Exercise Book

Mark Trede and Willi Mutschler

Version: November 16, 2015

Overview of Exercises

1 A short introduction to R

1 Quick overview of R

R, also called GNU S, is a strongly functional language and environment. It provides a wide variety of statistical (linear and nonlinear modeling, classical statistical tests, time-series analysis, classification, clustering, simulation, optimization...) and graphical techniques, and is highly extensible. To get a flavor of the possibilities just try the following commands:

- `example(lm)`
- `demo(graphics)`
- `demo(lm.glm)`
- `library(tcltk); demo(tkcanvas)`
- `RSiteSearch("GMM'')`

It is open-source and can be run under Windows and any UNIX System (MAC, Linux). It is a programming language, so we type commands („we ask") and R executes them („R answers").

R also has a rudimental GUI (Graphical User Interface), which can be used to organize the workspace, load packages and list and manipulate some objects. For instance look at `Verschiedenes`. There are buttons for the following functions:

- `objects()` or `ls()` lists all variables and objects
- `search()` lists all packages that are currently in use
- `rm(x,y,z)` removes the variables x,y and z
- `rm(list=ls())` removes everything (be careful!)

There are better GUIs like SciViewsR, JGR or RCommander. We, however, won't use them, but will instead type everything in the command window or even better in a script editor. TinnR, Notepad++ (in combination with NPPtoR) or Eclipse (in combination with StatET) are highly recommendable. In our opinion, one of the best editors is RStudio, so we will make use of its functionality! If you need assistance installing those editors, please contact us.

There are a couple of good books and manuals for R, for instance:

- Behr, Andreas (2011) - Einführung in die Statistik mit R
- Crawley, Michael J. (2007) - The R Book
- Farnsworth, Grant V. (2008) - Econometrics in R
- Verzani, John (2005) - Using R for Introductory Statistics
- The R Development Core Team (2010): An Introduction to R
- The command RSiteSearch("whatever-you-want-to-find") helps to find functions and references on the internet.

1.1 Starting and quitting

Start R. The window you see is the "R Console" and we will call it the command window in the following. Inside the command window, compute $1 + 1$, $2 - 1$, $3/2$, $2 * 4$ and 2^{10}. Quit R using the command `q()`, without saving the workspace.

1.1 Starting and quitting

During the start R searches for two files in the current working directory: .RData and .Rhistory. Those files contain your workspace and your history. You can create them by saving your workspace and saving your history (simply use the menu bar: `Datei`). The basic mathematical operators are `+,-,*,/,^`. Please try the following code:

```
##############################
### Starting and quitting ###
##############################
1+1
2-1
3/2
2*4
2^10
q()
# In order to save your workspace, you can also use
save.image(file.choose())
#In order to save your history, i.e. the commands you've executed, use
savehistory(file.choose())
#To load your workspace and history use
load(file.choose())
loadhistory(file.choose())
#better: use the GUI!!!
```

There are two very handy key shortcuts:

- ↑ and ↓ scroll through the history,i.e. all the commands you have typed so far.
- Tab (⇆) completes commands, functions and variable names

1.2 Scripts

Restart R. In the menu, choose "Datei", then "Neues Skript". A new window opens. Type the following four lines:

```
a <- 3
b <- 4
c <- a+b
print(c) pi Pi PI
```

Mark the lines and press Strg+R (or Ctrl+R). Save your script under any name (preferably with the extension `.R`) on your hard disk or USB flash drive. Quit R, restart, open the script, and execute it.

1.2 Scripts

Scripts are a great way to organize your code, since it can be very impractical to scroll through all the commands and execute them one by one. So please use an editor!

Variables: Variables are placeholders for any content you can imagine: numbers, vectors, matrices, text, dates, times, tables, data, etc. In order to assign a value or a content into a variable, use `x <- 5.6`. Note: `x <-5,6` produces an error. Try the following code:

```
###############
### Scripts ###
###############
a <- 3
b <- 4
c <- a+b
#there are three ways to print the contents of the variable
(c <- a+b) #Parentheses around a command prints its output
print(c)    #The command print()
c #Just type the name of the variable
pi
Pi
PI
```

Upper and lower cases, dot and comma: R differentiates between upper and lower cases, try: `pi`, `Pi` und `PI`. Keep that in mind when calling functions and commands, and also when naming variables. The decimal point is the dot.

1.3 Working directory

In R, you can obtain the current working directory using the command `getwd()` ("get working directory"). This is the directory where R saves files, and where it looks for files to read.

1. Find out where your working directory is.
2. You can change the working directory using the command `setwd("x:/path")` where `x:` is the drive (e.g. hard disk) and `path` the complete path. Note that the path does not contain the backslash ("\") which is usually used in Windows, but the slash ("/"). Change your working directory as you like. Check if the change was successful.

1.3 Working directory

To change the working directory you can also use the GUI: `(Datei - Verzeichnis wechseln)`.

Text and strings: You have to use quotation marks and put your text in between those. Missing quotation marks are the most common error you get. Try the following code:

```
#########################
### Working directory ###
#########################
getwd()
setwd("c:/") #important: Windows uses the backslash \, R uses the slash /
getwd()
```

1.4 Help and comments

1. A very important command in R is the question mark (`?`) followed by any name of a function. This way you can start the help function, giving you details about any R command. Read the help page for the command `mean`.

2. The hash sign (`#`) is the comment sign. Everything following the comment sign is ignored (until the end of line). Insert some comments into your script and re-execute it.

1.4 Help and comments

Help

- It is very important to get used to the extensive help functions in R.
- The most important one is the question mark: `?function` (example: `?mean`).
- If you can't remember the exact name of the function, try
  - The TAB-button ⇆.
  - `??mean` searches for aliases, concepts or titles that include `mean`.
  - `apropos("mean")` lists all objects that contain `mean`.
  - `example("mean")` gives you an example.
  - `find("mean")` gives you the name of the package containing the function mean.

Comments: You can put comments in your script using #. This is extremely important in order to keep an organized and understandable code. Please get used to comment what you are doing. Please have a look at the following code:

```
#########################
### Help and comments ###
#########################
?mean
??mean
apropos("mean")
example("mean")
find("mean")
```

1.5 Packages

An advantage of R is the large number of packages available on CRAN. Packages increase the functionality of R. If your computer is connected to the internet, you can install new packages by choosing the menu items "Pakete", "Installiere Paket(e)...".

1. Install the package `xlsx`. Then activate the package using the command `library(xlsx)`. Typing `library(help=xlsx)` will give you more information about the new commands.

2. Install the package `AER`. We will need it for the next exercises.

1.5 Packages

There are several methods to install new packages:

1. Using the GUI:
   - `Pakete - Installiere Pakete`, choose a close-by mirror.
   - Highlight AER and xlsx (using the CTRL button you can highlight several items).
2. Using the command window:
   - `install.packages("xlsx")`

After you downloaded the packages you can load them either by typing:
`library(xlsx)` and `library(AER)` or using the GUI: `Pakete - Lade Paket`.
Please have a look at the help files: `library(help=xlsx)` and `library(help=AER)`. Here's the code:

```
################
### Packages ###
#################
#If you want to specify a folder for the installation files of the package,
#use: .libPaths("X:/Your Folder")
install.packages("xlsx")
#very important: JAVA needs to be installed on your computer otherwise you get an error
library(xlsx)
library(help=xlsx)
library(AER)
```

2 Importing data into R

When learning a new computer language, the most basic standard problem is how to import data. In this exercise you will learn a number of ways to read datasets. You can type the commands either in the command window or, preferably, write and save a script and then execute it.

2.1 Reading text files

Download the file `bsp1.txt` from the course page and save it in the directory `c:/temp` (of course, you may use other directories). Change the working directory to `c:/temp`. Use the command

`bsp1 <- read.csv("bsp1.txt")`

to import data into object `bsp1` and type `print(bsp1)`, or simply `bsp1`, to see the dataset.

2.1 Reading text files

The output should look like this:

```
   Groesse Alter Tore Gehalt
1      168    21    0    1.9
2      186    20    0    1.6
3      158    21    4    3.3
4      170    20    6    1.6
(...)
```

CSV (*Comma-Separated Values*) is a format to describe structured data in a textfile (`.txt` or `.csv`). Each item is separated by a comma, semicolon or a tabstop. Often you can find the names of the variables in the first line (`header=TRUE` oder `header=FALSE`). The command `read.csv()` is used to import the data. You can change the separation symbol with `sep=`. The default value is the comma: `bsp1 <- read.csv("bsp1.txt",sep=",")`.

If you don't want to write the exact name of the file, simply use `file=file.choose()`. The code looks like this:

```
##########################
### Reading text files ###
##########################
bsp1 <- read.csv(file.choose())
```

2.2 Reading excel files

Download the excel file `bsp2.xlsx` from the course page and save it. Reading excel files is rather uncomfortable in R.

1. Open the file from Excel and save it as `bsp2.csv`. In contrast to the English version, the German version of Excel does not write a decimal point, but a comma, and entries are not separated by commas, but semicolons.[1] If your data are saved in the German format, you can read the data using one of the two following commands

   `bsp2 <- read.csv("bsp2.csv",dec=",",sep=";")`

   `bsp2 <- read.csv2("bsp2.csv")`.

   Import the dataset and have a look at it using `print(bsp2)`.

2. If you insist to read Excel files, the best way to do it is by means of the package `xlsx`. Activate the package using `library(xlsx)`. Read the help text of the command `read.xlsx`. Load the file `bsp2.xlsx` using the command `read.xlsx` and print the data.

2.2 Reading excel files

```
###########################
### Reading excel files ###
###########################
#specify the symbol for the decimal point and separator
bsp2 <- read.csv(file.choose(), dec=",",sep=";")
bsp2
bsp2 <- read.csv(file.choose()) # if you use the "English" decimal point
bsp2 <- read.csv2(file.choose()) # if you use the "German" decimal point
bsp2

#use the package xlsx
library(xlsx)
?read.xlsx
bsp2 <- read.xlsx(file.choose(),sheetIndex=1)
bsp2
```

[1]Please always check, if your Excel version uses the German or the English format.

2.3 Other data formats

1. There is a large number of packages to make foreign data formats readable in R. The most important package is `foreign`, which can be used to read SPSS and Stata files (but not Excel). Install and activate the package `foreign` and read the corresponding help with `library(help=foreign)`. Load `bsp3.dta` into the object `bsp3` and print it.
2. R also has got its own data format. You can save objects using the command `save` and then re-load them with `load`. Save the object `bsp3` in the file `bsp3.Rdata`, quit R, restart, and load the file `bsp3.Rdata`. Print `bsp3`.
3. Try `scandat <- scan()` and insert some data. Edit your data with `edit(scandat)`.

2.3 Other data formats

Please see the following code.

```
##########################
### Other data formats ###
##########################
install.packages("foreign")
library(foreign)
library(help=foreign)
bsp3 <- read.dta(file.choose())
bsp3

#R's data format
save(bsp3, file=file.choose())
rm(bsp3)
bsp3
load(file.choose())
bsp3

#Editing data
                    # you can input as many things as you like,
scandat <- scan()   # copy&paste works as well (very handy)
data.entry(scandat) # edits your data
edit(scandat)       # edits your data
```

2.4 Missing values and trimming

`NA` stands for a missing value. NaN stands for Not a Number (example `0/0`). Missing values can produce errors in some functions and you should either remove them (trimming) or replace them with a 0. Create a vector `y <- c(1:3,NA,NA,4:2)` and (i) trim or (ii) replace them with 0.

2.4 Missing values and trimming

Please see the following code.

```
###################################
### Missing values and trimming ###
###################################
```

```
0/0
y <- c(1:3,NA,NA,4:2)
y
mean(y)
is.na(y)        # a query to get NAs
which(is.na(y)) # another way to get the positions of the NAs
y[-4]           # removes the 4th entry

y <- c(1:3,NA,NA,4:2)
y[which(is.na(y))]=0 ;y # overwrites NA with 0, note: y changes!

y <- c(1:3,NA,NA,4:2)
y[-which(is.na(y))]; y # removes the NA, note: y has not changed!
```

3 Describing data in R

Imported datasets are usually stored as dataframe objects. A dataframe is almost the same as a matrix. Each row is an observation, and each column is a variable. Create a new script for the following exercises to be able to repeat the commands.

3.1 Head and tail

On the internet site of the course you will find the file `indices.csv`. It contains the daily index values of the two indices DAX and FTSE 100 from 8/9/2005 to 8/9/2010.[2] Load the data into R and save them as dataframe `indices`. Large dataframes cannot be printed nicely. A good way to learn about the structure of the dataframe is the command `head(indices)`. Try it (by the way, you can also use `tail(indices)`). If you are only interested in the variable names of the dataframe, just type `names(indices)`. For a thorough insight try also `str(indices)`, `class(indices)` and `attributes(indices)`.

3.1 Head and tail

Please have a look at the following code:

```
#####################
### Head and tail ###
#####################
indices <- read.csv2(file.choose())
head(indices)
tail(indices)
names(indices)
str(indices)        # gives you the structure and an overview of the object
class(indices)      # gives you the type of the object
attributes(indices) # gives you a very good overview of the attributes of the object
```

3.2 Attaching dataframes

If you use the `attach` command, the columns of the dataframe are accessible by the column names as ordinary variables. Now you can directly access the two variables `dax` and `ftse`. Type

[2]Since working with calendar dates is a bit cumbersome in R, the information about the dates has been omitted.

`attach(indices)`. *Note: The help page for* `attach` *notes that attach can lead to confusion: The possibilities for creating errors when using attach are numerous.* Therefore we are going to avoid it and use the $ sign to attach variables, i.e. `dax <- indices`$dax; ftse < -indices$`ftse`. Another way is to directly access the columns of the dataframe, i.e. `dax <- indices[,1]; ftse <- indices[,2]`. Save the DAX series into `dax` and the FTSE series into `ftse`.

3.2 Attaching dataframes

Please have a look at the following code:

```
############################
### Attaching dataframes ###
############################
#To access data there are at least 3 ways to do it
#1) With the dollar sign
dax <- indices$dax; ftse <- indices$ftse
print(dax)
#2) with braskets
dax <- indices[,1]; ftse <- indices[,2]
print(dax)
#3) with the attach command
attach(indices)
print(dax)
```

3.3 Simple plots

Type `plot(dax)` to create a graph showing the time series of the DAX index. Create a new graph of the time series of the logarithm of the DAX index.

3.3 Simple plots

```
####################
### Simple plots ###
####################
plot(dax)
plot(log(dax)) #compare the y-axis!
```

The graphs look the same, the y-axis, however, has changed. The range is smaller using logs.

3.4 Stock returns

1. Save the number of observations into the variable `n`. Hint: The command `dim(x)` returns the number of rows and columns of `x` as a vector.
2. Generate a new variable containing the daily returns of the DAX index:

   `rdax <- log(dax[2:n]/dax[1:(n-1)])`

   and plot them. Define and plot the returns of the FTSE in a similar way (`rftse`).
3. Activate the package `MASS`. Use the command `truehist` to draw the histogram of the DAX returns.

4. For the DAX returns, compute the mean (`mean`), the standard deviation (`sd`), the variance (`var`), the median, the 1%-and the 99% quantiles (`quantile`), and the range (`range`).
5. Sometimes boxplots are a nice way to present a dataset. Type `boxplot(rdax,rftse)`.
6. Plot the DAX returns against the FTSE returns using `plot(rdax,rftse)`.
7. Compute the correlation between the DAX returns and the FTSE returns (`cor`).
8. Compute the correlation of the DAX returns with its lagged (by one day) return.

3.4 Stock returns Please see the following code:

```
#####################
### Stock returns ###
#####################
dim(indices)
n <- dim(indices)[1]; n
head(dax)
head(dax[2:n])     # vector containing all elements of dax except the first one

tail(dax)
tail(dax[1:(n-1)]) # vector containing all elements of dax except the last one
rdax <- log(dax[2:n]/dax[1:(n-1)])
head(rdax)
length(rdax)
rftse <- log(ftse[2:n]/ftse[1:(n-1)])

plot(rdax)
library(MASS)
truehist(rdax)

mean(rdax)
var(rdax)
sd(rdax)
median(rdax)
quantile(rdax,probs=c(0.01,0.99))
range(rdax)

boxplot(rdax)
boxplot(rdax,rftse)

plot(rdax,rftse)

cor(rdax,rftse)

m <- length(rdax)
#Attention dim(rdax) doesn't work (dim(rdax)=NULL), so we have to use the length of the vector.
#Note: A vector has the dimension of NULL!
cor(rdax[2:m],rdax[1:(m-1)])
```

The plots, the histogram as well as the boxplot show the well-known stylized facts about stock returns:

- The mean is around 0, but positive (positive expected return).

- The standard deviation is a measure of risk.
- Compared to a normal distribution, one can see that the daily returns are not perfectly symmetric around the mean (weak asymmetry).
  - Large negative returns are more often than large positive ones.
  - Large positive returns are in absolute terms greater than large negative returns.
- There's more mass in the tails of the distribution (fat tails).
- The center is, compared to a normal distribution, higher (peakedness).

The computed statistics support the evidence for those stylized facts.

4 Graphics with R

A strength of R is its flexible way to create graphics. The following exercises illustrate that. Please write scripts for these exercises.

4 More on graphics

`plot()` is a very powerful command. Among other things it creates a graphical window, a Cartesian coordinate system and it plots the data. Most graphic functions expect x- and y-coordinates. You can load these from variables, enter them manually or use the function `locator(n)`, where you can simply click in the plot. The parameter `n` indicates the number of times you have to click for coordinates. Thus, `points(locator(4))` expects four clicks and after that it puts four points in the graph.

You can add other graphics like `points()`, `lines()`,`legend()`,... Please note that in order to use those functions you first have to call the `plot()` function. Thus, when plotting several graphics, write a script and execute all commands each time you change something.

There are several parameters that are pretty common for all graphical functions. Among others: `xlab=, ylab=, main=, col=, pch=, type=, ylim=, xlim=, lty=, lwd=` ...

`par()` creates a new windows for several graphics. It doesn't draw a coordinate system, however, so you have to use `plot()` again.

4.1 School data

1. Download the dataset `caschool.csv` into the object `caschool`. This dataset is discussed in great detail in the textbook of Stock and Watson. The codebook (`caschool.pdf`) is downloadable from the internet site of this course. Draw a scatterplot of the variable `testscr` against `str`.
2. Re-create the same plot with nicer and more informative axis labels (the axes options in the `plot` command are `xlab` and `ylab`).
3. Re-create the plot again and add a title (using the `main` option of the `plot` command).
4. The `col`-option can be used to change the colors of the points or lines. Try it. A list of all available color names is `colors()`. One can even color different parts of the plot differently, but we omit that here.

5. The command `points` adds one or more points into an existing plot. Add the point (mean of `str`, mean of `testscore`) to your last plot in red color. If you want to change the point symbol, you can use the option `pch`, see also `?points`.

6. The `text` command inserts text into an existing plot. Label the red point with the text "mean". The easiest way to position the text is by means of the mouse. Use the command `locator`, e.g. as in `text(locator(1),"mean")`.

7. One can partition the window into an array of small windows. You can prepare a partition using `par(mfrow=c(n,m))` where $n \times m$ is the number of plots (n rows and m columns). Prepare a window for four scatterplots (2×2). Plot the scatterplots of `testscr` against (a) the teacher-student ratio `str`, (b) the percentage of English language learners `el_pct`, (c) the percentage qualifying for reduced price lunch `meal_pct`, (d) the percentage qualifying for income assistance `calw_pct`.

