Practical Empirical Research Using Gretl

Artur Tarassow[*]
Department of Socioeconomics, University of Hamburg

August 27, 2018

Abstract

This paper examines

Keywords: XXX, XXX
JEL: XXX; XXX

[*]The author is grateful to XXX for helpful comments. Any remaining errors or omissions are strictly our own.

1 Introduction

Data analysis is a vital part of the economic training and of an economist's daily life. Economists are trained to be sophisticated users of data, and econometrics — the interchange of economic theory, statistics and mathematics — is a subject which usually all economics students have to attend. Undergraduates studying economics are typically exposed to at least once course in statistics and econometrics, covering the practice and interpretation of least-squares regression, data management and data visualisation. Graduate students learn the advanced treatments of the subject, likely involving Maximum Likelihood estimation, simulation and so on. Professional economists — whether in public or private institutions — are likely to find themselves using, or possibly developing, econometric methods.

Gretl is an open-source and freely available statistics and econometrics software attempting to bridge the different demands for teaching at different levels as well as professional work. It comprises a full-featured graphical interface but also a powerful scripting language called Hansl. Gretl's scripting language can be seen as a domain-specific language for statistics and econometrics. It handles datasets — a matrix but with a richer structure including information on the data type, eventual calendar date and recorded data frequency — similar to Eviews but can also deal with 'pure' matrices such as Matlab, Gauss and Julia for advanced programming purposes.

In the following, we will introduce Gretl — and to some extend Hansl — alongside a practical empirical example using time-series data. Nevertheless, it should be said that Gretl is also covers methods and estimators for cross-sectional as well as panel data structures.

The paper is structured as follows. The next section discusses the market of econometrics software and delivers some information on Gretl and its scripting language Hansl. Section 3 introduces Gretl graphical user interface and explains how to load, append and transform data and provides an overview about basic commands for analysing and plotting data. In Section 4 a dynamic time-series model will be estimated and it will be shown how to analyse such a model using Gretl. Section 5 briefly provides an introduction on how to work with matrices and how variables from the dataset can interact with the matrix-world before Section 6 summarises.

2 Econometric Software and Gretl

2.1 The 'Market' of econometrics software

The requirements for an econometric software are a delicate issue. For teaching purpose it makes sense to have undergraduate students work with reasonably user-friendly software — in the easiest case steering the software by a graphical user interface through point-and-click using the mouse. However, it makes little sense to introduce a software to undergraduates which does not support more advanced methods and does not offer a fully-fledged scripting and programming environment — aspects which are required for an advanced treatment in graduate classes and for professionals. And of course there's a premium on teaching "marketable skills" rather than dead-end expertise.

Gretl is a program which attempts to bridge these different demands. It comprises a full-featured graphical interface (the GUI): its underlying functionality (coded in C) can be driven either by *hansl* (Gretl's scripting language) or by the apparatus of menus, dialog boxes and so on. The developers try to ensure that almost everything that can be done via hansl can also be done via the GUI, and vice versa, with only few exceptions (Cottrell, 2017).

Gretl's main competitors are the major proprietary econometrics packages, Stata and Eviews, and also the major open-source statistical software project, R. As in gretl, datasets and series are also basic in Stata and Eviews, and econometric functionality is supported by a wide range of built-in commands. Even though both Stata and Eviews support scripting, their respective languages are quite odd from the point of view of a programmer used to general-purpose scripting languages or Matlab-like interfaces for matrix manipulation. One cannot define a function as such in either Stata or Eviews. Gretl, however, also offers the common apparatus of fully-fledged programming languages (function-calling, function-definition, declaration of and assignment to named variables of various types) similar of what is known from Matlab.

Another major plus of Gretl is its built-in capability to communicate with other software packages such as Stata, Python, R, Julia, Ox and Octave. The *foreign*-block is an interface to include statements of other languages into a hansl script allowing to send and receive information (in the form of matrices). Thus, users can call and execute functions available in other software packages which are, for instance, currently not available in Gretl. For details see the Appendix A.9.

2.2 Gretl and Hansl

Gretl is acronym for **G**nu **R**egression, **E**conometrics and **T**ime-series **L**ibrary. The software is available for Windows, Mac OS X as well as Linux through the official Gretl homepage: `http://gretl.sourceforge.net/`. It is free, open-source software which may be redistributed and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation.

Gretl comprises a large shared library, a common command line (CLI) program and a GUI client, and makes use of reliable free software packages, e.g. (multi-threaded) LAPACK/BLAS, fftw, GTK, gnuplot, etc. The first Gretl version was released in January 2000 and has been under active development, mainly using the C programming language, since then. Its user interface is available in 17 languages thanks to an active community. Gretl is shipped with a User Guide of already 380+ pages, a Command Reference of 210+ pages, and a "A Hansl Primer" tutorial to Gretl's scripting language.

Gretl comprises a full-featured graphical user interface for steering tasks through clicking. However, apart from the GUI, functions can also be driven either by scripting. As the unique selling point, Gretl offers a high-level matrix oriented language with a syntax similar to Matlab or Gauss, *and* an intuitive high-level scripting language that is attuned to econometrics. This makes Gretl especially interesting for lecturing statistics and econometrics but also for serious research.

Hansl — "**H**ansl's **A** **N**eat **S**cripting **L**anguage" — was developed over time and has become a very advanced high-level scripting language with more than 140 commands specifically tailored to the domain of econometrics to date. The lead developers, Allin Cottrell and Riccardo "Jack" Lucchetti, have pushed the progress of Gretl mainly through Hansl in recent times. Users and developers are asked to program and publish user-contributed function packages by means of Hansl rather than the low-level language C. A great plus of Hansl-based function packages is that it comes with a handy optional GUI integration making it possible for developers to add access to specific functions through the GUI rather than solely through commands. For more technical details on Hansl, we refer to Cottrell (2017).

