Dougherty: Introduction to Econometrics 4e

Manual for doing the computer exercises in

Dougherty, C. *Introduction to Econometrics*
(Oxford: Oxford University Press, fourth edition 2011) using

gretl

gretl

gretl (GNU Regression, Econometrics and Time-Series Library) is the official econometrics application for Christopher Doughert's University of London module "20 Elements of Econometrics". It is free (obviously, a major consideration for distance learning students) and very powerful. It does everything that is required in the computer regression exercises in the course text Introduction to Econometrics (C. Dougherty, Oxford University Press, fourth edition 2011). In fact, we use only a small fraction of its features. It is similar to major econometrics applications such as Stata, EViews, SPSS, and SAS, with the crucial difference that you do not have to pay for it. If you already have access to one of the commercial applications just mentioned, there is no need to change. However, if you are struggling to run regressions with a regression feature built into a spreadsheet application such as Microsoft Excel, quit now and switch to gretl. Spreadsheet applications can do only very basic regressions and are quite unsuitable for this course.

gretl should be downloaded from the its home page http://gretl.sourceforge.net/ See that page for its history and current status.

This manual provides just the information needed to undertake the regression exercises in the text. The information is provided as you need it, so that you are not overwhelmed by too much information too soon. By the time that you have completed the exercises, you will have a good basic working knowledge of the application. Comprehensive reference information can be found in the gretl manual gretl-guide.pdf located in the Help section of gretl. The Help section also provides direct access to information about the commands and functions used by gretl.

It is assumed that you have downloaded and installed gretl and that you have downloaded the necessary data sets from the LSE website or from the OUP site at

www.oup.com/uk/orc/dougherty4e

(The OUP link is likely to appear if you type 'Dougherty 4e' into your search engine.)

GUI/CLI/Batch mode

In common with other sophisticated econometrics packages, gretl may be operated in three ways: via a graphical user interface (GUI), via a command line interface (CLI), and in batch mode.

The GUI involves menus and windows, is intuitively designed, and provides an excellent starting point for experimenting with gretl. However, it will mostly be ignored here, for two reasons. One, the less important, is that operations with menus and windows can be cumbersome to describe. The other is that serious researchers always primarily use batch mode.

The CLI involves using a special command line window, called the 'command console' in the gretl manual, instead of the GUI. The command line window may be opened by clicking on the Tools tab in the main menu and then 'Gretl console'. Operations, such as opening a data file or running a regression, are effected by typing in a line of text into this window and then pressing the Enter key.

Batch mode is essentially a well-organized use of the CLI. Instead of entering the commands one by one, they are all written together, line by line, in a text file with extension `.inp`. The text file is then run by gretl, the commands in it being executed sequentially.

Batch mode has **huge** advantages over the interactive use of the GUI or the CLI:

- You have an exact and permanent record of your operations
- If you make a mistake somewhere in your operations, all you have to do is to correct the faulty line in the batch file rather than repeat all your pointing and clicking, and
- If at some future time you wish to modify the analysis or develop it further, you do not have to start from scratch. You modify the existing batch file or create a new one using it as a base. This is how research is done.

There is a further small advantage of batch mode over the GUI. If you are skilled at word processing, you may actually find it quicker to use a batch file than the GUI.

Note that gretl calls batch files 'script files'. To run one, you launch gretl, go to Files/Script files/User files and then browse to the location of your file. Double click on its name and a window listing its contents will open. To run the commands, click the cogs icon in the menu of this window.

Since the exercises in the text are relatively simple, each involving only a few commands, the instructions in this manual assume that you use the CLI directly. Indeed, if you prefer, you could use the GUI. However the explanation here is in terms of using the CLI because I can describe exactly what you need to do for each of the exercises that you are going to undertake. Short of leaving out a space where there should be one, or using a number '1' instead of a letter 'l', you should not be able to make mistakes.

Common operations

Here are a few commands that will be useful in the exercises: You will learn other commands as you work your way through the exercises. In fact, doing the exercises is a relatively painless way of acquiring a basic understanding of how to use gretl.

`ols` followed by a variable name, then `const`, and further variable names. The first variable is regressed on the rest. Omit `const` if there is no intercept in the model.

`summary` followed by a list of variable names. This produces a table giving the mean, maximum, minimum, and standard deviation of each variable listed.

`corr` followed by a list of variables. The correlation coefficients are calculated.

`genr` followed by an equation. This creates a new variable defined as the dependent variable of the equation.

`smpl` followed by a condition in parentheses. This restricts the observations to the subset that satisfy the condition. `smpl` is cumulative, so that a second use of it will select a subset of the subset selected by its first use. The command `smpl full` must be used to return to the full sample.

Examples of the use of these commands, and a few others, will be found in the following list of exercises.

Exercise 1.6 Opening a data file, running a simple regression, and saving and printing the output

Exercise 3.14 Using the `corr` command for computing correlation coefficients

Exercise 3.15 Creating new variables from existing ones

Exercise 4.4 Limiting the sample to those observations that satisfy some condition

Exercise 4.4 Creating a new variable as a function of an existing one. See Help/Function reference from the gretl main menu for a complete list of available functions.

Exercise 4.9 Saving a data file after defining new variables (overwriting the existing file).

Exercise 5.19 Restoring the full sample

Exercise 5.21 Using a backslash in a batch file to extend a long command that has wrapped to a second line.