4.1 School data

Please see the following code:

```
###################
### School data ###
###################
#1)
caschool  <- read.csv(file.choose())
head(caschool)
tail(caschool)
names(caschool)
str(caschool)
str <- caschool$str #Note that you have now overwritten the str() function!
testscr <- caschool$testscr
plot(str,testscr)

#2)
plot(str,testscr,xlab="Student Teacher Ratio", ylab="Test Score")

#3)
plot(str,testscr,xlab="Student Teacher Ratio", ylab="Test Score",main="CA Test Score Data")

#4)
print(colors())
plot(str,testscr,xlab="Student Teacher Ratio", ylab="Test Score",main="CA Test Score Data",
                                                                              col="violet")

#5)
?points #read the pch settings
plot(str,testscr,xlab="Student Teacher Ratio", ylab="Test Score",main="CA Test Score Data")
points(mean(str),mean(testscr), col= "red", pch=23)

#6)
#either
plot(str,testscr,xlab="Student Teacher Ratio", ylab="Test Score",main="CA Test Score Data")
points(mean(str),mean(testscr), col= "red", pch=19)
text(mean(str)-1,mean(testscr),"MEAN", col="red")
```

```
#or
plot(str,testscr,xlab="Student Teacher Ratio", ylab="Test Score",main="CA Test Score Data")
points(mean(str),mean(testscr), col= "red", pch=19)
text(locator(1),"MEAN", col="red") #locator(n) asks for n positions

#7)
el_pct <- caschool$el_pct
meal_pct <- caschool$meal_pct
calw_pct <- caschool$calw_pct
par(mfrow=c(2,2))
plot(testscr, str, xlab = "Test Score", ylab="Student-teacher-ratio")
plot(testscr, el_pct,xlab = "Test Score", ylab="Percentage English learners")
plot(testscr, meal_pct,xlab = "Test Score", ylab="Percentage reduced price lunch")
plot(testscr, calw_pct,xlab = "Test Score", ylab="Percentage income assistance")
```

4.2 Index returns

1. Download the dataset `indices.csv`. Generate a new variable with starting value 100 that represents the relative time series of the DAX index. Plot the time series of this normalized DAX index using the command `plot` with the options `type="l"` (for "line") and `col="blue"`.
2. Add the normalized FTSE index to the last plot. Use the command `lines` with the color option `col="red"`.
3. Use the `legend` command to add a legend explaining the meaning of the two colored lines. You may use the command `locator` to find a suitable position for the legend.

4.2 Index returns

Please see the following code:

```
#####################
### Index returns ###
#####################
#1)
indices <- read.csv2(file.choose())
head(indices)
tail(indices)
names(indices)
str(indices)

dax <- indices$dax
dax_norm <- 100*dax/dax[1]
par(mfrow=c(3,1))
plot(dax)
plot(dax_norm)
plot(dax,dax_norm) #Perfect linear correlation
cor(dax,dax_norm)  #Perfect linear correlation

#2)
par(mfrow=c(1,1))
```

```
ftse <- indices$ftse
ftse_norm <- 100*ftse/ftse[1]
plot(dax_norm,type="l", col="blue")
lines(ftse_norm, col="red")

#3)
plot(dax_norm,type="l", col="blue")
lines(ftse_norm, col="red")
legend(locator(1), legend=c("Normalized Dax", "Normalized FTSE"), fill = c("blue","red"))
```

5 Programming with R

5.1 Using functions

1. Functions are called by its name followed by the arguments in parentheses. The syntax is

   `function(Argument 1=arg1, Argument 2=arg2, ...)`

   You can specify arguments either by regarding the order they need to be called (`log(10,10)`) or by specifying the argument itself (`log(10,base=10)`).

   You can call several functions at a time using ";". Whenever you encounter the symbol + instead of >, you have forgotten to close parentheses. Just type ")" or hit ESC.

   Try:

   `sqrt(2); sin(pi); exp(1); log(10); log(10,10);log(10,base=10);`
   `sqrt(2` (without closing the bracket!)

2. The concatenation function c() creates vectors. You can pick the i-th item of a vector using square brackets. Try:

   ```
   simpsons <- c("Homer","Marge","Bart","Lisa","Maggie")
   x <- c(1,2,3,4,5,6,7,8,9,10)
   x <- c(1:10)
   length(simpsons); sum(x); mean(x)
   simpsons[3]
   ```

3. Consider the vector `x <- 0:10`. Use the function `sum()` to calculate the sum of all values that are smaller than 5, i.e. 0+1+2+3+4 = 10.

5.1 Using functions

Have a look at the following code:

```
#######################
### Using functions ###
#######################
#1)
log(10,10)
log(10,base=10)

sqrt(2); sin(pi); exp(1); log(10)
log(10,10);log(10,base=10)
sqrt(2
```

```
#2)
simpsons <- c("Homer","Marge","Bart","Lisa","Maggie")
x <- c(1,2,3,4,5,6,7,8,9,10)
x <- c(1:10)
length(simpsons); sum(x); mean(x)
simpsons[3]
simpsons[-3]

#3)
x <- 0:10
sum(x<5)
x<5

x*(x<5)
sum(x*(x<5)) #simple

x[x<5]
sum(x[x<5]) # more elegant
```

`sum(x<5)` is equal to 5, because `x<5` gives you a vector consisting of 1 and 0. These numbers indicate if the value in `x` is smaller than 5, i.e. TRUE(1) if it's smaller and FALSE(0) if not. `sum(x<5)` counts these 0's and 1's. The right solution is `sum(x*(x<5))` or `sum(x[x<5])`. Try to understand why!

5.2 Sequences and other vectors

In R, sequences are generated by the `seq`-command. An abbreviated form for integers is `from:to`. To generate a vector with repeated elements use the command `rep`.

1. Generate the vectors $x = (1, 2, \ldots, 100)$ and $y = (2, 4, 6, \ldots, 1000)$.
2. Generate an equi-spaced grid from -4 to 4 with 500 grid points.
3. Generate a vector of $n = 100$ missing values (`NA`).
4. Generate the vector $x = (0, 1, 2, 0, 1, 2, \ldots, 0, 1, 2)$ of length 300.
5. Generate the vector $x = (0, \ldots, 0, 1, 0, \ldots, 0)$ of length 100 with the 1 at position 40.

5.2 Sequences and other vectors

```
###################################
### Sequences and other vectors ###
###################################
?seq
?rep

# x=(1,2,...,100)
x <- 1:100; x

# x=(2,4,...,1000)
x <- 2*(1:500)
```

```
x <- seq(2,1000,by=2); x

# equi-spaced grid from -4 to 4 with 500 grid points
x <- seq(-4,4,length.out=500); x
length(x)
x[2]-x[1]
x[2]-x[1] == x[300]-x[299]

# 100 missing values
x <- rep(NA,100); x

# x=(0,1,2,0,1,2,...,0,1,2) with length 300
x <- rep(c(0,1,2),100); x
length(x)

# vector with 0 except a 1 at position 40, length 100
# several ways to do this
x <- c(rep(0,39),1,rep(0,60)); x ;length(x)
x <- rep(0,100); x[40] <- 1; x; length(x)
```

5.3 Random numbers

There are random number generators for a large number of distributions. The general syntax is

`rNAME(n,parameters)`

where `NAME` is an abbreviation of the distribution name (e.g. `norm`, `lnorm`, `binom`, etc.), `n` is the number of values to be drawn, and `parameters` are the parameter(s) of the distribution.

1. Activate the `MASS` package. Generate a vector `x` of $n = 10000$ random numbers drawn from the standard normal distribution and plot the histogram.
2. Generate a vector `r` of $n = 500$ random numbers drawn from the t-distribution with 3 degrees of freedom (see `?rt`). Execute `plot(r)`.
3. Cumulate the vector `r` using the command `cumsum`. Plot the cumulated series.

5.3 Random numbers

```
######################
### Random numbers ###
######################
library(MASS)
#1)
?rnorm
x <- rnorm(10000); x
truehist(x)

#2)
?rt
r <- rt(500,3); r
plot(r)
```

```
mean(r) # expectation of a student t-distribution is E(r) = 0
var(r)  # variance of a student t-distribution is Var(r) = (k)/(k-2)
        # with k degrees of freedom. Here: Var(r)=3
truehist(r)

#3)
x <- 1:10
x; cumsum(x)
plot(cumsum(r)) # we se a random walk!
```

5.4 Loops

In general, one should try to avoid loops in R as they often slow down the computations considerably. In this course, we will ignore this advice for didactical reasons. The type of loop that is used most often, is the `for`-loop. Unfortunately, the help function does not work for the loop commands, please type `?Control` to read the help text. The syntax of the `for`-loop is

`for( [var] in [sequence]) { [commands] }`

where `[var]` is an index variable and `[sequence]` is a vector of values to be assigned to the index variable. In our applications, we often need to store the results computed within the loop in a result vector. In this case, it is advisable to initiate an empty vector before the loop starts:

`Z <- rep(NA,100)`

`for(i in 1:100) { [compute something with result x]; Z[i] <- x }`

1. Generate a vector `r` of $n = 500$ random numbers drawn from the t-distribution with 3 degrees of freedom. Use a `for`-loop to compute the moving average of `r` within a window of length 21.
2. Write a program using a `for`-loop over $r = 1, \ldots, 10000$ to perform the following steps for every r: Generate a sample of size $n = 100$ from the lognormal distribution $LN(0, 1)$. Find the maximum and store it. After the loop is performed, plot the histogram of the maxima.

5.4 Loops

```
#############
### Loops ###
#############
#1)
?Control
r <- rt(500,3)
Z <- rep(NA,length(r))

for (i in 10:length(r)) {
    Z[i] <- mean(r[-10:10 + i])
}

plot(r)
lines(Z, col="red")
abline(h=mean(r), col="blue")
```

```
#2)
??"Log normal"
?Lognormal
Z <- rep(NA,10000)
for (r in 1:10000) {
Z[r] <- max(rlnorm(100))
}

library(MASS)
truehist(Z)
truehist(rlnorm(10000))
```

5.5 Functions

Functions are very powerful in R. Their general syntax is

```
f <- function(arg1,arg2,...)  { [commands to compute output var]; return(var)}
```

where the arguments can be scalars, vectors, matrices etc. For example, the following function computes and returns $x^2 + 2y^2$.

```
fexmpl <- function(x,y) { z <- x^2+2*y^2; return(z)}
```

Once the function has been defined it can be used like any other internal R function.

1. Define a function $f(x) = x^2 + \sin(x)$ where x can either be a scalar or a vector. Define a grid of length 500 on the interval $[-3, 3]$ and plot the function.
2. Define a function that computes the empirical raw moment of order p for a sample $x_1, \dots, x_n$, i.e. $m_p = \frac{1}{n}\sum_{i=1}^{n} x_i^p$.

5.5 Functions

```
#################
### Functions ###
#################
#1)
fexmpl1 <- function(x) {
    z <- x^2 + sin(x)
    return(z)
}
str(fexmpl1)

x <- seq(-3,3,length.out=500)
plot(x,fexmpl1(x))

#2)
mp <- function(x,p) {
n <- length(x)
m <- 1/n * sum(x^p)
return(m)
}

x <- c(1,2,3)
mp(x,1)
mp(x,2)
```

5.6 Numerical optimization

There are two commands for numerical optimization: `optimize` for univariate optimization and `optim` for multivariate optimization.

1. Numerically find the minimum of the function $f(x) = x^2 + \sin(x)$. *Hint: It lies between -1 and 0.*
2. Numerically find the minimum of the function $f(x, y) = x^2 + \sin(x) + y^2 - 2\cos(y)$. First get a view of the function using the following commands

   ```
   f <- function(x,y) x^2+sin(x)+y^2-2*cos(y)
   x <- seq(-5,5,by=.2);y <- seq(-5,5,by=.2);z <- outer(x,y,f)
   persp(x,y,z,phi=-45,theta=45,col="yellow",shade=.65 ,ticktype="detailed")
   ```

   You can try to edit phi and theta to get a better view.

5.6 Numerical optimization

```
###############################
### Numerical optimization ###
###############################
?optimize
?optim
#1)
optimize(fexmpl1, upper=0, lower=-1)

#2)
f <- function(x,y) x^2+sin(x)+y^2-2*cos(y)
x <- seq(-5,5,by=.2)
y <- seq(-5,5,by=.2)
z <- outer(x,y,f)
persp(x,y,z,phi=-45,theta=45,col="yellow",shade=.65 ,ticktype="detailed")
#the Minimum appears to be at (-.5,0)

#note: When using optim for multidimensional optimization,
#the argument in your definition of the function must be a single vector
fx <- function(x) x[1]^2+sin(x[1])+x[2]^2-2*cos(x[2])
optim(c(-.5,0),fx) # does work!
optim(c(-.5,0),f) #does not work
```

6 Probability theory

6.1 Moments

1. Show that the moments of the standard normal distribution $N(0, 1)$ are $\mu_r = 0$ for odd orders r, and $\mu_r = \prod_{i=1}^{r/2} (2i - 1)$ for even orders r.
2. Let $X \sim N(\mu, \sigma^2)$ and $Y = \exp(X)$. Derive the expectation of Y.

3. The distribution function of the Pareto distribution with parameters $K > 0$ and $\alpha > 0$ is

$$F_X(x) = 1 - \left(\frac{K}{x}\right)^{\alpha}.$$

where $x \geq K$. Derive the density f_X and the moment of order $p < \alpha$. Do moments of order $p \geq \alpha$ exist?

6.1 Moments

1. Since the density function of $N(0,1)$ is symmetric around 0, all odd moments are obviously 0. Proof:

$$E[X^r] = E[(-X)^r] = E[(-1)^r X^r] = E[-(X^r)] = -E[X^r] \Leftrightarrow E[X^r] = 0$$

We use without proof $\mu_2 = 1$ (variance). As to moments of even orders $r \geq 4$,

$$\mu_r = \int_{-\infty}^{\infty} x^r \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx$$

which can be integrated by parts to

$$\mu_r = \frac{1}{r+1} x^{r+1} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \Big|_{-\infty}^{\infty} - \int_{-\infty}^{\infty} \frac{1}{r+1} x^{r+1} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \left(-x\right) dx.$$

Since the exponential function goes to zero faster than any power of x goes to infinity, the first summand vanishes, and

$$\mu_r = \frac{1}{r+1} \underbrace{\int_{-\infty}^{\infty} x^{r+2} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx}_{\mu_{r+2}}$$

or

$$\mu_{r+2} = (r+1)\,\mu_r.$$

Odd moments are thus the product of odd numbers. Another way to write this product is given by $\mu_r = \prod_{i=1}^{r/2} (2i-1)$.

2. The expectation is

$$\begin{aligned} E(Y) &= E\left(\exp\left(X\right)\right) \\ &= \int_{-\infty}^{\infty} \exp\left(x\right) \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right) dx \\ &= \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2 - \frac{1}{2}\left(-2x\right)\right) dx. \end{aligned}$$

The term inside the exponential function can be written as

$$\begin{aligned}
& -\frac{1}{2}\left[\left(\frac{x-\mu}{\sigma}\right)^2-\frac{2x\sigma^2}{\sigma^2}\right] \\
= & -\frac{1}{2}\left(\frac{x^2-2\left(\mu+\sigma^2\right)x+\mu^2}{\sigma^2}\right) \\
= & -\frac{1}{2}\left(\frac{\left[x^2-2\left(\mu+\sigma^2\right)x+\left(\mu+\sigma^2\right)^2\right]-\left(\mu+\sigma^2\right)^2+\mu^2}{\sigma^2}\right) \\
= & -\frac{1}{2}\left(\frac{\left[x-\left(\mu+\sigma^2\right)\right]^2-\mu^2-2\mu\sigma^2-\sigma^4+\mu^2}{\sigma^2}\right) \\
= & -\frac{1}{2}\left(\frac{\left[x-\left(\mu+\sigma^2\right)\right]^2}{\sigma^2}-2\mu-\sigma^2\right)
\end{aligned}$$

Hence,

$$\begin{aligned}
E(Y) &= \int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2}\left(\frac{\left[x-\left(\mu+\sigma^2\right)\right]^2}{\sigma^2}\right)+\mu+\frac{\sigma^2}{2}\right)dx \\
&= e^{\mu+\sigma^2/2}\int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2}\left(\frac{\left[x-\left(\mu+\sigma^2\right)\right]^2}{\sigma^2}\right)\right)dx
\end{aligned}$$

The integrand is simply the density of a normal distribution with mean $\mu+\sigma^2$ and variance σ^2. The integral over the density is unity, and thus

$$E(Y)=e^{\mu+\sigma^2/2}.$$

3. The density of the Pareto distribution is

$$f_X(x)=\frac{dF(x)}{dx}=\alpha K^\alpha x^{-\alpha-1}$$

for $x\geq K$ (and zero elsewhere). The moment of order $p<\alpha$ is

$$\begin{aligned}
E\left(X^p\right) &= \int_{-\infty}^{\infty}x^p f_X(x)dx \\
&= \int_K^\infty x^p\alpha K^\alpha x^{-\alpha-1}dx \quad \text{since } x\geq K \\
&= \alpha K^\alpha\int_K^\infty x^{p-\alpha-1}dx \\
&= \alpha K^\alpha\cdot\left[\frac{1}{p-\alpha}x^{p-\alpha}\right]_K^\infty .
\end{aligned}$$

Since $p<\alpha$ the expression within the square brackets goes to zero as $x\to\infty$, and therefore

$$\begin{aligned}
E\left(X^p\right) &= \alpha K^\alpha\cdot\frac{1}{\alpha-p}K^{p-\alpha} \\
&= \frac{\alpha}{\alpha-p}K^p.
\end{aligned}$$

Moments of order $p\geq\alpha$ do not exist as the integral does not converge.

7 Multiple linear regression

Linear models are estimated in R by the command `lm`. This command has an unusual syntax and returns a rather complex object (called `lm` object). Be prepared: it takes some time to get used to that. We start with the simple linear regression model that is used in the textbook by Stock and Watson. The codebook `caschool.pdf` for the dataset can be downloaded from the course site.

7.1 Student teacher ratio (I)

1. Load the dataset `caschool.csv` into the object `caschool` and make `testscr` as well as `str` accessible. Perform the following commands:
   `regr <- lm(testscr~str)`
   `print(regr)`
   Create the scatterplot of `testscr` against `str` and then type `abline(regr)`. The color (`col`), the line type (`lty`), and the line width (`lwd`) can easily be changed by the options of the plot command. Try it.
2. The student teacher ratio `str` in the school district Antelope is 19.33 an. Predict the variable `testscore` for the district Antelope using the `predict` command. To do so, type `predict(regr,newdata=data.frame(str=19.33))`. Add the predicted value to the plot (in blue color).
3. Among other things, `lm` objects also contain the residuals of the regression. You can extract them using the function `residuals` with the `lm`-object as argument. Compute the sum of the residuals.
4. Create a plot showing the residuals. Add the horizontal axis using the command `abline(h=0)` (the `h` is for horizontal).
5. Plot the residuals against the variable `str`.
6. Load the `AER` package. Execute the commands `print(summary(regr))` and `print(coeftest(regr,vcov=vcovHC))`. Interpret the outputs.
7. Test the hypothesis $H_0 : \beta = -1$. Write down each step of the test procedure. *Hint: You can also make use of `linearHypothesis` function of the car package.*
8. Give a 95% confidence interval for β.