To date Gretl *natively* implements a variety of models, methods and functions among others such as:

1. Time series data: ARIMA, GARCH-type, (S)VARs and VECMs, unit root and cointegration tests, Kalman filter, Mixed time-series frequencies (MIDAS), etc.

2. Limited dependent variables: Logit, Probit, Tobit, Interval regression, Models for count and duration data, etc.

3. Panel-data: Instrumental variables, Probit and GMM-based panel models, etc.

4. Output models as RTF or LaTeX files, in tabular or equation format.

5. Using gnuplot for compiling high-quality graphs.

6. Support for machine learning via the LIBSVM library.

Besides Gretl's core functionality, several addons (`http://gretl.sourceforge.net/addons-data/addons.xml`) and numerous contributed function packages (`http://ricardo.ecn.wfu.edu/gretl/cgi-bin/gretldata.cgi?opt=SHOW_FUNCS`) are hosted online and are easily accessible through Gretl.

3 Getting Started in Gretl

3.1 Gretl's Graphical User Interface

There are several ways to work in Gretl. For starters, the most intuitive way is to use the program through its built-in graphical user interface (GUI). The GUI is shown in Figure 1 and consists of four separate windows. At the top of the main window (see Figure 1(a)) you find the menu bar comprising commands to import and manipulate data, analyse data, and manage output. Below the menu bar, the user finds information on the loaded dataset and the working directory currently selected. Most space is given to the variables list where each variable is associated with a unique ID number, the variable's name and an optional descriptive label. Below the variables list, information on the loaded dataset is given such as the total number of observations or the number (and eventual time-span) of the currently selected sub-sample, respectively. The toolbar containing a number of useful utilities can be found at the bottom of the window.

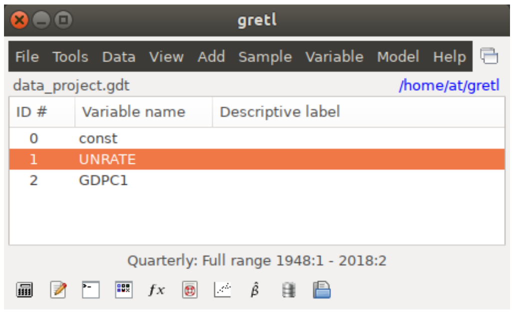
The console window, as depicted in Figure 1(b) accepts Gretl commands which can be directly executed pressing "Enter". The output will immediately appear in the console. From the command prompt, '?', you can type in commands from the gretl language or calculate something. Try for instance,

```
? eval sqrt(25)/5
```

The `eval` command evaluates an expression and prints its value.

The icon view window (see Figure 1(c)) comprises icons for eventually generated scalar values, matrices and model objects among others — more on this below. Lastly, the script editor, as depicted in Figure 1(d), collects several lines of programming code into a file which can be executed all at once in a script. Serious and reproducible research relies on such scripts for recording each step of the data analysis.

Lee Adkin's (2014) terrific and freely available econometrics Ebook using Gretl for the applications provides a much deeper introduction to Gretl, and is highly recommended for learning and mastering Gretl (4th edition downloadable here:
`http://www.learneconometrics.com/gretl/using_gretl_for_POE4.pdf`; the 5th edition

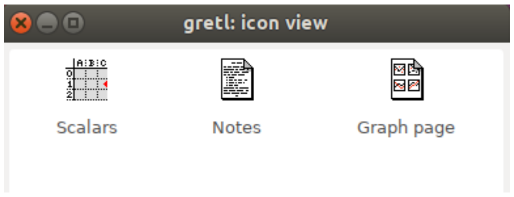

gretl
File Tools Data View Add Sample Variable Model Help
data_project.gdt
/home/at/gretl
Quarterly: Full range 1948:1 - 2018:2

ID #	Variable name	Descriptive label
0	const	
1	UNRATE	
2	GDPC1	

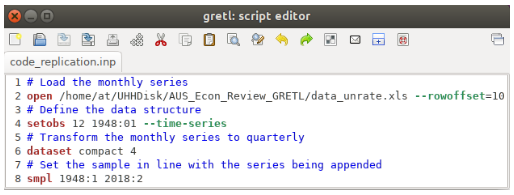
(a) Main Window

```
gretl console: type 'help' for a list of commands
?
```

(b) Console Window



(c) Icon View Window



```
# Load the monthly series
open /home/at/UHHDisk/AUS_Econ_Review_GRETL/data_unrate.xls --rowoffset=10
# Define the data structure
setobs 12 1948:01 --time-series
# Transform the monthly series to quarterly
dataset compact 4
# Set the sample in line with the series being appended
smpl 1948:1 2018:2
```

(d) Script Editor Window

Figure 1: The Gretl Interface

will be published very soon!). Also the manual, which can be opened from the "Help, User's guide" menu, is the first source you should check in case of questions or problems.

3.2 Loading and Appending Data

Gretl can import data from a variety of formats from the local hard-disk and from the internet. In the GUI program this can be done via the "File, Open Data, User file" menu. In script mode, simply use the `open` command. The supported import formats include (i) plain text files (typically comma-separated or "CSV"), (ii) spreadsheet formats as MS Excel, Gnumeric and Open Document, (iii) Stata data files (.dta), (iv) SPSS data files (.sav) and (v) Eviews workfiles (.wf1) among some others.

The following conditions must be satisfied to make sure Gretl reads data successfully:

1. The first row must contain valid variable names, otherwise the program will automatically add names, v1, v2 and so on.

2. Data values should constitute a rectangular block, with one variable per column (and one observation per row).

3. Numeric data are expected, but in the case of importing from plain text, the program offers handling of character (string) data.

