoing the process with the union model

Fifth step occurs only when there are more than one *third step* model not rejected against the *union* model.

In that case call the *union* model the new *general* model and restart the process with this new *general* model at the beginning of third step.

If, at the end of the process, there are still more than one *third step* model not rejected against the new *union* model, then select the final model on the basis of an information criterion.

*
* *

In practice, the implementation of the algorithm requires then:

- the choice of the set Z of potentially relevant variables;
- the choice of the specification tests; Hendry and Krozlig recommend the following ones: the Chow predictive failure tests with a break at 50% of the sample and 90 % of the sample; the Doornik and Hansen normality test; the LM autocorrelation test with 4 lags and, lastly, the heteroscedasticity test base upon the squared of all variables; these tests have

- the choices of the 6 significance levels α, η, φ, γ, φ_u and φ_b and of the scale Ψ; a convenient choice is α=0.05, η=0.01 (type I error of rejecting wrongly the true model on the basis of the specification tests is then equal to $1-0.99^5 \approx 0.05$), φ=0.05, γ=0.5 (cf. Hendry and Krozlig (2001)[4]);
- the choice of an information criterion: Hendry and Krozlig (2001, op.cit.) advocates the use of Schwarz BIC criterion.

3. The use of automatic.

As in many other programs in GROCER, there are in automatic a lot of default options. The simplest use of automatic looks then very much like the one of ols: automatic('endo','exo1',…,'exon'). The user can however choose her own options.

First, the user may want to choose her own specification tests: the 5 default ones are the Chow predictive failure tests with a break at 50% of the sample and 90 % of the sample; the Doornik and Hansen normality test; the LM autocorrelation test with 4 lags and, lastly, the heteroskedasticity test based upon the squared of all exogenous variables. These tests are performed by the mean of functions predfailin0, doornhans, arlm0 and hetero_sq0 (see. chapter 7 for a description). If the user wants to use other tests, then she has to enter the option 'test=xxx' where xxx is the name of the tests the user wants to use:

- doornhans for the Doornick and Hansen normality test;
- jbnorm for Jarque and Bera normality test;
- predfailin(p) for a Chow predictive failure test for a break at round(p*nobs)
- chowtest(p) for a Chow usual test for a break at round(p*nobs)
- arlm(p) for a LM test of p-th order residual autocorrelation
- arch(p) for a LM test of p-th order conditional heteroskedasticity test
- hetero_sq for the hederoscedasticity test quadratic in regressors.

If the user wants to use other tests, she can. There must exist a function performing this test. This function must have 2 or 3 arguments: the first argument must be a result tlist stemming from a regression program such as ols; the last argument must be 'noprint' (this argument can be not used in the function, if the function does never print anything); the function can have 3 arguments, if, for instance, the test involves some lag of residuals (for instance, the available option arlm needs the # of lags of residuals); this argument must be the second. The user must also enter in automatic the option 'newname(vec1,vec2)' where vec1 is vector of the names of the all new testing function and vec2 is the vector of corresponding names the user wants the program to display. Assume for instance that the name of this function is myfunc and that the user wants the program to display the name 'my beautiful function'. Then he has to enter: 'newnames(myfunc,my beautiful function) '. Note also that if the user wants to add permanently her test to the list of default tests, then she just has to modify the tlist at the beginning of function automatic.

The level of these test (η in part 2 notations), which is 0.01 by default, can also be adjusted by giving the following option: 'eta=xxx' where xxx is a (kx1) vector if k is the number of specification tests chosen by the user (default is 5).

The values of alpha, gamma, f0_tdo and f0_bup can be loaded automatically, through 2 strategies implemented as in pc-gets: the liberal and conservative strategy. The conservative strategy is more stringent than the liberal one (the default strategy): it will generally lead to a more parsimonious model. They correspond to the following values:

[4] D.F. Hendry and H-M Krozlig (2000): « New Developments in Automatic General-to-specific Modelling », in *Econometrics and the Philosophy of Economy*, edited by Berndt P. Stigum.

