Statistical Software Guide

for Introductory Econometrics

Antonio Bubbico antonio.bubbico@ucd.ie

Johan A. Elkink jos.elkink@ucd.ie

April 30, 2013

DRAFT VERSION

Contents

1 Introduction

This booklet is based on a module in the School of Politics & International Relations of University College Dublin that consists primarily of a typical introduction to econometrics. In this course, the statistical package used is R, but the intention is to allow students to choose their software freely. This booklet provides the command syntax necessary for a number of different statistical packages, originally developed for this course, but intended to be useful for anyone working in basic econometric analysis.

1.1 Syntax files

For the purpose of replication, whichever statistical package you use, it is always important to use command syntax files instead of just menu buttons. By using the graphical user interface without storing the individual commands (SPSS for example has a "paste" button on most screens that allows you to save the syntax instead of directly executing the command), it will be impossible to derive at a later date what steps exactly you took. This booklet therefore explains everything through code commands, not graphical user interfaces.

A number of small example data sets will be used in this booklet and distributed separately.

2 Data and file management

In this section we will discuss general commands related to data and file management. Many of those commands can be rather intricate, with various options that are only useful for very specific file formats. This section therefore only discusses common concerns and for further detail we would refer the reader to the relevant help files for each sofftware package.

2.1 Working directory

R code:

```
setwd("C:/Documents and Settings/data")
```

R has a working directory that is returned typing `getwd()` and set by typing `setwd()`. In Windows it is standard to use a backslash to separate folders, but R is originally designed for UNIX systems, and expects forward slashes as in the example. You can also use double backslashes.

Stata code:

```
cd "C:\Documents and Settings\data"
```

Stata has a working directory that is returned typing `pwd` and set by `cd`.

SPSS code:

```
cd 'C:\Documents and Settings\data'.
```

In SPSS we can use the `cd` command to set the working directory.

2.2 Importing data

In these examples, we assume the file of interest is in the working directory. Otherwise we need to specify the file's folder.

Importing a text file (`.txt`)

R code:

```
data <- read.table("file.txt")
```

Stata code:

```
insheet using file.txt
```

SPSS code:

```
get data
/type=txt
/file="file.txt"
/delcase=line
/delimiters="
/arrangement=delimited
/firstcase=2
/importcase=all
/variables=
var1 a9
var2 a6
var3 f4.0
var4 f8.6
cache.
```

where for each variable we indicate its desiderable format. In this case the variable var1 is a string with maximum 9 characters, variable var2 is a string with maximum 6 characters, var3 is a number with maximum 4 digits, var4 a number with maximum 8 digits and 6 decimals.

Importing a comma-separated spreadsheet file (`.csv`)

R code:

```
data <- read.csv("file.csv")
```

Stata code:

```
insheet using file.csv
```

In SPSS a .csv is imported equally to a .txt, also in the definition of the type of file.

Importing a Stata file (`.dta`)

R code:

```
library(foreign)
data <- read.dta("demdev.dta")
```

Stata code:

```
use demdev.dta
```

SPSS code:

```
get stata file='demdev.dta'.
```

SPSS is not able to read a `.dta` saved as last version of Stata. Then, it is better to save the `.dta` in older versions.

2.3 Importing data from a web page

R code:

```
library(foreign)
cigs <- read.dta("http://fmwww.bc.edu/ec-p/data/stockwatson/cig_ch10.dta"))
cigs <- read.table(file=http://fmwww.bc.edu/ec-p/data/stockwatson/cig_ch10.txt,
header=TRUE)
```

Stata code:

```
use http://fmwww.bc.edu/ec-p/data/stockwatson/cig_ch10.dta
```

In SPSS, you will have to download the file manually using your web browser at

`http://fmwww.bc.edu/ec-p/data/stockwatson/cig_ch10.dta`

then, it can be opened like a regular Stata file.

2.4 Exporting data to another format

From R to:

tbfa .csv or .txt file

R code:

```
write.csv(data, "data.csv")
write.csv(data, "data.txt")
```

an excel spreadsheet

R code:

```
install.packages("xlsReadWrite")
library(xlsReadWrite)
xls.getshlib()
write.xls(data, "data.xls")
```

a SPSS file

R code:

```
library(foreign)
write.foreign(data, "data.txt", "data.sps", package="SPSS")
```

a Stata file

R code:

```
library(foreign)
write.dta(data, "data.dta")
```

From Stata to:

a .csv or .txt file

Stata code:

```
outsheet country ccode year polity2 democracy laggdppc energy2 cwar ipyears
propdem catho80 ioscore using demdev.csv , comma
outsheet country ccode year polity2 democracy laggdppc energy2 cwar ipyears
propdem catho80 ioscore using demdev.txt , comma
```

an excel spreadsheet

Stata code:

```
outsheet country ccode year polity2 democracy laggdppc energy2 cwar ipyears
propdem catho80 ioscore using demdev.xls
```

a SPSS file

SPSS code:

```
outsheet country ccode year polity2 democracy laggdppc energy2 cwar ipyears
propdem catho80 ioscore using demdev.sps
```

From SPSS to:

a .csv or .txt file

SPSS code:

```
save translate outfile = 'data.csv'
/type=csv
/map
/replace
/fieldnames
/cells=values.
```

an excel spreadsheet

SPSS code:

```
save translate outfile = 'data.xls'
/type=xls
/map
/replace
/fieldnames
/cells=values.
```

a Stata file

SPSS code:

```
save translate outfile='data.dta'
/type=stata
/map
/replace.
```

2.5 Saving commands

Save your command history

R code:

```
savehistory(file="namefile")
```

In Stata it is necessary to create a log file in which are saved all the commands and all the results until the decision to close the log file In order to open the log file:

Stata code:

```
log using "demdevhistory.smcl"
```

In order to close the log file:

Stata code:

```
log close
```

SPSS code:

```
save outfile ='statistics2.spv'.
```

Recall your command history

R code:

```
loadhistory(file="namefile")
```

Stata code:

```
view "C:\Documents\STATA\demdevhistory.smcl"
```

SPSS does not permit to open a syntax file through another syntax file, then the way to use is through clicking on `File` →`Open`→`Syntax`

Save your workspace

R code:

```
save.image(file="filename.Rdata")
```

Stata code:

```
findit estsave
estsave, gen(name)
save filename
```

In Stata, after `findit estsave` it is necessary to download the proposed package in order to be able to use the command `estasave`. In SPSS we distinguish between the output file (where all the commands and the results are saved) and the Data editor (where all the variables are showed). In the case we have more than one data editor open, we need to specify which we want to use for our analysis, for example if we need to use the dataset1, we'll type, before other commands, `dataset activate dataset1`. In order to save the output:

SPSS code:

```
output save outfile='output.spv'.
```

In order to save the Data editor:

SPSS code:

```
save outfile='dataeditor.sav'.
```

Recall your workspace

R code:

```
load("~/filename.Rdata")
```

Stata code:

```
use filename
estsave, from(name)
```

In SPSS, in order to recall the output file, type:

SPSS code:

```
output open
file='output1.spv'
```

in order to recall the data editor file:

SPSS code:

```
get
file='filename.sav'.
```

2.6 Converting character to numeric and vice versa

We need to make a distinction between numeric, character (string) and factor data. There will be commands requiring one specific type of variable. Here a string variable is converted in numeric, where country is a string and idcountry is numeric:

R code:

```
idcountry<-as.numeric(country)
```

Stata code:

```
encode country, gen(idcountry)
```

SPSS code:

```
compute idcountry = number(country, f8.2).
```

or

SPSS code:

```
alter type country (f8.2).
```

Converting a numeric variable in a string

R code:

```
strcountry<-as.character(idcountry)
```

Stata code:

```
decode idcountry, gen(strcountry)
```

SPSS code:

```
string strcountry(a8).
compute strcountry = string(n, f2.0).
```

2.7 Renaming columns (variables)

R code:

```
names(data)[names(data)=="country"]<-"Country"
```

Stata code:

```
rename country Country
```

SPSS code:

```
rename variables country = Country.
```

SPSS code:

```
rename variables country year polity2 = Country Year PolityTwo.
```

2.8 Create a new variable

In R the creation of a new variable is not influenced by the variables already present in the data set, since in R we can have different objects and different data sets and to use them separately. In Stata and in SPSS this is not possible, since the number of observations of a new variable is determined by the number of observations of the current data set. Here are presented the commands to create a variable when a data set is already used. The new variable is the squared of "laggdppc", ("demdev" data set).

R code:

```
laggdppcSq <- data$laggdppc^2
```

Stata code:

```
generate laggdppcSq = laggdppc^2
```

SPSS code:

```
compute laggdppcSq = laggdppc * laggdppc.
```

2.9 Create a categorical variable

The following example creates a new variable, `laggdppcl`, that contains the GDP per capita information, but categorized by quantile.

R code:

```
laggdppc <- data$laggdppc
quantile(laggdppc, na.rm = TRUE)
laggdppcl <- laggdppc
laggdppcl[laggdppc <= 1454.925] <- 1
laggdppcl[laggdppc > 1454.925 & laggdppc < 7035.5] <- 2
laggdppcl[laggdppc >= 7035.5] <- 3
laggdppc.f <- factor(laggdppcl, labels=c("lowlaggdppc", "mediumlaggdppc",
"highlaggdppc"))
```

Stata code:

```
summarize laggdppc, detail
generate laggdppc.f = 0
replace laggdppc.f = lowlaggdppc if laggdppc <= 1454.85
replace laggdppc.f = mediumlaggdppc if laggdppc > 1454.85 & laggdppc < 7036
replace laggdppc.f = highlaggdppc if laggdppc >= 7036
```

SPSS code:

```
examine variables = laggdppc
/percentiles(5, 10, 25, 50, 75, 90, 95) haverage
/nototal.
do if ( laggdppc <= 1454.775).
compute laggdppc.f = lowlaggdppc.
else if (laggdppc > 1454.775 & laggdppc < 7036.5).
compute laggdppc.f = mediumlaggdppc.
else if (laggdppc >= 7036.5).
compute laggdppc.f = highlaggdppc.
end if.
```

2.10 Create dummy variables

The following code creates dummy variables (variables with only values 0 and 1) for each of the quantiles in income distribution.

R code:

```
laggdppc<-data$laggdppc
quantile(laggdppc,na.rm=TRUE)
laggdppc.1<-as.numeric(data$laggdppc<=1454.925)
laggdppc.2<-as.numeric(data$laggdppc>1454.925 & data$laggdppc<7035.5)
laggdppc.3<-as.numeric(data$laggdppc>7035.5)
```

Stata code:

```
summarize laggdppc,detail
generate laggdppc.1=0
replace laggdppc.1=1 if laggdppc<=1454.85
replace laggdppc.2=0
replace laggdppc.2=1 if laggdppc>1454.85 & laggdppc<7036
generate laggdppc.3=0
replace laggdppc.3=1 if laggdppc>=7036
```

SPSS code:

```
dataset activate dataset1.
examine variables=laggdppc
/percentiles(5,10,25,50,75,90,95) haverage
/nototal.
do if ( laggdppc<=1454.775).
compute laggdppc.1=1.
else if (laggdppc>1454.775 & laggdppc<7036.5).
compute laggdppc.2=1.
else if ( laggdppc>=7036.5).
compute laggdppc.3=1.
end if.
```

2.11 Merging data sets (adding variables to a data set)

While in Stata and in SPSS a new generated variable is automatically added to the used data set, in R it must be added explicitly; for example, to add the dummy variable "laggdppc.1" in the data set, we can use:

R code:

```
data$laggdppc.1 <- laggdppc.1
```

Here we show how to merge two panel data sets. Since this is panel data, we need to merge on both the country code ("ccode") and the year ("year"). The two data sets are obtained by dividing the dataset "demdev" in two parts. The effect of merging will be of re-obtaining the original data set.

R code:

```
data1<-data[,c("country","ccode", "year", "polity2", "democracy", "laggdppc")]
data2<-data[,c("ccode","year", "energy2","cwar", "ipyears", "propdem",
"catho80", "ioscore")]
complete.data<-merge(data1, data2, by=c("ccode","year"))
```

Stata requires the data to be sorted by the variable used for merging.