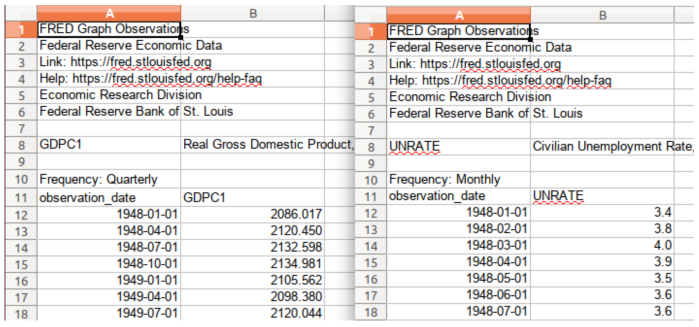
4. Optionally, the first column may contain strings such as dates, or labels for cross-sectional observations

One can append additional data either by the `append` command or via the "File, Append data" menu: Gretl will check the new data for conformability with the existing dataset and, if everything seems OK, will merge the data. For more complex tasks, there exists the powerful `join` command (execute `help join` in the console for details).

Once data is read-in by Gretl, it may be necessary to supply some information on the type of the dataset. The three types that can be distinguished are cross-section, time-series and panel data. The primary tool for doing this is the "Data, Dataset structure" menu entry in the GUI, or the `setobs` command when using scripts or the console.

In our real-world example, we deal with two macroeconomic time-series downloaded from the St, Louis Federal Reserve Bank database called FRED (`https://fred.stlouisfed.org/`). The first series is real GDP of the U.S. economy for which data values are collected at a quarterly frequency since 1948 (`https://fred.stlouisfed.org/series/GDPC`). The second series refers to the civilian unemployment rate observed at monthly frequency since 1948 (`https://fred.stlouisfed.org/series/UNRATE`).

The two separate spreadsheets look as depicted in Figure 2 with basic information on data characteristics reported in Table 1. The two files are stored online.



	A	B
1	FRED Graph Observations	
2	Federal Reserve Economic Data	
3	Link: https://fred.stlouisfed.org	
4	Help: https://fred.stlouisfed.org/help-faq	
5	Economic Research Division	
6	Federal Reserve Bank of St. Louis	
7		
8	GDPC1	Real Gross Domestic Product,
9		
10	Frequency: Quarterly	
11	observation_date	GDPC1
12	1948-01-01	2086.017
13	1948-04-01	2120.450
14	1948-07-01	2132.598
15	1948-10-01	2134.981
16	1949-01-01	2105.562
17	1949-04-01	2098.380
18	1949-07-01	2120.044

	A	B
1	FRED Graph Observations	
2	Federal Reserve Economic Data	
3	Link: https://fred.stlouisfed.org	
4	Help: https://fred.stlouisfed.org/help-faq	
5	Economic Research Division	
6	Federal Reserve Bank of St. Louis	
7		
8	UNRATE	Civilian Unemployment Rate,
9		
10	Frequency: Monthly	
11	observation_date	UNRATE
12	1948-01-01	3.4
13	1948-02-01	3.8
14	1948-03-01	4.0
15	1948-04-01	3.9
16	1948-05-01	3.5
17	1948-06-01	3.6
18	1948-07-01	3.6

Figure 2: Raw Data of Real GDP and Unemployment Rate for the U.S., Spreadsheet

Working with these two series involves at least two problems. First, the frequency

Series	Frequency	Starting Date	Ending Date
GDPC1	quarterly	1948Q1	2018Q2
UNRATE	monthly	1948M1	2018M7

Table 1: Data Information of Raw Data

among the variables differs which requires some method to equalize it. Second, both series are stored in separate Excel files and need being joined. Both tasks can easily be solved.

Before proceeding, you should download both Excel-files and save them in a folder:

- data_unrate.xls: `https://scm.darnold.org/atecon/gretl_aer/raw/master/data_unrate.xls`
- data_gdp.xls: `https://scm.darnold.org/atecon/gretl_aer/raw/master/data_gdp.xls`

First, we load the monthly UNRATE series stored in the "data_unrate.xls" file via the "File, Open data, User file" menu which opens the "spreadsheed import" window (see Figure 3). As the name of the first variable ('observation_date') has its entry in the spreadsheet in column A and row 11, we start the data import of the rectangular block at this position as depicted in Figure 3.

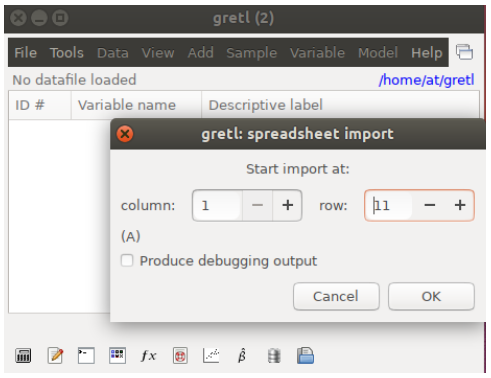


Figure 3: Spreadsheet Import Window

In case Gretl has not successfully recognized the time-series structure given some observation column (here 'observation_date'), the user can manually determine the frequency and starting date by the menus automatically showing up in the process. Alternative we can set the data structure by the following command through the console

```
setobs 12 1948:1 --time-series
```

where '12' refers to the number of observations per year ('12' for monthly data) followed by the initial observation date and an option saying that the data has a time series structure. The variables which appear in the main window are of the so called 'series' type.