Exercise 7.3 Sorting a data set

Exercise 8.10 Instrumental variables (IV) regression and Hausman test

Exercise 10.4 Defining a dummy variable using a condition

Exercise 10.4 logit regression

Exercise 10.6 probit regression

Exercise 10.9 tobit regression

Exercise 10.11 heckit (sample selection) regression

Exercise 11.2 Plotting a time series

Exercise 11.7 Creating lagged versions of existing variables

Exercise 13.9 Creating first differences of variables using the `diff` command

Exercise 13.9 Testing for a unit root (augmented Dickey–Fuller test) with the `adf` command

Exercise 14.2 Testing for the presence of fixed effects with an F test

Exercise 14.6 Testing for the presence of random effects with a Breusch–Pagan test

Exercise 14.6 Discriminating between fixed effects and random effects using a Hausman test

Exercise 1.6 (*EAEF*)

Opening a data set

This involves using one of the *EAEF* data sets described in detail in Appendix B of the text. There are 20 for the use of students working together with an instructor. The students should use different data sets and therefore get slightly different results. If you are working on your own, it does not matter which one you use (other than set 21 which is used for examples in the text). You should download your data set in Stata format from the OUP website. I shall represent the path to the subfolder with your data set as `path\`. To open the dataset, you should type the following command in the console box

```
open path\eaef**.gdt
```

where ** is the number of your data set. Thus if you are using data set 7 and have stored the data set in subfolder 'data' of folder 'gretl' on your C: drive, *path* will be `C:\gretl\data` and the command will be

```
open c:\gretl\data\eaef07.gdt
```

Running a simple ordinary least squares regression

The command is `ols`. It should be followed by the name of the dependent variable, then `const` (assuming you want an intercept in the regression) and then the name of the explanatory variable. In this exercise the command is

```
ols S const ASVABC
```

Note that gretl is case sensitive. `ols` and `const` must always be lower case. `S` and `ASVABC` must be upper case because they are defined as upper case in the data set..

Saving and printing the output

An easy way to print the output is to block off the regression output that appears in the console window and paste it into a Word document (or other word-processor document). Change the font to Courier so that the output lines up properly. If the lines wrap, reduce the size of the font. Print this document in the usual way.

Alternatively, you could save the whole console window as a text file and edit it afterwards.

Exercise 1.7 (*EAEF*)

The regression command is

```
ols EARNINGS const S
```

Exercises 1.19 and 1.20 (*EAEF*)

R^2 is part of the output (`unadjusted R-squared`) in Exercises 1.6 and 1.7.

Exercises 2.16, 2.17, 2.21, and 2.22 (*EAEF*)

The t statistics are part of the output in Exercises 1.6 and 1.7.

Exercise 2.27 (*EAEF*)

You will need to use the standard error of the coefficient. It is part of the output in Exercise 1.7.

Exercise 2.30 (*EAEF*)

F statistics are generally calculated using data on the sums of the squares of the residuals. Occasionally they may equivalently be calculated from data on R^2, but since this is not always the case it is better to give priority to the sums of squares approach. Accordingly, in the next edition of the text, in this exercise I will ask students to use that approach instead (and then maybe confirm that the same F statistic can be obtained via R^2).

In the present case, using equation (2.66) in the text, the expression for F is

$$F(k-1, n-k) = \frac{ESS/(k-1)}{RSS/(n-k)} = \frac{ESS}{RSS/(n-2)}$$

since $k = 2$. However the output from a gretl regression reports RSS but does not report ESS. Given equation (2.64), we could calculate it as ($TSS - RSS$), where TSS is the total sum of squares:

$$TSS = \sum_{i=1}^{n} (Y_i - \bar{Y})^2$$

but TSS is not reported either. To get around this, regress the dependent variable on a constant only, fitting the model

$$Y = \beta_1 + u$$

The estimator of β_1 will be $\bar{Y}$. The fitted value of Y in all observations will be $\bar{Y}$. The residual in observation i will be $Y_i - \bar{Y}$, and hence RSS for this regression will be $\sum_1^n (Y_i - \bar{Y})^2$. By definition, this is TSS for any regression with Y as the dependent variable. So run this regression first, and then compute ESS for the main regression as $TSS - RSS$ (RSS for the main regression). Thus the necessary commands are

```
ols EARNINGS const
ols EARNINGS const S
```

To compute F from R^2, use equation (2.68) in the text.

gretl does not report the F statistic for simple regressions run in batch mode or from the console box. The (unstated) reason is that there is no need. It is automatically equal to the square of the t statistic of the coefficient of the explanatory variable (see section 2.7 of the text). However you may produce the F statistic by placing the following command immediately after the regression:

```
genr F = ($rsq/($ncoeff-1))/((1-$rsq)/$df)
```

The expression uses the R^2 approach. (`$rsq`, `$ncoeff`, and `$df` are temporary scalars containing R^2, the number of coefficients, and the number of degrees of freedom, respectively in the most recent regression.)

In GUI mode the *F* statistic may be obtained directly via the analysis tab on the menu line.

