Gretl Examples Econometrics 4/4/2013

I have selected some material from Adkins' manual on GRETL to illustrate some of the concepts applied in this week's lecture.

Chapter 8

Heteroskedasticity

The simple linear regression models of chapter 2 and the multiple regression model in Chapter 5 can be generalized in other ways. For instance, there is no guarantee that the random variables of these models (either the y_i or the e_i) have the same inherent variability. That is to say, some observations may have a larger or smaller variance than others. This describes the condition known as **heteroskedasticity**. The general linear regression model is shown in equation (8.1) below.

$$y_i = \beta_1 + \beta_2 x_{i2} + \cdots + \beta_k x_{iK} + e_i \quad i = 1, 2, \ldots, N \tag{8.1}$$

where y_i is the dependent variable, x_{ik} is the i^{th} observation on the k^{th} independent variable, $k = 2, 3, \ldots, K$, e_i is random error, and $\beta_1, \beta_2, \ldots, \beta_K$ are the parameters you want to estimate. Just as in the simple linear regression model, e_i, have an average value of zero for each value of the independent variables and are uncorrelated with one another. The difference in this model is that the variance of e_i now depends on i, i.e., the observation to which it belongs. Indexing the variance with the i subscript is just a way of indicating that observations may have different amounts of variability associated with them. The error assumptions can be summarized as $e_i | x_{i2}, x_{i3}, \ldots x_{iK}$ iid $N(0, \sigma_i^2)$.

The intercept and slopes, β_1, β_2, ..., β_K, are consistently estimated by least squares even if the data are heteroskedastic. Unfortunately, the usual estimators of the least squares standard errors and tests based on them are inconsistent and invalid. In this chapter, several ways to detect heteroskedasticity are considered. Also, statistically valid ways of estimating the parameters of 8.1 and testing hypotheses about the βs when the data are heteroskedastic are explored.

8.1 Food Expenditure Example

First, a simple model of food expenditures is estimated using least squares. The model is

$$food\_exp_i = \beta_1 + \beta_2 income_i + e_i \quad i = 1, 2, \ldots, N \tag{8.2}$$

where $food\_exp_i$ is food expenditure and $income_i$ is income of the i^{th} individual. When the errors of the model are heteroskedastic, then the least squares estimator of the coefficients is consistent. That means that the least squares *point estimates* of the intercept and slope are useful. However, when the errors are heteroskedastic the usual least squares **standard errors** are **inconsistent** and therefore should not be used to form confidence intervals or to test hypotheses.

To use least squares estimates with heteroskedastic data, at a very minimum, you'll need a consistent estimator of their standard errors in order to construct valid tests and intervals. A simple computation proposed by White accomplishes this. Standard errors computed using White's technique are loosely referred to as **robust**, though one has to be careful when using this term; the standard errors are robust to the *presence of heteroskedasticity* in the errors of model (but not necessarily other forms of model misspecification).

Open the *food.gdt* data in **gretl** and estimate the model using least squares.

```
open "@gretldir\data\poe\food.gdt"
ols food_exp const income
gnuplot food_exp income --linear-fit
```

This yields the usual least squares estimates of the parameters, but produces the wrong standard errors when the data are heteroskedastic. To get an initial idea of whether this might be the case a plot of the data is generated and the least squares line is graphed. If the data are heteroskedastic with respect to income then you will see more variation around the regression line for some levels of income. The graph is shown in Figure 8.1 and this appears to be the case. There is significantly more variation in the data for high incomes than for low.

To obtain the heteroskedasticity robust standard errors, simply add the `--robust` option to the regression as shown in the following **gretl** script. After issuing the `--robust` option, the standard errors stored in the accessor `$stderr(income)` are the robust ones.

```
ols food_exp const income --robust
# confidence intervals (Robust)
scalar lb = $coeff(income) - critical(t,$df,0.025) * $stderr(income)
scalar ub = $coeff(income) + critical(t,$df,0.025) * $stderr(income)
printf "\nThe 95%% confidence interval is (%.3f, %.3f).\n",lb,ub
```

In the script, we have used the `critical(t,$df,0.025)` function to get the desired critical value from the t-distribution. Remember, the degrees of freedom from the preceding regression are stored in `$df`. The first argument in the function indicates the desired distribution, and the last is the desired right-tail probability ($\alpha/2$ in this case).