Liberal strategy

	T ≤ 60	60 < T ≤ 120	120 < T ≤ 200	200 < T ≤ 1000	1000 < T
alpha	0.125	0.075	0.075	0.05	0.025
gamma	0.1	0.05	0.05	0.025	0.01
f0-tdo	0.75	0.5	0.375	0.25	0.1
f0-bup	0.6	0.4	0.3	0.2	0.075

Conservative strategy

	T ≤ 60	60 < T ≤ 120	120 < T ≤ 200	200 < T ≤ 1000	1000 < T
alpha	0.05	0.02	0.015	0.01	0.002
gamma	0.025	0.01	0.01	0.005	0.001
f0-tdo	0.5	0.25	0.1	0.05	0.025
f0-bup	0.25	0.125	0.05	0.025	0.0125

But the level of the significance test for an individual coefficient alpha can also be set to value x by giving the option 'alpha=x', the level of the encompassing test of competing models gamma can be changed from its default value to value x by giving the option 'gam=x', the significance level of the bottom-up by entering the option 'f0_bup=v' where v is value defined by the user and likewise 'f0_tdo=v' for the top-down models

The user can also choose the level of the joint significance pre-test of step 2 can be changed from its default value, 0.9, to value x by giving the option 'st0_sig=x', the significance levels of the groups tested at step iv) of step 3 by entering 'groups_pval=[m1;…;mn]' where the mis are value chosen by the user (groups_pval=[] (°) by default (°°)[5]) and the Ψ used to adjust the level of the specification tests which are rejected for the initial model can be changed from 3 to x by giving the option 'descent=x'.

The user can change the default information criterion used to select models at step 5 of the process, BIC to x through the option 'crit=x' where x=aic (for Akaïke) or hq (for Hannan-Quinn).

There is an option in automatic that allows to impose the presence of variables in all models: the option 'comp=var1;var2;…;varn' where var1, var2,.., varn are the names of existing variables. Note that these variables must be entered only once and must not belong to the list of exogenous variables entered in the function.

The function automatic produces a lot of regressions which are stored in the results tlist of the package: initial model, final model, second step and third step models, step 3 union model,… So the user can define what she wants to print by given the option: 'prt=opt1,…,optn' where opti is either 'nothing', 'all', or the name of a result, such as 'initial', 'final', 'st1_mod',… Note that only final regression is displayed with the corresponding results of the chosen specification test and that when the final model is printed (a case which should be the rule!), the reason why the program has ended is given. (°)The reliability of each variable is also displayed (see chapter 7 for an explanation; in a few words, the reliability of a variable measures the number of times a variable is significant over the whole sample and two sub-samples; the greater they are,

[5](°)Note that this default value has been changed from the 1.1 version to the 1.2 one, because the introduction of models whose variables have significance level in the stage 0 greater than pre-specified significance level often led to choose a model with non significant variables; although this can be seen as the sign of multicollinearity and as such worthwhile to let it know to the user, I found it not very satisfactory; so it is left optional.(°°)

the greater the variable is reliable)(°°). All specification tests can however be displayed on screen by entering in the list of printing options the option 'test_inter' (see the description of automatic in part 4).

Here is an example that makes use of many options:

```
-->bounds('1964q1','1989q2'); load('SCI/macros/grocer/db/bdhenderic.dat') ;
--> [r3]=automatic('lm1', 'lagts(lm1)','lagts(2,lm1)','lagts(3,lm1)','lagts(4,lm1)',…
-->'lp', 'lagts(1,lp)','lagts(2,lp)','lagts(3,lp)','lagts(4,lp)','ly','lagts(1,ly)',…
-->'lagts(2,ly)','lagts(3,ly)','lagts(4,ly)','rnet','lagts(1,rnet)','lagts(2,rnet)',…
-->'lagts(3,rnet)','lagts(4,rnet)','cte','prt=initial,final,st2_mod,test_final', …
-->'test=predfailin(0.51),predfailin(0.862),doornhans,myautocor(5),hetero_sq', …
-->'alpha=0.01', 'st0_sig=0.8','gam=0.01','eta=0.02*ones(5,1)', 'descent=2',…
-->'newnames(myautocor,autocorrelation)','groups_pval=[]','crit=aic')
```

Assuming that myautocor produces the same results as arlm0[6], the results are the following:

```
_____________________________________________________
| results of the automatic regression package       |
 ____________________________________________________

initial model
```