7.1 Student teacher ratio (I)

```
####################################
#### Student teacher ratio (I) ####
####################################
#1)
caschool <- read.csv(file.choose())
View(caschool)
testscr <-caschool$testscr
str <- caschool$str

regr <- lm(testscr~str)
```

```
print(regr)
str(regr) #a very complex object! You can access those things using the $ sign.

plot(str,testscr, xlab="Student-Teacher-Ratio", ylab="Testscore", main="Scatterplot", pch=20)
abline(regr, col="red",lwd=3, lty=2)

#2)
prediction <- predict(regr, newdata=data.frame(str=19.33))
plot(str,testscr, xlab="Student-Teacher-Ratio", ylab="Testscore", main="Scatterplot", pch=20)
abline(regr, col="red",lwd=3, lty=2)
#add a point
points(19.33, prediction, col="blue", pch=20)
#lines connects two points with a straight line.
#Note: abline needs a slope and an intercept to draw a straight line
lines(c(19.33, 19.33), c(0,prediction),lty=3, col="blue")
lines(c(0, 19.33), c(prediction,prediction),lty=3, col="blue")

#3)
#Either use the function residuals()
sum(residuals(regr))
#Or extract them from the object regr with the dollar sign
sum(regr$residuals)

#4)
plot(residuals(regr))
abline(h=0)

#5)
plot(regr$residuals,str)

#6)
print(summary(regr)) # with this command you assume homoscedasticity

library(AER)
print(coeftest(regr,vcov=vcovHC)) #with this command you assume heteroscedasticity

#7)
# Access the variables
BETA <- coeftest(regr,vcov=vcovHC)[2,1]
SDBETA <- coeftest(regr,vcov=vcovHC)[2,2]
#t-Test
H0 <- -1
t <- (BETA- (H0))/SDBETA
t
#the critical values are 1.96 (5%) and 2.58 (1%)
abs(t) > 1.96; abs(t) > 2.58
#H0 can be rejected with a significance level of 5%!
#The estimate is significantly different from -1.

#There is also a function for linear hypothesis testing  (F-Test)
library(car)
linearHypothesis(regr,c("str=-1"))
```

```
#8)
lower_limit <- BETA - 1.96*SDBETA
upper_limit <- BETA + 1.96*SDBETA

names(lower_limit)="lower limit"; names(upper_limit)="upper limit"
print(c(lower_limit, upper_limit))
```

1. In the scatterplot you can see that the points scatter heterogeneously around the regression line, since there are several outliers. The assumption of homoscedasticity does not hold.
2. Note: `lines` connects points with a straight line. `abline`, however, needs a slope and an intercept to draw a straight line.
3. The $ sign is very handy to access data and variables from complex objects. Get used to using the $.
4. The plot shows that the residuals are heteroscedastic and most likely autocorrelated (not i.i.d.).
5. According to the plot, there is no obvious linear correlation between the exogenous variable and the residuals.
6. This is the standard output of a regression. In order to control for heteroscedasticity, use `coeftest()` and specify the variance-covariance matrix appropriately (e.g. `vcov=vcovHC`).
7. This is a standard t-Test. You can either compute it by hand or use `linearHypothesis()`.
8. The critical value for a 95% confidence interval is 1.96.

7.2 Capital asset pricing model

Load the dataset `capm.csv` and make the variables accessible. The variable `rdai` contains the daily returns (in %) of Daimler from 9/9/2009 to 8/9/2010, the variable `rdax` contains the DAX returns. The CAPM implies that the intercept of the simple linear regression

$$r_{DAI,t} = \alpha + \beta r_{DAX,t} + u_t$$

is zero.

1. Estimate the model and test the null hypothesis $H_0 : \alpha = 0$.
2. The coefficient β is a measure of the systemic risk. Give a 95% confidence interval for β.

7.2 Capital asset pricing model

Please see the following code:

```
#####################################
#### Capital asset pricing model ####
#####################################
#1)
capm <- read.csv2(file.choose())
head(capm)
```

```
rdai <- capm$rdai
rdax <- capm$rdax
regr <- lm(rdai~rdax)
regr
library(car)
linearHypothesis(regr,c("(Intercept) = 0")) #cannot be rejected

#2)
summary(regr)
BETA <- summary(regr)$coefficients[2,1]
SDBETA <- summary(regr)$coefficients[2,2]
lower_limit <- BETA - 1.96*SDBETA
upper_limit <- BETA + 1.96*SDBETA
names(lower_limit)="lower limit"; names(upper_limit)="upper limit"
print(c(lower_limit, upper_limit))
```

7.3 Student teacher ratio (II)

The `lm`-command is also used to perform multiple linear regressions. The syntax is close to the simple linear models. Put the endogenous variable to the left of the tilde. On the right of the tilde you list the exogenous variables, separated by plus signs. It looks like this: `lm(y~x1+x2+x3)`.

1. Load the dataset `caschool.csv` into the object `caschool` and make `testscr`, `str`, `el_pct` and `expn_stu` accessible. Perform the following commands:
   ```
   regr <- lm(testscr~str+el_pct)
   print(regr)
   ```
   Explain the output.
2. Regress `testscr` on `str`, assign the residuals of the regression into the variable `r1` and plot them. Now regress `testscr` on `str`, `el_pct` and `expn_stu`, put the residuals into the variable `r2` and add them to the plot. Compute the sum of squared residuals for both regressions.
3. Consider the regression of `testscr` on `str`, `el_pct` and `expn_stu`. Using the `predict`-command, predict the value of `testscr` for a school district with an average class size (`str`) of 25 students, a percentage of English learners (`el_pct`) of 60% and an average expenditures per student (`expn_stu`) of 4000$. How would the result change if the average class size was reduced to 17?
4. Reconsider the regression of `testscr` on `str`, `el_pct` and `expn_stu`. Let `regr` be the object containing the regression results. Execute the commands `print(summary(regr))` and `print(coeftest(regr,vcov=vcovHC))`. Interpret the output.
5. Test the null hypothesis that the coefficients on `str` and `expn_stu` both equal 0 and the coefficient on `el_pct` equals -0.7. *Hint: Use the linearHypothesis function of the car package.*

7.3 Student teacher ratio (II)

```
####################################
#### Student teacher ratio (II) ####
```

```
####################################
library(AER)
#1)
caschool <- read.csv(file.choose())
head(caschool)
testscr <- caschool$testscr
str <- caschool$str
el_pct <- caschool$el_pct
expn_stu <- caschool$expn_stu
regr <- lm(testscr~str+el_pct)
regr

#2)
simple <- lm(testscr~str)
r1 <- residuals(simple)
plot(r1)

multiple <- lm(testscr~str + el_pct + expn_stu)
r2 <- residuals(multiple)
plot(r2)

plot(r1,ylab="")
points(r2,col="red")

sum(r1^2)
sum(r2^2)
sum(r1^2) > sum(r2^2)

#3)
multiple <- lm(testscr~str + el_pct + expn_stu)
predict(multiple, newdata=data.frame(str=25, el_pct=0.6, expn_stu=4000))
predict(multiple, newdata=data.frame(str=17, el_pct=0.6, expn_stu=4000))

#4)
regr <- lm(testscr~str + el_pct + expn_stu)
summary(regr)
library(AER)
coeftest(regr, vcov=vcovHC)

#5)
library(car)
linearHypothesis(regr,c("str=0","expn_stu=0","el_pct=-.7"))
```

7.4 Omitted variable bias

Load the dataset `omitted.csv` into the object `omitted` and make it accessible. There are five variables: `y`, `x1`, `x2`, `x3` and `x4`. The sample has been generated in R; the sample size is $n = 500$. The true regression surface is

$$Y = 1 + 2X_1 + 3X_2 + 4X_3 + 5X_4.$$

The exogenous variable X_1 is uncorrelated with X_2, X_3, X_4. The variables X_2 and X_3 are positively correlated, so are X_3 and X_4. The variables X_2 and X_4 are uncorrelated.

1. Estimate the intercept and the slope coefficients for X_1, X_2, X_3, X_4 from the dataset (the estimates should be close to the true values 1,2,3,4,5).
2. Estimate a regression of Y on X_2, X_3 and X_4. Explain why the estimates are still close to the true values.
3. Estimate a regression of Y on X_1, X_2 and X_3. Which coefficients are still estimated accurately? And why?

7.4 Omitted variable bias

Please see the following code:

```
################################
#### Omitted variable bias ####
################################
library(car)
omitted <- read.csv2(file.choose())
head(omitted)
y <- omitted$y
x1 <- omitted$x1
x2 <- omitted$x2
x3 <- omitted$x3
x4 <- omitted$x4

cor(omitted)

#1
regra <-lm(y ~ x1+x2+x3+x4)
summary(regra)
linearHypothesis(regra,c("x1=2","x2=3","x3=4","x4=5"))
#2
regrb <- lm(y~x2+x3+x4)
summary(regrb)
linearHypothesis(regrb,c("x2=3","x3=4","x4=5"))
#3
regrc <- lm(y~x1+x2+x3)
summary(regrc)
linearHypothesis(regrc,c("x1=2","x2=3","x3=4"))
```

Multicollinearity: Strong correlation between the exogenous variables. The coefficients, however, are unbiased if the model is specified correctly, i.e. if no variables are omitted. Otherwise you get an omitted variable bias. Furthermore the estimators are not efficient, they have high standard errors. Also a ceteribus paribus interpretation of a single coefficient is not valid.

7.5 Asymptotic normality

1. Consider the multiple linear regression model $y = X\beta + u$. In R, generate the matrix X by executing the following commands:

   `library(MASS)`

   `set.seed(123)`

`X <- cbind(1,mvrnorm(n=100,c(5,10),matrix(c(1,0.9,0.9,1),2,2)))`

The true coefficient vector is

$$\beta = \begin{pmatrix} 3 \\ 2 \\ -1 \end{pmatrix}$$

and the error terms are i.i.d. uniformly distributed on the interval $[-1, 1]$. Hence, the assumption of normally distributed error terms is violated.

2. Write an R program that generates $R = 10000$ random samples of size $n = 100$ each (the easiest way to do so is to use a `for` loop). Generate an empty vector `V <- rep(NA,10000)`. For each sample $i = 1, \dots, R$, compute the OLS estimate $\hat{\beta}$ of β and store the second component of $\hat{\beta}$ in the i-th element of the vector `V`.

3. Plot the histogram of `V`.

4. Compute the mean m and standard deviation s of `V` and add the density of $N(m, s)$ to the plot. *Hint: You can use `curve(dnorm(x,mean=m,sd=s),add=T)` to add the Gaussian density with mean m and std. deviation s to the plot.*

5. Move the command that generates X into the loop (without the seed command). Now there is a new, random X for each sample. Is the normal approximation still valid?

6. Try if the approximation is worse for sample size $n = 10$ (you will have to shorten X in this case).

7.5 Asymptotic normality The code might look like this:

```
##############################
#### Asymptotic Normality ####
##############################
library(MASS)
beta_true <- c(3,2,-1)
TT <- 100
R <- 10000
set.seed(123)
X_fix <- cbind(1,mvrnorm(n=TT,c(5,10),matrix(c(1,0.9,0.9,1),2,2)))

V1 <- rep(NA,R); V2 <- rep(NA,R);
for (i in 1:R) {
  u <- runif(TT,-1,1)
  X_rand <- cbind(1,mvrnorm(n=TT,c(5,10),matrix(c(1,0.9,0.9,1),2,2)))
  y1 <- X_fix%*%beta_true + u
  y2 <- X_rand%*%beta_true + u
  beta_hat1 <- solve(t(X_fix)%*%X_fix)%*%t(X_fix)%*%y1
  beta_hat2 <- solve(t(X_rand)%*%X_rand)%*%t(X_rand)%*%y2
  V1[i] <- beta_hat1[2]
  V2[i] <- beta_hat2[2]
}
truehist(V1)
m1 <- mean(V1)
s1 <- sd(V1)
curve(dnorm(x,mean=m1,sd=s1),add=T)
```

```
truehist(V2)
m2 <- mean(V2)
s2 <- sd(V2)
curve(dnorm(x,mean=m2,sd=s2),add=T)
```

7.6 Pitfalls in the linear regression model (I)

A simple linear regression is very easily performed by any statistical program. However, there are many mistakes and misinterpretations that can be made. A critical inspection of your regression results is crucial. Three of the more common mistakes are illustrated in the following.

Load the dataset `gehaelter.csv` into the object `gehaelter` and make the variables accessible. The dataset contains observations on 100 graduates about their length of study (`dauer`), their initial salary (`gehalt`) and their major (`fach`, 1=chemistry, 2=economics).[3]

1. Draw the scatterplot of salary against length of study.
2. Perform a linear regression of salary on length of study and add the estimated regression line to the scatterplot. What is the effect of the length of study on the salary?
3. Repeat 1. and 2. separately for chemistry and economics graduates. What is the effect of the length of study on salary in each group?
4. Re-draw the scatterplot with economists colored in blue and chemists colored in red.

7.6 Pitfalls in the linear regression model (I)

Please have a look at the following code:

```
#####################################################
#### Pitfalls in the linear regression model (I) ####
#####################################################
gehaelter <- read.csv(file.choose())
names(gehaelter)
head(gehaelter)
str(x)

dauer <- gehaelter$dauer
gehalt <- gehaelter$gehalt
fach <- gehaelter$fach

#1
plot(dauer, gehalt, xlab="Duration", ylab="Salary", main = "Scatterplot")

#2
model <- lm(gehalt~dauer)
model
plot(dauer, gehalt, xlab="Duration", ylab="Salary", main = "Scatterplot")
abline(model)

#3
```

[3]The data are fictional and have been generated by a computer algorithm.

```
chem <- lm(gehalt~dauer, data=x[fach==1,])
chem
econ <- lm(gehalt~dauer, data=x[fach==2,])
econ

plot(dauer, gehalt, xlab="Duration", ylab="Salary", main = "Scatter")
points(x[fach==1,2],x[fach==1,1],col="blue") #mark chem students
points(x[fach==2,2],x[fach==2,1], col="red") #mark econ students
abline(model)
abline(chem, col="blue")
abline(econ, col="red")
legend("topright", legend=c("Total","Chemistry", "Economics"), fill = c("black","blue","red"))
```

7.7 Pitfalls in the linear regression model (II)

Load the dataset `storch.csv`. It contains observations on the stork population (eyries) in Lower Saxony and the number of births in Germany from 1958 to 2004.

1. Plot the scatterplot of the number of births against the number of storks and perform a linear regression. What is the effect of the number of storks on the number of births?
2. Repeat the exercise with the number of out-of-wedlock births.

7.7 Pitfalls in the linear regression model (II)

```
#######################################################
#### Pitfalls in the linear regression model (II) ####
#######################################################
storch <- read.csv2(file.choose())
head(storch)
names(storch)
str(storch)

Horstpaare <- storch$Horstpaare
Geburten <- storch$Geburten
nichtehelich<- storch$nichtehelich

#1)
plot(Horstpaare, Geburten, xlab="Number storks", ylab="Number of births")
regr <- lm(Geburten ~ Horstpaare)
regr
abline(regr)

#2)
regr1 <- lm(nichtehelich ~ Horstpaare)
regr1
plot(Horstpaare, nichtehelich, xlab="Number storks", ylab="Number of out-of-wedlock births")
abline(regr1)
```

7.8 Pitfalls in the linear regression model (III)

Load the `indices.csv` (this dataset has been used before, see exercise 3.1). Execute the following commands:

1.
   ```
   n <- dim(indices)[1]
   kdax <- dax[6:n]
   kftselag <- ftse[1:(n-5)]
   ```
   There are two new variables: the DAX index `kdax` and the FTSE-100 index lagged by five trading days (`kftselag`).
2. Regress the DAX index `kdax` on the lagged FTSE index `kftselag`. Interpret the estimated coefficients.
3. Test the null hypothesis that the DAX index does not depend on the lagged FTSE index (significance level 0.05).

7.8 Pitfalls in the linear regression model (III)

Please have a look at the following code:

```
#######################################################
#### Pitfalls in the linear regression model (III) ####
#######################################################
indices <- read.csv2(file.choose())
head(indices)
names(indices)
str(indices)
dax <- indices$dax
ftse <- indices$ftse

n <- dim(indices)[1]
kdax <- dax[6:n]
kftselag <- ftse[1:(n-5)]

plot(kftselag,kdax)
obj <- lm(kdax ~ kftselag)
obj
summary(obj)
library(AER)
coeftest(obj, vcov=vcovHC)
```

Be careful with time series data! The assumption of iid in the linear regression model is not valid anymore!

8 Multivariate random variables

The package `MASS` includes a command to generate i.i.d. draws from the multivariate normal distribution. Type `library(MASS)` to activate it.

8.1 Joint distributions

Consider the bivariate density

$$f(x,y) = 40 \cdot (x-0.5)^2 \cdot y^3 \cdot (3-2x-y)$$

for $(x,y) \in [0,1] \times [0,1]$ and $f(x,y) = 0$ else.

1. Show that $f(x,y)$ is really a density function.
2. Derive the marginal densities $f_X(x)$ and $f_Y(y)$ and plot them.
3. Derive the conditional density of X given $Y = y$ and plot it for $y = 0.01$ and $y = 0.95$.
4. Are X and Y independent?

8.1 Joint distributions

1. If $f(x,y)$ is a density function, then the double integral over the support must be equal to 1. So, first integrate for x and then for y:

$$\int_0^1 f(x,y)dx = \frac{10}{3}y^3(2-y)$$
$$\int_0^1 \frac{10}{3}y^3(2-y)dy = 1$$

2. The marginal densities are given by

$$f_Y(y) = \int_0^1 f(x,y)dx = \frac{10}{3}y^3(2-y)$$
$$f_X(x) = \int_0^1 f(x,y)dy = -20x^3 + 42x^2 - 27x + 55$$

3. The conditional density of X given $Y = y$ is given by

$$f_X(x|y) = \frac{f(x,y)}{f_Y(y)} = 12(x-0.5)^2\frac{3-2x-y}{2-y}$$

4. X and Y are dependent, because:

$$f(x,y) \neq f_X(x) \cdot f_Y(y)$$

8.2 Gaussianity or else?

Load the dataset `gaussian.csv` into the object `gaussian`. Each column of the dataframe `gaussian` is a variable (V1, V2, V3, V4).

1. Split the screen into 2×2 (see exercise 7). Plot the histogram for each variable and add the density of the standard normal distribution to each histogram. Are the variables normally distributed?
2. Compute the correlation matrix. Are the variables correlated?
3. Plot the 4×4 matrix of scatterplots (use the command `pairs`). Are the variables independent?
4. Compute the sum $Y = V1+V2+V3+V4$ and plot the histogram of Y. Is the sum normally distributed?