Exercise 3.2 (*EAEF*)

```
ols S const ASVABC SM
ols S const ASVABC SF
ols S const ASVABC SM SF
```

Exercise 3.3 (*EAEF*)

```
ols EARNINGS const S EXP
```

Exercise 3.4 (*EAEF*)

```
ols S const ASVABC SF
```

`$uhat` is an invisible temporary variable that contains the residuals from the most recent regression. The next command saves them permanently as `ES`.

```
genr ES = $uhat
ols SM const ASVABC SF
genr ESM = $uhat
ols ES const ESM
```

The exercise asks for a plot of `ES` on `ESM`. I have not managed to do this using console box commands.

Exercise 3.9 (*EAEF*)

Use the results from Exercises 3.2 and 3.3

Exercise 3.12 (*EAEF*)

An example might be

```
ols WEIGHT02 const HEIGHT S AGE
```

(What kinds of coefficient would you anticipate for `S` and `AGE`?)

Exercise 3.14 (*EAEF*)

```
ols S const SM SF ASVAB02 ASVAB03 ASVAB04
ols S const SM SF ASVABC
corr ASVAB02 ASVAB03 ASVAB04
```

Exercise 3.15 (EAEF)

The fourth command generates the new variable SP from the existing ones SM and SF.

```
genr CHILDREN = SIBLINGS + 1
ols CHILDREN const SM SF
corr SM SF
genr SP = SM + SF
ols CHILDREN const SP
```

Exercise 3.17 (*EAEF*)

```
genr PWE = AGE - S - 5
ols EARNINGS const S PWE
ols EARNINGS const S PWE AGE
corr S PWE AGE
```

Exercise 3.18 (*EAEF*)

```
ols S const ASVABC SM SF
```

Exercise 3.19 (*EAEF*)

```
ols S const ASVABC SM SF ASVAB05 ASVAB06
```

Exercise 3.20 (*EAEF*)

```
ols S const ASVABC SM SF ASVAB05
```

See Exercise 2.30 for a command that will produce the *F* statistic.

Exercise 4.4 (*CES*)

First you should download the CES data set CES.gdt from the OUP website, launch gretl, and open the data set with the open command (see Exercise 1.3). You should work with only one of the different categories of expenditure. (If the exercises are being done by a group, the instructor should assign a different category to each student.) The commands use *CAT* as a placeholder for the three or four letter name of your category. The first command restricts the sample to those observations with positive values for your category of expenditure.

```
smpl CAT > 0 --restrict
ols CAT const EXP
```

Just to avoid any doubt, if your category were CLOT (clothing), the commands would be

```
smpl CLOT > 0 --restrict
ols CLOT const EXP
```

To run the logarithmic regression, you first need to create the logarithmic variables

```
genr LGCAT = log(CAT)

genr LGEXP = log(EXP)

ols LGCAT const LGEXP
```

Note that log is log to base *e*, the natural logarithm.

```
smpl full

store path\CES.gdt
```

In the last command, you should replace `path\` by the path to the location where you have downloaded the CES data file. See Exercise 1.3 for further information on `path\`. The store command will overwrite the existing CES data file, preserving the new variables `LGCAT` and `LGEXP`. This means that you will not need to redefine `them` the next time that you need them. You should always save a data file after defining new variables if you think that you will need them again.

Exercise 4.5 (*CES*)

```
genr LGSIZE = log(SIZE)

ols LGCAT const LGEXP LGSIZE

store path\CES.gdt
```

Exercise 4.6 (*EAEF*)

```
genr LGWT85 = log(WEIGHT85)

genr LGHT = log(HEIGHT)

ols LGWT85 const LGHT
```

Exercise 4.9 (*EAEF*)

```
genr LGEARN = log(EARNINGS)

ols LGEARN const S EXP

store path\EAEF01.gdt
```

In the last command, you should replace `EAEF01` by the name of your particular data set and `path\` by the path to it. See Exercises 1.3 and 4.1 for further information. The store command will overwrite the existing data file, preserving the new variable `LGEARN`. This means that you will not need to redefine `LGEARN` the next time that you need it.

Exercise 4.10 (*EAEF*)

```
genr EARNSTAR = EARNINGS/exp(mean(LGEARN))
genr LGEARNST = log(EARNSTAR)
ols EARNSTAR const S EXP
ols LGEARNST const S EXP
```

`mean(LGEARN)` produces the mean of `LGEARN` and `exp(mean(LGEARN))` produces its exponential.

Exercise 4.11 (*CES*)

```
smpl CAT > 0 --restrict
genr CATSTAR = CAT/exp(mean(LGCAT))
genr LGCATST = log(CATSTAR)
ols CATSTAR const EXP SIZE
ols LGCATST const LGEXP LGSIZE
```

Exercise 4.13 (*EAEF*)

```
genr SMSQ = SM*SM
ols S const SM SMSQ
genr SM12 = SM - 12
genr SM12SQ = SM12*SM12
ols S const SM12 SM12SQ
```

Exercise 4.15 (*EAEF*)

```
ols WEIGHT02 const HEIGHT
genr YHAT = $yhat
genr YHATSQ = YHAT*YHAT
ols WEIGHT02 const HEIGHT YHATSQ
```

Exercise 5.1 (*EAEF*)

```
ols S const ASVABC SM SF MALE
```

Exercise 5.3 (*EAEF*)

```
ols LGEARN const S EXP MALE
```

See Box 5.1 in the text for a guide to the interpretation of dummy variable coefficients in semilogarithmic regressions.