// the user has entered the option prt=initial; the initial model is therefore printed

```
initial model


ols estimation results for dependent variable: lm1
estimation period: 1964q1-1989q2
number of observations: 102
number of variables: 20
R² = 0.9997911      adjusted R² =0.9997428
Overall F test: F(19,82) = 20660.156       p-value = 0
standard error of the regression: 0.0128214
sum of squared residuals: 0.0134798
DW(0) =2.1420694
Belsley, Kuh, Welsch Condition index: 40996

variable       coeff       t-statistic p value
lagts(lm1)     0.6265709   5.8462914   9.860D-08
lagts(2,lm1)   0.1744436   1.4059036   0.1635307
lagts(3,lm1)   -0.2084854 -1.6437741   0.1040519
lagts(4,lm1)   0.2815709   2.7651607   0.0070263
lp             0.1466538   0.6854111   0.4950177
lagts(1,lp)    0.3099395   0.8998303   0.3708456
lagts(2,lp)    -0.0557078 -0.1613026   0.8722518
lagts(3,lp)    -0.4272171 -1.2778863   0.2048953
lagts(4,lp)    0.1470595   0.7630921   0.4475983
ly             -0.0140862 -0.1297232   0.8971031
lagts(1,ly)    0.2946174   2.2751329   0.0255080
lagts(2,ly)    -0.1351352 -1.0352956   0.3035744
lagts(3,ly)    -0.1585411 -1.2075487   0.2306924
lagts(4,ly)    0.1693574   1.5594676   0.1227371
rnet           -0.4164073 -3.6848631   0.0004093
lagts(1,rnet)  -0.3253502 -1.9202116   0.0583079
lagts(2,rnet)  -0.0726672 -0.4207056   0.6750706
lagts(3,rnet)  -0.0346699 -0.2029840   0.8396504
lagts(4,rnet)  -0.0282974 -0.2363236   0.8137709
```

[6] For instance, by running on screen the instruction:
--> myautocor=arlm0

```
cte           -0.3145329 -0.7918287  0.4307453


                         *
                     *     *
```

// the user has given the option prt=st2_mod; so all (2 here) stage 2 models are printed

```
stage 2 models

model #1


ols estimation results for dependent variable: lm1
estimation period: 1964q1-1989q2
number of observations: 102
number of variables: 8
standard error of the regression: 0.0127701
sum of squared residuals: 0.0153292
DW(0) =1.9813115
Belsley, Kuh, Welsch Condition index: 5583

variable      coeff      t-statistic p value
lagts(lm1)    0.5997435  6.1361945   1.991D-08
lagts(2,lm1)  0.1870658  1.8592358   0.0661219
lagts(4,lm1)  0.1061773  1.8071436   0.0739389
lagts(1,lp)   0.4369634  5.8150952   8.294D-08
lagts(3,lp)   -0.3274466 -4.533529   0.0000171
lagts(1,ly)   0.1104072  6.9017627   5.949D-10
rnet          -0.3763906 -3.8429891  0.0002208
lagts(1,rnet) -0.4201485 -2.9465079  0.0040520


                         *
                     *     *


aic -8.6461073
bic -8.4402271
hq  -8.5627394



                         *
                     *     *

final model

ending reason: all stage 2 models rejected against their union


ols estimation results for dependent variable: lm1
estimation period: 1964q1-1989q2
number of observations: 102
number of variables: 8
standard error of the regression: 0.0127701
sum of squared residuals: 0.0153292
DW(0) =1.9813115
Belsley, Kuh, Welsch Condition index: 5583

variable      coeff      t-statistic p value
```

```
lagts(lm1)     0.5997435  6.1361945   1.991D-08
lagts(2,lm1)   0.1870658  1.8592358   0.0661219
lagts(4,lm1)   0.1061773  1.8071436   0.0739389
lagts(1,lp)    0.4369634  5.8150952   8.294D-08
lagts(3,lp)    -0.3274466 -4.533529   0.0000171
lagts(1,ly)    0.1104072  6.9017627   5.949D-10
rnet           -0.3763906 -3.8429891  0.0002208
lagts(1,rnet) -0.4201485 -2.9465079  0.0040520


                        *
                    *     *

tests results:
**************
test                    test value p-value
Chow pred. fail. (50%) 0.4574206  0.9961104
Chow pred. fail. (90%) 0.4065963  0.9399961
Doornik & Hansen       7.7701427  0.0205464
AR(1-4)                0.9617740  0.4325247
hetero x_squared       1.584211   0.1986298


                        *
                    *     *



variable       reliability

lagts(lm1)     1
lagts(2,lm1)   0
lagts(4,lm1)   0.3
lagts(1,lp)    0.4
lagts(3,lp)    0.4
lagts(1,ly)    1
rnet           1
lagts(1,rnet) 0.7

                        *
                    *     *
```