The script produces

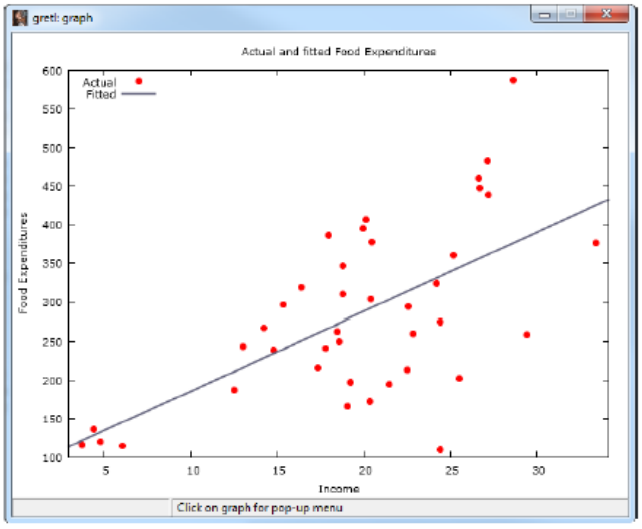


Figure 8.1: Plot of food expenditures against income with least squares fit.

```
The 95% confidence interval is (6.391, 14.028).
```

This can also be done from the pull-down menus. Select `Model>Ordinary Least Squares` (see Figure 2.6) to generate the dialog to specify the model shown in Figure 8.2 below. Note, the check box to generate 'robust standard errors' is circled. You will also notice that there is a button labeled **Configure** just to the right of the 'Robust standard errors' check box. Clicking this button reveals a dialog from which several options can be selected. In this case, we can select the particular method that will be used to compute the robust standard errors and even set robust standard errors to be the default computation for least squares. This dialog box is shown in Figure 8.3 below.

To reproduce the results in Hill et al. (2011), you'll want to select HC1 from the pull-down list. As you can see, other **gretl** options can be selected here that affect the default behavior of the program. The particular variant it uses depends on which dataset structure you have defined for your data. If none is defined, **gretl** assumes you have cross-sectional data.

The model results for the food expenditure example appear in the table below. After estimating the model using the dialog, you can use `Analysis>Confidence intervals for coefficients` to generate 95% confidence intervals. Since you used the `robust` option in the dialog, these will be based on the variant of White's standard errors chosen using the 'configure' button. In this case, I chose HC3, which some suggest performs slightly better in small samples. The result is:

VARIABLE	COEFFICIENT	95% CONFIDENCE INTERVAL	
const	83.4160	25.4153	141.417
income	10.2096	6.39125	14.0280

OLS, using observations 1–40
Dependent variable: food_exp
Heteroskedasticity-robust standard errors, variant HC3

	Coefficient	Std. Error	t-ratio	p-value
const	83.4160	28.6509	2.9115	0.0060
income	10.2096	1.88619	5.4128	0.0000

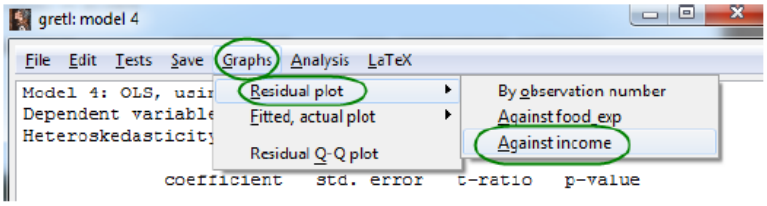
Mean dependent var	283.5735	S.D. dependent var	112.6752
Sum squared resid	304505.2	S.E. of regression	89.51700
R^2	0.385002	Adjusted R^2	0.368818
$F(1, 38)$	29.29889	P-value(F)	3.63e–06

Table 8.1: Least squares estimates with the usual and robust standard errors.

8.2 Detecting Heteroskedasticity

In the discussion above we used a graph of the data and the regression function to give us an initial reading of whether the data are heteroskedastic. Residual plots are equally useful, but some care must be taken in generating and interpreting them. By their very nature, plots allow you to 'see' relationships one variable at a time. If the heteroskedasticity involves more than one variable they may not be very revealing.

In Figure 8.4 is a plot of the least squares residuals against income. It appears that for larger levels of income there is much higher variance in the residuals. The graph was generated from the model window by selecting `Graphs>Residual plot>Against income`. I also right-clicked on the graph, chose **Edit** and altered its appearance a bit. Summoning the dialog looks like



Of course, you can also generate graphs from a script, which in this case is:

```
ols food_exp const income --robust
series res = $uhat
setinfo res -d "Least Squares Residuals" -n "Residual"
gnuplot res income --output=c:\Temp\olsres
```

8.3 Lagrange Multiplier Tests

There are many tests of the null hypothesis of homoskedasticity that have been proposed elsewhere. Two of these, based on Lagrange multipliers, are particularly simple to do and useful. The first is sometimes referred to as the Breusch-Pagan (BP) test. The second test is credited to White.