8.2 Gaussianity or else?

The program might look like this:

```
##############################
#### Gaussianity or else? ####
##############################
#1)
gaussian <- read.csv(file.choose())
View(gaussian)

library(MASS)
x <- seq(-4, 4, length=100)
par(mfrow=c(2,2))

truehist(gaussian$V1); lines(x,dnorm(x))
truehist(gaussian$V2); lines(x,dnorm(x))
truehist(gaussian$V3); lines(x,dnorm(x))
truehist(gaussian$V4); lines(x,dnorm(x))

#2)
cor(gaussian)

#3)
pairs(gaussian)

#4)
Y <- gaussian$V1+gaussian$V2+gaussian$V3+gaussian$V4
par(mfrow=c(1,1))
truehist(Y); lines(x,dnorm(x, mean=mean(Y),sd=sd(Y)))
```

1. Each variable seems to be normally distributed.
2. There is no (or just a very small) correlation between the variables.
3. The scatterplots show that the variables are NOT independent, even though they are normally distributed and uncorrelated. If one variable is (in absolute terms) very big, it is very likely that the other variables are (in absolute terms) big as well. This is dependence!

4. Compared to a normal distribution the sum has more mass in the center of the distribution. Because of the dependence the sum is per se not normally distributed.

This exercise illustrates that for the sum of normally distributed variables being also normally distributed requires the assumption of **independence**, not just uncorrelatedness; two separately (not jointly) normally distributed random variables can be uncorrelated without being independent, in which case their sum can be non-normally distributed.

8.3 Gaussian and uncorrelated, but dependent

Let $X \sim N(0,1)$ and define

$$Y = U \cdot X$$

where

$$U = \begin{cases} -1 & \text{with probability } 0.5 \\ 1 & \text{with probability } 0.5 \end{cases}$$

1. Determine the distribution of Y.
2. Derive the covariance between X and Y.
3. Generate a random sample of size $n = 1000$ from $(X, Y)'$ and show the scatterplot.
4. For the sample, compute the sum $X + Y$ and plot its histogram.

8.3 Gaussian and uncorrelated, but dependent

1. The distribution of Y is the same as X, because

$$\begin{aligned} &Pr(Y \leq x) = Pr(X \leq x) \cdot Pr(U = 1) + Pr(-X \leq x) \cdot Pr(U = -1) \\ &= \Phi(x) \cdot \frac{1}{2} + \Phi(x) \cdot \frac{1}{2} = \Phi(x) \end{aligned}$$

since X and $-X$ have the same distribution and Φ ist the distribution function of the normal distribution.

2. X and Y are uncorrelated, since the covariance is given by

$$Cov(X,Y) = E(X \cdot Y) - \underbrace{E(X)}_{=0} \cdot E(Y) = E[E(X \cdot Y|U)] = E[X^2] \cdot \frac{1}{2} + E[-X^2] \cdot \frac{1}{2} = 0$$

3. The program might look like this

```
####################################################
#### Gaussian and uncorrelated, but dependent ####
####################################################
library(MASS)
X <- rnorm(1000)
U <- sample(c(-1, 1), 1000, replace = TRUE)
Y <- U*X
truehist(X)
truehist(Y)
cov(X,Y)
plot(X,Y)
```

Even though X and Y are normal and uncorrelated, they are not independent, since it is very likely that if X is large, Y is in absolute terms also large. In fact: $|Y| = |X|$.

4. The additional code might look like this:

```
Z <- X+Y
mean(Z)
par(mfrow=c(1,1))
truehist(Z)
```

Because X and Y are dependent, the sum is not normally distributed (see also ex. *Gaussianity or else*).

9 Stochastic convergence and limit theorems

9.1 Law of large numbers

Let $X_1, X_2, \ldots$ be an i.i.d. sequence of arbitrarily distributed random variables with finite variance σ^2. Define the sequence of random variables

$$\bar{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i.$$

1. Write an R program to illustrate the law of large numbers.
2. Now suppose that the sequence $X_1, X_2, \ldots$ is an $AR(1)$ process:

   $$(X_i - \mu) = \rho\,(X_{i-1} - \mu) + \varepsilon_i$$

   where $\varepsilon_i \sim iid(0, \sigma_\varepsilon^2)$ is not necessarily normally distributed and $|\rho| < 1$. Show that the law of large numbers still holds despite the intertemporal dependence.

9.1 Law of large numbers

The programs might look like this:

```
###############################
#### Law of large numbers ####
###############################
#1)
z <- rep(NA,1000); u <- rep(NA,1000); g <- rep(NA,1000)
for (i in 1:1000) {
  z[i] <- mean(rnorm(i,mean=10,sd=2))
  u[i] <- mean(runif(i,min=0,max=1)) #expectation is (a+b)/2
  g[i] <- mean(rgeom(i,prob=0.2)) #expectation is (1-p)/p
}

par(mfrow=c(1,2))

truehist(z,main="Law of large numbers")
plot(z, main="for the normal distribution"); abline(h=10,lwd=2,col="red")

truehist(u,main="Law of large numbers")
plot(u,main="for the uniform distribution"); abline(h=0.5,lwd=2,col="red")

truehist(g,main="Law of large numbers")
plot(g,main="for the geometric distribution"); abline(h=4,lwd=2,col="red")

#2)
z <- rep(NA,1000)
rho=0.8
mu=2
for (i in 1:1000) {
  z[i] <- mean(filter((1-rho)*mu+rnorm(i,sd=2),rho,method="recursive",init=(1-rho)*mu))
  #try and use a different distribution
}
```

```
par(mfrow=c(1,2),pty="s")
truehist(z,main="Law of large numbers")
plot(z, main="for intertemporal dependence AR(1)"); abline(h=mu,lwd=2,col="red")
```

9.2 Law of large numbers for the variance

Let $X_1, X_2, \ldots$ be an i.i.d. sequence of arbitrarily distributed random variables with mean μ, variance σ^2, and finite kurtosis, i.e. $E(X_i^4) < \infty$. Define the sequence of random variables

$$S_n^2 = \frac{1}{n} \sum_{i=1}^{n} \left(X_i - \bar{X}\right)^2, \quad \text{where } \bar{X} = n^{-1} \sum_{i=1}^{n} X_i.$$

1. Write an R program to illustrate that $S_n^2 \to \sigma^2$ in probability.
2. Now draw the samples from a t-distribution with 3 degrees of freedom, i.e. $X_i \overset{iid}{\sim} t_3$. The kurtosis of the t_3-distribution is infinite. Use your R program to show that S_n^2 does no longer converge to σ^2 in probability.

9.2 Law of large numbers for the variance

The program might look like this:

```
################################################
#### Law of large numbers for the variance ####
################################################
#1)
library(MASS)
zn <- rep(NA,1000); zu <- rep(NA,1000); zg <- rep(NA,1000)
for (i in 1:1000) {
  rn <- rnorm(i,mean=10,sd=2) ; variance_n <- 2^2
  ru <- runif(i,min=0,max=6) ; variance_u <- 3
  rg <- rgeom(i,prob=0.2); variance_g <-20
  zn[i] <- sum((rn-mean(rn))^2)/i
  zu[i] <- sum((ru-mean(ru))^2)/i
  zg[i] <- sum((rg-mean(rg))^2)/i
}

par(mfrow=c(1,2))
truehist(zn,main="Law of large numbers")
plot(zn, main="for the variance (normal distrib.)"); abline(h=variance_n,lwd=2,col="red")

truehist(zn,main="Law of large numbers")
plot(zu, main="for the variance (uniform distrib.)"); abline(h=variance_u,lwd=2,col="red")

truehist(zn,main="Law of large numbers")
plot(zg, main="for the variance (geometr. distrib.)"); abline(h=variance_g,lwd=2,col="red")

#2)
zt <- rep(NA,1000)
for (i in 1:1000) {
  xt <- rt(i,df=3)
  zt[i] <- sum((xt-mean(xt))^2)/i; variance_t <-3/(3-2)
}

par(mfrow=c(1,2))

truehist(zt,main="Law of large numbers")
plot(zt, main="for the variance (t-distrib. df=3)");abline(h=variance,lwd=2,col="red")
```

9.3 Central limit theorem

Let $X_1, X_2, \ldots$ be an i.i.d. sequence of arbitrarily distributed random variables with mean μ and finite variance σ^2. Define the sequences of random variables

$$Y_n = \sum_{i=1}^{n} X_i, \qquad Z_n = \sqrt{n}\frac{\left(\frac{1}{n}Y_n\right) - \mu}{\sigma}.$$

1. Write an R program to illustrate the central limit theorem.
2. Show that the central limit theorem still holds if we replace the standard deviation σ in the denominator of Z_n by the estimated standard deviation

$$S_n = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(X_i - \bar{X}\right)^2}.$$

3. Now let $X_1, X_2, \ldots$ be an i.i.d. sequence of t-distributed random variables with 1.5 degrees of freedom. Show that the convergence in distribution breaks down.

9.3 Central limit theorem

The program might look like this:

```
################################
#### Central limit theorem ####
################################
#1) Illustration of the central limit theorem for the uniform distribution
#    with sigma as the standard deviation
library(MASS)
N <- 10000 # how many times to draw individual X_i's, note that i = 1,2,..,n
expect <- 0.5
vari <- 1/12
par(mfrow = c(1,2),pty = "s")
for(n in 1:20) {
  X <- runif(N*n) #N*n random numbers
  M <- matrix(X, N, n) #matrix of N rows and n columns filled with random numbers
  Yn <- rowSums(M) # calculate the sum of n random numbers for N rows
  Zn <- (Yn/n - expect)*sqrt(n/vari) #standardization
  #display the sequence of random variables
  truehist(Zn, xlim = c(-4,4),ylim = c(0,0.5), main = paste("n =", toString(n),sep =" "))
  coord <- par("usr")
  # par("usr") gives you a vector of the form c(x1, x2, y1, y2)
  # giving the extremes of the coordinates of the plotting region
  x <- seq(coord[1], coord[2], length.out = 500)
  lines(x, dnorm(x), col = "red")
  qqnorm(Zn, ylim = c(-4,4), xlim = c(-4,4), pch = ".", col = "blue")
  abline(0, 1, col = "red")
  Sys.sleep(1)
}

#2) Illustration of the central limit theorem for the uniform distribution
#     with the estimated standard deviation
library(MASS)
N <- 10000 # how many times to draw individual X_i's, note that i = 1,2,..,n
expect <- 0.5
par(mfrow = c(1,2),pty = "s")
#hint: it needs to start with n=2, for n=1 you get vari=0 and you cannot divide by 0
for(n in 2:20) {
  X <- runif(N*n) #N*n random numbers
  M <- matrix(X, N, n) #matrix of N rows and n columns filled with random numbers
  vari <- 1/n * rowSums((M-rowMeans(M))^2)
  Yn <- rowSums(M) # calculate the sum of n random numbers for N rows
  Zn <- (Yn/n - expect)*sqrt(n/vari) #standardization
  #display the sequence of random variables
  truehist(Zn, xlim = c(-4,4),ylim = c(0,0.5), main = paste("n =", toString(n),sep =" "))
  coord <- par("usr")
  x <- seq(coord[1], coord[2], length.out = 500)
  lines(x, dnorm(x), col = "red")
  qqnorm(Zn, ylim = c(-4,4), xlim = c(-4,4), pch = ".", col = "blue")
  abline(0, 1, col = "red")
  Sys.sleep(1)
}

#3)
library(MASS)
N <- 10000 # how many times to draw individual X_i's, note that i = 1,2,..,n
expect <- 0
par(mfrow = c(1,2),pty = "s")
#hint: it needs to start with n=2, for n=1 you get vari=0 and you cannot divide by 0
for(n in 2:20) {
  X <- rt(N*n,df=1.5)
  M <- matrix(X, N, n)
  vari <- 1/n * rowSums((M-rowMeans(M))^2)
  Yn <- rowSums(M)
  Zn <- (Yn/n - expect)*sqrt(n/vari)
  truehist(Zn, xlim = c(-4,4),ylim = c(0,0.5), main = paste("n =", toString(n),sep =" "))
  coord <- par("usr")
  x <- seq(coord[1], coord[2], length.out = 500)
  lines(x, dnorm(x), col = "red")
  qqnorm(Zn, ylim = c(-4,4), xlim = c(-4,4), pch = ".", col = "blue")
  abline(0, 1, col = "red")
  Sys.sleep(1)
}
```

9.4 Central limit theorem for dependent data

Suppose that the sequence $X_1, X_2, \ldots$ is an $AR(1)$ process, i.e.

$$(X_i - \mu) = \rho\,(X_{i-1} - \mu) + \varepsilon_i$$

where $\varepsilon_i \sim iid(0, \sigma_\varepsilon^2)$ is not necessarily normally distributed and $|\rho| < 1$.

1. Show that X_i has mean equal to μ and finite variance equal to $\sigma_\varepsilon^2/(1-\rho^2)$.
2. To derive the asymptotic distribution of the sample mean, do the following steps:
   (a) Derive the asymptotic distribution of $\frac{1}{\sqrt{n}}\sum_{i=1}^{n}\varepsilon_i$
   (b) Show that
   $$\frac{1}{\sqrt{n}}\sum_{i=1}^{n}\varepsilon_i = \sqrt{n}\left[(1-\rho)\left(\frac{1}{n}Y_n - \mu\right) + \rho\left(\frac{X_n - X_0}{n}\right)\right]$$
   with $Y_n = \sum_{i=1}^{n} X_i$.
   (c) Show that
   $$plim\left[\frac{\rho}{1-\rho}\left(\frac{X_n - X_0}{\sqrt{n}}\right)\right] = 0$$
   Hint: Use Tchebychev's Inequality.
   (d) Put your results of (a),(b) and (c) together and derive the asymptotic distribution of the sample mean. That is, show that
   $$Z_n = \sqrt{n}\frac{\left(\frac{1}{n}Y_n\right) - \mu}{\sigma} \xrightarrow{d} U \sim N(0,1)$$
   for $\sigma = \sqrt{\sigma_\varepsilon^2/(1-\rho)^2}$.
3. Write an R program to demonstrate the central limit theorem for the AR(1) process.

9.4 Central limit theorem for dependent data

1. First let's derive the expectation and variance of the AR(1) process with $|\rho| < 1$. For this, we use recursive substitution techniques given a starting value X_0:
   $$X_i = (1-\rho)(1+\rho+\rho^2+\cdots+\rho^N)\mu + \varepsilon_i + \rho\varepsilon_{i-1} + \rho^2\varepsilon_{i-2} + \cdots + \rho^N\varepsilon_{i-N} + \rho^{N+1}X_0$$
   Note that $\lim_{N\to\infty}\rho^{N+1} = 0$ and $\lim_{N\to\infty}\sum_{j=0}^{\infty}\rho^j = \frac{1}{1-\rho}$, since $|\rho| < 1$. The AR(1) process with $|\rho| < 1$ can therefore be equally represented by
   $$X_i = \mu + \sum_{j=1}^{\infty}\rho^j\varepsilon_{i-j}$$
   Its expectation and variance are then equal to
   $$E(X_i) = \mu + \sum_{j=1}^{\infty}\rho^j E(\varepsilon_{i-j}) = \mu$$
   $$Var(X_i) = \sum_{j=1}^{\infty}(\rho^j)^2 var(\varepsilon_{i-j}) = \sum_{j=1}^{\infty}(\rho^2)^j\sigma_\varepsilon^2 = \frac{\sigma_\varepsilon^2}{1-\rho^2}$$

2. Asymptotic distribution for mean

(a) Due to our assumptions on ε_i, we can use the central limit theorem such that

$$\sqrt{n}\left(\frac{1}{n}\sum_{i=1}^{n}\varepsilon_i\right) = \frac{1}{\sqrt{n}}\sum_{i=1}^{n}\varepsilon_i \xrightarrow{d} U_\varepsilon \sim N(0,\sigma_\varepsilon^2)$$

(b) Let's have a look at $\frac{1}{\sqrt{n}}\sum_{i=1}^{n}\varepsilon_i$:

$$\begin{aligned}
\frac{1}{\sqrt{n}}\sum_{i=1}^{n}\varepsilon_i &= \frac{1}{\sqrt{n}}\sum_{i=1}^{n}[(X_i-\mu)-\rho(X_{i-1}-\mu)] \\
&= \frac{1}{\sqrt{n}}\left[\sum_{i=1}^{n}(X_i-\mu)-\rho\sum_{i=1}^{n}(X_{i-1}-\mu)\right] \\
&= \frac{1}{\sqrt{n}}\left[\sum_{i=1}^{n}(X_i-\mu)-\rho\left[\sum_{i=1}^{n}(X_i-\mu)-(X_n-X_0)\right]\right] \\
&= \sqrt{n}\left[\frac{1}{n}\sum_{i=1}^{n}(X_i-\mu)-\rho\left[\frac{1}{n}\sum_{i=1}^{n}(X_i-\mu)-\left(\frac{X_n-X_0}{n}\right)\right]\right] \\
&= \sqrt{n}\left[\frac{1}{n}Y_n-\mu-\rho\left[\frac{1}{n}Y_n-\mu-\left(\frac{X_n-X_0}{n}\right)\right]\right] \\
&= \sqrt{n}\left[(1-\rho)\left(\frac{1}{n}Y_n-\mu\right)+\rho\left(\frac{X_n-X_0}{n}\right)\right]
\end{aligned}$$

(c) Using the definition of the probability limit, we have to show that for any $\delta>0$

$$\lim_{n\to\infty} P\left(\left|\frac{\rho}{1-\rho}\left(\frac{X_n-X_0}{\sqrt{n}}\right)\right|>\delta\right)=0$$

By Tchebychev's Inequality we have

$$P\left(\left|\frac{\rho}{1-\rho}\left(\frac{X_n-X_0}{\sqrt{n}}\right)\right|>\delta\right)\le\frac{1}{\delta^2}var\left[\frac{\rho}{1-\rho}\left(\frac{X_n-X_0}{\sqrt{n}}\right)\right]$$

Let's have a look at $var\left[\frac{\rho}{1-\rho}\left(\frac{X_n-X_0}{\sqrt{n}}\right)\right]$:

$$\begin{aligned}
var\left[\frac{\rho}{1-\rho}\left(\frac{X_n-X_0}{\sqrt{n}}\right)\right] &= \frac{1}{n}\left(\frac{\rho}{1-\rho}\right)^2 var(X_n-X_0) \\
&= \frac{1}{n}\left(\frac{\rho}{1-\rho}\right)^2 (var(X_n)+var(X_0)-2cov(X_n,X_0)] \\
&= \frac{1}{n}\left(\frac{\rho}{1-\rho}\right)^2\left[\frac{\sigma_\varepsilon^2}{1-\rho^2}+\frac{\sigma_\varepsilon^2}{1-\rho^2}-2corr(X_n,X_0)\sqrt{\frac{\sigma_\varepsilon^2}{1-\rho^2}}\sqrt{\frac{\sigma_\varepsilon^2}{1-\rho^2}}\right] \\
&\le \frac{1}{n}\left(\frac{\rho}{1-\rho}\right)^2 4\left(\frac{\sigma_\varepsilon^2}{1-\rho^2}\right)
\end{aligned}$$

since $corr(X_n,X_0)\ge -1$.
Thus for any $\delta>0$, we have

$$P\left(\left|\frac{\rho}{1-\rho}\left(\frac{X_n-X_0}{\sqrt{n}}\right)\right|>\delta\right)\le\frac{1}{\delta^2}\frac{1}{n}\left(\frac{\rho}{1-\rho}\right)^2 4\left(\frac{\sigma_\varepsilon^2}{1-\rho^2}\right)$$

In the limit

$$\lim_{n\to\infty} P\left(\left|\frac{\rho}{1-\rho}\left(\frac{X_n - X_0}{\sqrt{n}}\right)\right| > \delta\right) = 0.$$