(*) Remarks:

a) to execute automatic, you need a minimal amount of degrees of freedom, that is a maximal number of exogenous variables. If you try to use the program with more than this maximum number of variables, then you will have the message:

```
!--error  9999
too many exogenous variables
at line     232 of function automatic                called by :
```

The maximal number of degrees of freedom is determined: first, by the heteroskedasticity test and then by the Chow tests. So if you have too many exogenous variables, then you can withdraw the hetero_sq test from your list of specifications, with the option 'test=predfailin(0.5),predfailin(0.9),doornhans,myautocor(5)'; if this not sufficient, then withdraw the Chow tests: 'test= doornhans,myautocor(5)' (but in that case, you have

really many exogenous variables with respect to the time span and it would be much better to restrict the number of exogenous variables…).

b) you may be surprised to find that automatic selects a model with non significant variables. There are however three possible reasons why it may happen:

- there are less than 60 observations and you have chosen the liberal strategy (or let the program choose this strategy by default); in that case the significance level of individual Staudent's t is set to 0.1; you can change it, by entering the option 'alpha=0.05';
- there are more than 60 observations, but the program is obliged to choose a model with some non significant variables because withdrawing any of these variables leads to a rejection of one of the specification tests; you can impose the program to ignore the problem by entering the option 'eta=%eps'; then see what specification test is rejected by the chosen model and decide if it can be ignored or not;
- there are more than 60 observations, but the final model has been selected among the groups of pre-specified significance level; in that case, some variables can be individually non significant (but near so) and collectively significant; you can impose the program not to estimate these groups by entering the option 'groups_pval=[]'. (**)

4. The Automatic functions and their specifications.

auto_stage0_____elimination of non significant variables

CALLING SEQUENCE

[done,results,indxf_0]=auto_stage0(y,x,indcte,r00,rmod,alpha,f0_sig,eta,z,f_bup,results)

PARAMETERS

INPUT:
* y = vector of the endogenous variable
* x = matrix of the exogenous variables
* indcte = index of the constant variable (= nvar+1 if there is no constant in the r00 model)
* r00 = results of the estimation (by ols or ols2) of the model Y = X*b+U
* rmod = a tlist defined by def_results
* alpha = broad significance level (typically lower than f0_sig)
* f0_sig = maximum joint significance level allowing elimination of the corresponding variables
* eta = (px1) significance level for specification tests
 z = matrix of the compulsory variables (the ones that must be in the regression whatever significance they have)
* f_bup = the significance level for the bottom-up model
* res = the automatic results tlist (contains first results)

OUTPUT:
* done = %t if all variables are significant, %f otherwise
* r0 = results structure of the final model (the one without the eliminated variables) as provided by ols2
* indxf_0 = indexes in x of the remaining exogenous variables

DESCRIPTION

In a model Y=X*b+U, eliminates variables whose joint significance is lower than f0_sig under the condition that no specification test fails. The function (used in automatic) assumes that an empty result tlist exists and that the model Y==X*b+U with tlist results called r00 in the function. The user must give the index, if any, of the constant, the global significance level, the significance level under which elimination is done and the significance level for the specifications tests performed by function test_func.

Examples:
1) [done,r0,indxf]=auto_stage0(y,x,5,r00,rmod,0.05,0.9,0.01*ones(5,1))
2) [done,r0,indxf]=auto_stage0(grocer_y,grocer_x,indcte,r1_00,lrmod(1),grocer_alpha, grocer_f0_sig,grocer_eta)

In example 1, function auto_stage0 is called with a 0.05 general significance level, a 0.9 elimination level and a 0.01 significance level for 5 specification tests. Example 2 is the call to function auto_satge0 in automatic (the most natural use).

auto_stage1_____search of an statistically admissible regression

CALLING SEQUENCE

[listeq]=auto_stage1(y,x,listeq,indx,indcte,firstelim,r00,rmod,alpha,eta,z)

PARAMETERS

INPUT:
* y = vector of the endogenous variable
* x = matrix of the exogenous variables
* listeq = the list containing all preceding paths and the corresponding estimation results
* index = the indexes of the x variables remaining at the entry of the function
* indcte = the booelan indicating the presence or the absence of the constant in the regression
* firstelim = the index of the first variable which is withdrawn to begin the path
* r00 = the results tlist for the regression containing all exogenous variables whose index is in indx
* rmod = a tlist defined by def_results
* alpha = broad significance level (typically lower than f0_sig)
* eta = (px1) significance level for specification tests
z = matrix of the compulsory variables (the ones that must be in the regression whatever significance they have)