The null and alternative hypotheses for the Breusch-Pagan test are

$$H_0 : \sigma_i^2 = \sigma^2$$
$$H_1 : \sigma_i^2 = h(\alpha_1 + \alpha_2 z_{i2} + \ldots \alpha_s z_{iS})$$

The null hypothesis is that the data are homoskedastic. The alternative is that the data are heteroskedastic in a way that depends upon the variables z_{is}, $i = 2, 3, \ldots, S$. These variables are exogenous and correlated with the model's variances. The function $h()$, is not specified. It could be anything that depends on its argument, i.e., the linear function of the variables in z. Here are the steps:

1. Estimate the regression model
2. Save the residuals
3. Square the residuals
4. Regress the squared residuals on z_{is}, $i = 2, 3, \ldots, S$.
5. Compute NR^2 from this regression and compare it to the α level critical value from the $\chi^2(S-1)$ distribution.

The **gretl** script to perform the test manually is

```
ols food_exp const income
series sq_ehat = $uhat*$uhat
ols sq_ehat const income
scalar NR2 = $trsq
pvalue X 1 NR2
```

The only new item in this script is the use of the accessor, `$trsq`. This is the saved value of NR^2 from the previously estimated model. The output from the script is

```
Replaced scalar NR2 = 7.38442
Chi-square(1): area to the right of 7.38442 = 0.00657911
(to the left: 0.993421)
```

The p-value is less than 5% and we would reject the homoskedasticity null at that level. The heteroskedasticity seen in the residual plots appears to be confirmed.

```
ols food_exp const income
modtest --breusch-pagan
```

Produces

```
Breusch-Pagan test for heteroskedasticity
OLS, using observations 1-40
Dependent variable: scaled uhat^2

            coefficient   std. error   t-ratio   p-value
  ---------------------------------------------------------
  const      -0.756949     0.633618    -1.195    0.2396
  income      0.0896185    0.0305534    2.933    0.0057   ***

  Explained sum of squares = 14.6879

Test statistic: LM = 7.343935,
with p-value = P(Chi-square(1) > 7.343935) = 0.006729
```

The functionality of `modtest --breusch-pagan` is limited in that it will include every regressor in the model as a `z`. It matches the result we derived manually because the model only includes `income` as the regressor. The `modtest --breusch-pagan` uses it as `z`. This means that you can't test a subset of the regressors with this function, nor can you use it to test for heteroskedasticity of exogenous variables that are not included in the regression function. In either of these cases, use the manual method described above; it is very easy to do.

8.3.1 The White Test

White's test is in fact just a minor variation on the Breusch-Pagan test. The null and alternative hypotheses are

$$H_0 : \sigma_i^2 = \sigma^2 \quad \text{for all } i$$
$$H_1 : \sigma_i^2 \neq \sigma_j^2 \quad \text{for at least 1 } i \neq j$$

This is a composite alternative that captures every possibility other than the one covered by the null. If you know nothing about the nature of heteroskedasticity in your data, then this is a good place to start. The test is very similar to the BP test. In this test, the heteroskedasticity related variables ($z_{is}, i = 2, 3, \ldots, S$) include each non-redundant regressor, its square, and all cross products between regressors. See *POE4* for details. In the food expenditure model there is only one continuous regressor and an intercept. So, the constant squared and the cross product between the constant and income are redundant. This leaves only one unique variable to add to the model, income squared.

In **gretl** generate the squared value of income and regress the squared residuals from the model on income and its square. Compute NR^2 from this regression and compare it to α level critical value from the $\chi^2(S-1)$ distribution. As is the case in all the LM tests considered in this book, N is the number of observations in the second or auxiliary regression.

As with the BP test there is a built-in function that computes White's test. It generates all of the squares and unique cross-products to add to the model. The script to do both manual and built-in tests is found below:

```
ols food_exp const income
series sq_ehat = $uhat*$uhat
series sq_income = income^2
ols sq_ehat const income sq_income
scalar NR2 = $trsq
pvalue X 2 NR2

ols food_exp const income --quiet
modtest --white --quiet
```

The results from the two match perfectly and only that from the built-in procedure is produced below:

```
White's test for heteroskedasticity

Test statistic: TR^2 = 7.555079,
with p-value = P(Chi-square(2) > 7.555079) = 0.022879
```

The homoskedasticity null hypothesis is rejected at the 5% level.

8.4 Heteroskedastic-Consistent Standard Errors

The least squares estimator can be used to estimate the linear model even when the errors are heteroskedastic with good results. As mentioned in the first part of this chapter, the problem with using least squares in a heteroskedastic model is that the usual estimator of precision (estimated variance-covariance matrix) is not consistent. The simplest way to tackle this problem is to use least squares to estimate the intercept and slopes and use an estimator of least squares covariance that is consistent whether errors are heteroskedastic or not. This is the so-called heteroskcedasticity robust estimator of covariance that **gretl** uses.