As real GDP is only quarterly observed, we need to transform the monthly UNRATE series to quarterly observations. A standard way to do this is to compute the average of three consecutive months of a specific quarter. This is easily done via the "Data, Compact data" menu or the command

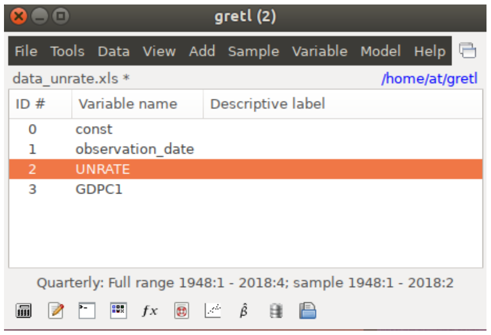
```
dataset compact 4
```

telling Gretl to transform the dataset from a monthly to a quarterly frequency. The resulting quarterly time-series will range from 1948Q1 to 2018Q2.

The next step involves appending the quarterly real GDP series from another source to the existing dataset. However, we need to make sure that the currently active dataset covers the same time span as the one wished to append. Thus, to avoid error, we restrict the currently selected sample for the period between 1948Q1 to 2018Q2 by the command

```
smpl 1948:1 2018:2
```

Appending a file can be done by using the "File, Append data" menu followed by the same request of the 'spreadsheet import' window as already shown in Figure 3. Gretl automatically recognizes that the appended dataset has the same length as the active one and will add the real GDP series to the dataset. This will finally result in Figure 4.



data_unrate.xls * /home/at/gretl

ID #	Variable name	Descriptive label
0	const	
1	observation_date	
2	UNRATE	
3	GDPC1	

Quarterly: Full range 1948:1 - 2018:4; sample 1948:1 - 2018:2

Figure 4: Appended Time-Series

Instead of using the GUI, all previous steps from data loading, frequency reduction and appending can be done using a collection of Gretl commands as shown in the Appendix A.2.

It should be said, that Gretl natively includes many example datasets, some of them taken from popular econometrics textbooks, which is especially useful for teaching purposes. These can be accessed through the "File, Open data, Sample file" menus. Additionally, since the latest stable Gretl version (2018b) available, users can easily download data from the DB.NOMICS database (`https://next.nomics.world/`). See for a *how-to* the following video on youtube: `https://youtu.be/B2KmX9A6ICg`. Also there exist user-written packages for downloading data from the `quandl.com` (see the `getQuandl` package: `http://ricardo.ecn.wfu.edu/gretl/cgi-bin/current_fnfiles/getQuandl.gfn`), and `yahoo.com` (see the `yahoo_get` package: `http://ricardo.ecn.wfu.edu/gretl/cgi-bin/current_fnfiles/yahoo_get.gfn`) databases.

3.3 Basic Commands

The purpose of this section is to introduce some basic commands which are frequently executed when doing data analysis. In the following, we proceed with the dataset compiled in the previous step. Once the data is loaded, one should check it for validity. Initially, we will generate a so called list which can (and should!) be used to make command scripts less verbose and repetitious, and more easily modifiable. In the following, a list named `L` comprises both variables of interest namely `UNRATE` and `GDPC1`:

```
list L = UNRATE GDPC1
```

For printing the values for each variable in a list by their respective observations, we use the `print` command with the `--byobs` option:

```
print L --byobs
```

The corresponding output for the first five observations looks like:

```
             UNRATE        GDPC1

1948:1      3.73333      2086.02
1948:2      3.66667      2120.45
1948:3      3.76667      2132.60
1948:4      3.83333      2134.98
1949:1      4.66667      2105.56
```

One can easily compute basic descriptive statistics for elements in a list by means of the `summary` (execute `help summary` in the console for obtaining additional options) command

```
summary L
```

which returns various statistics as well as information on eventual missing values:

```
                   Mean           Median         Minimum        Maximum
UNRATE             5.7771         5.5667          2.5667         10.667
GDPC1              8557.0         7013.6          2086.0          18507

                   Std. Dev.     C.V.          Skewness   Ex. kurtosis
UNRATE             1.6318        0.28245        0.62860       0.096705
GDPC1              4956.1        0.57919        0.44045        -1.1495

                   5% perc.      95% perc.        IQ range   Missing obs.
UNRATE             3.5050         9.0617        2.2083              0
GDPC1              2465.1          17234        9046.0              0
```

In case one wants to compute these descriptive statistics using only observations between 1980Q1 and 2007Q4 one simply needs to restrict the sample accordingly before calling the summary command:

```
smpl 1980:1 2007:4
summary L
```

One can also restrict the underlying sample in a more complicated way. For instance, running the command

```
smpl UNRATE>4 && UNRATE <= 9.6 --restrict --replace
```

would restrict the selected dataset to observations which fulfil the criteria that `UNRATE` is greater 4 but less or equal to 9.6, and replace any eventual previous restrictions imposed by the `--replace` option.[1] The user will be informed that the currently selected sample includes only 237 out of all 282 valid observations, as depicted in Figure 5. The information that the currently selected sample is "undated" states that due to the restrictions imposed, the time-series structure of consecutive observations has been broken. The original full sample with a time-series structure can be restored by the `smpl full` command.

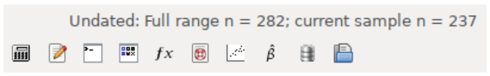

Undated: Full range n = 282; current sample n = 237

Figure 5: Sample Selection Information After Imposed Restriction

[1] An overview about logical operators can be found in Table A1 in the Appendix.

Sometimes one has to manipulate data entries due to wrong or missing values recorded. Let us assume that variable `UNRATE` includes a missing value for observation 1981Q2 but you know that the U.S. unemployment rate has been 7.4 % for this observation. Replacing a valid value or a missing value for a specific observation can be simply done by the command

```
UNRATE[1981:2] = 7.4
```

A basic statistics for a bivariate analysis is the correlation coefficient. The correlation coefficient can be easily computed for all pairs of variables in a list by means of the command

```
corr L
```

which computes Pearson's product-moment correlation. Other types of correlation such as Spearman's rho (using the option `--spearman`) and Kendall's tau (using `--kendall`) are also available. The correlation between $UNRATE_t$ and $GDPC1_t$ — where t is a subscript referring to the t-th observation for $t = 1, \ldots, T$ — is close to zero and is not significantly different from zero in this example:

```
corr(UNRATE, GDPC1) = -0.01416004
Under the null hypothesis of no correlation:
t(280) = -0.236967, with two-tailed p-value 0.8129
```

An overview about some of the available descriptive statistics are listed in Table 2.