OUTPUT:
* listeq = the entry list plus the path followed here and, if it leads to a new model, the corresponding estimation results

NOTES:
* used by automatic()

DESCRIPTION

Starting from the variable whose index in indx is firstelim, find a path where: the less significant variable which does not make the specification tests calculated in test_func fails is successively withdrawn until:
1) either the withdrawn variables as a group has been already encountered on another path
2) or all remaining variables are non significant or the regression does not pass the specification tests calculated in test_func.

Example:
This function is a procedure of automatic. Very specific indeed.

auto_test_____build specification tests function and vector of names

CALLING SEQUENCE

[test_func,name_test]=auto_test(auto_ names,auto_ltest,nobs)

PARAMETERS

INPUT:
* auto_names = the tlist that associates a name to each specification test function
* auto_ltest = the list of options provided by the user (default: an empty list)
* nobs = # of observations

OUTPUT:
* test_func = the function which gathers the specification tests used by automatic()
* name_test = the name of the specification tests which will be used for the printings

DESCRIPTION

From the specifications list given by the user, build the function test_func which performs these specification tests and the vector name_test of corresponding names.

Examples:
1) [func,names] = auto_test('test=arlm(6),jbnorm,chowtest(0.8),myautocor(5)')
2) [test_func,m2prt_test]=auto_test(grocer_ltest)

Example 1 creates a function that performs the Bresuch-Pagan test with 1 to6 lags, the Jarque and Bera normality test, the Chow stability test with a break at 80% of the sample and the matrix of corresponding names.
Example 2 is an application of auto_test in function automatic.

automatic_____automatic general to specific regression

CALLING SEQUENCE

[results]=automatic(grocer_namey,varargin)

PARAMETERS

INPUT:
* grocer_namey = a time series, a real (nx1) vector or a string equal to the name of a time series or a (nx1) real vector between quotes
* varargin = arguments which can be:
  . a time series
  . a real (nxp) vector
  . a string equal to the name of a time series or a (nx1) real vector between quotes
  . the string 'prt=opt1,opt2,...,optn'
  where opti is one of the following available options:
    - nothing (nothing printed !)
    - initial (results of initial model printed)
    - st0_mod (results of stage 0 model printed)
    - st1_mod (results of stage 1 models printed)
    - st1_union (results of stage 1 union model printed)
    - st2_mod (results of stage 2 models printed)
    - st2_union (results of stage 2 union model printed)
    - final (results of final model printed)
    - test_inter (results of specification tests for the intermediate results printed)
    - test_final (results of specification tests for the final model printed)
    - test (specification tests printed for every regression result printed)
    - st1_path (results of stage 1 paths printed)
    - st2_path (results of stage 2 paths printed)
    - all (all results printed: can be very long !)
  . the string 'test=opt1,opt2,...,optn'
  where opti is one of the following available options:
    - chowtest(p)
    - predfailin(p)
    - doornhans
    - bpagan(p)
    - hetero_sq
  . the string 'strategy=liberal' or 'strategy=conservative' if the user wants to use a predefined strategy
    (default = liberal)
  . the string 'alpha=p' for the simplification significance level (default = set by the chosen strategy)
  . the string 'f0_tdo=p' for the top_down pre-test significance level (default = set by the chosen strategy)
  . the string 'f0_bup=p' for the bottom-up pre-test significance level (default = set by the chosen strategy)
  . the string 'eta=p' for the specification tests significance level (default = 0.01)
  . the string 'gam=p' for the F-tests significance level (default = set by the chosen strategy)
  . the string 'groups_pval=m1;m2;...;mp' where the mi's are thresholds: coefficient whose p-values are greater than a given threshold are gathered to form a model included in stage 1 models
    (default = set by the chosen strategy)
  . the string 'crit=x' where x= aic, bic, or hq(default bic)
  . the string 'test=x1,...,xp' where xi is the name of a test function

. the string 'newname(x1,x2)' where x1 is the name of a test function and x2 is the corresponding name the user wants the program to display

. the string 'comp=x1;...;xn' where xi is the name of a variable that must be in the regression whatever significance it has