In this example, the food expenditure data is used to estimate the model using least squares with both the usual and the robust sets of standard errors. Start by estimating the food expenditure model using least squares and add the estimates to the model table the estimates (Usual). Reestimate the model using the `--robust` option and store the results (`modeltab add`).

```
ols food_exp const income --quiet
modeltab add
ols food_exp const income --robust --quiet
modeltab add
modeltab show
```

The model table, which I edited a bit, is

OLS estimates
Dependent variable: food_exp

	(Usual)	(HC3 Robust)
const	72.96*	72.96**
	(38.83)	(19.91)
income	11.50**	11.50**
	(2.508)	(2.078)
n	20	20

[1]Replace `sortby income` with `dsortby income` to sort the sample by income in descending order.

R^2	0.5389	0.5389
ℓ	−109.1	−109.1

Standard errors in parentheses
* indicates significance at the 10 percent level
** indicates significance at the 5 percent level

Notice that the coefficient estimates are the same, but that the estimated standard errors are different. Interestingly enough, the robust standard error for the slope is actually smaller than the usual one!

A number of commands behave differently when used after a model that employs the `--robust` option. For instance, the `omit` and restrict commands will use a Wald test instead of the usual one based on the difference in sum of squared errors.

The confidence intervals can be computed manually using saved results from the regression or from the model window of a model estimated through the GUI. Estimate the model using `ols` from the GUI. `Select Analysis > Confidence Intervals for coefficients` in the model window to generate confidence intervals based on the HCCME.

When you estimate the model, check the 'Robust standard errors' option (see Figure 8.2) and choose the 'Configure' button to select one of the options for bias correction using the pull-down menu for cross-sectional data as shown earlier in Figure 8.3.

These robust standard errors are obtained from what is often referred to as the heteroskedasticity-consistent covariance matrix estimator (HCCME) that was proposed by Huber and rediscovered by White. In econometrics, the HCCME standard errors may be referred to as White's standard errors or Huber/White standard errors. This probably accounts for the tab's name in the dialog box.

Since least squares is inefficient in heteroskedastic models, you'd think that there might be another unbiased estimator that is more precise. And, there is. The **generalized least squares** (GLS) estimator is, at least in principle, easy to obtain. Essentially, with the GLS estimator of the heteroskedastic model, the different error variances are used to reweigh the data so that they are all have the same (homoskedastic) variance. If the data are equally variable, then least squares is efficient!

8.5 Weighted Least Squares

If you know something about the structure of the heteroskedasticity, you may be able to get more precise estimates using a generalization of least squares. In heteroskedastic models, observations that are observed with high variance don't contain as much information about the location of

the regression line as those observations having low variance. The basic idea of generalized least squares in this context is to reweigh the data so that all the observations contain the same level of information (i.e., same variance) about the location of the regression line. So, observations that contain more **noise** are given small weights and those containing more **signal** a higher weight. Reweighing the data in this way is known in some statistical disciplines as **weighted least squares**. This descriptive term is the one used by **gretl** as well.

Suppose that the errors vary proportionally with x_i according to

$$var(e_i) = \sigma^2 x_i \tag{8.5}$$

The errors are heteroskedastic since each error will have a different variance, the value of which depends on the level of x_i. Weighted least squares reweighs the observations in the model so that each transformed observation has the same variance as all the others. Simple algebra reveals that

$$\frac{1}{\sqrt{x_i}} var(e_i) = \sigma^2 \tag{8.6}$$

So, multiply equation (8.1) by $1/\sqrt{x_i}$ to complete the transformation. The transformed model is homoskedastic and least squares and the least squares standard errors are statistically valid and efficient.

Gretl makes this easy since it contains a function to reweigh all the observations according to a weight you specify. The command is `wls`, which naturally stands for weighted least squares! The only thing you need to be careful of is how **gretl** handles the weights. **Gretl** takes the square root of the value you provide. That is, to reweigh the variables using $1/\sqrt{x_i}$ you need to use its square $1/x_i$ as the weight. **Gretl** takes the square root of `w` for you. To me, this is a bit confusing, so you may want to verify what **gretl** is doing by manually transforming y, x, and the constant and running the regression. The script file shown below does this.