Stata code:

```
use "demdev.dta", clear
keep country ccode year polity2 democracy laggdppc energy2
sort ccode year save data1
use "demdev.dta", clear
keep ccode year cwar ipyears propdem catho80 ioscore
sort ccode year save data2
merge m:m ccode year using data1
```

In the SPSS example we consider the complete data set already split in two data sets.

SPSS code:

```
get
stata file='data1.dta'.
dataset name dataset1 window=front.
get
stata file='data2.dta'.
dataset name dataset2 window=front.
dataset activate dataset1.
match files /file=*
/file='DataSet2'
/by ccode year.
```

2.12 Append data sets (add observations to a data set)

In order to show how to append two data sets, we divide the data set "demdev" in two parts with equal variables but different cases (or observations). Precisely here we want to show how to add some cases to a data set.

R code:

```
data1<-data[1:3035,]
data2<-data[3036:6071 ,]
dataappend<-rbind(data1,data2)]
```

Stata code:

```
use "demdev.dta", clear
drop in 1/3035
save data1
use "demdev.dta", clear
drop in 3036/6071
save data2
append using "data1.dta"
```

SPSS code:

```
get
stata file='demdev.dta'.
dataset name dataset3 window=front.
compute id=$casenum.
select if not (id<=3035).  save outfile='data2.sav'
/compressed.
get
stata file='data1.dta'.
dataset name dataset3 window=front.
select if not (id>3035).
save outfile='data1.sav'
/compressed.
dataset activate dataset3.
add files /file=*
/file='data2.sav'.
```

2.13 Selection of variables and observations

2.13.1 Selection of variables

In this example a new data set is created using some variables ("country", "democracy" and "laggdppc") from the"demdev" data set.

R code:

```
library(foreign)
data<-read.dta("demdev.dta")
variables<-c("country","democracy", "laggdppc")
dataN<- data[variables]
```

Stata code:

```
use "demdev.dta"
keep country democracy laggdppc
```

SPSS code:

```
get
stata file='demdev.dta'.
dataset name dataset1 window=front.
save outfile= 'dataN.sav'
/keep country democracy laggdppc.
get file='dataN.sav'.
```

2.13.2 Selection of observations

In this example a new data set is created by "demdev" using data on Ireland, using the country code.

R code:

```
demdevIre<- data[ which(data$ccode=="205"), ]
```

Stata code:

```
keep if ccode==205
```

SPSS code:

```
get file='demireland.sav'.
select if (ccode = 205).
```

2.14 Sorting Data

2.15 Sort a data set by a variable

Ascending order

R code:

```
attach(data)
sortlaggdppc<-data[order(laggdppc), ]
```

or

R code:

```
library(plyr)
sortlaggdppc<-arrange(data, laggdppc)
```

In R when sorting by a factor variable the second method is the only one to use.

Stata code:

```
sort laggdppc
```

SPSS code:

```
get
stata file='demdev.dta'.
dataset name dataset1 window=front.
dataset activate dataset1.
sort cases by laggdppc(a).
```

Descending order

R code:

```
rsortlaggdppc<-data[ order(-laggdppc), ]
```

or

R code:

```
rsortlaggdppc<-arrange(data,desc(laggdppc))
```

In R when sorting by a factor variable the second method is the only one to use.

Stata code:

```
gsort -laggdppc
```

SPSS code:

```
get
stata file='demdev.dta'.
dataset name dataset1 window=front.
dataset activate dataset1.
sort cases by laggdppc(d).
```

2.16 Panel data manipulation: long and wide data sets

To illustrate the difference between long and wide data, we use the "longdata.dta" file, that originally is in a long format.

Long file

country	year	laggdppc
Albania	1996	3331
Albania	1997	3729
Albania	1998	3321
Argentina	1996	10371
Argentina	1997	10805
Argentina	1998	11530
Armenia	1996	3024
Armenia	1997	3085
Armenia	1998	3236
Austria	1996	20696
Austria	1997	21068
Austria	1998	21305

Wide file

country	laggdppc1996	laggdppc1997	laggdppc1998
Albania	3331	3729	3321
Argentina	10371	10805	11530
Armenia	3024	3085	3236
Austria	20696	21068	21305

From long to wide

R code:

```
install.packages("reshape2")
library(reshape2)
datalong<-read.dta("longdata.dta")
data.wide<-dcast(datalong, country~year, value.var="laggdppc")
```

Stata code:

```
use "longdata.dta", clear
reshape wide laggdppc, i(country) j(year)
```

SPSS code:

```
get
stata file='longlong.dta'.
casestovars
/id=country
/index = year.
list.
```

From wide to long

R code:

```
data.long<-melt(data.wide,id.vars=c("country"))
```

Stata code:

```
reshape long laggdppc, i(country) j(year)
```

SPSS code:

```
varstocases
/make laggdppc from laggdppc.1996 laggdppc.1997 laggdppc.1998
/index = year.
list country year laggdppc.
```

3 Missing values

Missing values may represent a problem since in that case for some software (as in R) the analysis is not possible. In other software (Stata and SPSS) the analysis is possible also at the presence of missing values, but it is important to check for them, in order to have a complete understanding on which data are used. R represents missing values by the symbol NA (not available) and impossible values (e.g., divided by zero) by the symbol NaN (not a number), while Stata and SPSS denote missing value with a point.
The "demdev.dta" file is used in the following examples.

3.1 Check for missing values

R code:

```
library(foreign)
data<-read.dta("demdev.dta")
complete.cases(data)
is.na(data$polity2)
```

and

R code:

```
install.packages("VIM")
library(VIM)
a<-aggr(data)
```

Stata code:

```
use "demdev.dta", clear
findit nmis
egen nmis=rmiss2( country ccode year polity2 democracy laggdppc energy2 cwar
ipyears propdem catho80 ioscore)
tab nmis
findit tabmiss
tabmiss country ccode year polity2 democracy laggdppc energy2 cwar ipyears
propdem catho80 ioscore
```

In Stata, once the "findit" command finds information on the package "tabmiss", we need to download it.

SPSS code:

```
get
stata file ='demdev.dta'.
count
cmiss = country ccode year polity2 democracy laggdppc energy2 cwar ipyears
propdem catho80 ioscore (missing).
frequencies variables=cmiss
/order= analysis.
```

The commands above in SPSS require numeric variables.

SPSS code:

```
frequencies variables=country ccode year polity2 democracy laggdppc energy2
cwar ipyears propdem catho80 ioscore
/format=notable
/order= analysis.
```

3.2 Create a data set without missing data

Note that this is listwise deletion for the entire data set: if one variable missing a data point, all data on that observation is removed. Often you might want to use an alternative strategy, such

as missing data imputation.[1] Furthermore, while for manual matrix algebra and many simple functions in R, you need data without missing values, for most regression analysis, listwise deletion is automatically applied and only on those variables that are included in the regression model.

R code:

```
library(foreign)
data<-read.dta("demdev.dta")
data1<- na.omit(data)
```

Stata code:

```
foreach var of varlist country ccode year polity2 democracy laggdppc energy2
cwar ipyears propdem catho80 ioscore {
drop if `var'==.
}
save datanomissing
```

[1] See the course syllabus for references on this topic.

SPSS code:

```
get stata file='demdev.dta'.
dataset name dataset2 window=front.
count nmissing= ccode to ioscore (missing).
execute.
dataset activate dataset2.
use all.
compute filter_$=(nmissing=0).
variable labels filter_$ 'nmissing=0 (filter)'.
value labels filter_$ 0 'not selected' 1 'selected'.
formats filter_$ (f1.0).
filter by filter_$.
execute.
filter off.
use all.
select if (nmissing=0).
execute.
save outfile='datanomissing.sav'.
```

3.3 Working explicitely on data used in the regression (in R)

Let's consider the "demdev.dta" data set for rumming a regression where "polity2" is the dependent and "laggdppc" is the independent variable.

R code:

```
m <- lm(polity2~laggdppc, data=data)
```

Since when running a regression the software does not consider the missing values, sometimes may happen we are interested only on data used in that regression. The data frame as provided by the regression output is accessible by typing

R code:

```
m$model
```

The regression output will also save a list of elements that it removed, which can be accessed with:

R code:

```
m$na.action
```

If we want to obtain just the data used in the regression for a specific variable ("polity2" in our example), we need to type:

R code:

```
polity2m <- data$polity2[-m$na.action]
```

4 Matrices

In Stata, a separate programming language has been created besides the regular Stata syntax for more flexible statistical programming. This language is called *mata* and this is what we will use in the example code below. In order to enter in the mata environment, type `mata` and use the command `end` to return to the regular Stata interface. All example code in this section assumes that it is within this mata environment.

In SPSS, to enter in a matrix environment, the first command to type in the Syntax editor is `matrix.`, while the last command must be `end matrix`. After the code has been completed, it is necessary to run the commands choosing the option "Run → all" in order to visualize the output. Note that output is often not printed on the screen, unless explicitly a `print variablename.` command is provided. All example code in this section assumes that it is within this matrix environment.

4.1 How to create a vector

$$x = \begin{bmatrix} 2 & 10 & 3 & 4 \end{bmatrix}'$$

R code:

```
x <- c(2, 10, 3, 4)
```

Stata code:

```
x = (2 \10 \3\4)
```

SPSS code:

```
compute x = {2; 10; 3; 4}.
```

4.2 How to crate a sequence

To generate a sequence between -3 and 3, with an increment of 0.5,

$$s = \begin{bmatrix} -3 & -2.5 & \dots & 2.5 & 3 \end{bmatrix},$$

you can use:

R code:

```
s <- seq(-3, 3, .5)
```

Stata code:

```
s = range(-3, 3, .5)
```

In SPSS, it is only possible to create sequences with integer steps. So to create a sequence

$$si = \begin{bmatrix} -3 & -1 & 1 & 3 \end{bmatrix},$$

you can use:

SPSS code:

```
compute si = {-3:3:2}.
```

To create the s sequence above, you could of course use a trick:

SPSS code:

```
compute s = {-6:6} / 2.
```

4.3 How to create a matrix

The commands below show how to create a matrix with 4 rows and 2 columns, such as:

$$X = \begin{bmatrix} 2 & 10 \\ 3 & 4 \\ 12 & 8 \\ 5 & 3 \end{bmatrix}.$$

R code:

```
X <- matrix(c(2, 10, 3, 4, 12, 8, 5, 3), nrow = 4, byrow = TRUE)
```

Stata code:

```
X = (2, 10 \3, 4 \12, 8 \5, 3)
```

SPSS code:

```
compute X = {2, 10; 3, 4; 12, 8; 5, 3}.
```

4.4 Identity matrix

An identity matrix of dimension 4×4:

$$I_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

R code:

```
identity <- diag(4)
```

Stata code:

```
identity = I(4)
```

SPSS code:

```
compute identity = ident(4).
```

4.5 Naming columns and rows

It is usually helpful to provide names to columns and rows of matrices, to help the interpration of later displays of the data.

R code:

```
colnames(X) <-c("Car", "Bike")
rownames(X) <-c("Child", "Young", "Adult", "OldestOld")
```

Stata code:

```
matrix colnames X= Car Bike
matrix rownames X= Child Young Adult OldestOld
```

SPSS code:

```
matrix.
compute X= {2,10;3,4;12,8;5,3}.
print X.
save X/outfile=*/variables Car Bike.
end matrix.
```

In SPSS it is not permitted to give names to columns and rows but only to the vectors/matrices.

4.6 Display matrix data on screen

R code:

```
X
```

Stata code:

```
X
```

SPSS code:

```
print X.
```

4.7 Basic matrix operations

Transpose

Transpose a vector or a matrix such as

$$Xt = X' = \begin{bmatrix} 2 & 3 & 12 & 5 \\ 10 & 4 & 8 & 3 \end{bmatrix}.$$

R code:

```
Xt <- t(X)
```

Stata code:

```
Xt = X'
```

SPSS code:

```
compute Xt = T(X).
```

Vector sum

$$z = \sum_{i=1}^{n} x_i = 2 + 10 + 3 + 4 = 19$$

R code:

```
z <- sum(x)
```

Stata code:

```
z = sum(x)
```

SPSS code:

```
compute z = csum(x).
```

Note that if x was a row-vector, the command `rsum` has to be used in SPSS.