Exercise 5.6 (*EAEF*)

```
ols S const ASVABC SM SF MALE ETHBLACK ETHHISP
```

Exercise 5.8 (*EAEF*)

```
ols LGEARN const S EXP MALE ETHBLACK ETHHISP
```

Exercise 5.10 (*EAEF*)

Use the results from Exercises 5.1 and 5.6.

Exercise 5.11 (*EAEF*)

Use the results from Exercises 5.3 and 5.8.

Exercise 5.12 (*EAEF*)

```
ols S const ASVABC SM SF MALE ETHHISP ETHWHITE
```

Exercise 5.11 (*EAEF*)

```
ols LGEARN const S EXP MALE ETHHISP ETHWHITE
```

Exercise 5.14 (*EAEF*)

```
ols LGEARN const S EXP MALE FEMALE
```

Exercise 5.15 (*EAEF*)

```
genr MALEASVC = MALE*ASVABC
ols S const ASVABC SM SF MALE ETHBLACK ETHHISP MALEASVC
store path\EAEF**.gdt
```

In the last command, you should replace `**` by the number of your particular data set and *`path\`* by the path to it. See Exercises 1.6 and 4.4 for further information. The `store` command will overwrite the existing data file, preserving the new variable `MALEASVC`. This means that you will not need to redefine `MALEASVC` the next time that you need it.

Exercise 5.17 (*EAEF*)

```
genr MALES = MALE*S
ols LGEARN const S EXP MALE ETHHISP ETHWHITE MALES
store path\EAEF**.gdt
```

In the last command, you should replace `**` by the number of your particular data set and *`path\`* by the path to it. See Exercises 1.6 and 4.4 for further information.

Exercise 5.18 (*EAEF*)

```
genr MALEBLAC = MALE*ETHBLACK
genr MALEHISP = MALE*ETHHISP
ols S const ASVABC SM SF MALE ETHBLACK ETHHISP MALEBLAC MALEHISP
store path\EAEF**.gdt
```

In the last command, you should replace `**` by the number of your particular data set and `path\` by the path to it. See Exercises 1.6 and 4.4 for further information.

Exercise 5.19 (*EAEF*)

```
ols S const ASVABC SM SF ETHBLACK ETHHISP
smpl MALE = 1 --restrict
ols S const ASVABC SM SF ETHBLACK ETHHISP
smpl full
smpl MALE = 0 --restrict
ols S const ASVABC SM SF ETHBLACK ETHHISP
smpl full
```

The `smpl` command is cumulative. `smpl MALE = 1 --restrict` eliminates all the female observations. If it is followed by `smpl MALE = 0 --restrict`, all the male observations are eliminated as well, leaving a null data set. To avoid this, any `smpl` command that is temporary and no longer needed should be cancelled with the `smpl full` command. This restores the sample to the full sample.

Exercise 5.20 (*EAEF*)

```
ols LGEARN const S EXP MALE ETHBLACK ETHHISP
smpl MALE = 1 --restrict
ols LGEARN const S EXP MALE ETHBLACK ETHHISP
smpl full
smpl MALE = 0 --restrict
ols LGEARN const S EXP MALE ETHBLACK ETHHISP
smpl full
```

Exercise 5.21 (*EAEF*)

```
genr MALESM = MALE*SM

genr MALESF = MALE*SF

ols S const ASVABC SM SF MALE ETHBLACK ETHHISP MALEASVC MALESM MALESF
MALEBLAC MALEHISP

ols S const ASVABC SM SF ETHBLACK ETHHISP

corr ASVABC SM SF MALE ETHBLACK ETHHISP MALEASVC MALESM MALESF
MALEBLAC MALEHISP

store path\EAEF**.gdt
```

Note that the command for the first regression is so long that it has wrapped to a second line. This is fine is you are using the command line (console box). However, if you are writing a batch file, gretl treats each line as a separate command. To indicate that a second line is a continuation of the first, one needs to insert a backslash at the end of the first line.

The `corr` command computes the correlations between the variables listed. It has also wrapped and would need a backslash at the end of the first line if you were writing a batch file.

In the last command, you should replace `**` by the number of your particular data set and *path* by the path to it. See Exercises 1.6 and 4.4 for further information.

Exercise 5.22 (*EAEF*)

```
genr MALEEXP = MALE*EXP

ols LGEARN const S EXP MALE ETHBLACK ETHHISP MALES MALEEXP MALEBLAC
MALEHISP

ols LGEARN const S EXP ETHBLACK ETHHISP

corr S EXP MALE ETHBLACK ETHHISP MALES MALEEXP MALEBLAC MALEHISP

store path\EAEF01.gdt
```

In the last command, you should replace `**` by the number of your particular data set and *path* by the path to it. See Exercises 1.6 and 4.4 for further information.