In the example, you first have to create the weight, then call the function `wls`. The script appears below.

```
open "@gretldir\data\poe\food.gdt"

#GLS using built in function
series w = 1/income
wls w food_exp const income
scalar lb = $coeff(income) - critical(t,$df,0.025) * $stderr(income)
scalar ub = $coeff(income) + critical(t,$df,0.025) * $stderr(income)
printf "\nThe 95%% confidence interval is (%.3f, %.3f).\n",lb,ub

#GLS using OLS on transformed data
series wi = 1/sqrt(income)
series ys = wi*food_exp
series xs = wi*x
series cs = wi
ols ys cs xs
```

The first argument after `wls` is the name of the weight variable. Then, specify the regression to which it is applied. **Gretl** multiplies each variable (including the constant) by the ***square root*** of the given weight and estimates the regression using least squares.

In the next block of the program, $w_i = 1/\sqrt{x_i}$ is created and used to transform the dependent variable, x and the constant. Least squares regression using this manually weighted data yields the same results as you get with **gretl**'s `wls` command. In either case, you interpret the output of weighted least squares in the usual way.

The weighted least squares estimation yields:

Model 6: WLS, using observations 1–40
Dependent variable: food_exp
Variable used as weight: w

	Coefficient	Std. Error	t-ratio	p-value
const	78.6841	23.7887	3.3076	0.0021
income	10.4510	1.38589	7.5410	0.0000

Statistics based on the weighted data:

Sum squared resid	13359.45	S.E. of regression	18.75006
R^2	0.599438	Adjusted R^2	0.588897
$F(1, 38)$	56.86672	P-value(F)	4.61e–09
Log-likelihood	−172.9795	Akaike criterion	349.9591
Schwarz criterion	353.3368	Hannan–Quinn	351.1804

Statistics based on the original data:

Mean dependent var	283.5735	S.D. dependent var	112.6752
Sum squared resid	304611.7	S.E. of regression	89.53266

and the 95% confidence interval for the slope β_2 is (7.645, 13.257).

Chapter six model specification

6.5 Model Selection: Introduction to gretl Functions

Choosing an appropriate model is part art and part science. Omitting relevant variables that are correlated with regressors causes least squares to be biased and inconsistent. Including irrelevant variables reduces the precision of least squares. So, from a purely technical point, it is important to estimate a model that has all of the necessary relevant variables and none that are irrelevant. It is also important to use a suitable functional form. There is no set of mechanical rules that one can follow to ensure that the model is correctly specified, but there are a few things you can do to increase your chances of having a suitable model to use for decision-making.

Here are a few rules of thumb:

1. Use whatever economic theory you have to select a functional form. For instance, if you are estimating a short-run production function then economic theory suggests that marginal returns to factors of production diminish. That means you should choose a functional form that permits this (e.g., log-log).
2. If the estimated coefficients have the wrong signs or unreasonable magnitudes, then you probably want to reevaluate either the functional form or whether relevant variables are omitted.
3. You can perform joint hypothesis tests to detect the inclusion of irrelevant sets of variables. Testing is not fool-proof since there is always positive probability that type 1 or type 2 error is being committed.
4. You can use model selection rules to find sets of regressors that are 'optimal' in terms of an estimated bias/precision trade-off.
5. Use a RESET test to detect possible misspecification of functional form.

In this section, I will give you some **gretl** commands to help with the last two: model selection and RESET.

In this section we consider three model selection rules: $\bar{R}^2$, AIC, and SC. I'm not necessarily recommending that these be used, since there are plenty of statistical problems caused by using the sample to both specify, estimate, and then test hypotheses in a model, but sometimes you have little other choice. Lag selection discussed later in this book is a reasonable application for these.

6.5.1 Adjusted R^2

The adjusted R^2 was introduced in chapter 5. The usual R^2 is 'adjusted' to impose a small penalty when a variable is added to the model. Adding a variable with any correlation to y always reduces SSE and increases the size of the usual R^2. With the adjusted version, the improvement in fit may be outweighed by the penalty and it could become smaller as variables are added. The formula is:

$$\bar{R}^2 = 1 - \frac{SSE/(N-K)}{SST/(N-1)} \tag{6.8}$$

This sometimes referred to as "R-bar squared," (i.e., $\bar{R}^2$) although in **gretl** it is called "adjusted R-squared." The biggest drawback of using $\bar{R}^2$ as a model selection rule is that the penalty it imposes for adding regressors is too small on average. It tends to lead to models that contain irrelevant variables. There are other model selection rules that impose larger penalties for adding regressors and two of these are considered below.