4.8 Basic matrix algebra

Matrix addition

$$Y = \begin{bmatrix} 7 & 12 \\ 8 & 5 \\ 1 & 13 \\ 3 & 1 \end{bmatrix} \qquad Z = X + Y = \begin{bmatrix} 9 & 22 \\ 11 & 9 \\ 13 & 21 \\ 8 & 4 \end{bmatrix}$$

R code:

```
Y <- matrix(c(7, 12, 8, 5, 1, 13, 3, 1), nrow = 4, byrow = TRUE)
Z <- X + Y
```

Stata code:

```
Y = (7, 12 \8, 5 \1, 13 \3, 1)
Z = X + Y
```

SPSS code:

```
compute Y = {7, 12; 8, 5; 1, 13; 3, 1}.
compute Z = X + Y.
```

Subtraction of two matrices or vectors is done analogously.

Matrix multiplication with a scalar

$$xf = 5 \cdot x = \begin{bmatrix} 10 & 50 & 15 & 20 \end{bmatrix}'$$

R code:

```
xf <- x * 5
```

Stata code:

```
xf = x * 5
```

SPSS code:

```
compute xf = x * 5.
```

Vector inner product

$$y = \begin{bmatrix} 3 \\ 7 \\ 2 \\ 4 \end{bmatrix} \qquad ip = x'y = 2 \cdot 3 + 10 \cdot 7 + 3 \cdot 2 + 4 \cdot 4 = 98$$

R code:

```
y <- c(3, 7, 2, 4)
ip <- t(x) %*% y
```

Stata code:

```
y = (3 \7 \2 \4)
ip = x'y
```

SPSS code:

```
compute y = {3; 7; 2; 4}.
compute ip = T(x) * y.
```

Vector outer product

$$op = xy' = \begin{bmatrix} 6 & 14 & 4 & 8 \\ 30 & 70 & 20 & 40 \\ 9 & 21 & 6 & 12 \\ 12 & 28 & 8 & 16 \end{bmatrix}$$

R code:

```
op <- x %*% t(y)
```

Stata code:

```
op = x * y'
```

SPSS code:

```
compute op = x * T(y).
```

Matrix multiplication

$$V = \begin{bmatrix} 12 & 22 & 2 \\ 3 & 4 & 8 \end{bmatrix} \qquad M = X \cdot V = \begin{bmatrix} 54 & 84 & 84 \\ 48 & 82 & 38 \\ 168 & 296 & 88 \\ 69 & 122 & 34 \end{bmatrix}$$

R code:

```
M <- X %*% V
```

Stata code:

```
M = X * V
```

SPSS code:

```
compute M = X * V.
```

4.9 Further matrix algebra

Matrix inverse

$$R = \begin{bmatrix} 2 & 10 \\ 3 & 4 \end{bmatrix} \qquad Ri = R^{-1}$$

R code:

```
R <- matrix(c(2, 10, 3, 4), nrow = 2, byrow = TRUE)
Ri <- solve(R)
```

Stata code:

```
R = (2, 10 \3, 4)
Ri = inv(R)
```

SPSS code:

```
compute R = {2, 10; 3, 4}.
compute Ri = inv(R).
```

Matrix trace

$$A = \begin{bmatrix} 1 & 6 & 1 & 0 \\ 2 & 4 & 6 & 10 \\ 1 & 3 & 9 & 3 \\ 10 & 20 & 0 & 8 \end{bmatrix} \qquad trA = tr(A) = 1 + 4 + 9 + 8 = 22$$

R code:

```
trA <- sum(diag(A))
```

Stata code:

```
trA = trace(A)
```

SPSS code:

```
compute trA = trace(A)
```

4.10 Exchange between matrix and data

In Stata, to save a vector as a variable to the current data set, it is necessary to save it from the mata environment to the regular Stata environment. Then, use `svmat` to save the vector as a variable:

Stata code:

```
st_matrix("x", x)
end
svmat x
```

In SPSS, to save a vector created in the matrix environment in your dataset, you need to use the following commands, before going out the matrix environment:

SPSS code:

```
save X/outfile=*/variables X.
```

In R, there is no difference between data and a matrix, although often a data frame is used for data analysis:

R code:

```
X <- as.data.frame(X)
```

5 Probability distributions

5.1 Calculate normal density at a point d

R code:

```
dnorm(-1)
```

Stata code:

```
display normalden(-1)
```

SPSS code:

```
compute c=pdf.normal(-1,0,1).
execute.
```

5.2 Calculate the normal area up to a point p

R code:

```
pnorm(-1)
```

Stata code:

```
display normal(-1)
```

SPSS code:

```
comp a=cdfnorm(-1).
```

5.3 Normal distribution: calculate the area beyond the point 1-p

R code:

```
1-pnorm(-1)
```

Stata code:

```
display 1-normal(-1)
```

SPSS code:

```
comp a=1-cdfnorm(-1).
```

5.4 Normal distribution: knowing the area, calculate point on the x-axis

R code:

```
qnorm(.1586553)
```

Stata code:

```
display invnormal(.1586553)
```

SPSS code:

```
compute q=idf.normal(0.1586553,0,1).
execute.
```

5.5 Normal distribution: draw 100 random numbers

R code:

```
x.random<-rnorm(100)
hist(x.random, freq=FALSE)
```

Stata code:

```
drop _all
set up 100
gen r=invnormal(uniform())
```

SPSS code:

```
new file.
input program.
loop #i=1 to 100.
compute id=#i.
compute x=rv.normal(0,1).
end case.
end loop.
end file.
end input program.
execute.
```

5.6 Area under χ^2-distribution with k degree of freedom

R code:

```
pchisq(n,k)
```

Stata code:

```
display chi2(k,n)
```

SPSS code:

```
compute chi=cdf.chisq(n,k).
execute.
```

5.7 Area under F-distribution

R code:

```
1-pf(n,df1,df2)
```

Stata code:

```
display 1-F(df1,df2,n)
```

SPSS code:

```
compute f=1-cdf.f(0.2,3,4).
execute.
```

5.8 Binomial distribution: draw 10 random numbers

Here we make an example with n=1 and p=0.5 (ten throws with a coin).

R code:

```
x<-rbinom(10,1,.5)
```

Stata code:

```
drop _all
set up 10
mata:
n = 1
p = .5
draw1 = J(st_nobs(),1,.)
for(i=1; i<=rows(draw1); i++) {
trials = uniform(1,n)
successes = trials :< p
draw1[i,1] = rowsum(successes)
}
idx = st_addvar("int", "draw1")
st_store(.,idx,draw1)
end
```

SPSS code:

```
new file.
input program.
loop #I=1 to 10.
compute binom= rv.binom(1, 0.5).
end case.
end loop.
end file.
end input program.
execute.
```

5.9 P-value from t-distribution (one-tailed)

R code:

```
1-pt(x,df)
```

Stata code:

```
display ttail(df, x)
```

SPSS code:

```
compute t=1-cdf.t(x,df).
execute.
```

6 Ordinary Least Squares (OLS)

In this section we will look at a simple regression model by way of example, using the data on the Irish Lisbon Referendum (`"lisbon.dta"`). We will look at the following model:

$$IMMIGRATION_i = \beta_0 + \beta_1 DISSATECON_i + \beta_2 ABORTION_i + \beta_3 NEUTRALITY_i$$

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	4.379	1.736	2.522	0.018
dissatecon	2.189	0.879	2.490	0.019
abortion	0.189	0.144	1.309	0.202
neutrality	0.04	0.239	0.015	0.988

6.1 Estimation of a linear model

R code:

```
summary(lm(immigratio ~ dissatecon + abortion + neutrality, data = lisbon))
```

Stata code:

```
reg immigration dissatecon abortion neutrality
```

SPSS code:

```
regression
/missing listwise
/statistics coeff outs r anova
/criteria=pin(.05) pout(.10)
/noorigin
/dependent immigration
/method=enter dissatecon abortion neutrality.
```

Estimation of a linear model without constant

R code:

```
summary(lm(immigration ~ 0 + dissatecon + abortion + neutrality, data =
lisbon))
```

Stata code:

```
reg immigration dissatecon abortion neutrality, noconstant
```

SPSS code:

```
regression
/missing listwise
/statistics coeff outs r anova
/criteria=pin(.05) pout(.10)
/origin
/dependent immigration
/method=enter dissatecon abortion neutrality.
```

6.2 Outliers, leverage, influence

Fitted values against residuals

R code:

```
library (faraway)
m<-lm(immigration ~ dissatecon + abortion + neutrality, data = lisbon)
plot(m, which=1,bty="n",pch=19)
```

Stata code:

```
rvfplot, yline(0)
```

SPSS code:

```
/save pred resid.
graph
/scatterplot(bivar)=pre_1 with res_1
/missing=listwise.
```

Normal Q-Q plot

R code:

```
plot(m, which=2, bty="n", pch=19)
```

Stata code:

```
predict e, residuals
qnorm e
```

SPSS code:

```
pplot
/variables=res_1
/nolog
/nostandardize
/type=q-q
/fraction=blom
/ties=mean
/dist=normal.
```

Scale-location plot

R code:

```
plot(m, which=3, bty="n", pch=19)
```

Stata code:

```
predict estud, rstudent
predict yhat
gen rstscale = sqrt(abs(estud))
graph twoway (scatter rstscale yhat)
```

SPSS code:

```
/save pred sdresid.
compute rstscale=sqrt(abs(sdr_1)).
execute.
Graph
/scatterplot(bivar)=pre_1 with rstscale
/missing=listwise.
```

Leverage

R code:

```
h <- influence(m)$hat
h <- h[order(h)]
plot(h, type="n", bty="n", xlab="", ylab="Leverage")
text(h, label=names(h), cex=.7, pos=2)
points(h, col="red", pch=19)
```

Stata code:

```
predict lev, leverage
stem lev
sort lev
gen index=_n
graph twoway (scatter lev index)
```

SPSS code:

```
/save lever .
sort cases lev_1.
loop num_id=1 to 30.
end case.
end loop.
execute.
graph
/scatterplot(bivar)=num_id with lev_1
/missing=listwise.
```

Residuals vs leverage

R code:

```
plot(m, which=5, bty="n", pch=19)
```

Stata code:

```
lvr2plot,mlabel()
```

SPSS code:

```
/save lever zresid.
graph
/scatterplot(bivar)=lev_1 with zre_1
/missing=listwise.
```

Cook's distance

R code:

```
cook<-cooks.distance(m)
```

Stata code:

```
predict cooksd if e(sample), cooksd
```

SPSS code:

```
/save cook.
```

Cook's distance against leverage

R code:

```
plot(m, which=6, pch=19, bty="n")
```

Stata code:

```
graph twoway (scatter cooksd lev)
```

SPSS code:

```
/save cook lever.
graph
/scatterplot(bivar)=lev_1 with coo_1
/missing=listwise.
```

6.3 Multicollinearity

Variance Inflation Factor

R code:

```
library(car)
vif(m)
```

Stata code:

```
vif
```

or

Stata code:

```
estat vif
```

SPSS code:

```
regression /statistics=defaults tol
/dependent immigration
/method=enter dissatecon abortion neutrality.
```

6.4 Heteroskedasticity

White's HCCM

$$var(\hat{\beta}^{OLS}) = (\mathbf{X'X})^{-1}\mathbf{X'}\Omega\mathbf{X}(\mathbf{X'X})^{-1}$$

$$HC0 = (\mathbf{X'X})^{-1}\mathbf{X'}diag(e_i^2)\mathbf{X}(\mathbf{X'X})^{-1}$$

$$HC1 = \frac{n}{n-k}(\mathbf{X'X})^{-1}\mathbf{X'}diag(e_i^2)\mathbf{X}(\mathbf{X'X})^{-1} = \frac{n}{n-k}HC0$$

$$HC2 = (\mathbf{X'X})^{-1}\mathbf{X'}diag(\frac{e_i^2}{1-h_{ii}})\mathbf{X}(\mathbf{X'X})^{-1}$$

$$HC3 = (\mathbf{X'X})^{-1}\mathbf{X'}diag(\frac{e_i^2}{(1-h_{ii})^2})\mathbf{X}(\mathbf{X'X})^{-1}$$

R code:

```
library(car)
vcov <- hccm(m, type="hc1")
sqrt(diag(vcov))
```

or

R code:

```
vcov <- hccm(m, type="hc3")
sqrt(diag(vcov))
```

Stata offers only the possibility to calculate, in a simple way, HC1.