Exercise 6.1 (*EAEF*)

```
ols S const ASVABC SM

ols S const ASVABC

ols S const SM

corr ASVABC SM
```

Exercise 6.2 (*EAEF*)

```
ols LGEARN const S EXP
ols LGEARN const S
ols LGEARN const EXP
corr S EXP
```

Exercise 6.3 (*EAEF*)

```
ols LGEARN const S EXP MALE ETHBLACK ETHHISP
ols LGEARN const S EXP MALE ETHBLACK ETHHISP ASVABC
corr S EXP MALE ETHBLACK ETHHISP ASVABC
```

Exercise 6.8 (*EAEF*)

```
ols LGEARN const S EXP ASVABC MALE ETHBLACK ETHHISP
ols LGEARN const S EXP ASVABC MALE ETHBLACK ETHHISP SIBLINGS
corr S EXP ASVABC MALE ETHBLACK ETHHISP SIBLINGS
```

Exercise 6.9 (*EAEF*)

If you did Exercise 3.14 and saved your data set afterwards, you do not need to define PWE again here.

```
genr PWE = AGE - S - 5
ols LGEARN const S ASVABC MALE ETHBLACK ETHHISP
ols LGEARN const S ASVABC MALE ETHBLACK ETHHISP PWE
ols LGEARN const S ASVABC MALE ETHBLACK ETHHISP EXP
corr S EXP PWE
```

Variation

```
smpl MALE = 1 --restrict
ols LGEARN const S ASVABC MALE ETHBLACK ETHHISP
ols LGEARN const S ASVABC MALE ETHBLACK ETHHISP PWE
ols LGEARN const S ASVABC MALE ETHBLACK ETHHISP EXP
corr S EXP PWE
smpl full
smpl MALE = 0 --restrict
ols LGEARN const S ASVABC MALE ETHBLACK ETHHISP
ols LGEARN const S ASVABC MALE ETHBLACK ETHHISP PWE
ols LGEARN const S ASVABC MALE ETHBLACK ETHHISP EXP
corr S EXP PWE
smpl full
```

Exercise 6.11 (*EAEF*)

The text should have specified using ASVABC as an explanatory variable in the second regression as well as in the first.

```
genr PREVEXP = EXP - TENURE
ols LGEARN const S ASVABC EXP MALE ETHBLACK ETHHISP
ols LGEARN const S ASVABC PREVEXP TENURE MALE ETHBLACK ETHHISP
smpl MALE = 1 --restrict
ols LGEARN const S ASVABC EXP MALE ETHBLACK ETHHISP
ols LGEARN const S ASVABC PREVEXP TENURE MALE ETHBLACK ETHHISP
smpl full
smpl MALE = 0 --restrict
ols LGEARN const S ASVABC EXP MALE ETHBLACK ETHHISP
ols LGEARN const S ASVABC PREVEXP TENURE MALE ETHBLACK ETHHISP
smpl full
```

Exercise 6.12 (*EAEF*)

```
ols LGEARN const S ASVABC EXP TENURE MALE ETHBLACK ETHHISP
smpl MALE = 1 --restrict
ols LGEARN const S ASVABC EXP TENURE MALE ETHBLACK ETHHISP
smpl full
smpl MALE = 0 --restrict
ols LGEARN const S ASVABC EXP TENURE MALE ETHBLACK ETHHISP
smpl full
```

Exercise 6.14 (*EAEF*)

```
ols S const SM SF ASVABC
ols S const SM SF ASVAB02 ASVAB03 ASVAB04
```

Exercise 7.3 (*EAEF*)

As should be obvious, the first command sorts the observations in (increasing) size of `S`. The second command restricts the sample to the first 203 of the sorted observations (approximately three-eights of 540). The fourth command redefines the sample as those observations with the 203 largest values of `S` .

The specification has deliberately been kept rather simple, in anticipation of Exercise 7.4. Obviously it would be a good idea to add `ASVABC`, `ETHBLACK`, and `ETHHISP`.

```
dataset sortby S
smpl 1 203
ols EARNINGS const S EXP MALE
smpl 338 540
ols EARNINGS const S EXP MALE
smpl full
```

Exercise 7.4 (*EAEF*)

It is assumed that `MALES` and `MALEEXP` have been defined and saved in previous exercises. The residuals from the first regression are saved as `EEARN`. As in Exercise 7.3, it would be a good idea to add `ASVABC`, `ETHBLACK`, and `ETHHISP` to the specification ... and their squares and interactions with each other and the other variables to the test equation. The final command is a special gretl function that produces the test statistic, nR^2, for the White test.

```
ols EARNINGS const S EXP MALE
genr EEARN = $uhat
genr EEARNSQ = EEARN*EEARN
genr SSQ = S*S
genr EXPSQ = EXP*EXP
genr SEXP = S*EXP
ols EEARNSQ const S EXP MALE SSQ EXPSQ SEXP MALES MALEEXP
genr WHITE = $trsq
```

Exercise 7.6 (*EDUC*)

It is assumed that you have downloaded the *EDUC* data set and have opened it within gretl.

```
dataset sortby GDP
smpl 1 14
ols EDUC const GDP
smpl full
smpl 25 38
ols EDUC const GDP
smpl full
```

Exercise 7.8 (*EAEF*)

It is assumed that the variables defined in Exercise 7.4 have been saved.