Command	Description
`min`	Prints the minimum of the series
`max`	Prints the maximum of the series
`mean`	Prints the mean of the series
`median(X)`	Prints the median of the series X
`summary`	Prints summary statistics of the series
`xtab`	Displays a contingency table or cross-tabulation for series
`freq`	Prints/ Plots the frequency distribution of the series
`corr`	Prints the pairwise correlation coefficients for series
`xcorrgm`	Prints and graphs the cross-correlogram between two series
`pergm`	Computes and displays the spectrum of the series

Table 2: Examples of Descriptive Statistics in Gretl

3.4 Plotting

Visualising data marks an important step in empirical work. Gretl generates graphs by means of the reliable open-source tool gnuplot which is available for many platforms. Gnuplot is a very full-featured graphing program with myriad options (for details see ch. 6 in the gretl manual). Gretl gives you direct access, via a graphical interface, to a subset of gnuplot's options and it tries to choose sensible values for you; it also allows you to take complete control over graph details if you wish.

In a time-series context, an obvious informative graph is the time-series plot which depicts data over time. This is easily done in Gretl by (the backslash allows for line breaks)

```
gnuplot L --with-lines=GDPC1 --with-impulses=UNRATE \
--time-series --output=display
```

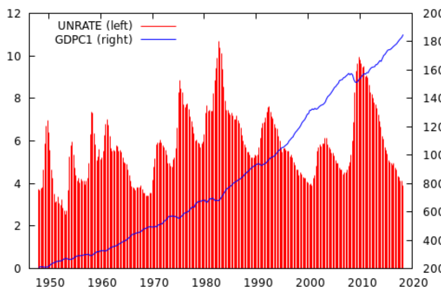
and the graph can be seen in Figure 6(a). We ask `gnuplot` to plot the variables in list `L` over time using the `--time-series` option. Furthermore, we want series GDPC1 being drawn as a line but UNRATE as impulses in this example. The option `--output=display` returns immediately a plot onto the screen.

Also scatter-plots serve as standard tools for visualising relationships between variables. A simple scatter-plot between the two series included in list `L` — augmented by a linear OLS-based regression fit — is called by (see Figure 6(b)):

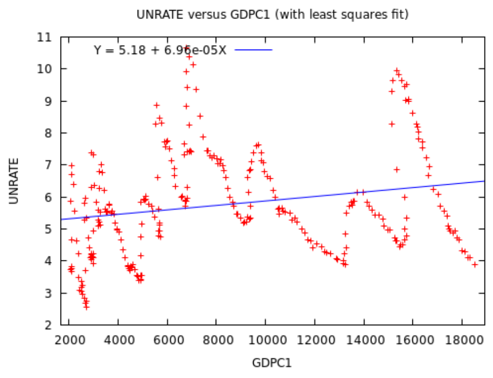
```
gnuplot L --fit=linear --output=display
```

The option `--output=filename` controls the filename used, and at the same time allows you to specify a particular output format. Supported output formats are *.eps* (Encapsulated PostScript), *.pdf* (PDF) and *.png* (PNG) among others. For instance, for storing a figure in png-format in a specific directory just used the expression:

```
--output="C:\Project\Figures\Fig1.png"
```



(a) Time-Series Plot



(b) Scatter-Plot

Figure 6: Time-Series and Scatter-Plot using Gretl

Another useful type of plot is the Boxplot which summarizes a number of descriptive statistics for the researcher. The `boxplot` command produces this type of graph. Before showing an example, however, let us generate a new series named `GreatModeration` which shall be a binary dummy variable taking the value of one for observations between 1985Q1 and 2007Q4, and otherwise zero:

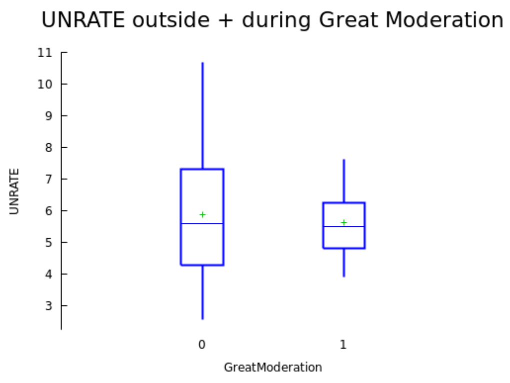
```
series GreatModeration = (obs>="1985:1" && obs<="2007:4")
```

The ones indicate the well-known Great Moderation period during which business cycle indicators such as GDP growth and price inflation rates followed a rather stable path associated with only small variance around some mean value.

Next, we draw the boxplot for the `UNRATE` series and examine its distribution conditional on the value of the dummy variable `GreatModeration` by using the `--factorized` option:

```
boxplot UNRATE GreatModeration --factorized --output=display \
{ set title 'Distribution of UNRATE outside and during Great Moderation' ; }
```

The graph is depicted in Figure 7 showing that the variance of the unemployment rate has indeed been smaller during the Great Moderation in the U.S. Also, note that we added a self-explanatory title to the graph.



(a) Boxplot

Figure 7: Factorized Boxplot using Gretl

4 Let's estimate a dynamic time-series model

In this section we want to illustrate how to estimate a dynamic single-equation time-series model using Gretl. We will to examine the dynamic relationship between the change in the unemployment rate and growth in aggregate output (as proxied by the Gross Domestic Product). The famous macroeconomist Arthur Melvin Okun formulated a negative relationship between these two variables which has become known as 'Okun's Law'.