Stata code:

```
regress immigration dissatecon abortion neutrality, vce(robust)

matrix list e(V)
```

Code in SPSS for calculating HC0:

SPSS code:

```
regression
/missing listwise
/statistics coeff outs r anova
/criteria=pin(.05) pout(.10)
/noorigin
/dependent immigration
/method=enter dissatecon abortion neutrality
/save resid(res_1).
compute esq = res_1 * res_1.
execute.
filter off.
use all.
execute .
compute constant = 1.
execute.
matrix.
filter off.
use all.
select if(missing(esq) = 0).
execute .
get y / variables = immigration.
get x / variables = constant, dissatecon, abortion, neutrality
/ names = xtitles.
get residual / variables = res_1.
get esq / variables = esq.
compute xrtitles = transpos(xtitles).
compute n = nrow(esq).
compute k = ncol(x).
```

SPSS code:

```
compute o = mdiag(esq).
compute whitev = (inv(transpos(x) *x)) *transpos(x)* o *x*inv(transpos(x) *
x).
compute wdiag = diag(whitev).
compute white_se = sqrt(wdiag).
print white_se
/ format = "f5.6"
/ title = "HC0 white's corrected standard errors"
/ rnames = xrtitles.
end matrix.
```

Code in SPSS for calculating HC3:

SPSS code:

```
regression
/missing listwise
/statistics coeff outs r anova
/criteria=pin(.05) pout(.10)
/noorigin
/dependent immigration
/method=enter dissatecon abortion neutrality
/save resid(res_1) .
compute esq = res_1 * res_1.
execute.
filter off.
use all.
execute.
compute constant = 1.
execute.
```

SPSS code:

```
filter off.
use all.
select if(missing(esq) = 0).
execute.
matrix.
get y / variables = immigration.
get x / variables = constant, dissatecon, abortion, neutrality
/ names = xtitles.
get residual / variables = res_1.
get esq / variables = esq.
compute xrtitles = transpos(xtitles).
compute n = nrow(esq).
compute k = ncol(x).
compute o = mdiag(esq).
compute xx = transpos(x) *x.
compute xx_1 = inv(xx).
compute x_1 = transpos(x).
compute h = x * xx_1 * x_1.
compute h_m = h * -1.
compute one_h = h_m + 1.
compute one_h_sq = one_h & ** 2.
compute o_hc3 = o &/ one_h_sq.
compute hc3_a = xx_1 * x_1 *o_hc3.
compute hc3 = hc3_a * x*xx_1.
compute hc3diag = diag(hc3).
compute hc3_se = sqrt(hc3diag).
print hc3_se
/ format = "f5.6"
/ title = "HC3 white's corrected standard errors"
/ rnames = xrtitles.
end matrix.
```

Residual plots

Residuals against fitted values

R code:

```
plot(residuals(m) ~ fitted(m))
```

Stata code:

```
rvfplot, yline(0)
```

SPSS code:

```
/save pred resid.
graph
/scatterplot(bivar)=pre_1 with res_1
/missing=listwise.
```

Residuals against dependent variable

R code:

```
plot(residuals(m) ~ immigration)
```

Stata code:

```
graph twoway scatter e immigration
```

SPSS code:

```
/save sresid.
graph
/scatterplot(bivar)=immigration with sre_1
/missing=listwise.
```

Residuals against independent variable(s)

R code:

```
plot(residuals(m)~abortion, data=lisbon)
```

Stata code:

```
graph twoway scatter e abortion
```

SPSS code:

```
graph
/scatterplot(bivar)= abortion with res_1
/missing=listwise.
```

Known groups F-test for heteroskedasticity

$$\frac{SSR_1/(n_1-k)}{SSR_2/(n_2-k)} \sim F(n_1-k, n_2-k)$$

$$H_0 : \sigma_1^2 = \sigma_2^2$$

R code:

```
f<-immigration~dissatecon+abortion+neutrality
m1<-lm(f, data=lisbon, subset=(abstain==1))
m2<-lm(f,data=lisbon, subset=(abstain==0))
ssr1<-sum(residuals(m1)^2)
ssr2 <- sum(residuals(m2) ^2)
f <- (ssr1/m1$df) / (ssr2/m2$df)
1 - pf(f, m1$df, m2$df)
```

Stata code:

```
regress immigration dissatecon abortion neutrality if abstain==1
gen df_res=e(df_r)
gen rss=e(rss)
regress immigration dissatecon abortion neutrality if abstain==0
gen df_res2=e(df_r)
gen rss2=e(rss)
generate f=(rss/df_res)/(rss2/df_res2)
display 1-F(df_res,df_res2,f)
```

SPSS code:

```
regression
/select=abstain eq 1
/missing listwise
/statistics coeff outs r anova
/criteria=pin(.05) pout(.10)
/noorigin
/dependent immigration
/method=enter dissatecon abortion neutrality.
regression
/select=abstain eq 0
/missing listwise
/statistics coeff outs r anova
/criteria=pin(.05) pout(.10)
/noorigin
/dependent immigration
/method=enter dissatecon abortion neutrality
/save resid.
compute f=(0.879/2)/(106.75/20).
compute i=cdf.f(f,2,20).
compute m=1-i.
execute.
```

Breush-Pagan test

$$\sigma_i^2 = f(\mathbf{Z}\alpha)$$

$$\alpha = [\alpha_0 \alpha^*]'$$

$$H_0 : \alpha* = 0$$

$$H_1 = \alpha^* \neq 0$$

Assuming that:

$$e_i^2 \sim N(0, \sigma_i^2)$$

$$\eta = \frac{\mathbf{q'Z(Z'Z)^{-1}Z'q}}{2\hat{\sigma^4}} \sim \chi^2(s-1) asymptotically$$

R code:

```
bptest(m, studentize=F)
```

Stata implements a slightly different version of the Breusch-Pagan test, the Breusch-Pagan / Cook-Weisberg test for heteroskedasticity, where the null hypothesis is that the error is homoskedastic:

Stata code:

```
estat hettes
```

In order to obtain the "classic" Breush-Pagan test in Stata:

Stata code:

```
reg immigration dissatecon abortion neutrality
predict uhat, resid
gen uhatsq=uhat ^ 2
generate g=uhatsq/(e(rss)/e(N))
regress g dissatecon abortion neutrality
generate b=0.5*e(mss)
display 1-chi2(3,b)
```

SPSS code:

```
regression
/missing listwise
/statistics coeff outs r anova
/criteria=pin(.05) pout(.10)
/noorigin
/dependent immigration
/method=enter dissatecon abortion neutrality
/residuals hist(zresid) norm(zresid)
/save resid zresid.
descriptives variables=zre_1
/statistics=mean stddev min max kurtosis skewness.
compute res_1sq=res_1*res_1.
variable labels res_1sq "square of saved residuals res_1".
execute.
descriptives
variables=res_1sq
/statistics=sum.
compute g=res_1sq/(119.02/30).
execute.
regression
/missing listwise
/statistics coeff outs r anova
/criteria=pin(.05) pout(.10)
/noorigin
/dependent g
/method=enter dissatecon abortion neutrality.
compute b=0.5 * 0.752.
compute b_pchisq=1-cdf.chisq(b, 3).
execute.
```

Goldfeld-Quandt test

R code:

```
X<-model.matrix(m)
n<-dim(X)[1]
gqtest(m, n-20)
```

Stata code:

```
gen index=_n
regress immigration dissatecon abortion neutrality if index<11
regress immigration dissatecon abortion neutrality if index>=11
scalar GQ=1.8047/7.0947
scalar crit=invFtail(16,6,.05)
scalar pvalue=Ftail(16,6,GQ)
scalar list GQ pvalue crit
```

SPSS code:

```
loop num_id = 1 to 30.
end case.
end loop.
execute.
use all.
compute filter_$=(num_id < 11).
variable labels filter_$ 'num_id < 11 (filter)'.
value labels filter_$ 0 'not selected' 1 'selected'.
formats filter_$ (f1.0).
filter by filter_$.
execute.
```

SPSS code:

```
regression
/missing listwise
/statistics coeff outs r anova
/criteria=pin(.05) pout(.10)
/noorigin
/dependent immigration
/method=enter dissatecon abortion neutrality
/save resid.
compute res_1sq=res_1*res_1.
variable labels res_1sq "squared of save residuals res_1".
execute.
descriptives variables=res_1sq
/statistics=sum.
execute.
use all.
compute filter_$=(num_id >= 11).
variable labels filter_$ 'num_id >= 11 (filter)'.
value labels filter_$ 0 'not selected' 1 'selected'.
formats filter_$ (f1.0).
filter by filter_$.
execute.
regression
/missing listwise
/statistics coeff outs r anova
/criteria=pin(.05) pout(.10)
/noorigin
/dependent immigration
/method=enter dissatecon abortion neutrality
/save resid.
```

SPSS code:

```
compute res_2sq=sum(res_2*res_2).
variable labels res_2sq "squared of save residuals res_2".
execute.
descriptives variables=res_2sq
/statistics=sum.
execute.
compute GQ=1.805/7.095.
execute.
compute m=cdf.f(GQ,16,6).
compute p=1-m.
execute.
```

White's test:

R code:

```
library(lmtest)
bptest(m,~ dissatecon * abortion+dissatecon * neutrality+abortion *
neutrality+I(dissatecon ^ 2)+I(abortion ^ 2)+I(neutrality ^ 2), data=lisbon)
```

Stata code:

```
quietly reg immigration dissatecon abortion neutrality
estat imtest, white
```

SPSS code:

```
regression
/missing listwise
/statistics coeff outs r anova
/criteria=pin(.05) pout(.10)
/noorigin
/dependent immigration
/method=enter dissatecon abortion neutrality
/save resid.
```

SPSS code:

```
compute esq = res_1 * res_1.
execute.
save outfile= 'test.sav'
keep= esq dissatecon abortion neutrality.
get file = 'test.sav'.
vector v=dissatecon TO neutrality /cp(3 f8.0).
compute #idx=1.
loop #cnt1=1 to 2.
loop #cnt2=#cnt1 +1 to 3.
compute cp(#idx)=v(#cnt1) * v(#cnt2).
compute #idx=#idx+1.
end loop.
end loop.
execute.
compute dissatecon2=dissatecon*dissatecon.
compute abortion2=abortion*abortion.
compute neutrality2=neutrality*neutrality.
execute.
regression
/missing listwise
/statistics coeff outs r anova
/noorigin
/dependent esq
/method=enter dissatecon abortion neutrality dissatecon2 abortion2 neutrality2
cp1 cp2 cp3
/save resid(res_2).
matrix.
compute p = 8.
get esq / variables = esq.
get res_2 / variables = res_2.
```

SPSS code:

```
compute res2_sq = res_2 &**2.
compute n = nrow(esq).
compute rss = msum(res2_sq).
compute ii_1 = make(n, n, 1).
compute i = ident(n).
compute m0 = i - ((1/n) * ii_1).
compute tss = transpos(esq)*m0*esq.
print rss
/ format = "f5.6".
print tss
/ format = "f5.6".
compute r_sq = 1-(rss / tss).
print r_sq
/ format = "f5.6".
print n
/ format = "f5.6".
print p
/ format = "f5.6".
compute wh_test = n * (1-(rss / tss)).
print wh_test
/ format = "f5.6"
/ title = "white's test".
compute sig = 1 - chicdf(wh_test,p).
print sig
/ format = "f5.6"
/ title = "significance level".
end matrix.
```

7 Spatial Autocorrelation

In these examples we use a contiguity binary matrix W and we calculate the spatial autocorrelation for the variable "energy2" of the "demdev.dta" data set for 1988.