(d) Now, let's go back to

$$\frac{1}{\sqrt{n}}\sum_{i=1}^{n}\varepsilon_i = \sqrt{n}\left[(1-\rho)\left(\frac{1}{n}Y_n - \mu\right) + \rho\left(\frac{X_n - X_0}{n}\right)\right]$$

Let's divide by $(1-\rho)$

$$\frac{\frac{1}{\sqrt{n}}\sum_{i=1}^{n}\varepsilon_i}{1-\rho} = \sqrt{n}\left[\frac{1}{n}Y_n - \mu\right] + \frac{\rho}{1-\rho}\left(\frac{X_n - X_0}{\sqrt{n}}\right)$$

For the left-hand-side we have

$$\frac{\frac{1}{\sqrt{n}}\sum_{i=1}^{n}\varepsilon_i}{1-\rho} \xrightarrow{d} \tilde{U_\varepsilon} \sim N\left(0, \frac{\sigma_\varepsilon^2}{(1-\rho)^2}\right)$$

Since $plim\left[\frac{\rho}{1-\rho}\left(\frac{X_n - X_0}{\sqrt{n}}\right)\right] = 0$, we have

$$\sqrt{n}\left[\frac{1}{n}Y_n - \mu\right] \xrightarrow{d} \tilde{U} \sim N\left(0, \frac{\sigma_\varepsilon^2}{(1-\rho)^2}\right)$$

and we're done. That is, set $\sigma^2 = \frac{\sigma_\varepsilon^2}{(1-\rho)^2}$, then

$$Z_n = \sqrt{n}\frac{\left(\frac{1}{n}Y_n\right) - \mu}{\sigma} \xrightarrow{d} U \sim N(0,1)$$

3. The program might look like this:

```
##################################################
### Central limit theorem for dependent data ###
##################################################
# AR(1) process
library(MASS)
n <- 10000
N <- 5000 # how many times to draw individual X_i's, note that i = 1,2,..,n
mu <- 0
rho=0.8
sigma_eps <- 0.5 #this is the standard deviation of the random term in the AR(1) process
var_x <- sigma^2/(1-rho^2) # analytical variance of an AR(1)-process
factr <- var_x*(1+rho)/(1-rho) # this is the required adjustment
X <- matrix(NA,N,n)
for (j in 1:N) {
  X[j,] <- filter((1-rho)*mu+rnorm(n,sd=sigma),rho,method="recursive",init=(1-rho)*mu)
}
Yn <- rowSums(X)
Zn <- sqrt(n)*(Yn/n - mu)/sqrt(factr) #standardization
truehist(Zn, main = paste("n =", toString(n),sep =" "))
coord <- par("usr")
x <- seq(coord[1], coord[2], length.out = 500)
lines(x, dnorm(x), col = "red")
```

9.5 Delta method

A very important linear transformation is a first order Taylor approximation. First, we consider the univariate case. Suppose $X_1, X_2, \ldots$ are iid with common mean μ and variance σ^2. Define $\bar{X}_n = (X_1 + \cdots + X_n)/n$.

1. Derive the asymptotic distribution of
$$\sqrt{n}(f(\bar{X}_n) - f(\mu)),$$
where f is differentiable (at least at μ) and its derivative is continuous.

2. Now turn to the multivariate case and let $X_1, X_2, \ldots$ be random iid vectors of length K with common mean vector μ and covariance matrix Σ. Define $Y = f(\bar{X}_n)$ where f is a scalar valued differentiable function.[4] Denote the gradient of f as D_f and suppose it is continuous. Write down the first order Taylor approximation of f around μ and determine the approximate distribution of Y.

9.5 Delta method

1. According to a central limit theorem, we have
$$\sqrt{n}\left(\bar{X}_n - \mu\right) \overset{d}{\to} U \sim N(0, \sigma^2).$$
The first order Taylor expansion of $f(\bar{X}_n)$ around μ is
$$f\left(\bar{X}_n\right) \approx f\left(\mu\right) + f'\left(\mu_n\right)\left(\bar{X}_n - \mu\right),$$
where μ_n is an intermediate point between X and μ. Since $|\mu_n - \mu| \leq |\bar{X}_n - \mu|$ and $X_n \overset{p}{\to} \mu$, $\mu_n \overset{p}{\to} \mu$. Since f' is continuous we have $f'(\mu_n) \overset{p}{\to} f'(\mu)$.
We see that $f(\bar{X}_n)$ is just a linear transformation of $\bar{X}_n$. Since $\bar{X}_n$ is asymptotically normal, so is $f(\bar{X}_n)$. The asymptotic mean is equal to
$$E\left(f(\bar{X}_n)\right) = f\left(\mu\right) + f'\left(\mu\right)\left(E\left(X\right) - \mu\right) = f\left(\mu\right)$$
and the asymptotic variance is
$$\begin{aligned} Var\left(f(\bar{X}_n)\right) &= Var\left(f\left(\mu\right) + f'\left(\mu\right)\left(X - \mu\right)\right) \\ &= \left[f'\left(\mu\right)\right]^2 Var\left(X\right) = \left[f'\left(\mu\right)\right]^2 \sigma^2. \end{aligned}$$
So we have
$$\sqrt{n}\left(f(\bar{X}_n) - f(\mu)\right) = f'(\mu_n)\left[\sqrt{n}\left(\bar{X}_n - \mu\right)\right] \overset{d}{\to} f'(\mu)U \sim N(0, [f'(\mu)]^2\sigma^2)$$

2. The first order Taylor expansion of f around μ is
$$Y \approx f\left(\mu\right) + D_f\left(\mu\right)\left(\bar{X}_n - \mu\right)$$
Because Y is a linear transformation of $\bar{X}_n$, and $\bar{X}_n$ is asymptotically normal, so is Y. The asymptotic mean of Y is
$$E\left(Y\right) = f\left(\mu\right)$$
and the asymptotic variance is
$$\begin{aligned} Var\left(Y\right) &= Var\left(f\left(\mu\right) + D_f\left(\mu\right)\left(\bar{X}_n - \mu\right)\right) \\ &= D_f\left(\mu\right) Var\left(\bar{X}_n\right) D_f\left(\mu\right)' \\ &\to D_f\left(\mu\right) \Sigma D_f\left(\mu\right)'. \end{aligned}$$

[4] Of course, one could also consider vector-valued functions.

9.6 Limits of maxima (I)

Let $X_1, X_2, \ldots$ be an i.i.d. sequence of standard normally distributed random variables. Define the random variable

$$M_n = \max_{i=1,\ldots,n} X_i$$

and its normalized version $R_n = (M_n - d_n)/c_n$ where

$$\begin{aligned} d_n &= \sqrt{2\ln n} - \frac{\ln(4\pi) + \ln\ln n}{2\sqrt{(2\ln n)}} \\ c_n &= (2\ln n)^{-1/2}. \end{aligned}$$

1. Write an R program to illustrate that R_n converges in distribution.
2. The limit distribution of R_n is the Gumbel distribution. Add the Gumbel density $\exp(-x - e^{-x})$ to a histogram of R_n.

9.6 Limits of maxima (I)

The program might look like this:

```
#############################
### Limits of maxima (I) ###
#############################
library(MASS)
n <- 100
N <- 1000 # how many times to draw individual X_i's, note that i = 1,2,..,n
X <- matrix(rnorm(N*n),N, n) #matrix of N rows and n columns filled with random numbers
dn <- sqrt(2*log(n))-(log(4*pi)+log(log(n)))/(2*sqrt(2*log(n)))
cn <- (2*log(n))^(-1/2)
Mn <- apply(X, 1, max) # get max of each row
Rn <- (Mn - dn)/cn #standardization
#display the sequence of random variables
truehist(Rn, main = paste("n =", toString(n),sep =" "))
coord <- par("usr")
x <- seq(coord[1], coord[2], length.out = 500)
lines(x, exp(-x-exp(-x)), col = "red")
```

See also the Fisher-Tippett-Galambos Theorem which states that a sample of iid random variables after proper standardization can only converge in distribution to one of 3 possible distributions: the Gumbel distribution, the Fréchet distribution, or the Weibull distribution.

9.7 Limits of maxima (II)

Let $X_1, X_2, \ldots$ be an i.i.d. sequence of t-distributed random variables with 1.5 degrees of freedom. Define the random variables

$$M_n = \max_{i=1,\ldots,n} X_i$$

and its normalized version $R_n = M_n/c_n$ with

$$c_n = F_{t_{1.5}}^{-1}\left(1 - \frac{1}{n}\right)$$

where $F_{t_{1.5}}^{-1}$ is the quantile function of the $t_{1.5}$-distribution (see the R command `qt`).

1. Write an R program to illustrate that R_n converges in distribution.
2. The limit distribution of R_n is the Frechet distribution (with tail index 1.5). Add the Frechet density $1.5x^{-2.5}\exp\left(-x^{-1.5}\right)$ to a histogram of R_n.

9.7 Limits of maxima (II) The program might look like this:

```
################################
#### Limits of maxima (II) ####
################################
library(MASS)
n <- 100
N <- 1000 # how many times to draw individual X_i's, note that i = 1,2,..,n
X <- matrix(rt(n*N,df=1.5), N, n) #matrix of N rows and n columns filled with random numbers
Mn <- apply(X, 1, max) # get max of each row
Rn <- Mn/qt(1-1/n,1.5) #standardization
#display the sequence of random variables
truehist(Rn, main = paste("n =", toString(n),sep =" "))
coord <- par("usr")
x <- seq(coord[1], coord[2], length.out = 500)
lines(x, 1.5*x^(-2.5)*exp(-x^(-1.5)), col = "red")
```

See also the Fisher-Tippett-Galambos Theorem which states that a sample of iid random variables after proper standardization can only converge in distribution to one of 3 possible distributions: the Gumbel distribution, the Fréchet distribution, or the Weibull distribution.

9.8 Limits of maxima (III)

Let $X_1, X_2, \ldots$ be an i.i.d. sequence of random variables uniformly distributed on the interval $[0,1]$. Define the random variables

$$M_n = \max_{i=1,\ldots,n} X_i$$

and its normalized version $R_n = (M_n - d_n)/c_n$ where

$$\begin{aligned} d_n &= 1 \\ c_n &= \frac{1}{n}. \end{aligned}$$

1. Write an R program to illustrate that R_n converges in distribution.
2. The limit distribution of R_n is the Weibull distribution. Add the Weibull density $\exp(x)$ to a histogram of R_n.

9.8 Limits of maxima (III)

The program might look like this:

```
#################################
#### Limits of maxima (III) ####
#################################
library(MASS)
n <- 100
```

```
N <- 1000 # how many times to draw individual X_i’s, note that i = 1,2,..,n
X <- matrix(runif(n*N), N, n) #matrix of N rows and n columns filled with random numbers
Mn <- apply(X, 1, max) # get max of each row
dn <- 1
cn <- 1/n
Rn <- (Mn-dn)/cn #standardization
#display the sequence of random variables
truehist(Rn, main = paste("n =", toString(n),sep =" "))
coord <- par("usr")
x <- seq(coord[1], coord[2], length.out = 500)
lines(x, exp(x), col = "red")
```

See also the Fisher-Tippett-Galambos Theorem which states that a sample of iid random variables after proper standardization can only converge in distribution to one of 3 possible distributions: the Gumbel distribution, the Fréchet distribution, or the Weibull distribution.

10 Estimators and their properties

10.1 Counter examples

Let $X_1, X_2, \ldots$ be a sample from some random variable X with $E(X) = \mu$ and $Var(X) = 1$. While the variance is known, we would like to estimate the expectation μ.

1. Give an example of an estimator that is unbiased but inconsistent.
2. Give an example of an estimator that is biased but consistent.
3. Give an example of an estimator that is asymptotically biased but consistent.

10.1 Counter examples

We assume in addition that X is normally distributed, $X \sim N(\mu, 1)$.

1. The sequence of estimators $\hat{\mu}_n = X_1$ is unbiased since $E(\hat{\mu}_n) = E(X_1) = \mu$. But it is obviously not consistent.
2. The sequence of estimators $\hat{\mu}_n = \left(\frac{1}{n}\sum_{i=1}^n X_i\right) + 1/n$ is biased with bias $1/n$. For given n

$$\begin{aligned} E(\hat{\mu}_n) &= \mu + \frac{1}{n} \\ Var(\hat{\mu}_n) &= \frac{1}{n}, \end{aligned}$$

   and thus $\lim_{n\to\infty} E(\hat{\mu}_n) = \mu$ and $\lim_{n\to\infty} Var(\hat{\mu}_n) = 0$. **These are the sufficient conditions for consistency.**

3. This is the most tricky case. Consider the sequence of estimators

$$\hat{\mu}_n = \mathbf{1}\left(X_1 - \frac{1}{n}\sum_{i=1}^n X_i < \Phi^{-1}(1/n)\right) \cdot n + \left[1 - \mathbf{1}\left(X_1 - \frac{1}{n}\sum_{i=1}^n X_i < \Phi^{-1}(1/n)\right)\right] \cdot \frac{1}{n}\sum_{i=1}^n X_i$$

   where $\mathbf{1}(\cdot)$ is an indicator function which is 1 if the condition is true and 0 else, $\Phi^{-1}(1/n)$ is the $1/n$-quantile of the standard normal distribution. For large n, the term $X_1 - \frac{1}{n}\sum_{i=1}^n X_i$ is approximately $N(0,1)$ and the indicator variable will be 1 with probability $1/n$. Thus, the expectation of the first summand of $\hat{\mu}_n$ is

$$E\left(\mathbf{1}\left(X_1 - \frac{1}{n}\sum_{i=1}^n X_i < \Phi^{-1}(1/n)\right) \cdot n\right) = \frac{1}{n} \cdot n = 1.$$

   The expectation of the second summand is

$$E\left(\left[1 - \mathbf{1}\left(X_1 - \frac{1}{n}\sum_{i=1}^n X_i < \Phi^{-1}(1/n)\right)\right] \cdot \frac{1}{n}\sum_{i=1}^n X_i\right) = \left(1 - \frac{1}{n}\right) \cdot \mu,$$

   and hence the asymptotic bias of $\hat{\mu}_n$ is 1. The estimator is consistent because, as $n \to \infty$, the probability mass concentrates on the second summand which converges to μ.

11 Least Squares and Method of Moments

11.1 Nonlinear least squares

1. Consider the exponential model

$$y_i = \exp(\alpha + \beta x_i) + u_i$$

where $u_i \sim N(0, \sigma^2)$. Since the error term is additive one cannot simply take logarithms to make the model linear. Load the dataset `expgrowth.csv` from the course site and estimate the parameters α and β by minimizing

$$\sum_{i=1}^{n} (y_i - \exp(a + bx_i))^2$$

numerically with respect to a and b.

2. Consider the following example from Davidson and MacKinnon (2004),

$$y_t = \beta_1 + \beta_2 x_{t1} + \frac{1}{\beta_2} x_{t2} + u_t.$$

Assume that $u_t \sim N(0, 1)$. Load the dataset `DMacK1.csv` and estimate the parameters β_1 and β_2.

11.1 Nonlinear least squares

The two programs could look like this:

```
############################################
#### Nonlinear least squares estimation ####
############################################

#1)
# Load the dataset
dat <- read.csv(file.choose())

# Definition of the objective function
# The first argument must be the parameter vector
squarediffs <- function(param,dat) {
  alpha <- param[1]
  beta <- param[2]
  u <- dat$y-exp(alpha+beta*dat$x)
  return(sum(u^2))
}

# Minimize the objective function
obj <- optim(c(1,0),squarediffs,dat=dat)
estimates <- obj$par
alphahat <- estimates[1]
betahat <- estimates[2]

# Plot the data and the estimated regression curve
```

```
plot(dat$x,dat$y)
g <- seq(0,40,length=500)
lines(g,exp(alphahat+betahat*g))


#2)

# Load the dataset
dat <- read.csv(file.choose())

# Definition of the objective function
# The first argument must be the parameter vector
squarediffs <- function(param,dat) {
  beta1 <- param[1]
  beta2 <- param[2]
  u <- dat$y-(beta1+beta2*dat$x1+1/beta2*dat$x2)
  return(sum(u^2))
}

# Minimize the objective function
obj <- optim(c(1,1),squarediffs,dat=dat)
estimates <- obj$par
beta1hat <- estimates[1]
beta2hat <- estimates[2]

# Plot the data and the estimated regression plane
library(rgl)
plot3d(dat$x1,dat$x2,dat$y)
gx1 <- seq(min(dat$x1),max(dat$x1),length=60)
gx2 <- seq(min(dat$x2),max(dat$x2),length=60)
yhat <- outer(gx1,gx2,function(x1,x2) beta1hat+beta2hat*x1+1/beta2hat*x2)
surface3d(gx1,gx2,yhat,col="light green")
```

11.2 Method of moments for the binomial distribution

Consider the binomial distribution $Binom(n,\theta)$ with parameters $n > 0$ and $0 < \theta < 1$. The expectation and variance of $X \sim Binom(n,\theta)$ are

$$\begin{aligned} E(X) &= n\theta \\ Var(X) &= n\theta\,(1-\theta)\,. \end{aligned}$$

Derive the moment estimators of n and θ. Ignore the restriction $n \in \mathbb{N}$.

11.2 Method of moments for the binomial distribution

The first step is already given in the text,

$$\begin{aligned} \mu_1 &= n\theta \\ \mu_2' &= n\theta\,(1-\theta)\,. \end{aligned}$$

The second step is the inversion of the two equations,

$$\begin{aligned}\theta &= 1-\frac{\mu_2'}{\mu_1}\\ n &= \frac{\mu_1}{\theta}.\end{aligned}$$

Finally, the theoretical moments are replaced by their empirical counterparts, and the moment estimators are

$$\begin{aligned}\hat{\theta} &= 1-\frac{\hat{\mu}_2'}{\hat{\mu}_1}\\ \hat{n} &= \frac{\hat{\mu}_1}{\hat{\theta}}.\end{aligned}$$

11.3 Method of moments for the geometric distribution

Consider the geometric distribution with parameter λ. The expectation of $X \sim Geom(\lambda)$ is $E(X) = 1/\lambda$.

1. Give a moment estimator of λ.
2. Explain why the moment estimator is biased.
3. Explain why the moment estimator is consistent.

11.3 Method of moments for the geometric distribution

1. From $\mu_1 = \lambda^{-1}$ we derive $\lambda = \mu_1^{-1}$, and hence the moment estimator of λ is $\hat{\lambda} = \hat{\mu}_1^{-1}$.
2. The empirical moment $\hat{\mu}_1$ is an unbiased estimator of μ_1. The estimator $\hat{\lambda} = 1/\hat{\mu}_1$ is a nonlinear (convex) transformation. According to Jensen's inequality $E\left(\hat{\lambda}\right) = E(1/\hat{\mu}_1) > 1/E(\hat{\mu}_1) = \lambda$.
3. The rules of calculus for the probability limit are simple,

$$plim\hat{\lambda} = plim\frac{1}{\hat{\mu}_1} = \frac{1}{plim\hat{\mu}_1} = \frac{1}{\mu_1} = \lambda.$$

11.4 Method of moments for the Gumbel distribution

Consider the Gumbel distribution (also called extreme value distribution) with parameters $\alpha \in \mathbb{R}$ and $\beta \in \mathbb{R}$. The expectation and variance of $X \sim Gumbel(\alpha, \beta)$ are

$$\begin{aligned}E(X) &= \alpha + 0.5772 \cdot \beta\\ Var(X) &= \frac{1}{6}\beta^2\pi^2.\end{aligned}$$

Derive the moment estimators.