Before estimating the regression model, we need to generate the two variables of interest. The following lines generate two new series objects where `dU` refers to the period-change in the unemployment rate, $dU_t = U_t - U_{t-1}$, and `gY` refers to the (exact) period-growth rate of real GDP (in percent), $gY_t = 100 \times \left(\frac{Y_t - Y_{t-1}}{Y_{t-1}}\right) = 100 \times \left(\frac{Y_t}{Y_{t-1}} - 1\right)$:

```
series dU = UNRATE - UNRATE(-1)
series gY = 100 * (GDPC1/GDPC1(-1) - 1)
```

As can be seen, in a time-series (but also panel) context, Gretl easily handles lags and leads by using an expression such as `x(p)` where p generates the p-th lag of series `x` if p takes a negative and the p-th lead if it takes a positive integer value.

For illustration, we estimate the following linear autoregressive distributed lag (ARDL(p,q)) model

$$dU_t = \beta_0 + \sum_{i=1}^{p} \beta_i dU_{t-i} + \sum_{j=0}^{q} \theta_j gY_{t-j} + u_t \ , t = 1, \ldots, T \tag{1}$$

where dU_t is a function of its own p lagged values, and the contemporary as well as lagged values up to order q of growth of output, gY_t. Scalar β_0 refers to the intercept, and the error-term u_t follows a white-noise zero mean process with constant variance $u_t \sim WN(u, \sigma_u^2)$.

The objective is to estimate the unknown parameters β, θ and σ_u^2. Assuming, that gY_t is indeed exogenous w.r.t. dU_t meaning that gY_t 'causes' contemporaneously dU_t but not *vice versa*, we can estimate (under stationarity of both series) eq. (1) by OLS.

For simplicity, we will base the estimation on a specific sub-sample ranging from 1985Q1 to 2007Q4 comprising in total $T = 92$ valid observations as this sample is associated with parameter stability. As a consequence of two oil crises in the mid and late 1970s as well as the recent Great Financial Crisis, Okun's Law has been associated with changed model coefficients and heteroskedasticity — so called structural breaks which violate some of the standard Gauss Markov assumptions required for an unbiased and efficient OLS estimator. The sample restriction is called by the command:

```
smpl 1985:1 2007:4
```

We restrict the lag-orders p and q to be equal — which of course doesn't actually have to be but simplifies things. The unknown optimal finite lag-order is usually determined by means of information criteria (see every good introductory textbook on time-series analysis). We use Gretl's built-in `var` command with the option `--lagselect` to determine the optimal lag order by setting the maximum lag order to test twice the data frequency (=8 for quarterly data). The command and the corresponding output are:

```
var 8 dU gY --lagselect

lags    loglik    p(LR)       AIC          BIC          HQC

1      -7.70739              0.297987     0.462451*    0.364366
2      -0.56199  0.00642     0.229608*    0.503716     0.340241*
3       1.35475  0.42901     0.274897     0.658647     0.429782
4       3.11277  0.47544     0.323635     0.817029     0.522773
5       4.19395  0.70592     0.387088     0.990124     0.630479
6       6.09821  0.43254     0.432648     1.145327     0.720291
7       8.02423  0.42640     0.477734     1.300056     0.809630
8      14.89473  0.00817     0.415332     1.347297     0.791481
```

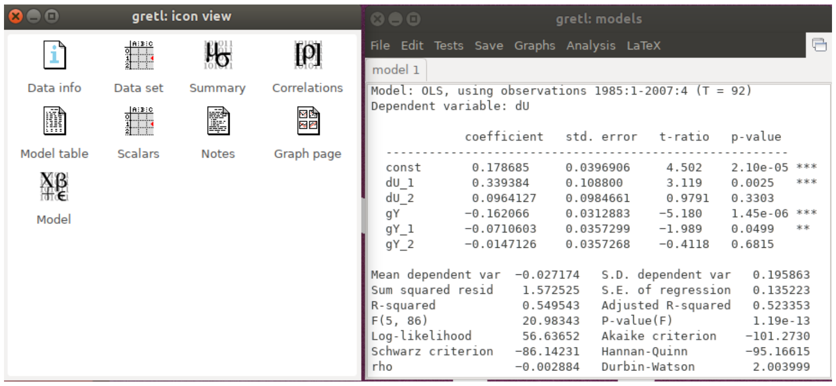
While the Akaike (AIC) and Hannan–Quinn (HQC) information criteria select the optimal lag-order to be $p^* = q^* = 2$ — resulting in a ARDL(2,2) model — the Schwarz (BIC) criteria selects a parsimonious ARDL(1,1) model. We stick to the ARDL(2,2) in the following example but the reader is invited to re-run the analysis for a ARDL(1,1) model.

Estimating the ARDL(2,2) model by means of OLS can be done by

```
Model <- ols dU const dU(-1 to -2) gY(0 to -2)
# equal to: Model <- ols dU const dU(-1) dU(-2) gY gY(-1) gY(-2)
```

which illustrates the ease of handling leads and lags in Gretl. The `ols` command is followed by the name of the endogenous variable and a set (or list) of exogenous series. The intercept is not included per default but can be added by the `const` series (the built-in identifier for a constant or y-intercept).