7.1 Moran's I

$$Moran'sI = \frac{Cov(x_i, m(x_i)}{Var(x_i)}$$

R code:

```
library(ape)
dem <- read.dta("demdev.dta")
dem88 <- dem[ which(dem$year==1988), ]
W <- read.dta("W.dta", header=FALSE)
moran <- Moran.I(dem88$energy2, W)
```

In STATA we need to install the package of spatial analysis tools. To be adressed to the download page, type: `findit spatgsa`

Stata code:

```
spatwmat using W.dta , name(W)
use demdev.dta, clear
keep if year==1988
spatgsa energy2, w(W) moran
```

Unfortunately, SPSS does not give the possibility to calculate the Moran.s I test

8 Time Series

As an example of a simple univariate time-series, we will use data on energy consumption of Ireland extracted by the "demdev.dta" data set. When using other variables, those will be always extracted from the same data set.

8.1 Create and plot a time series

Here, before creating and plotting a time series, we extrapolate the "energy2" variable for Ireland from the data set.

R code:

```
library(foreign)
dem <- read.dta("demdev.dta")
varse <- c("ccode", "energy2")
energy2 <- dem[varse]
energyIre <- energy2[ which(energy2$ccode=="205"), ]
energyIre <- energyIre$energy2
ene.Ire.ts <- ts(energyIre, frequency=1, start=1951)
library(tseries)
plot.ts(ene.Ire.ts)
```

Stata code:

```
use "demdev.dta"
keep ccode year energy2
keep if ccode==205
drop ccode
tsset year
scatter energy2 year, connect(l)
```

SPSS code:

```
get
stata file='demdev.dta'.
dataset name dataset1 window=front.
save outfile= 'demireland.sav'
/keep ccode year energy2.
get file='demireland.sav'.
select if (ccode = 205).
graph
/scatterplot(bivar)=year with energy2
/missing=listwise.
```

8.2 Plot current against lagged values

R code:

```
lag.plot(ene.Ire.ts, lags = 1, do.lines = FALSE)
```

Stata code:

```
scatter energy2 l.energy2
```

SPSS code:

```
compute energy2.1=lag(energy2).
execute.
graph
/scatterplot(bivar)=energy2.1 with energy2
/missing=listwise.
```

Plot the variable at the time $(-T)$ against itself at the time (-1)

R code:

```
lag.plot(ene.Ire.ts[-1],lags=10, do.lines=FALSE)
```

Stata code:

```
gen energy2_10=energy2[_n-10]
scatter l.energy2 energy2_10
```

SPSS code:

```
compute energy2.10=lag(energy2,10).
execute.
graph
/scatterplot(bivar)=energy2.1 with energy2.10
/missing=listwise.
```

8.3 Autocorrelation and Partial Autocorrelation functions (ACF and PCF)

Autocorrelation

R code:

```
acf(ene.Ire.ts)
```

Stata code:

```
ac energy
```

SPSS code:

```
acf variables=energy2
/nolog
/mxauto 16
/serror=ind.
```

Partial Autocorrelation

R code:

```
pacf(ene.Ire.ts), lag.max=20
```

Stata code:

```
pac energy, lags(20)
```

SPSS code:

```
pacf variables=energy2
/nolog
/mxauto 16.
```

8.4 Dickey-Fuller test

$$\Delta y_t = (\alpha - 1) y_{t-1} + \varepsilon_t = \beta y_{t-1} + \varepsilon_t$$

where the null hypothesis is that the series has a unit root, that is :

$$H_0 : (\beta) = 0$$

$$H_1 : (\beta) < 0$$

comment:

Since here there is not problem for critical values, and the regression in SPSS can be implemented without problems

R code:

```
adf.test(energyIre, k=0)
```

Stata code:

```
dfuller energy, lags(0)
```

In SPSS does not print test of stationarity, but it can be reached making a regression

SPSS code:

```
compute energy2.1=lag(energy2,1).
create
/energy2_1=diff(energy2.1 1).
regression
/missing listwise
/statistics coeff outs r anova
/criteria=pin(.05) pout(.10)
/origin
/dependent energy2_1
/method=enter energy2.1.
```

Augmented Dickey-Fuller test

$$\Delta y_t = \beta y_{t-1} + \sum_{k=1}^{p} \delta_k \Delta y_{t-k} + \varepsilon_t$$

Under the null hypothesis of a unit root, $\Delta y_t \sim I(0)$ and $y_t \sim I(1)$, so t-test is invalid. The critical values are different of the one-sided critical values from the standard normal distribution (they are larger, that is more negative). Here we show Large-sample critical values of the Augmented Dickey-Fuller Statistic Stock and Watson (2011, ch 12). Then, when making the ADF test in SPSS, we need to consider these critical values instead of those given in default by the software.

Table 1: Large-sample critical values of the Augmented Dickey-Fuller Statistic

Deterministic regressors	10%	5%	1%
Intercept only	-2.57	-2.86	-3.43
Intercept and time trend	-3.12	-3.41	-3.96

R code:

```
adf.test(energyIre)
```

Stata code:

```
dfuller energy, lags(2)
```

SPSS code:

```
compute energy2.1=lag(energy2,1).
compute energy 2.2=lag(energy2.1,1).
execute.
create
/energy2_1=diff(energy2.1 1).
create
/energy2_2=diff(energy2.2 1).
regression
/missing listwise
/statistics coeff outs r anova
/criteria=pin(.05) pout(.10)
/origin
/dependent energy2_1
/method=enter energy2.1 energy2_2.
```

8.5 Cointegration test

Here we test cointegration between "energy2" and "laggdppc" for Ireland from the data set "demdev.dta". In R, while energy2 for Ireland has been already defined (8.1), the "laggdppc" has to be defined. In Stata and SPSS, data for Ireland have been already selectioned ate the beginning of the analysis (8.1).

Philips-Perron and Engle & Granger - ADF test procedure (for $\sim CI(1,1)$)

For both test the null hypothesis is that there is not cointegration. The Philips Perron test is implemented in R, while Engle-Granger ADF test is implemented in Stata and SPSS. Critical values for Engle-Granger ADF test are showed in the next table Stock and Watson (2011, ch 12).

Table 2: Critical values for the Engle-Granger ADF Statistic

Number of X's	10%	5%	1%
1	-3.12	-3.41	-3.96
2	-3.52	-3.80	-4.36
3	-3.84	-4.16	-4.73
4	-4.20	-4.49	-5.07

R code:

```
install.packages("urca")
library(urca)
vars <- c("ccode", "laggdppc") laggdppcIre <- dem[vars] laggdppcIre <-
laggdppcIre[ which(laggdppcIre$ccode=="205"), ]
laggdppcIre <- laggdppcIre$laggdppc
Ire <- cbind(laggdppcIre, energyIre)
Ire <- data.frame(Ire)
Ire.po <- ca.po(Ire, type="Pz")
summary(Ire.po)
```

Stata code:

```
regress energy2 laggdppc
predict e, resid
dfuller e, lags(2)
```

SPSS code:

```
regression
/missing listwise
/statistics coeff outs r anova
/criteria=pin(.05) pout(.10)
/noorigin
/dependent energy2
/method=enter laggdppc
/save resid.
compute res_1.1=lag(res_1,1).
compute energy res_1.2=lag(res_1.1,1).
execute.
create
/res_1_1=diff(res_1.1 1).
create
/res_1_2=diff(res_1.2 1).
regression
/missing listwise
/statistics coeff outs r anova
/criteria=pin(.05) pout(.10)
/origin
/dependent res_1_1
/method=enter res_1.1 res_1_2.
```

8.6 Durbin-Watson test for autocorrelation of residuals

$$d = \frac{\sum_{i=2}^{n}(e_i - e_{i-1})^2}{\sum_{i=1}^{n} e_i^2}$$

where the null hypothesis is that there is not serial autocorrelation between errors.

R code:

```
library(lmtest)
dwtest(energyIre~laggdppcIre, data=Ire))
```

Stata code:

```
tsset year
reg energy2 laggdppc
dwstat
```

SPSS code:

```
regression
/missing listwise
/statistics coeff outs r anova
/criteria=pin(.05) pout(.10)
/noorigin
/dependent energy2
/method=enter laggdppc
/residuals durbin.
```

9 Causal analysis

9.1 Instrumental variables

Theory and examples here showed find a source in Stock and Watson (2011, ch 12). The data set on annual per capita cigarettes sales for 48 states between 1985 and 1995, which is available for the textbook by Stock and Watson (2011, ch 12). This can be found directly online and opened as follows:

R code:

```
library(foreign)
cigs <- read.dta("http://fmwww.bc.edu/ec-p/data/stockwatson/cig_ch10.dta")
```

Stata code:

```
use http://fmwww.bc.edu/ec-p/data/stockwatson/cig_ch10
```

In SPSS, you will have to download the file manually using your web browser at `http://fmwww.bc.edu/ec-p/data/stockwatson/cig_ch10.dta`, after which you can open it with:

SPSS code:

```
get
stata file = 'cig_ch10.dta'.
```

The description of data set can be visualised in .doc by: "http://fmwww.bc.edu/ec-p/data/stockwatson/cigarette.doc". The instrumental variable model is estimated because of the endogeneity between price and consumption. I.e. the price of cigarettes increases based on higher demand and demand reduces if the price goes up. So if we want to estimate the effect of price on consumption, we need an instrument that explains price, but not consumption, so that if there is a relation between the instrument and consumption, it must have been because of the impact through price. Such an instrument would be taxes on cigarettes: higher taxes means a higher price, but taxes are not themselves influencing cigarette consumption. In other words, higher taxes influence lower consumption only through the effect on price. Here we consider only the data on 1995:

R code:

```
cigs95<-cigs[which(cigs$year==1995),]
```

Stata code:

```
keep if year==1995
```

SPSS code:

```
select if (year = 1995).
```

Data preparation

Since we will analyse real prices and incomes, we need to generate new variables. In particular:

- "avgprsr" is the real average price during fiscal year, including sales taxes
- "taxr" is the real average state, federal, and real average local excise taxes for fiscal year
- "taxsr" is the real average excise taxes for fiscal year, including sales taxes
- "taxsor" is the real average sales tax per pack during fiscal year
- "incrpc" is the real per capita state income

R code:

```
cigs95 <- within(cigs95, { avgprsr <- avgprs / cpi
taxr <- tax / cpi
taxsr <- taxs / cpi
taxsor <- taxsr - taxr
lpackpc <- log(packpc)
lavgprsr <- log(avgprsr)
incrpc <- income / (pop * cpi)
lincrpc <- log(incrpc) })
```

Stata code:

```
gen avgprsr = avgprs/cpi
gen taxr = tax/cpi
gen taxsr = taxs/cpi
gen taxsor = taxsr-taxr
gen lpackpc = log(packpc)
gen lavgprsr = log(avgprsr)
gen incrpc = income/(pop*cpi)
gen lincrpc = log(incrpc)
```

SPSS code:

```
compute avgprsr = avgprs/cpi.
compute taxr = tax/cpi.
compute taxsr = taxs/cpi.
compute taxsor = taxsr-taxr.
compute lpackpc = ln(packpc).
compute lavgprsr = ln(avgprsr).
compute incrpc=income/(pop*cpi).
compute lincrpc=ln(incrpc).
```

9.1.1 The IV Estimator with a single regressor and a single instrument: the model

The basic model we are interested in is:

$$log(packpc_i) = \beta_0 + \beta_1 log(avgprsr_i) + u_i$$

First Stage Regression

Because of the endogeneity, we use "taxsor" as an instrument to estimate the price: log(avgprs). The first stage regression is the following:

$$log(avgprs_i) = \beta_0 + \beta_1 taxsor_i + u_i$$

Second Stage Regression

The second stage regression is then:

$$log(packpc_i) = \beta_0 + \beta_1 lavg\hat{p}rsr_i + u_i$$

Because just running the OLS estimates separately will lead to incorrect standard errors, we will use the packages for 2SLS as implemented.