---

OUTPUT:

a results tlist with the following fields:

* results('meth') = 'automatic'
* results('strategy') = estimation strategy (liberal, conservative or expert)
* results('alpha') = simplification significance level (default = set by the chosen strategy)
* results('f0_tdo') = top_down pre-test significance level (default = set by the chosen strategy)
* results('f0_bup') = bottom-up pre-test significance level (default = set by the chosen strategy)
* results('gam') = F-tests significance level (default = set by the chosen strategy)
* results('eta') = specification tests significance level (default = 0.01)
* results('comp') = matrix of values for the compulsory variables
* results('f_test') = the function used to perform the specification tests
* results('m_test') = the names of the specification tests
* results('initial model') = the estimation results of the unrestricted model
* results('ending reason') = the reason why the final model has been chosen
* results('final model') = the estimation results of the final model
* results('top-down model') = the top-down model
* results('bottom-up model') = the bottom-up model
* results('stage 0 model') = the estimation results of the stage 0 restricted model (all variables whose individual tstat level is lower than 1 and whose joint significance level is lower than f0_tdo are withdrawn)
* results('selected stage 0 model') = the way the stage 0 model has been selected
* results('checking process model') = the model resulting from the group checking process
* results('stage i models') = the estimation results of the stage i (i=1, 2.0, 2.1) models and the corresponding paths
* results('stage i union model') = the estimation results of the model built from the union of stage i (i=1, 2.0, 2.1) models

DESCRIPTION

The implementation of Krolzig, H.-M. and Hendry, D.F. (2001): "Computer Automation of General-to-Specific Model Selection Procedures", Journal of Economic Dynamics and Control, 25 (6-7), 831-866. The program selections a model by least-squares regressions: starting from the list of exogenous variables provided by the user, the function select the "best" regression, that is one where all variables are significant, all specification tests are passed, and, if more than one regression has these properties, the one with the smallest aic criterion. The program stores results in a tlist (itself built with regression tlists) and prints the results along the options given by the user.

Examples:

1) [r1]=automatic('del(lm1-lp)','lagts(lm1-lp-ly)','del(lp)','rnet','del(lagts(lm1-lp))', 'del(lagts(2,lm1-lp))','del(lagts(3,lm1-lp))','del(lagts(4,lm1-lp))','del(ly)','del(lagts(1,ly))', 'del(lagts(2,ly))',del(lagts(3,ly))','del(lagts(4,ly))','del(del(lp))','del(del(lagts(1,lp)))','del(del(lagts(2,lp)))','del(del(lagts(3,lp)))','del(del(lagts(4,lp)))','del(rnet)','del(lagts(1,rnet))','del(lagts(2,rnet))','del(lagts(3,rnet))','del(lagts(4,rnet))','cte','prt=initial,final','test=predfailin(0.5),predfailin(0.9),doornhans,arlm(5),hetero_sq')

2) [r2]=automatic('del(lm1-lp)','lagts(lm1-lp-ly)', 'del(lagts(lp))','lagts(rnet)','del(lagts(lm1-lp))','del(lagts(2,lm1-lp))','del(lagts(3,lm1-lp))','del(lagts(4,lm1-lp))','del(ly)','del(lagts(1,ly))', 'del(lagts(2,ly))','del(lagts(3,ly))','del(lagts(4,ly))','del(del(lp))','del(del(lagts(1,lp)))','del(del(lagts(2,lp)))','del(

del(lagts(3,lp))','del(del(lagts(4,lp))','del(rnet)','del(lagts(1,rnet))','del(lagts(2,rnet))','del(lagts(3,rnet))','del(lagts(4,rnet))','cte','prt=final','test=predfailin(0.5),predfailin(0.9),doornhans,arlm(5),hetero_sq')
3) [r3]=automatic('lm1','lagts(lm1)','lagts(2,lm1)','lagts(3,lm1)','lagts(4,lm1)','lp','lagts(1,lp)','lagts(2,lp)','lagts(3,lp)','lagts(4,lp)','ly','lagts(1,ly)','lagts(2,ly)','lagts(3,ly)','lagts(4,ly)','rnet','lagts(1,rnet)','lagts(2,rnet)','lagts(3,rnet)','lagts(4,rnet)','cte','prt=initial,final,st2_mod','test=predfailin(0.51),predfailin(0.862),doornhans,arlm(5),hetero_sq','alpha=0.05')