Note the expression `Model <-` preceding the actual `ols` command. This expression generates a so called model object (here named "Model") which instantly appears in the icon view. A double-click on the "Model"-icon opens the "models"-window (see Figure 8) where the user can proceed with a GUI for executing various tests, (bootstrap-based) inference, change the model specification, plot several estimation-related graphs such as residuals and fitted values etc. This shows the nice implementation of graphical objects which are especially useful for beginners and teaching.



```
Model: OLS, using observations 1985:1-2007:4 (T = 92)
Dependent variable: dU

             coefficient   std. error   t-ratio   p-value
  --------------------------------------------------------
  const       0.178685     0.0396906     4.502    2.10e-05 ***
  dU_1        0.339384     0.108800      3.119    0.0025   ***
  dU_2        0.0964127    0.0984661     0.9791   0.3303
  gY         -0.162066     0.0312883    -5.180    1.45e-06 ***
  gY_1       -0.0710603    0.0357299    -1.989    0.0499   **
  gY_2       -0.0147126    0.0357268    -0.4118   0.6815

Mean dependent var  -0.027174   S.D. dependent var   0.195863
Sum squared resid    1.572525   S.E. of regression   0.135223
R-squared            0.549543   Adjusted R-squared   0.523353
F(5, 86)             20.98343   P-value(F)           1.19e-13
Log-likelihood       56.63652   Akaike criterion    -101.2730
Schwarz criterion   -86.14231   Hannan-Quinn        -95.16615
rho                 -0.002884   Durbin-Watson        2.003999
```

Figure 8: Model Window After OLS-Estimation

The well-formatted estimation output covers standard model information:

```
Model 1: OLS, using observations 1985:1-2007:4 (T = 92)
Dependent variable: dU

           coefficient   std. error   t-ratio   p-value
  --------------------------------------------------------
const       0.178685     0.0396906     4.502    2.10e-05 ***
dU_1        0.339384     0.108800      3.119    0.0025   ***
dU_2        0.0964127    0.0984661     0.9791   0.3303
gY         -0.162066     0.0312883    -5.180    1.45e-06 ***
gY_1       -0.0710603    0.0357299    -1.989    0.0499   **
gY_2       -0.0147126    0.0357268    -0.4118   0.6815

Mean dependent var  -0.027174   S.D. dependent var   0.195863
Sum squared resid    1.572525   S.E. of regression   0.135223
R-squared            0.549543   Adjusted R-squared   0.523353
F(5, 86)             20.98343   P-value(F)           1.19e-13
Log-likelihood       56.63652   Akaike criterion    -101.2730
Schwarz criterion   -86.14231   Hannan-Quinn        -95.16615
rho                 -0.002884   Durbin-Watson        2.003999

Excluding the constant, p-value was highest for variable 9 (gY_2)
```

Before proceeding with the model interpretation, one needs to run specification tests to check whether the Gauss-Markov criteria are fulfilled by the estimated model. Using commands instead of the GUI, the estimated residuals are depicted in Figure 9(a) and can be directly accessed after having executed the `ols` command by the $\$uhat$ accessor (for a list of available accessors, click the menu "Help, function reference"):

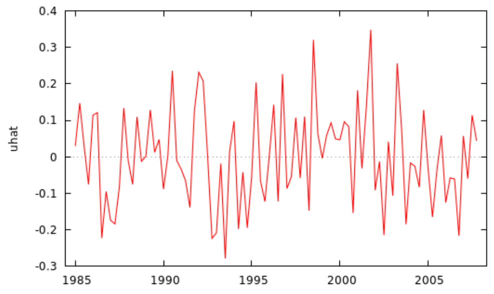
```
series resid = $uhat        # store residuals as series object
gnuplot resid --with-lines --time-series --output=display
```

The following four lines run a battery of specification tests. The `modtest` command calls various built-in specification tests such as tests on no remaining serial correlation in the residuals of order k (using `k --autocorr`), the White's test on homoskedasticity (`--white`) and more (see `help modtest` for details). The command `reset` calls Ramsey's RESET test on the correct functional form while `qlrtest` performs the Quandt likelihood ratio test for no structural break at an unknown point in time. Many more standard tests are easily accessible for cross-sectional, time-series and panel-data models. The `--quiet` option suppresses auxiliary regressions which may be deactivated if one wants to understand how the respective tests are actually performed.

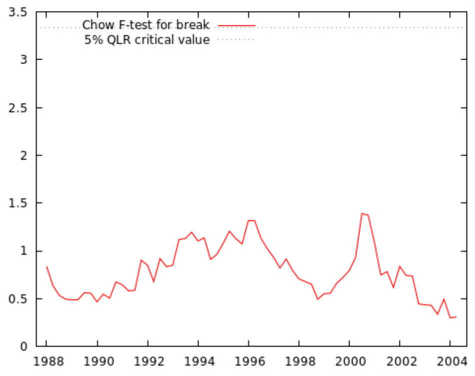
```
modtest 1 --autocorr --quiet        # test on 1st order serial correlation
modtest --white --quiet
reset --quiet
qlrtest --plot=display
```

The output indicates that the model does not seem to suffer from any misspecification issues and also there is no hint on parameter breaks at any point in time (see Figure 9(b)):

```
Breusch-Godfrey test for first-order autocorrelation
Test statistic: LMF = 0.030358,
with p-value = P(F(1,85) > 0.0303584) = 0.862

White's test for heteroskedasticity
Test statistic: TR^2 = 12.437424,
with p-value = P(Chi-square(20) > 12.437424) = 0.900198

RESET test for specification (squares and cubes)
Test statistic: F = 2.831376,
with p-value = P(F(2,84) > 2.83138) = 0.0646

Quandt likelihood ratio test for structural break at an unknown point,
with 15 percent trimming:
The maximum F(6, 80) = 1.38908 occurs at observation 2000:3
Asymptotic p-value = 0.870269 for chi-square(6) = 8.33445
```



(a) Residuals



(b) QLR test results

Figure 9: Estimated residuals, $\hat{u}_t$ and Quandt likelihood ratio test results on parameter stability