Repeat of Exercise 7.3 with LGEARN

```
dataset sortby S
smpl 1 203
ols LGEARN const S EXP MALE
smpl 338 540
ols LGEARN const S EXP MALE
smpl full
```

Repeat of Exercise 7.4 with LGEARN

```
ols LGEARN const S EXP MALE
genr EELG = $uhat
genr EELGSQ = EELG*EELG
genr SSQ = S*S
genr EXPSQ = EXP*EXP
genr SEXP = S*EXP
ols EELGSQ const S EXP MALE SSQ EXPSQ SEXP
MALES MALEEXP
genr WHITELG = $trsq
```

Exercise 7.9 (*EDUC*)

Scaling by population

```
genr EDUCPOP = EDUC/POP
genr GDPPOP = GDP/POP
genr RECPOP = 1/POP
dataset sortby GDPPOP
smpl 1 14
ols EDUCPOP RECPOP GDPPOP
smpl full
smpl 25 38
ols EDUCPOP RECPOP GDPPOP
smpl full
```

Scaling by GDP

```
genr EDUCGDP = EDUC/GDP
genr RECGDP = 1/GDP
dataset sortby RECGDP
smpl 1 14
```

```
ols EDUCGDP const RECGDP
smpl full
smpl 25 38
ols EDUCGDP const RECGDP
smpl full
```

Logarithmic specification

```
genr LGEDUC = log(EDUC)
genr LGGDP = log(GDP)
dataset sortby LGGDP
smpl 1 14
ols LGEDUC const LGGDP
smpl full
smpl 25 38
ols LGEDUC const LGGDP
smpl full
```

Whole sample regressions, for comparison

```
ols EDUCPOP RECPOP GDPPOP
ols EDUCGDP const GDPREC
ols LGEDUC const LGGDP
```

Exercise 8.14 (*EAEF*)

The command for undertaking an instrumental variable regression is `tsls` (short for two stage least squares; all IV regressions may be considered to be special cases of this estimation procedure). The first item after the command is, as usual, the dependent variable, and this is followed by const (assuming that you do want a constant) and the list of the explanatory variables appearing in the regression. Then follows a semicolon, and finally the list of the instrumental variables being used. gretl follows the convention that any variable that does not need an instrument is said to be acting as its own instrument and so should be included in the list of instruments following the semicolon. This applies even to the constant. Hence the list of instruments is the same as the list of the explanatory variables except (1) those variables needing instruments do not appear, and the external instruments do appear. In this case `ASVABC` is dropped from the list of instruments because it is being instrumented, and `SM`, `SF`, `SIBLINGS`, and `LIBRARY` are included in the list of instruments.

The Durbin–Wu–Hausman test is automatically included in the `tsls` regression output. It is implemented by gretl in a way that is different from that used in Stata and so may give somewhat different results.

```
tsls LGEARN const S EXP ASVABC ; const S EXP SM SF SIBLINGS LIBRARY
corr ASVABC SM SF SIBLINGS LIBRARY
```

The second command has been included to investigate the strength of the correlation between `ASVABC` and the instruments. Although the exercise in the text does not suggest including `MALE`, `ETHBLACK`, and

ETHHISP in the regression, obviously this would improve the specification.

```
tsls LGEARN const S EXP ASVABC MALE ETHBLACK ETHHISP ; const S EXP MALE
ETHBLACK ETHHISP SM SF SIBLINGS LIBRARY
```

Exercise 9.10 (*EAEF*)

```
ols ASVABC const S

tsls ASVABC const S ; const SM

cor S SM
```

Exercise 10.4 (*EAEF*)

```
genr COLLEGE = (S > 12)

ols COLLEGE const ASVABC SM SF MALE

logit COLLEGE const ASVABC SM SF MALE --p-values

logit COLLEGE const ASVABC SM SF MALE
```

The first command generates COLLEGE as a dummy variable that is equal to 1 when the condition in parentheses is satisfied and equal to 0 when it is not. So COLLEGE is equal to 1 for those who had more than 12 years of schooling (which means that they must have at least one year of college) and it is 0 for the rest. The second command performs a linear probability model regression. The third performs the corresponding logit regression. The fourth does the same, except that the output reports the marginal effects instead of the p values. As can be seen, this is actually the default for the logit regression in gretl. The marginal effects are calculated at the mean values of tall of the explanatory variables.

Exercise 10.6 (*EAEF*)

```
probit COLLEGE const ASVABC SM SF MALE --p-values

probit COLLEGE const ASVABC SM SF MALE
```

Exercise 10.8 (*LFP*)

```
genr MARL06 = MARRIED*CHILDL06

smpl MALE = 0 --restrict

probit WORKING const S AGE CHILDL06 CHILDL16 MARRIED
MARL06 ETHBLACK ETHHISP --p-values

probit WORKING const S AGE CHILDL06 CHILDL16 MARRIED
MARL06 ETHBLACK ETHHISP
```

Exercise 10.9 (*CES*)

Remember, *CAT* is a placeholder for your category of expenditure. See Exercise 4.4.

```
ols CAT const EXP
```

```
smpl CAT > 0 --restrict
ols CAT const EXP
smpl full
tobit CAT const EXP
```

The first command fits the regression using all the observations, including the zero observations. The next two commands do the same, restricting the sample to the non-zero observations. The last performs the tobit regression.

Exercise 10.11 (*EAEF*)

```
smpl S > 12 --restrict
genr COLLYEAR = S - 12
genr LGEARNCL = LGEARN
ols LGEARNCL const COLLYEAR EXP ASVABC MALE ETHBLACK ETHHISP
smpl full
genr D = (S > 12)
heckit LGEARNCL const COLLYEAR EXP ASVABC MALE ETHBLACK ETHHISP ;
D const ASVABC MALE ETHBLACK ETHHISP SM SF SIBLINGS
```

The third command defines the logarithm of earnings for those who went to college. It is not defined for the rest. The sixth command defines a dummy variable `D` that is equal to 1 for observations that are selected and 0 for the rest. The first part of the `heckit` command regresses the logarithm of earnings on the usual explanatory variables. The second part (after the semicolon) lists the dummy variable D followed by the variables used to explain selection. If this command is being used in a batch file, a backslash should be added at the end of the first line to indicate that the command continues to a second line.