All these examples start from a rather extensive set of exogenous variables: from 'lagts(lm1-lp-ly)' to 'cte' in examples 1 and 2 and from 'lagts(lm1)' in example 3.

check_groups_____estimation of model whose variables have predefined significance levels

CALLING SEQUENCE

[results,r_checks,nbmodst1,list1_2]=check_groups(results,y,z,groups_pval,gpvalue,indpvalue1_0,gam,eta,lrmod,list1_2,nbmodst1,indcte)

PARAMETERS

INPUT:
* results = an automatic tlist results
* y = vector of the endogenous variable
* z = matrix of the compulsory variables (the ones that must be in the regression whatever significance they have)
* x = matrix of the exogenous variables
* groups_pval = a vector of significance levels ordered from the lowest to the greatest
* gpvalue = the vector of p-values from the previous (stage 0) regression
* indpvalue1_0 = the corresponding vector of indexes in the list of initial variables
* gam = the significance-level of F-tests
* eta = a vector equal to the significance levels for the specification tests
* rmod = a tlist defined by def_results
* list1_2 = the list of models selected before
* nbmodst1 = the corresponding number of models
* indcte = the booelan indicating the presence or the absence of the constant in the regression
---
OUTPUT:
* results = an automatic tlist results
* r_checks = the first estimated model that is accepted against the initial one or the last that passes the specification tests
* nbmodst1 = the number of models selected at the end of stage 1
* list1_2 = the list of models selected at the end of stage 1

DESCRIPTION

Creates a list of 2 results tlist, each tlist having the formal name of their elements defined and the invariable elements defined (y,namey,...) ; the second list has the names 'rsqr', 'rbar', 'f' and 'fvalue' that the first one doesn't have. Useful when you have to perform numerous regressions with the same variables.

Example:
check_groups(results,grocer_y,grocer_z,grocer_groups_pval,…
gpvalue,xpath,grocer_gam,grocer_eta,lrmod(1),list1_2,nbmodst1,indcte1)

This example is taken from automatic.

def_results_____creation of two tlists for subsequent regressions

CALLING SEQUENCE

[lrmod,indcte]=def_results(y,namey,nobs,prests, prescte,indcte,boundsvarb,m2prt_test)

PARAMETERS

INPUT:
* y = y data vector
* namey = name of the y variable
* nobs = # of observations
* prests = boolean indicating the presence or absence of a time series in the regression
* prescte = boolean indicating the presence or absence of a constant in the regression
* indcte = index of the constant in the regression+1 (= nvar+1 if there is none)
* boundsvarb = if there is a timeseries in the regression, the bounds of the regression
* m2prt_test = the column describing the name of the specification tests use in the process
---
OUTPUT:
* lmod = a list of 2 tlists
* indcte = the index of the constant variable in the x matrix (nvar+1 if there is no constant variable)

DESCRIPTION

Creates a list of 2 results tlist, each tlist having the formal name of their elements defined and the invariable elements defined (y,namey,...) ; the second list has the names 'rsqr', 'rbar', 'f' and 'fvalue' that the first one doesn't have. Useful when you have to perform numerous regressions with the same variables.

Example:
1) [lrmod,indcte]=def_results(grocer_y,grocer_namey,nobs,grocer_prests,prescte,indcte, grocer_boundsvarb,m2prt_test)

This example is taken from automatic.

ols1a_____ordinary least squares

CALLING SEQUENCE

[results]=ols1a(y,x,results)

PARAMETERS

INPUT:
* y = dependent variable vector (nobs x 1)
* x = independent variables matrix (nobs x nvar)
* results: an existing tlist of regression results

OUTPUT:
a tlist with:

```
results('meth')  = 'ols'
results('beta')  = bhat
results('tstat') = t-stats
results('yhat')  = yhat
results('resid') = residuals
results('sige')  = e'*e/(n-k)
results('rsqr')  = rsquared
results('rbar')  = rbar-squared
results('dw')    = Durbin-Watson Statistic
results('nobs')  = nobs
results('nvar')  = nvars
results('y')     = y data vector
results('x')     = x data matrix
```

DESCRIPTION

One of the numerous functions performing ordinary least squares: this one assumes that x et y are already a matrix and a vector respectively, that a result tlist exists which can be filled and does not provide DW, R^2 statistics.