After having checked the model's specification, we compute the dynamic multipliers using the ARDL(2,2) coefficient estimates. The so called impact multiplier refers to the immediate change in the endogenous, dU_t after a unit-change in gY_t: $\frac{\partial dU_t}{\partial gY_t} = \hat{\theta}_0 = -0.162$ in period t. A percentage point increase in growth of real GDP leads to a reduction of the change in the unemployment rate by about 0.162 percentage points in the same quarter. The long-run multiplier, $\hat{\theta}^{lr}$, is a non-linear expression computed as $\hat{\theta}^{lr} = \frac{\sum_{j=0}^{2} \hat{\theta}_j}{1-\sum_{i=1}^{2} \hat{\beta}_i} = -0.439$. Using Gretl we can directly compute these two multiplier values by grabbing the OLS coefficient estimates using the `$coeff` accessor as follows:

```
printf "Impact multiplier = %g\n", $coeff(gY)
scalar lr_theta = ($coeff(gY)+$coeff(gY_1)+$coeff(gY_2)) / \
(1-$coeff(dU_1)-$coeff(dU_2))
printf "Long-run multiplier = %g\n", lr_theta
```

yielding the following values

```
Impact multiplier = -0.162066
Long-run multiplier = -0.439273
```

5 Working with matrices

Gretl's matrix language has similarity to the syntax of widespread commercial software packages such as MATLAB and GAUSS. The Gretl project allows an easy transfer of series objects to the matrix-world, and *vice versa*. For details on matrix manipulations, matrix algebra, statistics and transformations see Ch. 16 in the Gretl manual.

The following example shows how we could have estimated the ARDL(2,2) model easily by means of standard linear algebra using the solution to the minimization problem underlying standard OLS. The solution is the famous equation $(X'X)^{-1}X'y$ where y is a $T \times 1$ vector of the endogenous and X is the $T \times k$ matrix of k regressors.

We declare the two matrix objects `y` and `X` by directly transforming either a series or a list of series to a vector and matrix, respectively. The vector with the solution, `bhat`, is easily computed by using the `inv()` function for computing the inverse of a matrix, and `'` is a shortcut for the transpose of a matrix (equal to the `transp()` function). The required commands are:

```
matrix y = {dU}     # takes eventually restricted sample into account
list xlist = const dU(-1 to -2) gY(0 to -2)
matrix X = {xlist}
bhat = inv(X'X)*X'y
printf "%12.6g\n", bhat'
```

The `printf` command prints the result

```
    0.178685     0.339384    0.0964127    -0.162066   -0.0710603   -0.0147126
```

which is exactly the coefficient vector we have obtained before by the `ols` command.

6 Summary

References

Adkins, L. C. (2014). *Using gretl for Principles of Econometrics, 4th Edition*. Number 1412.

Cottrell, A. (2017). Hansl: A dsl for econometrics. In *Proceedings of the 2Nd International Workshop on Real World Domain Specific Languages*, RWDSL17, pages 1:1–1:10, New York, NY, USA. ACM.

A Appendix

A.1 Further Resources

- youtube tutorials ???
- ???

A.2 Data Preparation and Cleaning Process

Below is a script for loading, defining, transforming, appending and storing the dataset used in Section 3.2. The script replicates all steps which were done using the GUI elements in Section 3.2 before. Copy and paste the following commands into a new script file in the editor window (see Figure 1(d)) and execute all commands by pressing "Ctrl+R".

```
clear                          # clear memory
set verbose off          # avoid printing commands

# Define a string variable for the URL where files are located
string url = "https://scm.darnold.org/atecon/gretl_aer/raw/master"
print url

# Load the monthly series from the server
open "@url/data_unrate.xls" --rowoffset=10 --preserve

# Define the data structure
setobs 12 1948:01 --time-series

# Transform the monthly series to quarterly
dataset compact 4

# Set the sample in line with the series being appended
smpl 1948:1 2018:2

# Append the real GDP series from the server
append "@url/data_gdp.xls" --rowoffset=10

# Drop the series 'observation_date'
delete observation_date

# Store the two time-series in native Gretl data format (*.gdt)
store "data_project.gdt"        # alternatively one can use "*.csv"
# Open the clean project file
open "data_project.gdt"
```

A.3 Add-ons and Contributed Packages

Both add-ons and (user-contributed) function packages add functionalities such as estimators, hypothesis tests or other analytical procedures to gretl's repertoire of built-in procedures. A list (including manuals) of available add-ons can be found here: `http://gretl.sourceforge.net/addons.html`. A list of contributed packages including a help test or manual and sample script can be accessed here: `http://ricardo.ecn.wfu.edu/gretl/cgi-bin/gretldata.cgi?opt=SHOW_FUNCS`

For details on function packages read the "Gretl Function Package Guide" which can be downloaded by clicking the "Help, Function package guide" menu.

Function packages can either be acquired through the GUI or the command line. Using the command line, one needs to download (at least once to store it on the hard-disk) the package from the package server, and load it into memory at the beginning of each gretl session. A function package ships new functions which can be called. For details one has to read the help text or manual of a specific package. The following commands illustrate the steps:

```
install ADMBP                 # download package from server
include ADMBP.gfn             # load into memory
help ADMBP                    # opens help text or pdf-manual (if available)
```

A.4 Econometric Models

TBW

A.5 Logical Operators

?? The following logical operators are supported in Gretl.

Logical symbol	Meaning
==	Is equal to
!=	Is not equal to
&&	And
||	Or
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to

Table A1: Logical Operators

A.6 *if*-conditions

TBW

A.7 Loops

TBW

A.8 Functions

TBW

A.9 Communication with other software packages

Gretl support the communication with other popular software packages. Through the so called "foreign mechanism" the user can interpolate into a hansl script a set of statements to be executed by another program, with apparatus available to ferry data between the programs. This facility is supported for Octave, R, Python, Ox, Stata and Julia. This is especially useful to exploit functionality in the "foreign" program that is not currently available in Gretl. For details see ch. 39 to 44 in the Gretl manual.