Exercise 10.13 (*LFP*)

```
genr MARL06 = MARRIED*CHILDL06
genr LGEARN = ln(EARNINGS)
smpl MALE = 0 --restrict
heckit LGEARN const S ASVABC ETHBLACK
ETHHISP ; WORKING const S ASVABC ETHBLACK
ETHHISP AGE CHILDL06 CHILDL16 MARRIED MARL06
```

The data set is `lfp.gdt`. Compare the regression results with those in Table 10.8 in the textbook. The basic regression specification is the same. The only difference is in the specification of the selection model. Note that `WORKING` is a dummy variable that indicates that the individual has been selected into the sample. It is not an explanatory variable. It has the same function as `D` in Exercise 10.11.

Exercise 11.2 Construction of a relative price index (*DF*)

In the commands in this and the following exercises, *CAT* is a place holder and should be replaced by the three or four letter name of your category of expenditure.

```
open path\demand.gdt

genr PRELCAT = 100*PCAT/PTPE

plot PRELCAT
```

Thus, if your category were FLOW, the second and third commands would be

```
genr PRELFLOW = 100*PFLOW/PTPE

plot PRELFLOW

store path\demand.gdt
```

The plot is part of the output file and in ASCII format, so somewhat unsophisticated. Better plots can be obtained using the gnuplot facility bundled with gretl. I do not know how to use it from the command line, but it can be accessed easily through the GUI via the View/Graph specified vars tab on the main menu.

You will need PREL*CAT* in many of the following exercises, so save the data set once it has been defined, overwriting the existing one.

Exercise 11.3 Linear regression (*DF*)

```
ols CAT const DPI PRELCAT
```

Exercise 11.4 Logarithmic multiple regression (*DF*)

```
genr LGDPI = log(DPI)

genr LGCAT = log(CAT)

genr LGPRCAT = log(PRELCAT)

ols LGCAT const LGDPI LGPRCAT

store path\demand.gdt
```

Exercise 11.5 Multicollinearity (*DF*)

```
ols LGCAT const LGDPI LGPRCAT TIME

corr LGCAT LGDPI LGPRCAT TIME
```

Exercise 11.9 Naive attempts to model dynamics (*DF*)

The lags command creates lagged versions of existing variables. The command is followed by the number of lags desired, then a semicolon, and then a list of the variables in question. Thus LGDPI_1 is LGDPI lagged one time period (in this case, one year), and LGDPI_2 is LGDPI lagged two time periods.

```
lags 2 ; LGCAT LGDPI LGPRCAT
```

```
ols LGCAT_1 const LGDPI_1 LGPRCAT_1
ols LGCAT_2 const LGDPI_2 LGPRCAT_2
```

Exercise 11.11 Simple dynamics: a partial adjustment model (*DF*)

If the lags command is followed by a variable name or names, it is assumed that only the first lag is required.

```
lags CAT
ols CAT const DPI PRELCAT CAT_1
ols LGCAT const LGDPI LGPRCAT LGCAT_1
```

Exercise 12.3 Tests for autocorrelation (*DF*)

When it was defined, the Demand Functions data set was declared to be a time series set with annual data from 1959 to 2003. This accounts for the inclusion of the Durbin–Watson *d* statistic among the diagnostic statistics. The last three commands below are for the Breusch–Godfrey test. Remember that `$uhat` is a temporary variable containing the residuals from the most recent regression.

```
ols LGCAT const LGDPI LGPRCAT
genr ECAT = $uhat
lags ECAT
ols ECAT const LGDPI LGPRCAT ECAT_1
```

Exercise 12.4 Plot of residuals (*DF*)

See Exercise 11.2 for remarks about plots.

```
ols LGCAT const LGDPI LGPRCAT
genr ECAT = $uhat
plot ECAT
```

Exercise 12.5 Comparison of OLS and AR(1) specifications (*DF*)

ar1 implements the Cochrane–Orcutt iterative procedure for fitting a regression assumed to be subject to AR(1) autocorrelation. Adding `--pwe` at the end changes the estimation method to the Prais–Winsten variation. This retains the use of the first observation, which is lost under C–O, and may lead to a gain in efficiency.

```
ols LGCAT const LGDPI LGPRCAT
ar1 LGCAT const LGDPI LGPRCAT
ar1 LGCAT const LGDPI LGPRCAT --pwe
```

Exercise 12.8 Common factor test (*DF*)

The third, fourth, and fifth commands are for performing the Breusch–Godfrey test.

```
ar1 LGCAT const LGDPI LGPRCAT
ols LGCAT const LGCAT_1 LGDPI LGDPI_1 LGPRCAT LGPRCAT_1
genr ECAT = $uhat
lags ECAT
ols ECAT const LGCAT_1 LGDPI LGDPI_1 LGPRCAT LGPRCAT_1 ECAT_1
ols LGCAT const LGCAT_1 LGDPI LGPRCAT
```