11.4 Method of moments for the Gumbel distribution

Step 1:

$$\begin{aligned}\mu_1 &= \alpha + 0.5772 \cdot \beta \\ \mu_2' &= \frac{1}{6}\beta^2\pi^2.\end{aligned}$$

Step 2:

$$\begin{aligned}\beta &= \sqrt{\frac{6\mu_2'}{\pi^2}} \\ \alpha &= \mu_1 - 0.5772 \cdot \beta.\end{aligned}$$

Step 3:

$$\begin{aligned}\hat{\beta} &= \sqrt{\frac{6\hat{\mu}_2'}{\pi^2}} \\ \hat{\alpha} &= \hat{\mu}_1 - 0.5772 \cdot \hat{\beta}.\end{aligned}$$

11.5 Method of moments for the Pareto distribution

The Pareto distribution has two parameters, $K \leq x < 0$ and $\alpha > 0$ and density $f_X(x) = \alpha K^\alpha x^{-\alpha-1}$. The expectation and variance of $X \sim Pareto(K, \alpha)$ are

$$\begin{aligned}E(X) &= \frac{\alpha K}{\alpha - 1} \\ Var(X) &= \frac{\alpha K^2}{(\alpha-2)(\alpha-1)^2}.\end{aligned}$$

1. Derive the moment estimators. What happens if $\alpha < 2$?
2. Write an R program to simulate the distribution of the moment estimator of $\alpha = 5$ (with $K = 1$ fixed). Generate $R = 10000$ samples $X_1, \ldots, X_n$ of size $n = 100$ each. What happens if you increase the sample size to $n = 1000$? What happens if you consider an $\alpha < 2$?

11.5 Method of moments for the Pareto distribution

1. For a given K the moment estimator of α is given by

$$\begin{aligned}\mu_1 &= \frac{\alpha K}{\alpha - 1} \\ \alpha &= \frac{\mu_1}{\mu_1 - K} \\ \hat{\alpha} &= \frac{\hat{\mu_1}}{\hat{\mu_1} - K}\end{aligned}$$

For both parameters unknown one has to solve the system of equations

$$\begin{aligned}\mu_1 &= \frac{\alpha K}{\alpha - 1} \\ \mu_2' &= \frac{\alpha K^2}{(\alpha-2)(\alpha-1)^2}\end{aligned}$$

for α and K. Inserting the squared first expression into the second yields

$$\begin{aligned}
\mu_2' &= \frac{\mu_1^2}{\alpha(\alpha-2)} \\
\Leftrightarrow \alpha^2 - 2\alpha - \frac{\mu_1^2}{\mu_2'} &= 0 \\
\Leftrightarrow \alpha &= \frac{2}{2} \pm \frac{\sqrt{4(1+\frac{\mu_1^2}{\mu_2'})}}{2} = 1 \pm \sqrt{1+\frac{\mu_1^2}{\mu_2'}}
\end{aligned}$$

The variance is negative for $0 < \alpha < 2$ $(Var(X) < 0)$. Thus, α must be greater than 2 and the moment estimators are

$$\begin{aligned}
\hat{\alpha} &= 1 + \sqrt{1+\frac{\hat{\mu_1}^2}{\hat{\mu_2}'}} \\
\hat{K} &= \frac{\alpha-1}{\alpha}\hat{\mu}_1
\end{aligned}$$

2. The program might look like this:

```
####################################################
#### Moment estimator for Pareto distribution ####
####################################################

install.packages("VGAM") # in order to get the Pareto distribution
library(VGAM)
library(AER)

a <- 5              #shape parameter
K <- 1              #location parameter
R <- 10000          # index for the loop
n <- 100            # number of random variables
ahat <- rep(NA,R)   # initialize estimator
ahat1 <- rep(NA,R)  # initialize estimator

for (i in 1:R) {
  x <- rpareto(n,location=K,shape=a)      #generate n random variables
  ahat[i] <- mean(x)/(mean(x)-1)          #moment estimator of alpha if K is known
  ahat1[i] <- 1+sqrt(1+mean(x)^2/var(x)) #moment estimator of alpha if K is unknown
}
#graphical output
par(mfrow=c(1,2))
truehist(ahat)
g <- seq(min(ahat),max(ahat),length=300)
lines(g,dnorm(g,mean(ahat),sd(ahat)))
truehist(ahat1)
h <- seq(min(ahat1),max(ahat1),length=300)
lines(h,dnorm(h,mean(ahat1),sd(ahat1)))
```

For $\alpha < 2$ the second estimator is not asymptotically normal distributed, since the moment of order $2p$ does not exist (see the lecture for the necessary conditions regarding moment estimators).

11.6 Method of moments for the uniform distribution

1. Consider the uniform distribution with parameters a and b (where $b > a$). The expectation and variance of $X \sim unif(a,b)$ are

$$\begin{aligned} E(X) &= \frac{a+b}{2} \\ Var(X) &= \frac{(b-a)^2}{12}. \end{aligned}$$

Derive the moment estimators.

2. Write an R program to simulate the distribution of the moment estimators of $a = 0$ and $b = 1$. Generate $R = 10000$ samples $X_1, \ldots, X_n$ of size $n = 40$ each. Are the estimators approximately normally distributed? Check if the moment estimator of a is always smaller than (or equal to) the minimum in the sample.

11.6 Method of moments for the uniform distribution

1. The theoretical moments μ_1 and μ_2' are already given as functions of the unknown parameters. Solving the system

$$\begin{aligned} 2\mu_1 &= a+b \\ 12\mu_2' &= a^2 - 2ab + b^2 \end{aligned}$$

for a and b yields two solutions. Since $b > a$ only one solution is valid,

$$\begin{aligned} a &= \mu_1 - \sqrt{3\mu_2'} \\ b &= \mu_1 + \sqrt{3\mu_2'} \end{aligned}$$

and the moment estimators are

$$\begin{aligned} \hat{a} &= \hat{\mu}_1 - \sqrt{3\hat{\mu}_2'} \\ \hat{b} &= \hat{\mu}_1 + \sqrt{3\hat{\mu}_2'}. \end{aligned}$$

2. The program might look like this:

```
####################################################
#### Moment estimator for uniform distribution ####
####################################################
a <- 0
b <- 1
R <- 10000
ahat <- rep(NA,R)
bhat <- rep(NA,R)
minx <- rep(NA,R)
for(r in 1:R) {
  x <- runif(n=40,a,b)
  ahat[r] <- mean(x)-sqrt(3*var(x))
  bhat[r] <- mean(x)+sqrt(3*var(x))
  minx[r] <- min(x)
  }
par(mfrow=c(2,1))
```

```
truehist(ahat)
g <- seq(min(ahat),max(ahat),length=300)
lines(g,dnorm(g,mean(ahat),sd(ahat)))
truehist(bhat)
g <- seq(min(bhat),max(bhat),length=300)
lines(g,dnorm(g,mean(bhat),sd(bhat)))
print("Proportion of impossible estimates:")
print(sum(minx<ahat)/R)
```

11.7 Method of moments for the linear regression model

Consider the linear regression model under standard assumptions

$$y = X\beta + u.$$

Suppose that X may be non-stochastic but independent of u, i.e. $E(u|X) = 0$. Left-multiply the model equation by X' and take expectations. Show that the method of moment estimator of β is identical to the OLS estimator.

11.7 Method of moments for the linear regression model

First have a look at $X'y$:

$$\underset{k\times T}{X}' \underset{T\times 1}{y} = \begin{pmatrix} \sum_{t=1}^{T} X_{t1}\cdot y_t \\ \vdots \\ \sum_{t=1}^{T} X_{tk}\cdot y_t \end{pmatrix}$$

Taking the expectation yields for each row j: $\sum_{t=1}^{T} E[X_{tj}\cdot y_t] = T\cdot E(X_{1j}\cdot y_1)$ or any other vector of y and item X_{tj}.

Now, left-multiplication of $y = X\beta + u$ with X' and taking expectations yields

$$E\left(X'y\right) = E\left(X'X\beta\right) + E\left(X'u\right) = T\begin{pmatrix} E(X_{11}\cdot y_1) \\ \vdots \\ E(X_{1k}\cdot y_1) \end{pmatrix}.$$

Step 1: We relax the standard assumptions and assume that X may be non-stochastic but independent of u, i.e. $E(u|X) = 0$. This implies that $E\left(X'X\beta\right) = E\left(X'X\right)\beta$ and $E\left(X'u\right) = E\left(X'\right)E(u)$. Of course, $E(u) = 0$, and thus

$$\begin{aligned} E\left(X'y\right) &= E\left(X'X\beta\right) + E\left(X'u\right) \\ E\left(X'y\right) &= E\left(X'X\right)\beta. \end{aligned}$$

Step 2: The system is solved for the unknown parameters β,

$$\begin{aligned} \beta = & \underbrace{\left[E\left(X'X\right)\right]^{-1}}_{\text{Matrix of expectations}} \cdot \underbrace{E\left(X'y\right)}_{\text{Vector of expectations}} \\ = & \frac{1}{T}\cdot\begin{bmatrix} E(X_{11}^2) & \dots & E(X_{11}X_{1k}) \\ \vdots & \ddots & \vdots \\ E(X_{11}X_{1k}) & \dots & E(X_{1k}^2) \end{bmatrix}^{-1} \cdot T\begin{pmatrix} E(X_{11}\cdot y_1) \\ \vdots \\ E(X_{1k}\cdot y_1) \end{pmatrix}. \end{aligned}$$

Step 3: Replace the theoretical moments (expectations) by their empirical counterparts,

$$\hat{\beta} = \begin{bmatrix} \frac{1}{T}\sum_{t=1}^{T} X_{t1}^2 & \dots & \frac{1}{T}\sum_{t=1}^{T} X_{t1}X_{tk} \\ \vdots & \ddots & \vdots \\ \frac{1}{T}\sum_{t=1}^{T} X_{t1}X_{tk} & \dots & \frac{1}{T}\sum_{t=1}^{T} X_{tk}^2 \end{bmatrix}^{-1} \cdot \begin{pmatrix} \frac{1}{T}\sum_{t=1}^{T} X_{t1}\cdot y_t \\ \vdots \\ \frac{1}{T}\sum_{t=1}^{T} X_{tk}\cdot y_t \end{pmatrix} = (X'X)^{-1}X'y.$$

This is equal to the OLS estimator of β (and also identical to the maximum likelihood estimators, as we will see later).

12 Maximum likelihood estimation

12.1 Extreme values

Let $X \sim Pareto(K, \alpha)$ where the parameter $K \leq x < 0$ is known but the tail parameter α is unknown. The density function of Pareto distribution is

$$f_X(x) = \alpha K^\alpha x^{-\alpha-1}.$$

1. Derive the maximum likelihood estimator of α.
2. The Pareto distribution is an excellent approximation of large daily stock return losses (of, say, more than 2%). Load the dataset `daxreturns.csv`. It contains the daily DAX returns (in %) from 16/7/2001 to 13/7/2011 (without holidays). Multiply all DAX returns by (-1) in order to make losses positive, delete all losses that are smaller than 2%, and estimate the tail parameter α for the remaining observations.
3. Plot the likelihood of the observations as a function of α.

12.1 Extreme values

1. The maximum likelihood estimator of α is given by maximizing the loglikelihood function:

$$\begin{aligned}
L(\alpha; x, K) &= \prod_{i=1}^{n} \alpha K^\alpha x_i^{-\alpha-1} \\
log(L(\alpha; x, K)) &= \sum_{i=1}^{n} log(\alpha K^\alpha) + \sum_{i=1}^{n} log(x_i^{-\alpha-1}) \\
&= n(log(\alpha) + \alpha log(K)) - (\alpha+1)\sum_{i=1}^{n} log(x_i) \\
\frac{\partial log(L)}{\partial \alpha} &= \frac{n}{\alpha} + n\ log(K) - \sum_{i=1}^{n} log(x_i) = 0 \\
\Rightarrow \hat{\alpha} &= \frac{n}{\sum_{i=1}^{n}(log(x_i) - log(K))}
\end{aligned}$$

2./3. The code might look like this. The ML-estimator is also estimated by numerically optimizing the likelihood.

```
########################
#### Extreme values ####
########################
library(MASS)
library(VGAM)

daxreturns <- read.csv(file.choose(), sep=";", dec=",")
View(daxreturns)
daxret <- daxreturns$daxret
plot(daxret)
tmp <- daxret*(-1)
daxlosses <- tmp[tmp>=2]
```

```
alphahat <- length(daxlosses)/(sum(log(daxlosses) - log(2)))
plot(daxlosses)
truehist(daxlosses)
coord <- par("usr")
x <- seq(coord[1], coord[2], length.out = length(daxlosses))
lines(x,dpareto(x,scale=2,shape=alphahat),col="red",lwd=2)

#Likelihood of the observations as a function of alpha
loglikelihood <- function(alpha,K,x) {
  n <- length(x)
  z <- n*(log(alpha)+alpha*log(K)) - (alpha+1)*sum(log(x))
  return(z)
}
alpha <- seq(0,10,length.out=100)
plot(alpha,loglikelihood(alpha,2,daxlosses))
abline(v=alphahat)

#Just for comparison, let’s do it numerically
likelihood <- function(alpha,K,x) {
  z <- prod(dpareto(x,K,alpha))
  return(z)
}

f <- function(x) -likelihood(x,2,daxlosses)
optimize(f,lower=0,upper=3)
```

12.2 Parameters of the uniform distribution

Consider the uniform distribution on the interval $[a, b]$ with density

$$f_X(x) = \begin{cases} \frac{1}{b-a} & \text{for } a \leq x \leq b \\ 0 & \text{else.} \end{cases}$$

1. Derive the maximum likelihood estimators of a and b.
2. Write an R program to generate $R = 10000$ samples of size $n = 100$ each. For each sample compute and store the maximum likelihood estimates $\hat{a}$ and $\hat{b}$. Plot their histograms.

12.2 Parameters of the uniform distribution

1. Consider a sample $x_1, \ldots, x_n$ and let the order statistics be: $x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(n)}$. For $\alpha \leq x_{(1)}$ and $b \geq x_{(n)}$ the likelihood equals:

$$\begin{aligned} L(a, b; x) &= \prod_{i=1}^{n} f_X(x_i|a, b) = (b-a)^{-n} \\ log(L(a, b; x)) &= -n \cdot log(b-a) \\ \frac{\partial log(L)}{\partial a} &= \frac{n}{b-a} > 0 \\ \frac{\partial log(L)}{\partial b} &= -\frac{n}{b-a} < 0 \end{aligned}$$

Thus the loglikelihood is a strictly increasing function in a and a strictly decreasing function in b. Hence the ML-estimator is given by:

$$\hat{a} = x_{(1)} \text{ and } \hat{b} = x_{(n)}$$

2. The program might look like this:

```
#################################################
#### Parameters of the uniform distribution ####
#################################################
R <- 10000
n <- 100
a <- 3
b <- 6
ahat <- numeric(R)
bhat <- numeric(R)
for (r in 1:R) {
  x <- runif(n,min=a,max=b)
  ahat[r] <- min(x)
  bhat[r] <- max(x)
}
par(mfrow=c(1,2))
hist(ahat,prob=T)
hist(bhat,prob=T)
```

Clearly, the ML estimators are not asymptotically normal distributed. See also execise 'Limits of maxima (III)'.

12.3 Censored lognormal distribution

Let $X \sim LN(\mu, \sigma^2)$ and let $X_1, \ldots, X_n$ be a sample drawn from X. The X_i are not observable. Instead one can only observe

$$Y_i = \begin{cases} X_i & \text{if } X_i < c \\ c & \text{if } X_i \geq c \end{cases}$$

where c is a known constant. The likelihood of $Y_1, \ldots, Y_n$ is the product of all densities $f_X(y_i)$, for observations with $Y_i < c$, times the product of all probabilities that $Y_i = c$ for observations with $Y_i = c$.

1. Write an R function that computes the likelihood of μ and σ^2 given the observations $Y_1, \ldots, Y_n$ (and given c).
2. Load the dataset `censoredln.csv`.
3. Numerically maximize the likelihood function. The censoring value is $c = 12$.
4. Compute the asymptotic covariance matrix of $\hat{\mu}$ and $\hat{\sigma}^2$.

12.3 Censored lognormal distribution

First, let's consider the probability of $Y_i = c$:

$$Pr(Y_i = c) = Pr(X_i \geq c) = 1 - Pr(X_i \leq c) = 1 - F_X(c)$$

The likelihood function is now a mixture between the product of all densities $f_X(y_i)$, for observations with $Y_i < c$, times the product of all probabilities that $Y_i = c$ for observations with $Y_i = c$:

$$L(\mu, \sigma; y) = \prod_{i=1}^{n} \{f_X(y_i; \mu, \sigma)\}^{\delta_i} \{1 - F_X(c; \mu, \sigma)\}^{1-\delta_i}$$

with $\delta_i = 1$ for exact observations and $\delta_i = 0$ for a censored observation. The loglikelihood is thus the sum of those two components (let n_1 be the number of non-censored observations and n_2 the number of censored observations, $n_1 + n_2 = n$):

$$\log L(\mu, \sigma; y) = \sum_{i=1}^{n_1} \log f_X(y_i; \mu, \sigma) + \sum_{i=1}^{n_2} \log(1 - F_X(c; \mu, \sigma))$$

The code might look like this:

```
##########################################
#### Censored lognormal distribution ####
##########################################
# Definition of loglikelihood
neglogl <- function(param,dat,cens) {
  mu <- param[1]
  sigma <- param[2]
  y <- dat
  c <- cens
  z <- sum(log(dlnorm(y[y<c],meanlog=mu,sdlog=sigma)))
  + sum(log(1-plnorm(y[y==c],meanlog=mu,sdlog=sigma)))
  return(-z)
}
# Get data
censoredln <- read.table(file.choose(), header=T, quote="\"")
x <- censoredln$x
# Optimization
obj <- optim(c(0,1),neglogl,dat=x,cens=12,hessian=T)
print(obj$par)  # Point estimates
print(solve(obj$hessian)) # Numerical covariance matrix
```

12.4 Exponential model

Consider the exponential model

$$y_i = \exp(\alpha + \beta x_i) + u_i$$

and load the dataset `expgrowth.csv` from the course site.

1. Assume that the error terms are i.i.d. and $u_i \sim N(0, \sigma^2)$. Write an R function that calculates the log-likelihood of α, β and σ^2.
2. Numerically find the maximum likelihood estimates of α, β and σ^2. Compare your results with exercise 11.1.
3. Compute the asymptotic covariance matrix of $\hat{\alpha}$, $\hat{\beta}$ and $\hat{\sigma}^2$.

4. Assume that the error terms are i.i.d. with known density

$$f_{u_i}(u) = \frac{1}{2}\exp\left(-|u|\right).$$

Numerically find the estimates of α and β.