6.5.2 Information Criteria

The two model selection rules considered here are the Akaike Information Criterion (AIC) and the Schwarz Criterion (SC). The SC is sometimes called the Bayesian Information Criterion (BIC). Both are computed by default in **gretl** and included in the standard regression output. The values that **gretl** reports are based on maximizing a log-likelihood function (normal errors). There are other variants of these that have been suggested for use in linear regression and these are presented in the equations below:

$$AIC = \ln(SSE/N) + 2K/N \tag{6.9}$$

$$BIC = SC = \ln(SSE/N) + K\ln(N)/N \tag{6.10}$$

The rule is, compute AIC or SC for each model under consideration and choose the model that minimizes the desired criterion. The models should be evaluated using the same number of observations, i.e., for the same value of N. You can convert the ones **gretl** reports to the ones in (6.9) using a simple transformation; add $(1 + \ln(2\pi))$ and then multiply everything by N. Since sample size should be held constant when using model selection rules, you can see that the two different computations will lead to exactly the same model choice.

Since the functions have to be evaluated for each model estimated, it is worth writing a function in **gretl** that can be reused. The use of functions to perform repetitive computations makes programs shorter and reduced errors (unless your function is wrong, in which case every computation

is incorrect!) In the next section, I will introduce you to **gretl** functions and offer one that will compute the three model selection rules discussed above.

6.5.3 A gretl Function to Produce Model Selection Rules

Gretl offers a mechanism for defining **functions**, which may be called via the command line, in the context of a script, or (if packaged appropriately via the programs graphical interface. The syntax for defining a function looks like this:

```
function return-type function-name (parameters)
   function body
end function
```

The opening line of a function definition contains these elements, in strict order:

1. The keyword `function`.
2. return-type, which states the type of value returned by the function, if any. This must be one of void (if the function does not return anything), scalar, series, matrix, list or string.
3. function-name, the unique identifier for the function. Names must start with a letter. They have a maximum length of 31 characters; if you type a longer name it will be truncated. Function names cannot contain spaces. You will get an error if you try to define a function having the same name as an existing **gretl** command. Also, be careful not to give any of your variables (scalars, matrices, etc.) the same name as one of your functions.
4. The functionss parameters, in the form of a comma-separated list enclosed in parentheses. This may be run into the function name, or separated by white space as shown.

The model selection function is designed to do two things. First, we want it to print values of the model selection rules for $\bar{R}^2$, AIC and SC. While we are at it we should also print how many regressors the model has (and their names) and the sample size. The second thing we want is to be able to send the computed statistics to a matrix. This will allow us to collect results from several candidates into a single table.

The basic structure of the model selection function is

```
function matrix modelsel (series y, list xvars)
    [some computations]
    [print results]
    [return results]
end function
```

As required, it starts with the keyword `function`. The next word, `matrix`, tells the function that a matrix will be returned as output. The next word is `modelsel`, which is the name that we are giving to our function. The `modelsel` function has two arguments that will be used as inputs. The first is a data `series` that we will refer to inside the body of the function as `y`. The second is a `list` that will be referred to as `xvars`. The inputs are separated by a comma and there are spaces between the list of inputs. Essentially what we are going to do is feed the function a dependent variable and a list of the independent variables as inputs. Inside the function a regression is estimated, the criteria are computed based on it, the statistics are printed to the screen, and collected into a matrix that will be returned. The resulting matrix is then available for further manipulation outside of the function.

```
1  function matrix modelsel (series y, list xvars)
2      ols y xvars --quiet
3      scalar sse = $ess
4      scalar N = $nobs
5      scalar K = nelem(xvars)
6      scalar aic = ln(sse/N)+2*K/N
7      scalar bic = ln(sse/N)+K*N/N
8      scalar rbar2 = 1-((1-$rsq)*(N-1)/$df)
9      matrix A = { K, N, aic, bic, rbar2 }
10     printf "\nRegressors: %s\n",varname(xvars)
11     printf "K = %d, N = %d, AIC = %.4f, SC = %.4f, and\
12 Adjusted R2 = %.4f\n", K, N, aic, bic, rbar2
13     return A
14 end function
```

In line 2 the function inputs `y` and the list `xvars` are used to estimate a linear model by least squares. The `--quiet` option is used to suppress the least squares output. In lines 3-5 the sum of squared errors, *SSE*, the number of observations, *N*, and the number of regressors, *K*, are put into scalars. In lines 6-8 the three criteria are computed. Line 9 puts various scalars into a matrix called `A`. Lines 10 and 11 sends the names of the regressors to the screen. Line 11 sends formatted output to the screen. Line 12 sends the matrix `A` as a return from the function. The last line closes the function.[2]

At this point, the function can be highlighted and run.

To use the function create a `list` that will include the desired independent variables (called `x` in this case). Then to use the function you will create a matrix called `a` that will include the output from `modelsel`.

```
1 list x = const he we xtra_x5 xtra_x6
2 matrix a = modelsel(faminc,x)
```

[2]To get the **gretl** value of AIC: `scalar aic_g = (1+ln(2*pi)+aic)*N`

The output is:

```
Regressors: const,he,we,kl6,xtra_x5,xtra_x6
K = 6, N = 428, AIC = 21.2191, SC = 27.1911, and Adjusted R2 = 0.1681
```

You can see that each of the regressor names is printed out on the first line of output. This is followed by the values of K, N, AIC, SC, and $\bar{R}^2$.