2SLS estimates

[2] The output will provide both the first stage results and the second stage results.

R code:

```
install.packages("sem")
library(sem)
tsls95<-tsls(lpackpc~lavgprsr, instruments=~taxsor, data=cigs95)
```

Stata code:

```
ivregress 2sls lpackpc (lavgprsr = taxsor), first
```

SPSS code:

```
tset newvar=none.
2sls lpackpc with lavgprsr
/instruments taxsor
/constant.
```

9.1.2 The general IV regression model

Since the cigarettes demand is affected by income too, it is part of the error term of the population regression. If the real sales tax per pack (our instrument) is related to the real income per capita, it is correlated with a variable in the error term of the cigarette demand equation, which violates the instrument exogeneity condition. Consequently, the obtained estimator is

[2] Alternatively to the `sem` package, the `AER` package provides the `ivreg` command.

inconsistent. This means that the IV regression above suffers of an omitted variable bias. To solve this problem, we include the real income per capita in the regression, where the logarithm of the real per capita state income ($log(incrpc_i)$) is the exogenous variable, and the instrument is the real sales tax per pack ($taxsor$):

$$log(packpc_i) = \beta_0 + \beta_1 log(avgprsr_i) + log(incrpc_i) + u_i$$

R code:

```
reg95g<-tsls(lpackpc~lavgprsr+lincrpc , instruments=~taxsor+lincrpc,data=cigs95)
```

Stata code:

```
ivregress 2sls lpackpc lincrpc (lavgprsr = taxsor)
```

SPSS code:

```
tset newvar=none.
2sls lpackpc with lavgprsr lincrpc
/instruments taxsor lincrpc
/constant.
```

We can use another instrument for log of average price. In particular, we can use the cigarette specific tax ($taxr$) since it may provoke an increase of the price of cigarettes, and the condition for instrument relevance is probably obtained.

R code:

```
tsls95.2<-tsls(lpackpc~lavgprsr+lincrpc, instruments=~ taxsor+
taxr+lincrpc,data=cigs95)
```

Stata code:

```
ivregress 2sls lpackpc lincrpc (lavgprsr = taxsor taxr)
```

SPSS code:

```
tset newvar=none.
2sls lpackpc with lavgprsr lincrpc
/instruments taxsor taxr lincrpc
/constant.
```

9.1.3 Instrument Validity

Instrument relevance

In the case the instruments are weak, the 2SLS estimator is biased, the $t$statistic is unreliable. To test if instruments are not weak, in the case of a single endogenous regressor, the rule of thumb is that the F-statistic in the first-stage regression must be > 10. In our case, the F-statistic must be evaluated in the following regression:

$$log(avgprsr_i) = taxsor_i + taxr_i + u_i$$

Instrument Exogeneity

In the case the instruments are not exogenous, the 2SLS estimator is not consistent. Wheter it is not possible to test the hypothesis that the instruments are exogenous when the coefficient are exactly identified (the number of instruments equals the number of endogenous regressors), it is possible to test the hypothesis that the extra instruments are exogenous (overidentification restriction test), under the hypothesis that there are enough valid instruments to identify the endogenous regressor [3].

Let's consider the residuals of our 2SLS model:

$$\hat{u}_i^{2SLS} = log(packpc_i) - (\hat{\beta}_0{}^{2SLS} + \hat{\beta}_1{}^{2SLS} log(avgprsr_i) + \hat{\beta}_2{}^{2SLS} log(incrpc_i))$$

The idea is to test whether the instruments are not correlated with these reasiduals. Then, we need to check the F-test for the following model:

$$\hat{u}_i^{2SLS} = \delta_0 + \delta_1 taxsor_i + \delta_2 taxr_i + \delta_3 log(incrpc_i) + e_i$$

[3] If there is exact identification, the only possible test for exogeneity is a theorical study of the empirical problem.

The J-test is

$$J = mF$$

where m is the number of instruments. If the errors of this model are homoschedastic, J is distributed, in large samples, as a χ^2_{m-k} where k is the number of endogenous regressors. The null hypothesis is that the instruments are exogenous.

The idea behind the J-statistic is that of estimating different 2SLS estimators using each instrument separately. If these estimators are similar, it means that instruments are exogenous. In the case the estimates are very different, it means that at least one of the instruments is not exogenous. This comparison is not possible if there is only one instrument and, in fact, if $m = 1$, the J-statistic is equal to 0.

R code:

```
tsls95.2 <- tsls(lpackpc∼lavgprsr+lincrpc , instruments=∼taxsor+taxr+lincrpc,
data=cigs95)
regJ <- lm(residuals(tsls95.2)∼taxsor+ taxr+lincrpc, data=cigs95)
summary(regJ)
Jstatistic <- 2*summary(regJ)$fstatistic[1]
m <- 2
k <- 1
1-pchisq(Jstatistic,m-k)
```

Stata code:

```
ivregress 2sls lpackpc lincrpc (lavgprsr = taxsor taxr)
predict ehat, res
reg ehat taxsor taxr linprc
gen m=2
gen k=1
generate Jstatistic=e(F)*m
display 1-chi2(m-k,Jstatistic)
```

SPSS code:

```
tset newvar=none.
2sls lpackpc with lavgprsr lincrpc
/instruments taxsor lincrpc taxr
/constant.
compute Jstatistic=0.1023*2
compute m=2
compute k=1
compute Jdistchi=1-cdf.chisq(Jstatistic,1).
```

Hausman test

$$H_0 : \hat{\beta_0}^{OLS} = \hat{\beta}_{IV}^{OLS}$$

$$H_1 : \hat{\beta_0}^{OLS} \neq \hat{\beta}_{IV}^{OLS}$$

R code:

```
ols <- lm( lpackpc ~ lavgprsr )
iv <-tsls(lpackpc~lavgprsr , instruments=~taxsor+taxr,data=cigs95)
cf_diff <- coef(iv) - coef(ols)
vc_diff <- vcov(iv) - vcov(ols)
x2_diff <- as.vector(t(cf_diff) %*% solve(vc_diff) %*% cf_diff)
1-pchisq(x2_diff, df = 2)
```

Stata code:

```
ssc install ivreg2, replace
ssc install ranktest
ivreg2 lpackpc (lavgprsr=taxsor txsr)
estimates store iv
ivreg2 lpackpc lavgprsr
estimates store ols
hausman iv ols, force
```

Unfortunately, the Hausman test is not implemented in SPSS.

9.2 Matching

Here we use the `ess.dta` data set. It is extrapulated from the European Social Survey of 2010 for Ireland. In particular we consider 500 observations for the variable "union" as treatment variable; "turnout" as outcome and "trust_pol" and "eduyrs" as covariates.

9.2.1 Average Treatment Effect

As difference in means between treated and control units

R code:

```
library(foreign)
ess <- read.dta("ess.dta")
DiM <- lm(turnout~union, data=ess)
```

Stata code:

```
use "ess.dta",clear
ttest turnout, by(union)
```

As a regression estimate

R code:

```
RE <- lm(turnout~union+educ+trust_pol, data=ess)
```

Stata code:

```
reg turnout union educ trust_pol
```

9.2.2 Mahalanobis Distance Nearest Neigbor Matching

Here we perform 1:1 matching with replacement on the Mahalanobis distance to estimate the ATET.

Mahalanobis distance matching 1:1 and checking for balance

Note that, in R, the `Match()` command cannot deal with missing data

R code:

```
library(Matching)
vars.use <- c("turnout", "union", "educ", "trust_pol")
cc <- complete.cases(ess[, vars.use])
ess.use <- ess[cc, vars.use]
X <- cbind(ess.use$trust_pol, ess.use$educ)
mahaout <- Match(Tr=ess.use$union, X=X, estimand="ATT", M=1, ties=TRUE,
distance.tolerance = 1e-15, replace=TRUE, Weight=2)
```

Stata code:

```
ssc install nnmatch, replace
ssc install psmatch2, replace
psmatch2 union, mahalanobis(trust_pol educ)
```

Balance test

R code:

```
mb.mahaout <- MatchBalance (union~educ+trust_pol, nboots=0, ks=FALSE,
match.out=mahaout, data=ess.use)
```

Stata code:

```
pstest trust_pol educ
```

Average Treatment Effect on Treated

R code:

```
result.maha <- Match(Y=ess.use$turnout, Tr=ess.use$union, X=X, estimand="ATT",
M=1, ties=TRUE, distance.tolerance = 1e-15, replace=TRUE, Weight=2,
Var.calc=0, BiasAdjust = TRUE)
summary(result.maha)
```

Stata code:

```
rename trust_pol trustpol
nnmatch turnout union trustpol educ, tc(att) metric(maha) biasadj(bias) m(1)
robust(0)
```

9.2.3 Propensity Score Nearest Neighbor Matching

Predict the propensity score and match

R code:

```
library(Matching)
vars.use <- c("turnout", "union", "educ", "trust_pol")
cc <- complete.cases(ess[, vars.use])
ess.use <- ess[cc, vars.use]
logit <- glm(union~trust_pol+educ, family=binomial, data=ess.use)
summary(logit)
pscore <- logit$fitted
psout<-Match(Tr=ess.use$union, X=pscore, estimand="ATT", M=1, ties=FALSE,
distance.tolerance=1e-15, replace=TRUE
```

Stata code:

```
psmatch2 union educ trust_pol , logit
```

Balance test

R code:

```
mb.psout <- MatchBalance (union~educ+trust_pol, nboots=0, ks=FALSE,
match.out=psout, data=ess.use)
```

Stata code:

```
pstest trust_pol educ
```

Average Treatment Effect on Treated

R code:

```
result.ps <- Match(Y=ess.use$turnout, Tr=ess.use$union, X=pscore,
estimand="ATT", M=1, ties=TRUE, distance.tolerance = 1e-15, replace=TRUE,
Var.calc=0)
summary(result.ps)
```

Stata code:

```
nnmatch turnout union _pscore, tc(att)
```

10 Maximum Likelihood

Since SPSS does not permit to run a Maximum Likelihood estimation, we will consider only R and STATA.

10.1 Maximum Likelihood in pills

- In a regression analysis we estimate the β-coefficients by finding the maximum point of the (log) likelihood function;
- looking first at just univariate models, we can express the model as the probability density function $f(\mathbf{y}, \theta)$, where $\mathbf{y}$ is the dependent variable and θ the set of parameters and $f(\cdot)$ expresses the model specification;
- the Maximum Likelihhod estimator is purely parametric. We need to make strong assumptions about the shape of $f(\cdot)$ before we can estimate θ;
- given θ, $f(\cdot, \theta)$ is the Probability Density Function of $\mathbf{y}$, while given $\mathbf{y}$, $f(\mathbf{y}, \cdot)$ is the likelihood function. Our aim is to find out θ that maximizes the likelihood function;
- we consider the joint density of the entire sample. Since we consider observations are independent, the likelihood function is $f(\mathbf{y}, \theta) = \Pi_{i=1}^{n} f(y_i, \theta)$;
- it is easier to consider the log since sums are easier to deal with than products and the logarithmic trasformation is monotic:$l(\mathbf{y}, \theta) \equiv log(\mathbf{y}, \theta) = \sum_{i=1}^{n} log f_i(y_i, \theta) = \sum_{i=1}^{n} l_i(y_i, \theta)$.