Exercise 13.8 Tests for unit roots (*DF*)

The `diff` command creates new variables defined as the first differences of the variables listed. The names are the names of the variables preceded by `d_`. The `adf` command produces the augmented Dickey–Fuller test statistic for the variable listed. The number after `adf` gives the number of lagged differences to be included in the regression specification. Thus `adf 1` presupposes a model of the type given by equation (13.70) in the text with the addition of a deterministic time trend. Options include `--nc` (no constant, no time trend), `--c` (constant, no time trend), and `--ct` (both constant and time trend). Adding `--verbose` produces the full output for the regression.

```
diff LGCAT LGPRCAT
adf 1 LGCAT --ct
adf 1 d_LGCAT --ct
adf 1 LGPRCAT --ct
adf 1 d_LGPRCAT --ct
```

Exercise 13.12 Test for cointegration (*DF*)

Remember that `$uhat` is a temporary variable that contains the residuals from the most recent regression.

```
ols LGCAT const LGDPI LGPRCAT
genr ECAT = $uhat
adf 1 ECAT --nc
```

Exercise 13.14 Error correction model (*DF*)

```
diff LGDPI
lags 1 ECAT
ols d_LGCAT ECAT_1 d_LGDPI d_LGPRCAT
```

Exercise 14.1 Pooled OLS regression (*OECD2000*)

```
open path\OECD2000.gdt
ols E const G TIME2 TIME3
```

Exercise 14.2 (*OECD2000*)

There are various ways of indicating to gretl that a data set has the dual structure required for panel regressions. One, used here, is to specify the unit indicator and the time indicator, followed by `--panel-vars`. Here the units are countries and `ID` is the country indicator. The time indicator is simply `TIME` itself. The panel command fits a within-groups fixed effects model by default. It automatically includes a test for the presence of fixed effects, the null hypothesis being that they do not exist and that a pooled OLS regression is adequate (and preferable, being more efficient because it does not involve the estimation of unnecessary dummy variable coefficients). The `panel` command does this by performing an *F* test of the joint explanatory power of the dummy variables in the least squares dummy variables (LSDV) version of the fixed effects model, the specification with an intercept and no dummy for one of the units of observation. It compares *RSS* for the LSDV model with *RSS* for the pooled OLS regression. At least this is what is **says** it is doing. It does not actually have to fit the LSDV model itself because the within-groups regression is equivalent.

```
setobs ID TIME --panel-vars
panel E const G TIME2 TIME3
```

Exercise 14.3 (*OECD2000*)

`D*` is interpreted by gretl as shorthand for a list of all the variables in the data set starting with D and is a convenient way of including all the country dummies in the specification. The `delete` command is supposed to cause the listed variable(s) to be dropped from the data set. If the specification includes an intercept, it is necessary to drop one of the dummies. However I have not succeeded in getting this command to work. So in the second regression, `DAUS` still exists and the specification is subject to exact multicollinearity. To avoid this, gretl drops the final dummy variable (that for the USA).

```
ols E G TIME2 TIME3 D*

delete DAUS

ols E const G TIME2 TIME3 D*
```

Exercise 14.4 (*OECD2000*)

Use *RSS* from Exercise 14.1 and second regression in 14.3

Exercise 14.5 (*NLSY2000*)

```
open path\NLSY2000.gdt

setobs ID TIME --panel-vars

genr LGEARN = log(EARNINGS)

ols LGEARN const S ASVABC EXP MALE ETHBLACK ETHHISP

panel LGEARN const S ASVABC EXP MALE ETHBLACK ETHHISP

store path\NLSY2000.gdt
```

Exercise 14.6 (*NLSY2000*)

The `--random-effects` switch indicates to gretl that the panel regression should be random effects, rather than the default within-groups fixed effects. After performing a random effects regression, gretl automatically compares the results with those of pooled OLS and fixed effects regressions. It performs a Breusch–Pagan test to test for the presence of random effects, the null hypothesis being that pooled OLS is adequate. It performs a Hausman test to discriminate between the presence of random and fixed effect, the null hypotheses being random effects.

```
ols LGEARN const S ASVABC EXP MALE ETHBLACK ETHHISP UNION

panel LGEARN const S ASVABC EXP MALE ETHBLACK ETHHISP UNION --random-
effects
```

Exercise 14.7 (*NLSY2000*)

If a data set has been declared to be panel data,, placing the hausman command after a pooled OLS regression will automatically cause the corresponding fixed and random effects models also to be fitted and the three tests already described to be performed: the F test discriminating between pooled OLS and fixed effects (see Exercise 14.2), the Breusch–Pagan test discriminating between pooled OLS and random effects, and the Hausman test itself discriminating between fixed effects and random effects.

```
ols LGEARN const S ASVABC EXP MALE ETHBLACK ETHHISP
hausman
panel LGEARN const S ASVABC EXP MALE ETHBLACK ETHHISP
panel LGEARN const S ASVABC EXP MALE ETHBLACK ETHHISP --random-effects
```

Exercise 14.8 (*NLSY2000*)

```
ols LGEARN const S ASVABC EXP MALE ETHBLACK ETHHISP UNION
hausman
panel LGEARN const S ASVABC EXP MALE ETHBLACK ETHHISP UNION
panel LGEARN const S ASVABC EXP MALE ETHBLACK ETHHISP UNION --random-
effects
```