12.4 Exponential model

For the log-likelihood you have to consider the distribution of the error term:

$$u_i = y_i - exp\{\alpha + \beta x_i\} \sim N(0, \sigma^2)$$

1)/2)/3) The distributional assumption for the likelihood is thus the density of the normal distribution (f_{u_i}) and the log-likelihood is given by

$$\log L = \sum_{i=1}^{n} f_u(y_i - exp\{\alpha + \beta x_i\})$$

4) Now the log-likelihood is given by

$$\log L = -n\log(2) - \sum_{i=1}^{n} |\alpha + \beta x_i|$$

The code might look like this:

```
############################
#### Exponential model ####
############################
# Get data
expgrowth <- read.csv(file.choose())
View(expgrowth)

#1: Definition of negative loglikelihood
neglogl <- function(param,dat){
  a <- param[1]
  b <- param[2]
  sigma <- param[3]
  x <- dat[,2]
  y <- dat[,1]
  u <- y-exp(a+b*x)
  z <- sum(log(dnorm(u,mean=0,sd=sigma)))
  zz <- sum(log(dnorm(y,mean=exp(a+b*x),sd=sigma))) #works as well
  return(-z)
}

#2: Numerical Optimization
# Optimization, for start values one has to consider that the exponential function is very
# sensitive to the parameters a and b -> use small ones
obj <- optim(c(0,0.1,1),neglogl,dat=expgrowth,hessian=T)
print(obj$par)  # Point estimates

#3: Asymptotic numerical covariance matrix
```

```
print(solve(obj$hessian))

#4: An exponential density function for the error terms
#Definition of negative loglikelihood
neglogl2 <- function(param,dat){
  a <- param[1]
  b <- param[2]
  x <- dat[,2]
  y <- dat[,1]
  u <- y-exp(a+b*x)
  z <- -n*log(2) - sum(abs(u))     #analytically
  zz <- sum(log(0.5*exp(-abs(u)))) #alternatively
  return(-z) #or return(-zz)
}
obj2 <- optim(c(0,0.1,1),neglogl2,dat=expgrowth,hessian=T)
print(obj2$par)  # Point estimates
```

12.5 Tobit model

The Tobit model is a linear regression model where observations are censored from below at zero. A latent (unobservable) variable y_t^* is assumed to depend linearly on a vector x_t of exogenous variables,

$$y_t^* = x_t'\beta + u_t$$

where $u_t \sim N(0, \sigma^2)$. The observations are

$$y_t = \begin{cases} y_t^* & \text{if } y_t^* > 0 \\ 0 & \text{else.} \end{cases}$$

1. Given the vector of exogenous variables x_t, derive the probability that $y_t = 0$.
2. The likelihood of $y_1, \ldots, y_T$ is the product of all densities $f_{y_t}(y_t)$, for observations with $y_t > 0$, times the product of all probabilities that $y_t = 0$ for observations with $y_t = 0$. Derive the log-likelihood.
3. Load the dataset `tobitbsp.csv`. The dataset contains the observed endogenous variable `y` and three exogenous variables `x1`, `x2`, `x3` (where `x1` is just a vector of ones). The data are simulated but have similar means, crossproducts etc. as the data in "Estimation of Relationships for Limited Dependent Variables" by James Tobin, *Econometrica*, 26 (1958) 24-36.[5] Numerically compute the maximum likelihood estimates $\hat{\beta}$ and $\hat{\sigma}^2$.
4. Estimate an OLS regression without taking into account the censoring at zero. Compare the OLS estimates with the Tobit estimates.
5. Compute the standard errors of $\hat{\beta}$ and $\hat{\sigma}^2$.

12.5 Tobit model

First, let's consider censored observations, for $y_t^* \le 0$:

$$Pr(y_t = 0) = Pr(y_t^* \le 0) = Pr(u_t \le -x_t'\beta) = Pr\left(\frac{u_t}{\sigma} \le \frac{-x_t'\beta}{\sigma}\right) = \Phi\left(\frac{-x_t'\beta}{\sigma}\right)$$

[5]This was the first article to use Tobit estimation (although the name was coined later); it can be downloaded from the course site.

with Φ being the cdf of the standard normal distribution. For uncensored observations we have a linear model, i.e. the likelihood for $y_t^* > 0$:

$$f(u_t) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(y_t^*-x_t'\beta)^2}{2\sigma^2}} = \frac{1}{\sigma}\frac{1}{\sqrt{2\pi}} e^{\frac{-1}{2}\left(\frac{y_t^*-x_t'\beta}{\sigma}\right)} = \frac{1}{\sigma}\phi\left(\frac{y_t^*-x_t'\beta}{\sigma}\right)$$

with ϕ being the pdf of the standard normal distribution. The likelihood function is now a mixture:

$$L(\mu,\sigma;y) = \prod_{t=1}^{n}\left\{\frac{1}{\sigma}\phi\left(\frac{y_t^*-x_t'\beta}{\sigma}\right)\right\}^{\delta_t}\left\{\Phi\left(\frac{-x_t'\beta}{\sigma}\right)\right\}^{1-\delta_t}$$

with $\delta_t = 1$ for exact observations and $\delta_t = 0$ for a censored observation. The loglikelihood is thus the sum of those two components:

$$\log L(\mu,\sigma;y) = \sum_{y_t>0}\log\left(\frac{1}{\sigma}\phi\left(\frac{y_t^*-x_t'\beta}{\sigma}\right)\right) + \sum_{y_t=0}\log\left(\Phi\left(\frac{-x_t'\beta}{\sigma}\right)\right)$$

The code might look like this:

```
#####################
#### Tobit model ####
#####################
# Definition of negative loglikelihood
neglogl <- function(param,dat) {
  y <- dat$y
  x1 <- dat$x1
  x2 <- dat$x2
  x3 <- dat$x3
  beta1 <- param[1]
  beta2 <- param[2]
  beta3 <- param[3]
  sigm   <- param[4]
  idx_cens <- which(y==0)
  idx_uncens <- which(y>0)
  uncens <- sum(log(1/sigm*dnorm((y[idx_uncens]-x1[idx_uncens]*beta1
                             -x2[idx_uncens]*beta2-x3[idx_uncens]*beta3)/sigm)))
  cens   <- sum(log(pnorm((-x1[idx_cens]*beta1-x2[idx_cens]*beta2-x3[idx_cens]*beta3)/sigm)))
  z <- uncens+cens
  return(-z)
}
# Get data
tobitbsp <- read.csv(file.choose())
# Note that if tobitbsp.csv has negative y values,
# set all negative values to zero, and proceed with the exercise
View(tobitbsp)
tobitbsp$y[tobitbsp$y<0] <- 0
# Optimization
obj <- optim(c(0.2,-0.02,0.005,0.125),neglogl,dat=tobitbsp,hessian=T)
print(obj$par)  # Point estimates
print(diag(solve(obj$hessian))) # Numerical covariance matrix
#OLS
X <- as.matrix(cbind(tobitbsp$x1,tobitbsp$x2,tobitbsp$x3))
y <- as.matrix(tobitbsp$y)
beta_ols <- solve(t(X)%*%X)%*%t(X)%*%y
```

- Compared to Logit or Probit, we do not throw away information in y.
- Note that OLS is not a consistent estimator, however Maximum Likelihood is given some regularity conditions always consistent.

12.6 Probit model

Suppose the endogenous variable y_t can only take two values,

$$y_t = \begin{cases} 1 & \text{with probability } p_t \\ 0 & \text{with probability } 1 - p_t. \end{cases}$$

We would like to model the probability p_t as a function of a vector of exogenous variables x_t. In particular, assume that the probability of $y_t = 1$ equals the value of the cdf of $N(0,1)$ at $x_t'\beta$:

$$p_t \quad = \quad \Phi\left(x_t'\beta\right) = \int_{-\infty}^{x_t'\beta} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2} dz.$$

1. Derive the log-likelihood of β. The distribution function of $N(0,1)$ is `pnorm` in R.
2. Load the dataset `mroz.csv`. The file contains the data used in the article "The Sensitivity of an Empirical Model of Married Women's Hours of Work to Economic and Statistical Assumptions" by Thomas Mroz, *Econometrica*, 55 (1987) 765-799.[6]

   Use `inlf` ("**in l**abour **f**orce") as endogenous variable and `nwifeinc`, `educ`, `exper`, `exper2`, `age`, `kidslt6`, and `kidsge6` as exogenous variables. Add a vector of constants (ones). Numerically compute the maximum likelihood probit estimate $\hat{\beta}$.
3. Interpret the parameter estimates.
4. Predict the probability of $inlf = 1$ for a woman with the following covariates

   nwifeinc = 30, educ = 14, exper = 10, age = 44, kidslt6 = 0, kidsge6 = 3.

5. Calculate the standard errors of $\hat{\beta}$. Is $\hat{\beta}_{educ}$ significantly different from zero?
6. Suppose that the true distribution of the disturbances is not $N(0,1)$ but a uniform distribution on the interval $[-1,1]$. Write a simulation program to show that the maximum likelihood estimator is no longer consistent under this kind of misspecification.

12.6 Probit model

Since the probability that $y_t = 1$ is $\Phi(x_t'\beta)$, the contribution to the loglikelihood function for observation t when $y_t = 1$ is $log(\Phi(x_t'\beta))$, that is the log-likelihood function is equal to

$$\sum_{i=1}^{n} (y_t log(\Phi(x_t'\beta)) + (1 - y_t) log(1 - \Phi(x_t'\beta)))$$

The code might look like this:

[6]The article can be downloaded from the course site. The data are available on the internet site of Jeffrey Wooldridge, Econometric Analysis of Cross Section and Panel Data, 2nd ed., 2010. A short description of the dataset can be found on the course site.

```
######################
#### Probit model ####
######################
rm(list=ls())
dev.off()
cat("\014")

mroz <- read.csv(file.choose())
inlf <- mroz$inlf
nwifeinc <- mroz$nwifeinc
educ <- mroz$educ
exper <- mroz$exper
exper2 <- exper^2
age <- mroz$age
kidslt6 <- mroz$kidslt6
kidsge6 <- mroz$kidsge6


numTimeVals <- length(nwifeinc)
zuerkl <- inlf
numVar <- 8
erkl <- matrix(0,numTimeVals,numVar)
erkl[,1] <- nwifeinc
erkl[,2] <- educ
erkl[,3] <- exper
erkl[,4] <- exper2
erkl[,5] <- age
erkl[,6] <- kidslt6
erkl[,7] <- kidsge6
erkl[,8] <- rep(1,numTimeVals)

probitLL2 <- function(betas,x,y){
  vals <- x %*% betas
  TT <- length(y)
  summanden <- y*log(pnorm(vals, mean=0, sd=1)) + (1-y)*log(1-pnorm(vals, mean=0, sd=1))
  return (-sum(summanden))
}

beta_anf <- 0

obj2 <- optim(par=rep(beta_anf,numVar),probitLL2,x=erkl, y=zuerkl, hessian=T)
#print(obj)
obj2$par
solve(obj2$hessian)

plot(zuerkl)
points(seq(1,numTimeVals),pnorm(erkl%*%obj2$par),col = 2)

pred <- obj2$par%*%c(30,14,10,10^2,44,0,3,1)
pnorm(pred)

#t value
```

```
abs(obj2$par[2]/obj2$hessian[2,2])
# -> significantly different from 0

#simulate data
#datenpunkte
N <- 250

beta1_wahr <- 2
beta0_wahr <- -1.5

testen <- function(N, norm = T){
  #Daten produzieren
  if(norm == T){x <- rnorm(N)
  p <- pnorm(x*beta1_wahr + beta0_wahr  + rnorm(N,0,10^-1))
  }
  else{x <- rnorm(N)
  p <- pnorm(x*beta1_wahr + beta0_wahr  + runif(N,-1,1))
  }
  y <- runif(N)
  y2 <- runif(N)
  y[y2<=p] = 1
  y[y2>p] = 0
  #zu x konstante hinzufügen
  x <- cbind(x,rep(1,N))

  #Parameter schaetzen
  obj2 <- optim(par=c(0,0),probitLL2, x=x, y=y, hessian=T)
  return(obj2$par)
}

sampleSizes <- seq(100,10000,100)
#Inkonsistenz veranschaulichen bei Gleichverteilung
param = NULL
for(numbers in sampleSizes){
  param = rbind(param,testen(numbers,F))
}

par(mfrow=c(2,2))
plot(sampleSizes, param[,1], xlab="Stichprobengröße", ylab="beta1", main="Gleichverteilung",
                                                                                   ylim=c(1,3))
 abline(h = beta1_wahr, col=2)

plot(sampleSizes, param[,2],, xlab="Stichprobengröße", ylab="beta0", main="Gleichverteilung",
                                                                                   ylim=c(-2,-1))
abline(h = beta0_wahr, col=2)

#Konsistenz veranschaulichen
param = NULL
for(numbers in sampleSizes){
  param = rbind(param,testen(numbers,T))
}
```

```
plot(sampleSizes, param[,1], xlab="Stichprobengröße", ylab="beta1", main="Normalverteilung",
                                                                                    ylim=c(1,3))
abline(h = beta1_wahr, col=2)

plot(sampleSizes, param[,2],, xlab="Stichprobengröße", ylab="beta0", main="Normalverteilung",
                                                                                    ylim=c(-2,-1))
abline(h = beta0_wahr, col=2)
```

- Probit models are used to model binary outcomes, such as consumption decisions, labor decisions, agricultural decisions,..., and are among the most used models in applied econometrics.
- The probit model forecasts probabilities, which are in between 0 and 1: $Pr(Y = 1|x) = F(x'\beta)$
- Interpretation of coefficients:
  - Increase in x makes outcome of 1 more (or less) likely, we therefore interpret the sign of the coefficients.
  - Marginal effects at a specific x (most commonly at mean $\bar{x}$): $\frac{\partial p}{\partial x_j} = \phi(x'\beta)\beta_j$. Note that coefficients and marginal effects have the same sign.
  - Average marginal effects are computed over all x.
- Probit and Logit have usually very similar marginal effects, which model depends on an evaluation of the data generating process. Most of the times it doesn't matter though.

12.7 Logit model

Suppose the endogenous variable y_t can only take two values,

$$y_t = \begin{cases} 1 & \text{with probability } p_t \\ 0 & \text{with probability } 1 - p_t. \end{cases}$$

We would like to model the probability p_t as a function of a vector of exogenous variables x_t. In particular, assume that the probability of $y_t = 1$ equals the value of the logistic function at $x_t'\beta$:

$$p_t = \Lambda(x_t'\beta) = \frac{\exp(x_t'\beta)}{1 + \exp(x_t'\beta)}.$$

1. Derive the log-likelihood of β. In R, the distribution function Λ of the logistic distribution is computed by `plogis`.
2. Redo the application of exercise 12.6.2. Numerically compute the maximum likelihood logit estimate $\hat{\beta}$.
3. Predict the probability of $y = 1$ for the values of the exogenous variables given in exercise 12.6.4.
4. Calculate the standard errors of $\hat{\beta}$. Is $\hat{\beta}_{educ}$ significantly different from zero (at significance level 0.05)?

12.7 Logit model

Since the probability that $y_t = 1$ is $\Lambda\left(x_t'\beta\right)$, the contribution to the loglikelihood function for observation t when $y_t = 1$ is $log(\Lambda\left(x_t'\beta\right))$, that is the log-likelihood function is equal to

$$\sum_{i=1}^{n}\left(y_t log(\Lambda\left(x_t'\beta\right)) + (1-y_t) log(1-\Lambda\left(x_t'\beta\right))\right))$$

The code might look like this:

```
#####################
#### Logit model ####
#####################
rm(list=ls())
dev.off()
cat("\014")

mroz <- read.csv(file.choose())
inlf <- mroz$inlf
nwifeinc <- mroz$nwifeinc
educ <- mroz$educ
exper <- mroz$exper
exper2 <- exper^2
age <- mroz$age
kidslt6 <- mroz$kidslt6
kidsge6 <- mroz$kidsge6


numTimeVals <- length(nwifeinc)
zuerkl <- inlf
numVar <- 8
erkl <- matrix(0,numTimeVals,numVar)
erkl[,1] <- nwifeinc
erkl[,2] <- educ
erkl[,3] <- exper
erkl[,4] <- exper2
erkl[,5] <- age
erkl[,6] <- kidslt6
erkl[,7] <- kidsge6
erkl[,8] <- rep(1,numTimeVals)

neglogitloglik <- function(betas,x,y){
  vals <- x %*% betas
  TT <- length(y)
  summanden <- y*log(plogis(vals)) + (1-y)*log(1-plogis(vals))
  return (-sum(summanden))
}

beta_anf <- 0

obj2 <- optim(par=rep(beta_anf,numVar),neglogitloglik,x=erkl, y=zuerkl, hessian=T)
print(obj2$par)
solve(obj2$hessian)
```

```
pred <- obj2$par%*%c(30,14,10,10^2,44,0,3,1)
plogis(pred)

#t value
abs(obj2$par[2]/obj2$hessian[2,2])
# -> significantly different from 0
```

- Logit models are used to model binary outcomes, such as consumption decisions, labor decisions, agricultural decisions,. . . , and are among the most used models in applied econometrics.
- The logit model forecasts probabilities, which are in between 0 and 1: $Pr(Y = 1|x) = F(x'\beta)$
- Interpretation of coefficients:
  - Increase in x makes outcome of 1 more (or less) likely, we therefore interpret the sign of the coefficients.
  - Marginal effects at a specific x (most commonly at mean $\bar{x}$): $\frac{\partial p}{\partial x_j} = \frac{e^{x'\beta}}{(1+e^{x'\beta})^2}\beta_j$. Note that coefficients and marginal effects have the same sign.
  - Average marginal effects are computed over all x.
- Probit and Logit have usually very similar marginal effects, which model depends on an evaluation of the data generating process. Most of the times it doesn't matter though.

12.8 Sample selectivity: Heckman regression

If the sample is not selected randomly standard OLS methods cease to be consistent. Consistent estimators are, however, still possible. We consider the following simple sample selection model (see Davidson and MacKinnon, 2004, p. 486),

$$\begin{bmatrix} y_t^* \\ z_t^* \end{bmatrix} = \begin{bmatrix} X_t\beta \\ W_t\gamma \end{bmatrix} + \begin{bmatrix} u_t \\ v_t \end{bmatrix}, \qquad \begin{bmatrix} u_t \\ v_t \end{bmatrix} \sim N\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma^2 & \rho\sigma \\ \rho\sigma & 1 \end{bmatrix} \right)$$

where X_t and W_t are vectors of exogenous variables, β and γ are unknown parameter vectors, σ is the standard deviation of u_t and ρ is the correlation between u_t and v_t. Both y_t^* and z_t^* are latent (unobservable). Actually observed are

$$\begin{matrix} y_t = y_t^* \\ z_t = 1 \end{matrix} \qquad \text{if } z_t^* > 0$$

and

$$\begin{matrix} y_t \text{ unobserved} \\ z_t = 0 \end{matrix} \qquad \text{if } z_t^* \leq 0.$$

1. Derive the log-likelihood of $(\beta, \gamma, \rho, \sigma)'$. Hint: If y_t is not observed, its contribution to the log-likelihood is $\ln P(z_t = 0)$, else it is $\ln P(z_t = 1) f(y_t^*|z_t = 1)$.
2. Load the dataset `womanwk.dta`.[7] The dataset consists of 2,000 women, 1,343 of whom work. Create a new variable `inlf` *(In-Labor-Force)*, which is equal to 1, if the wage is observed, and equal to 0, if the wage is unobserved.

[7]This is a dataset used in the Stata Manual, Section Heckman selection model.