10.2 Example: Maximum Likelihood estimator of a linear model

$$\mathbf{y} = \mathbf{X}\beta + \varepsilon$$

$$\varepsilon \sim N(\mathbf{0}, \sigma^2 \mathbf{I})$$

$$f_i\left(y_i, \beta, \sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - \mathbf{x}_i\beta)^2}{2\sigma^2}}$$

$$f\left(\mathbf{y}, \beta, \sigma^2\right) = \Pi_i^n f_i\left(y_i, \beta, \sigma^2\right)$$

$$l\left(\mathbf{y}, \beta, \sigma^2\right) = -\frac{n}{2} log\sigma^2 - \frac{n}{2} log 2\pi - \frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\beta)'(\mathbf{y} - \mathbf{X}\beta)$$

We use the data set `ESS.dta`. In particular the variable *trust_pol* as dependent and *happy* and $safe$ as independent. In R it is necessary to eliminate missing values from the data set we are using for the analysis.

R code:

```
ESS <- read.dta("ESS.dta")
vars <- c("trust_pol", "happy", "safe")
ESSML <- ESS[vars]
ESSMLnm <- na.omit(ESSML)
y <- ESSMLnm$trust_pol
X <- cbind(1,ESSMLnm$happy, ESSMLnm$safe)
ll <- function(par, X, y) {
s2 <- exp(par[1])
beta <- par[-1]
n <- length(y)
r <- y - X %*% beta
- n/2 * log(s2) - n/2 * log(2*pi) - 1/(2*s2) * sum(r^2)
}
mlest <- optim(c(1,0,0,0), ll, NULL, X=X, y=y, control = list(fnscale = -1),
hessian=TRUE)
beta <- mlest$par[-1]
se <- sqrt(diag(solve(-mlest$hessian))[-1])
s2 <- exp(lmest$par[1])
```

Practical tricks for R: by default, *optim*() minimizes rather than maximizes, so we need the option `control = list(fnscale = -1)` to convert it to a maximizing algorithm. To get $\hat{\sigma}^2$, we need to take the exponent of the first parameter. This is done to make sure σ^2 is always positive.

Stata code:

```
program ols
version 12
args lnf xb lnsigma
local y "$ML_y1"
quietly replace `lnf' = ln(normalden(`y', `xb',exp(`lnsigma')))
end
ml model lf ols (xb:  trust_pol = happy safe) (lnsigma:)
ml maximize
```

10.3 Example: Maximum Likelihood estimator of binary logistic regression

Here we consider the data set `lisbon.dta`, that does not contain missing values. The dependent variable is *yesvote* and independent are *immigration* and $female$. The systematic part of a logistic regression is represented by:

$$\pi_i = \frac{1}{1 + e^{-\mathbf{x}_i\beta}}$$

The stochastic part is:

$$P(y_i = 1|\mathbf{x}_i, \beta) = \pi_i^{y_i}(1 - \pi_i)^{1-y_i}$$

This leads to the likelihood function:

$$f(\mathbf{y}|\pi_i) = \Pi_{i=1}^{n}\pi_i^{y_i}(1 - \pi_i)^{1-y_i}$$

and the loglikelihood function is:

$$l(\mathbf{y}) = \sum_{i=1}^{n} y_i log\pi_i + \sum_{i=1}^{n}(1 - y_i)log(1 - \pi_i)$$

Loglikelihood and estimation

R code:

```
ll <- function(theta,x,y){
beta <- theta[1:ncol(x)]
p <- (-x%*%beta)
sum(y*log(1/(1+exp(p))))+sum((1-y)*log(1-(1/(1+exp(p)))))} y <- d$yesvote
x <- cbind(d$immigration, d$female)
X <- cbind(1,x)
mlest <- optim(c(0,0,0), ll, NULL, control = list(fnscale = -1), x=X, y=y,
hessian=TRUE)
beta <- mlest$par
se <- sqrt(abs(diag(solve(mlest$hessian))))
t <- mlest$par/se
pval <- 2 * (1-pnorm(abs(t)))
```

Stata code:

```
capture program drop lflogit
program lflogit
version 1
args lnf xb
local y "$ML_y1"
quietly replace `lnf' = ln( invlogit(`xb')) if `y'==1
quietly replace `lnf' = ln(1-invlogit(`xb')) if `y'==0
end
ml model lf lflogit (yesvote = immigration female)
ml maximize
```

11 Limited dependent variables

When a dependent variable is not continuous, or is truncated for some reason, a linear model would lead to implausible predictions.

Furthermore, estimating limited dependent variable data with a linear model implies serious heteroscedasticity.

For these reasons we need different models. In this chapter we use the `ess.dta` data set. In particular we consider the following model:

$$Turnout_i = \beta_0 + \beta_1 Trust\_pol_i + \beta_2 Female_i + \beta_3 Happy_i + \beta_4 Interest_i + \beta_5 Educ_i + \varepsilon_i$$

11.1 Binary Dependent Variables: Logistic Regression

Stochastic component:

$$y_i = Bernoulli(\pi_i) = \pi_i^{y_i}(1-\pi_i)^{1-y_i}$$

Systematic component:

$$\pi_i = g(\mathbf{x}_i\beta) = \frac{1}{1+e^{-\mathbf{x}_i\beta}}$$

R code:

```
library(foreign)
ess <- read.dta("ess.dta")
m<- glm(turnout ~ happy + satisf_eco + age + union, data=ess,
family=binomial(link="logit"))
summary(m)
```

Stata code:

```
logit turnout happy satisf_eco age union
```

SPSS code:

```
logistic regression variables turnout
/method=enter happy satisf_eco age union
/criteria=pin(.05) pout(.10) iterate(20) cut(.5).
```

Grafical interpretation: Plot predicted probabilities by a variable

We plot the logistic regression against the variable `age`.

R code:

```
invlogit <- function(x) { 1 / (1 + exp(-x))}
plot(jitter(turnout) ~ jitter(age), data = ess, pch = 19, bty = "n")
curve(invlogit( cbind(1, median(ess$happy, na.rm = TRUE),
median(ess$satisf_eco, na.rm = TRUE), x, median(ess$union,na.rm=TRUE)) %*%
coef(m)), from = 15, to = 101, lwd = 2, col = "red", add = TRUE)
```

Stata code:

```
egen Mhappy=median(happy)
egen Msat=median(satisf_eco)
egen Munion=median(union)
generate
f = 1 / ( 1 + exp ( - ( ( _b[_cons] ) + ( _b[age] ) * age + ( _b[happy] ) *
Mhappy ) + ( _b[satisf_eco] ) * Msat ) + ( _b[union] ) * Munion )
twoway (line f age)
```

Unfortunately, in SPSS we need to take up numbers manually.

SPSS code:

```
frequencies variables= happy satisf_eco union
/statistics=median
/order=analysis.
```

It results that the median for *happy*, *satisf_eco* and *union* are 7, 2, 0 respectively. Taking the coefficients form the regression,

SPSS code:

```
compute fun=1/(1 + exp(-(-1.99 + .044 * age + 0.184 * 7 - .137 * 2 + 0.88 *
0))).
execute.
graph
/line(simple)=value(fun) by age.
```

Predicted differences between interesting cases

Here we calculate the predicted differences between being part and not be part of an union.

R code:

```
invlogit <- function(x) { 1 / (1 + exp(-x))}
x <- cbind(1,
median(ess$happy, na.rm = TRUE),
median(ess$satisf_eco, na.rm = TRUE),
c(0,1),
median(ess$age,na.rm=TRUE))
xb <- x %*% coef(m)
phat <- invlogit(xb)
```

Stata code:

```
logit turnout happy satisf_eco age i.union
margins union, atmeans
```

Again, SPSS requires to make calculations writing data manually. After verifying that the meadian for the variable `age` is 44, we look for the predicted differences between being part and not be part of an union in the following way:

SPSS code:

```
compute predicted_logit=1/(1 + exp(-( -1.992 + (.184 * 7) - ( .137 * 2 ) +
(.045 * 44) + (.881 * union)))).
execute.
```

11.2 Threshold models: Probit Model

Let consider a latent, unobserved variable:

$$\mathbf{y}* \sim f(\mu)$$

$$\mu = \mathbf{X}\beta$$

And the following observation mechanism:

$$y_i = \begin{cases} 1 & \text{if } y_i^* > 0 \\ 0 & \text{if } y_i^* \leq 0 \end{cases}$$

If $f(\mu_i)$ is the standardised logistic distribution, we replicate the logit model.

$$f(\mathbf{x}_i\beta) = \frac{e^{y_i*-\mu_i}}{(1+e^{y_i*-\mu_i})^2}$$

If $f(\mu_i)$ is the cumulative standard normal distribution ($\sigma^2 = 1$), we have the probit model. Hence β can be now interpreted as the effect of an increase by 1 in $\mathbf{x}$ on $\mathbf{y}*$, whereby the unit of $\mathbf{y}*$ is one standard deviation.

$$P(y_i = 1|\beta, x_i) = \Phi(x_i\beta)$$

whereby $\Phi(x)$ is the cumulative standard normal distribution. Stochastic component:

$$y_i = Bernoulli(\pi_i) = \pi_i^{y_i}(1-\pi_i)^{1-y_i}$$

Systematic component:

$$\pi_i = \Phi(x_i\beta)$$

Let's consider the same model of before:

$$turnout_i = \beta_0 + \beta_1 trust\_pol_i + \beta_2 female_i + \beta_3 happy_i + \beta_4 interest_i + \beta_5 educ_i + \varepsilon_i$$

R code:

```
m2 <- glm(turnout ~ happy + satisf_eco + age + union, data=ess, family =
binomial(link = "probit"))
```

Stata code:

```
probit turnout happy satisf_eco age union
```

SPSS code:

```
genlin turnout (reference=0) with happy satisf_eco age union
/model happy satisf_eco age union
distribution=binomial
link=probit
/print cps history fit solution.
```

11.3 Ordinal Dependent Variable

11.3.1 Order Probit

In this case we consider a categorical dependent variable, with categories in a particular order. We can thus take a similar latent variable approach.

Stochastic Component:

$$\mathbf{y}* \sim N(\mu_i, 1)$$

Systematic Component:

$$\mu_\mathbf{i} = \mathbf{x}_i\beta$$

with the following observation mechanism:

$$y_i = j \quad if \quad \tau_{j-1} \leq y_i^* \leq \tau_j$$

In R, we need to trasform our dependent variable in a factor

R code:

```
satisf_gov <- factor(ess$satisf_gov)
library(MASS)
o <- polr(satisf_gov ~ ess$immigrants + ess$satisf_eco + ess$trust_pol +
ess$female + ess$health + ess$traffic, method="probit")
ctable <- coef(summary(o))
p <- pnorm(abs(ctable[, "t value"]), lower.tail = FALSE) * 2
ctable <- cbind(ctable, `p value` = p)
```

Stata code:

```
oprobit satisf_gov immigrants satisf_eco trust_pol female health traffic
```

SPSS code:

```
plum satisf_gov with immigrants satisf_eco trust_pol female health traffic
/criteria=cin(95) delta(0) lconverge(0) mxiter(100) mxstep(5)
pconverge(1.0e-6) singular(1.0e-8)
/link=probit
/print=fit parameter summary.
```

11.4 Multinomial Logit

Here the dependent variable consists of multiple categories, without particular order. A latent variable approach will thus not work.

Estimation and predicted probabilities

R code:

```
library(nnet)
v <- multinom(party ~ female + religion + trust_pol + union, data=ess)
summary(v)
probs <- predict(v, type="probs")
```

Stata code:

```
mlogit party female religion trust_pol union, base(1)
predict FineGae GreenParty Independent Labour PeopleBeforeProfit SinnFein
SocialistParty UnitedLeftAlliance
```

In Stata, we use FiannaFai as base.

SPSS code:

```
nomreg party (base=first order=ascending) with female religion trust_pol union
/criteria cin(95) delta(0) mxiter(100) mxstep(5) chksep(20) lconverge(0)
pconverge(0.000001)
singular(0.00000001)
/model
/stepwise=pin(.05) pout(0.1) mineffect(0) rule(single) entrymethod(lr)
removalmethod(lr)
/intercept=include
/print=parameter summary lrt cps step mfi.
```

11.5 Count Models

Here the dependent variable is a count (of events). The dependent variable values are truncated at zero (no negative outcomes are possible), they have not an upper limit and are limited by time, place or both.

$$Patent_i = log(Gdp_i) + rdemploy_i + rdexp_i + educ_i + \varepsilon_i$$

11.5.1 Poisson Regression

Stochastic component:

$$y_i = Poisson(\lambda_i)$$

Systematic component:

$$\lambda_i = e^{\mathbf{x}_i\beta}$$

Possion distribution:

$$f(k, \lambda) = \frac{\lambda^k e^{-\lambda}}{k!}$$

Here we use the dataset `Patent.dta`, where the dependent variable is the number of patents presented at the European Patent Office by scientists living in European NUTS 2 Regions for 2009.