6.5.4 RESET

The **RESET** test is used to assess the adequacy of your functional form. The null hypothesis is that your functional form is adequate. The alternative is that it is not. The test involves running a couple of regressions and computing an F-statistic.

Consider the model

$$y_i = \beta_1 + \beta_2 x_{i2} + \beta_3 x_{i3} + e_i \tag{6.11}$$

and the hypothesis

$$\begin{aligned} H_0: &\quad E[y|x_{i2}, x_{i3}] = \beta_1 + \beta_2 x_{i2} + \beta_3 x_{i3} \\ H_1: &\quad \text{not } H_0 \end{aligned}$$

Rejection of H_0 implies that the functional form is not supported by the data. To test this, first estimate (6.11) using least squares and save the predicted values, $\hat{y}_i$. Then square and cube $\hat{y}$ and add them back to the model as shown below:

$$\begin{aligned} y_i &= \beta_1 + \beta_2 x_{i2} + \beta_3 x_{i3} + \gamma_1 \hat{y}_i^2 + e_i \\ y_i &= \beta_1 + \beta_2 x_{i2} + \beta_3 x_{i3} + \gamma_1 \hat{y}_i^2 + \gamma_2 \hat{y}_i^3 + e_i \end{aligned}$$

The null hypotheses to test (against alternative, 'not H_0') are:

$$\begin{aligned} H_0: &\quad \gamma_1 = 0 \\ H_0: &\quad \gamma_1 = \gamma_2 = 0 \end{aligned}$$

Estimate the auxiliary models using least squares and test the significance of the parameters of $\hat{y}^2$ and/or $\hat{y}^3$. This is accomplished through the following script. Note, the `reset` command issued

after the first regression computes the test associated with $H_0 : \gamma_1 = \gamma_2 = 0$. It is included here so that you can compare the 'canned' result with the one you compute using the two step procedure suggested above. The two results should match.

```
ols faminc x3 --quiet
reset --quiet
reset --quiet --squares-only
```

The results of the RESET for the family income equation is

```
RESET test for specification (squares and cubes)
Test statistic: F = 3.122581,
with p-value = P(F(2,422) > 3.12258) = 0.0451

RESET test for specification (squares only)
Test statistic: F = 5.690471,
with p-value = P(F(1,423) > 5.69047) = 0.0175
```

The adequacy of the functional form is rejected at the 5% level for both tests. It's back to the drawing board!

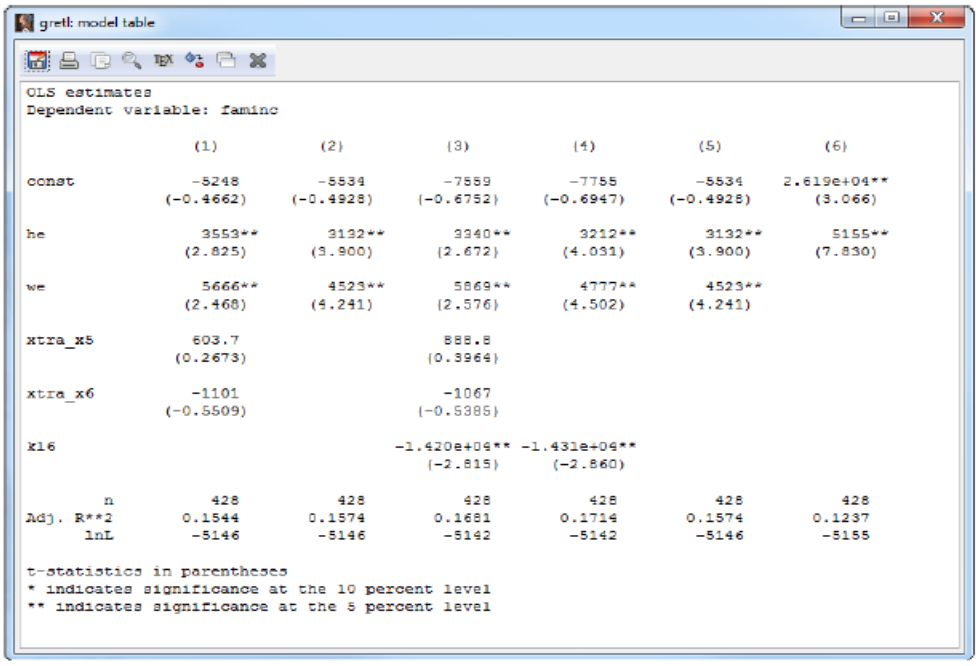


Figure 6.13: Save each model as an icon. Open the session window and drag each model to the model table icon. Click on the model table icon to reveal this output.