Example:
1) r=ols1a(y,xd,rmod)

This example is taken from auto_stage0.

ols2a_____ordinary least squares

CALLING SEQUENCE

[results]=ols2a(y,x ,prescte,results)

PARAMETERS

INPUT:
* y = dependent variable vector (nobs x 1)
* x = independent variables matrix (nobs x nvar)
* prescte = a boolean indicating whether the model contains a cte term or not
* results = a tlist containing all necessary fields, with the fields 'meth','nobs','y','namey','prests', 'ym' are already filled
---

RETURNS: a tlist
results = a tlist with
. results('meth') = 'ols'
. results('y') = y data vector
. results('x') = x data matrix
. results('nobs') = nobs
. results('nvar') = nvars
. results('beta') = bhat
. results('yhat') = yhat
. results('resid') = residuals
. results('vcovar') = estimated variance-covariance matrix of beta
. results('sige') = estimated variance of the residuals
. results('sige') = estimated variance of the residuals
. results('ser') = standard error of the regression
. results('tstat') = t-stats
. results('pvalue') = pvalue of the betas
. results('dw') = Durbin-Watson Statistic
. results('condindex') = multicolinearity cond index
. results('prescte') = boolean indicating the presence or absence of a constant in the regression
. results('rsqr') = rsquared
. results('rbar') = rbar-squared
. results('f') = F-stat for the nullity of coefficients other than the constant
. results('pvaluef') = its significance level

DESCRIPTION

One of the numerous functions performing ordinary least squares: this one assumes that x et y are already a matrix and a vector respectively, that the existence or absence of a constant has already been determined (and stored in a boolean) and that a result tlist exists which can be filled.

Example:
1) r1_00=ols2a(grocer_y,grocer_x,prescte,lrmod(2))

This example is taken from automatic.

ols3a_____ordinary least squares

CALLING SEQUENCE

[results]=ols3a(y,x,namexos,numx,prescte,results)

PARAMETERS

INPUT:
* y = dependent variable vector (nobs x 1)
* x = independent variables matrix (nobs x nvar)
* namexos = name of the exogenous variables
* numx = index of the chosen exogenous variables
(nobs x nvar')

* prests = boolean equal to %T if there is a constant
among the variables
* results = a tlist already filled with

OUTPUT:
results = a tlist with
. results('meth') = 'ols'
. results('y') = y data vector
. results('x') = x data matrix
. results('nobs') = nobs
. results('nvar') = nvars
. results('beta') = bhat
. results('tstat') = t-stats
. results('pvalue') = pvalue of the betas
. results('resid') = residuals
. results('vcovar') = estimated variance-covariance matrix of beta
. results('sige') = estimated variance of the residuals
. results('sigu') = sum of squared residuals
. results('ser') = standard error of the regression
. results('yhat') = yhat
. results('dw') = Durbin-Watson Statistic
. results('condindex') = multicolinearity cond index
. results('prescte') = boolean indicating the presence or absence of a constant in the regression
. results('prests') = boolean indicating the presence or absence of a ts in the regression
. results('namey') = name of the endogenous variable
. results('namex') = names of the exogenous variables
. results('ym') = mean of vector y
. results('spec_test') = vector of specification tests p-values
. results('m_test') = vector of names of specification tests
. results('ym') = mean of vector y
. results('rsqr') = rsquared
. results('rbar') = rbar-squared
. results('f') = F-stat for the nullity of coefficients other than the constant
. results('pvaluef') = its significance level

test_spec0_____five specification tests

CALLING SEQUENCE

[v,p]=test_spec0(r)

PARAMETERS

INPUT:
* r = a results tlist from a regression

OUTPUT:
* v = (5x1) vector of statistics of specification tests
* p = (5x1) vector of p-values of specification tests

DESCRIPTION

Provides the vectors of statistics values for 5 specification tests (2 Chow tests, the Doornik and Hansen normality test, the Breusch-Pagan autocorrelation test, the heteroscedasticity test) and their p-values

Examples:
1) load('grocer/bdexamples/bdhenderic.dat') ; bounds('1964q3','1989q2') ; he=ols('del(lm1-lp)','del(lp)','del(lagts(1,lm1-lp-ly))','rnet','lagts(1,lm1-lp-ly)','cte'); [v,p]=test_spec0(he)
2) [val,p]=test_func(r1_00)

Example 1 calculates the 5 specification tests and their p-values for Hendry and Ericsson's preferred model. Example 3 is taken from automatic.