In many cases a patent is the results of scientists living in different regions. For this reason,

the number of patents appears with decimals, since one patent is divided by the number of scientists concurred to develop it[4].

R code:

```
library(MASS)
patent <- read.dta("Patent.dta")
model <- glm(patent ~ log(gdp) + rdemploy + rdexp + educ, family=poisson,
data=Patent)
```

Stata code:

```
poisson patent ln(gdp) rdemploy rdexp educ, vce(robust)
```

SPSS code:

```
genlin patent with log(gdp) rdemploy rdexp educ
/model log(gdp) rdemploy rdexp educ intercept=yes
distribution=poisson link=log
/criteria method=fisher(1) scale=1 covb=model maxiterations=100
maxstephalving=5
pconverge=1e-006(absolute) singular=1e-012 analysistype=3(wald) cilevel=95
citype=wald
likelihood=full
/missing classmissing=exclude
/print cps descriptives modelinfo fit summary solution.
```

11.5.2 Negative Binomial

Stochastic component:

$$y_i \sim NegBin(\phi, \sigma^2)$$

Systematic component:

$$\phi = e^{\mathbf{x}_i\beta} = E(y_i)$$

[4]For more information, consult: http://epp.eurostat.ec.europa.eu/statistics_explained/index.php/Science_and_technology_at_regior

R code:

```
library(MASS)
model <- glm.nb(patent ~ log(gdp) + rdemploy + rdexp + educ, data=Patent)
```

Stata code:

```
nbreg patent ln(gdp) rdemploy rdexp educ
```

SPSS code:

```
genlin patent with log(gdp) rdemploy rdexp educ
/model log(gdp) rdemploy rdexp educ intercept = yes
distribution=negbin(1) link=log
/criteria method=fisher(1) scale=1 covb=model maxiterations=100
maxstephalving=5
pconverge=1e-006(absolute) singular=1e-012 analysistype=3(wald) cilevel=95
citype=wald
likelihood=full
/missing classmissing=exclude
/print cps descriptives modelinfo fit summary solution.
```

12 Multilevel Data

12.1 Fixed effects Models

In essence, we thus have different intercepts for each group.

$$y_i = \mathbf{x}_i\beta + \mu_{j[i]} + \varepsilon_i$$

whereby i denotes the individual unit, j the group, and $j[i]$ the group of i.

In this section we use the data set "ess.dta". In particular we consider the following model:

$$satisf\_gov_i = \beta_0 + \beta_1 trust\_pol_i + \beta_2 age_i + \mu_{party_i}$$

While R and Stata consider the first party as baseline by default, SPSS considers the last one.

R code:

```
summary(lm(satisf_gov ~ trust_pol + age + factor(party), data=ess))
```

Stata code:

```
xi:reg satisf_gov trust_pol age i.party
```

SPSS code:

```
mixed satisf_gov by party with trust_pol age
/fixed = party trust_pol age
/print=solution.
```

12.2 Random Effects

For the random effects model we still have:

$$y_i = \mathbf{x}_i\beta + \mu_{j[i]} + \varepsilon_i$$

However, this time we assume $\mu_j \sim N(0, \sigma^2_\mu)$

R code:

```
library(arm)
lmer(satisf_gov ~ trust_pol + age + (1|party), data=ess)
```

Stata code:

```
xtmixed satisf_gov trust_pol age, || party:, covariance(independent) reml
```

SPSS code:

```
mixed satisf_gov by party with trust_pol age
/fixed=trust_pol age party
/print=solution.
varcomp satisf_gov by party with trust_pol age
/random = party.
```

13 Panel data

Here we use the `demdev.dta` data set. The model considered is the following:

$$ipyears_{it} = \beta_0 + energy2_{it} + polity2_{it} + \varepsilon_{it}$$

Where:

- `ipyears` is the number of years without interstate conflict;
- `energy2` is the energy consumption, used as proxy for economic development;
- `polity2` is the score from autocracy (-10) to democracy (10).

13.1 Fixed effects

A panel data structure is similar to a multilevel structure, but with a time element.
Thus, we can think of the variation between units and within one unit over time.

$$y_{it} = \alpha_i + \mathbf{x}_{it}\beta + \varepsilon_{it}$$

Therefore, fixed unit effects are one way of dealing with the between variation.

Entity-specific intercepts

R code:

```
library(plm)
panel <- read.dta("demdev.dta")
summary(fixed <- plm(ipyears ~ energy2 + polity2, data=panel,
index=c("country", "year"), model="within")
fixef(fixed)
```

In Stata we need to "advice" that we are using panel data. In order to do that we need to have numeric variables for identifying countries and years. It is for that reason that here we use the variable `ccode` instead of `country`.

Stata code:

```
xtset ccode year
xtreg ipyears energy2 polity2, fe
estimates store fixed
```

Least squares dummy variable model

R code:

```
summary(fix.dummy <- lm( ipyears ~ energy2 + polity2 + factor(country) - 1,
data = panel)
```

Stata code:

```
reg ipyears energy2 polity2 i.ccode
```

SPSS code:

```
mixed ipyears by country with energy2 polity2
/fixed= country energy2 polity2
/print=solution.
```

13.2 Time - fixed effects

When common shocks are of concern, time fixed effects are possible:

$$y_{it} = \gamma_t + \mathbf{x}'_{it}\beta + \varepsilon_{it}$$

And one can control for both at once:

$$y_{it} = \alpha_i + \gamma_t + \mathbf{x}'_{it}\beta + \varepsilon_{it}$$

R code:

```
summary(fixed.time <- plm( ipyears ~ energy2 + polity2 + factor(year),
data=panel, index=c("ccode", "year"), model="within")
```

Stata code:

```
xi:  xtreg ipyears energy2 polity2 i.year, fe
```

Or, if you want to see the coefficient estimates for the fixed effects for the countries:

Stata code:

```
reg ipyears energy2 polity2 i.ccode i.year
```

SPSS code:

```
mixed ipyears by country year with energy2 polity2
/fixed= country year energy2 polity2
/print=solution.
```

13.3 Random effects

When we can assume α_i and $\mathbf{x}'_{it}$ are uncorrelated, we can use random effects.

$$y_{it} = \alpha_i + \mathbf{x}'_{it}\beta + \varepsilon_{it}$$

$$\alpha_i \sim N(0, \sigma_\alpha^2)$$

R code:

```
random <- plm(ipyears ~ energy2 + polity2, data=panel, index=c("country",
"year"), model="random")
```

Stata code:

```
xtreg ipyears energy2 polity2, re
estimates store random
```

13.4 Test for random effects: Breush-Pagan LM test

The null hypothesis is that $\sigma_\mu^2 = 0$, that is there is no significant difference across entities.

R code:

```
summary(pooling <- plm(ipyears ~ energy2 + polity2, data=panel,
index=c("country", "year"), model="pooling"))
plmtest(pooling, type=c("bp"))
```

Stata code:

```
xtreg ipyears energy2 polity2, re
xttest0
```

13.5 Choose between Fixed and Random Effects: Hausman Test

The random effects model assumes that $E(\mu_i \mathbf{x}_{it}) = \mathbf{0}$. Whether this is valid can be tested by looking at the differences in coefficients between the random and fixed models, using a Hausman test, where the null hypothesis is that both model are consistent.

R code:

```
phtest(fixed, random)
```

Stata code:

```
xtreg ipyears energy2 polity2, fe
estimates store fixed
xtreg ipyears energy2 polity2, re
estimates store random
hausman fixed random
```

13.6 Test for time fixed effects

Whereby the null hypothesis is that there is no significative difference between fixed effects and time fixed effects.

R code:

```
fixed <- plm(ipyears ~ energy2 + polity2, data=panel, index=c("country",
"year"), model="within")
fixed.time <- plm( ipyears ~ energy2 + polity2 + factor(year), data=panel,
index=c("ccode", "year"), model="within")
pFtest(fixed.time, fixed )
```

Stata code:

```
xi:  xtreg ipyears energy2 polity2 i.year, fe
testparm _Iyear*
```

[5]

13.7 Between Model

Where $\bar{x}_{i\cdot} = \frac{1}{T}\sum_{t=1}^{T} x_{it}$:

$$\bar{y}_{i\cdot} = \beta_0 + \bar{\mathbf{x}}'_{i\cdot}\beta + \varepsilon_i$$

R code:

```
summary(be <- plm(ipyears ~ energy2 + polity2, data = panel,model = "between",
index = ("country", "year"))
```

Stata code:

```
xtreg ipyears energy2 polity2, be
```

[5] In Stata 11 this command becomes `tesparm i.year`.

14 Simulation and Bootstrap

In this part we use the `asiabaro.dta` data set. We will consider a logit estimation and the following model:

$$abstain_i = \beta_0 + \beta_1 trustgovt_i + \beta_2 satisfdemoc_i + \beta_3 female_i + \beta_4 urban_i + \beta_5 prefdemoc_i + \varepsilon_i$$

In particular, we will use simulation and bootstrap for predicting, with a 95 % of confidence interval, the probability of someone abstaining who lives in an urban area and is "typical" on all other variables (we use the median).

14.1 Simulation

R code:

```
x.star <-c(1,median(na.omit(ab$trustgovt)),median(na.omit(ab$satisfdemoc)),
median(na.omit(ab$female)), 1 ,median(na.omit(ab$prefdemoc)))
b.star <- mvrnorm(1000, coef(m), vcov(m))
y.star <- x.star %*% t(b.star)
pi.star <- 1 / (1 + exp(-y.star))
(ci <- quantile(pi.star, c(.025, .975)))
```

In Stata is very useful the "clarify" program, that we need to download.

Stata code:

```
net from http://gking.harvard.edu/clarify/
net install clarify
use "asiabaro.dta"
logit abstain trustgovt satisfdemoc female urban prefdemoc
estsimp logit abstain trustgovt satisfdemoc female urban prefdemoc
setx trustgovt median satisfdemoc median female median urban 1 prefdemoc
median
simqi, pr listx
```

14.2 Bootstrap

Here we use 20 replications.

R code:

```
R <- 20
n <- dim(ab)[1]
x.star <- c(1, 1, 2, 1, 1, 9)
pi.star <- rep(NA, R)
for (m in 1:R) {
      bs.sample <- sample(1:n, n, replace = TRUE)
      m.bs <- glm(abstain ~ trustgovt + satisfdemoc + female + urban +
      prefdemoc, data = ab[bs.sample, ], family = binomial(link = "logit"))
      y.star <- x.star %*% coef(m.bs)
      pi.star[m] <- 1 / (1 + exp(-y.star))
     }
(ci <- quantile(pi.star, c(.025, .975)))
```

Stata code:

```
egen Mtrustgovt = median(trustgovt)
egen Msatisfdemoc = median(satisfdemoc)
egen Mfemale = median(female)
gen urban1=1
egen Mprefdemoc = median(prefdemoc)
bootstrap pred = (_b[_cons] + _b[trustgovt]*Mtrustgovt +
_b[satisfdemoc]*Msatisfdemoc + _b[female]*Mfemale + _b[urban]*urban1 +
_b[prefdemoc]*Mprefdemoc), reps(20) seed(1):  logit abstain trustgovt
satisfdemoc female urban prefdemoc
scalar low = el(e(ci_normal),1,1)
scalar high = el(e(ci_normal),2,1)
display 1/(1 + exp(-low))
display 1/(1 + exp(-high))
```

References

Stock, James H. and Mark W. Watson. 2011. *Introduction to econometrics*. 3rd ed. Prentice Hall.