1. Use whatever economic theory you have to select a functional form. For instance, if you are estimating a short-run production function then economic theory suggests that marginal returns to factors of production diminish. That means you should choose a functional form that permits this (e.g., log-log).
2. If the estimated coefficients have the wrong signs or unreasonable magnitudes, then you probably want to reevaluate either the functional form or whether relevant variables are omitted.
3. You can perform joint hypothesis tests to detect the inclusion of irrelevant sets of variables. Testing is not fool-proof since there is always positive probability that type 1 or type 2 error is being committed.
4. You can use model selection rules to find sets of regressors that are 'optimal' in terms of an estimated bias/precision trade-off.
5. Use a RESET test to detect possible misspecification of functional form.

In this section, I will give you some **gretl** commands to help with the last two: model selection and RESET.

I have taken the following descriptions of some of the standard diagnostics tests from: "Introduction to Probability and Statistics Using R", G. Jay Kerns, available at http://cran.r-project.org/web/packages/IPSUR/vignettes/IPSUR.pdf

11.4.3 Independence Assumption

One of the strongest of the regression assumptions is the one regarding independence. Departures from the independence assumption are often exhibited by correlation (or autocorrelation, literally, self-correlation) present in the residuals. There can be positive or negative correlation.

Positive correlation is displayed by positive residuals followed by positive residuals, and negative residuals followed by negative residuals. Looking from left to right, this is exhibited by a cyclical feature in the residual plots, with long sequences of positive residuals being followed by long sequences of negative ones.

On the other hand, negative correlation implies positive residuals followed by negative residuals, which are then followed by positive residuals, *etc*. Consequently, negatively correlated residuals are often associated with an alternating pattern in the residual plots. We examine the residual plot in Figure 11.4.3. There is no obvious cyclical wave pattern or structure to the residual plot.

Testing the Independence Assumption

We may statistically test whether there is evidence of autocorrelation in the residuals with the Durbin-Watson test. The test is based on the statistic

$$D = \frac{\sum_{i=2}^{n}(E_i - E_{i-1})^2}{\sum_{j=1}^{n} E_j^2}. \tag{11.4.6}$$

11.5 Other Diagnostic Tools

There are two types of observations with which we must be especially careful:

Influential observations are those that have a substantial effect on our estimates, predictions, or inferences. A small change in an influential observation is followed by a large change in the parameter estimates or inferences.

Outlying observations are those that fall fall far from the rest of the data. They may be indicating a lack of fit for our regression model, or they may just be a mistake or typographical error that should be corrected. Regardless, special attention should be given to these observations. An outlying observation may or may not be influential.

We will discuss outliers first because the notation builds sequentially in that order.

11.5.1 Outliers

There are three ways that an observation (x_i, y_i) may be an outlier: it can have an x_i value which falls far from the other x values, it can have a y_i value which falls far from the other y values, or it can have both its x_i and y_i values falling far from the other x and y values.

Leverage

Leverage statistics are designed to identify observations which have x values that are far away from the rest of the data. In the simple linear regression model the leverage of x_i is denoted by h_{ii} and defined by

$$h_{ii} = \frac{1}{n} + \frac{(x_i - \overline{x})^2}{\sum_{k=1}^{n}(x_k - \overline{x})^2}, \quad i = 1, 2, \ldots, n. \tag{11.5.1}$$

The formula has a nice interpretation in the SLR model: if the distance from x_i to $\overline{x}$ is large relative to the other x's then h_{ii} will be close to 1.

Leverages have nice mathematical properties; for example, they satisfy

$$0 \leq h_{ii} \leq 1, \tag{11.5.2}$$

and their sum is

$$\begin{aligned} \sum_{i=1}^{n} h_{ii} &= \sum_{i=1}^{n}\left[\frac{1}{n} + \frac{(x_i - \overline{x})^2}{\sum_{k=1}^{n}(x_k - \overline{x})^2}\right], && (11.5.3)\\ &= \frac{n}{n} + \frac{\sum_i (x_i - \overline{x})^2}{\sum_k (x_k - \overline{x})^2}, && (11.5.4)\\ &= 2. && (11.5.5) \end{aligned}$$

A rule of thumb is to consider leverage values to be large if they are more than double their average size (which is $2/n$ according to Equation 11.5.5). So leverages larger than $4/n$ are suspect. Another rule of thumb is to say that values bigger than 0.5 indicate high leverage, while values between 0.3 and 0.5 indicate moderate leverage.