3. Assume that the hourly wage is a function of education and age, whereas the likelihood of working (the likelihood of the wage being observed) is a function of marital status, the number of children at home, and (implicitly) the wage (via the inclusion of age and education). Explain why this is a sample selection problem and the wage would be upward biased if one performed ordinary least squares.

4. Use `wage` as endogenous variable y_t and `inlf` as selection variable z_t. Define the vectors

$$W_t = \begin{bmatrix} \text{constant} \\ \text{married} \\ \text{children} \\ \text{age} \\ \text{educ} \end{bmatrix}, \quad X_t = \begin{bmatrix} \text{constant} \\ \text{educ} \\ \text{age} \end{bmatrix}$$

and numerically compute the maximum likelihood estimates $\hat{\beta}, \hat{\gamma}, \hat{\rho}$ and $\hat{\sigma}$.

5. Calculate the standard errors for all parameters.

12.8 Heckman regression

1. Each observation contributes a factor to the likelihood function:

$$I(z_t = 0)Pr(z_t = 0) + I(z_t = 1)Pr(z_t = 1)f(y_t^*|z_t = 1)$$

where $I(z_t = 0)$ is the indicator function and $f(y_t^*|z_t = 1)$ denotes the density of y_t^* conditional on $z_t = 1$. The loglikelihood function is

$$\sum_{z_t=0} logPr(z_t = 0) + \sum_{z_t=1} logPr(z_t = 1)f(y_t^*|z_t = 1) = \sum_{z_t=0} logPr(z_t = 0) + \sum_{z_t=1} log\left(Pr(z_t = 1|y_t^*)f(y_t^*)\right)$$

where $f(y_t^*)$ is the normal density with mean $X_t\beta$ and variance σ^2. The first term, where $z_t = 0$, is the same as in a probit model.

Lets calculate $Pr(z_t = 1|y_t^*)$. Since u_t and v_t are bivariate normal, we can write $v_t = \rho u_t/\sigma + \varepsilon_t$, where ε_t is a normally distributed random variable with mean 0 and variance $(1 - \rho^2)$. Thus

$$z_t^* = W_t\gamma + \rho(y_t^* - X_t\beta)/\sigma + \varepsilon_t$$

Because $y_t = y_t^*$ for $z_t = 1$, it follows that

$$Pr(z_t = 1|y_t^*) = \Phi\left(\frac{W_t\gamma + \rho(y_t - X_t\beta)/\sigma}{(1-\rho^2)^{1/2}}\right)$$

Combining everything the loglikelihood is given by

$$\sum_{z_t=0} log\Phi(-W_t\gamma) + \sum_{z_t=1} log\left(\frac{1}{\sigma}\phi\left((y_t - X_t\beta)/\sigma\right)\right) + \sum_{z_t=1} log\Phi\left(\frac{W_t\gamma + \rho(y_t - X_t\beta)/\sigma}{(1-\rho^2)^{1/2}}\right)$$

2. Make variable in-labor-force if wage is observed:

```
library(foreign); womanwk <- read.dta(file.choose());
womanwk$inlf <- rep(1,dim(womanwk)[1]); womanwk$inlf[is.na(womanwk$wage)] <- 0
```

3. In the underlying model women choose whether to work and if they do, we observe their wages. If women made this decision randomly, we could ignore that not all wages are observed and use ordinary regression to fit a wage model. Such an assumption of random participation, however, is unlikely to be true; women who would have low wages may be unlikely to choose to work, and thus the sample of observed wages is biased upward. In the jargon of economics, women choose not to work when their personal reservation wage is greater than the wage offered by employers. Thus women who choose not to work might have even higher offer wages than those who do work—they may have high offer wages, but they have even higher reservation wages. We could tell a story that competency is related to wages, but competency is rewarded more at home than in the labor force In any case, in this problem—which is the paradigm for most such problems—a solution can be found if there are some variables that strongly affect the chances for observation (the reservation wage) but not the outcome under study (the offer wage). Such a variable might be the number of children in the home. (Theoretically, we do not need such identifying variables, but without them, we depend on functional form to identify the model. It would be difficult for anyone to take such results seriously because the functional form assumptions have no firm basis in theory.)

4. The code might look like this

```
#################################################
#### Sample selectivity: Heckman regression ####
#################################################
rm(list=ls())
dev.off()
cat("\014")
library(foreign)
womanwk <- read.dta(file.choose())
# make variable in-labor-force if wage is observed
womanwk$inlf <- rep(1,dim(womanwk)[1])
womanwk$inlf[is.na(womanwk$wage)] <- 0

negloglik <- function(param,dat){
  y <- as.matrix(dat$wage)
  X <- as.matrix(cbind(1,dat$educ, dat$age))
  z <- as.matrix(dat$inlf)
  W <- as.matrix(cbind(1, dat$married, dat$children, dat$age, dat$educ))

  bet <- param[1:dim(X)[2]]
  gam <- param[(dim(X)[2]+1):(dim(X)[2]+dim(W)[2])]
  rho_aux <- param[dim(X)[2]+dim(W)[2]+1] #rho_aux is unconstrained
  #rho <- -1 + 0.5*(1+1)*(1+tanh(rho_aux)) #this constraints rho to be between -1 and 1
  rho <- (exp(2*rho_aux)-1)/(1+exp(2*rho_aux)) #this constraints rho to be between -1 and 1
  sigm_aux <- param[dim(X)[2]+dim(W)[2]+2]#sigm_aux is unconstrained
  sigm <- exp(sigm_aux) #sigm is positive
  X_bet <- X%*%as.matrix(bet)
  W_gam <- W%*%as.matrix(gam)
  sum1 <- sum(log(pnorm(-W_gam[z==0])))
  sum2 <- sum(log(1/sigm*dnorm((y[z==1]-X_bet[z==1])/sigm)))
  sum3 <- sum(log(pnorm( (W_gam[z==1]+rho*(y[z==1]-X_bet[z==1])/sigm)/(sqrt(1-rho^2)) )))
  return (-(sum1+sum2+sum3))
}
```

```
#Finding good starting values is hard, an alternative method is Heckman’s two-step method
#Step1: Probit regression of selection equation
y <- as.matrix(womanwk$wage)
X <- as.matrix(cbind(1,womanwk$educ, womanwk$age))
z <- as.matrix(womanwk$inlf)
W <- as.matrix(cbind(1, womanwk$married, womanwk$children, womanwk$age, womanwk$educ))
step1 <- glm(z ~ -1 + W, family=binomial(link="probit"))
gam_anf <- coef(step1)
#Step 2: OLS regression with inverse Mills ratio as additional regressor
aux <- dnorm(W%*%as.matrix(gam_anf))/pnorm(W%*%as.matrix(gam_anf))
step2 <- lm(y ~ -1 + X + aux)
bet_anf <- coef(step2)[1:dim(X)[2]]
rho_anf <- c(0.7)
sigm_anf <- coef(step2)[dim(X)[2]+1]/rho_anf
par_anf <- c(bet_anf,gam_anf,rho_anf,sigm_anf); names(par_anf) <- NULL;

#Maximum Likelihood with constraints: rho in (-1;1) and sigm>0
#par_anf[9] <- tanh( 2*(par_anf[9]+1)/(1+1)-1) #rescale rho
par_anf[9] <- 1/2*log((1+par_anf[9])/(1-par_anf[9])) #rescale rho
par_anf[10] <- log(par_anf[10]) #rescale sigm
obj <- optim(par=par_anf,negloglik,dat=womanwk, hessian=T)
obj$value

#Show parameters, rescale rho and sigm
estimates <- obj$par
#estimates[9] <- -1 + 0.5*(1+1)*(1+tanh(estimates[9]))
estimates[9] <- (exp(2*estimates[9])-1)/(1+exp(2*estimates[9]))
estimates[10] <- exp(estimates[10])
print(estimates)
diag(solve(obj$hessian))
```

12.9 Count data

The standard multiple linear regression model is not working properly if the endogenous variable y_i takes on only small integer values. In this case one should use "count data" regression methods. We consider a fictional application of the simplest count data regression model – the Poisson regression model. Let y_i denote the number of goals scored by soccer player i (e.g. during a championship). Assume that y_i has a Poisson distribution with probability function

$$P\left(y_i = k\right) = \frac{e^{-\mu_i}\mu_i^k}{k!}.$$

The parameter μ_i of the Poisson distribution depends on exogenous variables such that,

$$\mu_i = \exp\left(X_i'\beta\right).$$

The vector of exogenous variables X_i includes a constant of unity, position (1=striker, 0=else), age, age^2, training time (in hours per week), fixed salary, and goal bonus (both in 1000 Euro).

1. Load the artificial dataset `players.csv`. It contains information about 300 players.
2. Write an R program to estimate the vector of coefficients β by maximum likelihood.
3. Compute $\hat{\beta}$ and its standard errors.
4. What is the probability that a striker aged 25 scores more than 3 goals if he is training 15 hours per week, has a fixed salary of 700,000 Euro and receives no bonuses.

12.9 Count data

The loglikelihood function is

$$\sum_{i=1}^{n}\left(-exp(x_i\beta) + y_i X_i\beta - log(y_i!)\right)$$

The code might look like this:

```
####################
#### Count data ####
####################
rm(list=ls()); dev.off(); cat("\014");

players <- read.csv(choose.files())

## Negative Log-likelihood
neg_ll <- function(bet,dat){
  position <- dat$position
  age <- dat$age
  training <- dat$training
  salary <- dat$salary
  bonus <- dat$bonus
  y <- dat$goals
  X <- cbind(1,position,age,age^2,training,salary,bonus)
  ll <- sum(-exp(X%*%bet)+y*X%*%bet-log(factorial(y)))
  return(-ll)
}
#Log-likelihood maximieren
out <- optim(par=c(-1,0.01,0.01,-0.01,0.01,0.01,0.01), fn=neg_ll,dat=players,hessian=T)
par <- out$par

## Standardabweichung der einzelnen Parameter
sdbeta <- round(sqrt(diag(solve(out$hessian))),8)
sdbeta

#Wahrscheinlichkeit, über drei Tore
mu <- (1*par[1]+1*par[2]+25*par[3]+25^2 * par[4] + 15*par[5]+ 700 *
         par[6] + 0)

1-ppois(3,mu)
#0.001064403
# Sehr unwahrscheinlich unter diesen Gegebenheiten über drei Tore zu schießen.
```

12.10 Stochastic frontier analysis

See Greene, 2008, section 16.9.5a. Consider the Cobb-Douglas production function

$$y = Ax_1^{\alpha}x_2^{\beta}.$$

By definition, the production function returns the maximal output for given inputs, and actual production cannot be larger than y. Due to inefficiencies, actual production could be modeled (in logs) as

$$\ln y = \ln A + \alpha \ln x_1 + \beta \ln x_2 - u$$

where u is a non-negative random variable. Since other disturbances (e.g. measurement errors) can enter the production function, it is more common to add another, symmetrically distributed, disturbance term,

$$\ln y = \ln A + \alpha \ln x_1 + \beta \ln x_2 - u + v.$$

Assume that $u \sim Exp(\lambda)$ and $v \sim N(0, \sigma^2)$ are independent.

1. Show that the density function of $\varepsilon = v - u$ is

$$f_{\varepsilon}(x) = \lambda \exp\left(\lambda x + \frac{1}{2}\lambda^2\sigma^2\right)\Phi\left(\frac{-x}{\sigma} - \lambda\sigma\right)$$

   where Φ is the cdf of $N(0,1)$. Hint: $f_{v-u}(x) = \int_0^{\infty} f_v(u+x) f_u(u)\, du$.

2. Write an R program to estimate the parameters $A, \alpha, \beta, \lambda$ and σ by maximum likelihood.

3. Load the dataset `sfa.csv`. This dataset is an abbreviated version of table F7.2 of Greene, 2008. The original data appeared in Zellner and Revankar, "Generalized Production Functions", *Review of Economic Studies*, 36 (1969) 241-250. Reported is the value added in the transportation equipment manufacturing industries of 25 US states and capital and Labour inputs. Compute the ML estimates and their standard errors.

4. Tabulate the estimated inefficiencies for the 25 states.

12.10 Stochastic frontier analysis

1. The density of the merged error term ε is

$$\begin{aligned}
f_\varepsilon(x) &= f_{v-u}(x) \\
&= \int_0^\infty f_v(u+x)\cdot f_u(u)\,du \\
&= \int_0^\infty \phi\left(\frac{u+x}{\sigma}\right)\cdot \lambda e^{-\lambda u}du \\
&= \int_0^\infty \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{u+x}{\sigma}\right)^2}\lambda e^{-\lambda u}du \\
&= \frac{\lambda}{\sqrt{2\pi}}\int_0^\infty \exp\left(-\frac{1}{2}\left(\frac{u+x}{\sigma}\right)^2-\lambda u\right)du. \qquad (1)
\end{aligned}$$

The integral in (1) can be manipulated in a way similar to exercise 6.1.2,

$$\begin{aligned}
&\int_0^\infty \exp\left(-\frac{1}{2}\left(\frac{u+x}{\sigma}\right)^2-\lambda u\right)du \\
=&\int_0^\infty \exp\left(-\frac{u^2+2u\left(x+\lambda\sigma^2\right)+x^2}{2\sigma^2}\right)du \\
=&\int_0^\infty \exp\left(-\frac{u^2+2u\left(x+\lambda\sigma^2\right)+\left(x+\lambda\sigma^2\right)^2+x^2-\left(x+\lambda\sigma^2\right)^2}{2\sigma^2}\right)du \\
=&\int_0^\infty \exp\left(-\frac{\left(u+\left(x+\lambda\sigma^2\right)\right)^2+x^2-\left(x+\lambda\sigma^2\right)^2}{2\sigma^2}\right)du \\
=&\exp\left(-\frac{x^2-\left(x+\lambda\sigma^2\right)^2}{2\sigma^2}\right)\int_0^\infty \exp\left(-\frac{1}{2}\left(\frac{u-\left(-x-\lambda\sigma^2\right)}{\sigma}\right)^2\right)du.
\end{aligned}$$

Substituting the integral in (1) we arrive at

$$\begin{aligned}
f_\varepsilon(x) &= \lambda\exp\left(-\frac{x^2-\left(x+\lambda\sigma^2\right)^2}{2\sigma^2}\right) \\
&\quad\times\int_0^\infty \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left(\frac{u-\left(-x-\lambda\sigma^2\right)}{\sigma}\right)^2\right)du.
\end{aligned}$$

The first factor can be simplified to

$$\lambda\exp\left(-\frac{x^2-\left(x+\lambda\sigma^2\right)^2}{2\sigma^2}\right)=\lambda\exp\left(\lambda x+\frac{\sigma^2\lambda^2}{2}\right),$$

and the integrand in the second factor is simply the density of a normal distribution with mean $\left(-x-\lambda\sigma^2\right)$ and variance σ^2. Thus the value of the integral can be derived from the cdf of $N(0,1)$ as follows,

$$\int_0^\infty \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left(\frac{u-\left(-x-\lambda\sigma^2\right)}{\sigma}\right)^2\right)du=\Phi\left(\frac{-x-\lambda\sigma^2}{\sigma}\right).$$

In sum, the density of ε is

$$f_\varepsilon(x)=\lambda\exp\left(\lambda x+\frac{\sigma^2\lambda^2}{2}\right)\Phi\left(\frac{-x-\lambda\sigma^2}{\sigma}\right)$$

and its logarithm is

$$\ln f_\varepsilon(x) = \ln\lambda + \lambda x + \frac{\sigma^2\lambda^2}{2} + \ln\Phi\left(\frac{-x-\lambda\sigma^2}{\sigma}\right).$$

2. The code might look like this:

```
######################################
#### Stochastic Frontier Analysis ####
######################################

##Log-Likelihood Funktion, die maximiert werden soll
neg_log_likeli <- function(theta, daten){
  A <- theta[1]
  alpha <- theta[2]
  beta <- theta[3]
  lambda <- theta[4]
  sigma <- theta[5]

  x1 <- daten[,3]
  x2 <- daten[,4]
  y <- daten[,2]

  logA <- log(A)
  logy <- log(y)
  logx1 <- log(x1)
  logx2 <- log(x2)

  eps <- logy - logA - alpha*logx1 - beta*logx2

  log_likeli <- sum(log(lambda*exp(lambda*eps+lambda^2/2*sigma^2)*pnorm(
                                -eps/sigma-lambda*sigma,mean=0,sd=1)))
  return(-log_likeli)
}

daten <- read.csv(file.choose())
head(daten)

##Schätzung der Parameter
opt <- optim(c(7, 0.3, 0.7, 10, 3), neg_log_likeli, daten=daten, hessian=T)
opt

A <- opt$par[1]
alpha <- opt$par[2]
beta <- opt$par[3]
lambda <- opt$par[4]
sigma <- opt$par[5]

##Standardfehler: Problem mit negativen Werten, hier kleine Stichprobe das Problem
diag(solve(opt$hessian))

##Geschätzte Ineffizienzen
epsilon <- log(daten$ValueAdd)-log(A)-alpha*log(daten$Capital)-beta*log(daten$Labor)
```

```
plot(epsilon)
hist(epsilon, freq=F)
curve(lambda*exp(lambda*x+lambda^2/2*sigma^2)*pnorm(-x/sigma-lambda*sigma, mean=0, sd=1),
                                                        from=-1, to=1, add=T, col="red")
```

- SFA are also called *composed error models.* In a production context efficiency loss is modelled as $-u$, in a cost context as $+u$.
- SFA models are also used to measure efficiency of the banking system, level of competitiveness of a market, quality of inputs and outputs, regulation, management evaluation,...
- Basic idea is that the ratio of observed output to maximum possible output is less or equal to 1.
- How to estimate inefficiency?
  - We can compute residuals $\hat{\varepsilon_t} = \widehat{u_t + v_t}$. Since $E(v_t) = 0$ we can conclude that if the residual is high, then so is the inefficiency.
  - Jondrow, Lovell, Materov, Schmidt (JLMS) approach: Look at the mean (or mode) of the conditional distribution.

12.11 ARCH models

Models with autoregressive conditional heteroscedasticity have many applications in empirical finance. We only consider the simple case of an $ARCH(1)$-process. Let X_t denote the stock return in period t. Suppose

$$X_t = \sigma_t \varepsilon_t$$

with $\varepsilon_t \sim N(0,1)$ and

$$\sigma_t^2 = \omega + \alpha X_{t-1}^2.$$

1. Factorize the joint density function of $X_1, \ldots, X_T$.
2. Ignore the marginal density of X_1 and write an R function to compute the log-likelihood of $X_2, \ldots, X_T$.
3. Load the (artificial) dataset `arch1bsp.csv` and estimate ω and α by maximizing the log-likelihood numerically.
4. Compute the covariance matrix of $\hat{\omega}, \hat{\alpha}$.

12.11 ARCH models

For ARCH models we have:

$$E(X_t|X_{t-1}) = E(\varepsilon_t \sigma_t|X_{t-1}) = E(\varepsilon_t \sqrt{\omega + \alpha X_{t-1}^2}|X_{t-1}) = \sqrt{\omega + \alpha X_{t-1}^2} E(\varepsilon_t|X_{t-1}) = 0$$

and

$$Var(X_t|X_{t-1}) = Var(\varepsilon_t \sigma_t|X_{t-1}) = Var(\varepsilon_t \sqrt{\omega + \alpha X_{t-1}^2}|X_{t-1}) = (\omega + \alpha X_{t-1}^2) \underbrace{E(\varepsilon_t^2|X_{t-1})}_{=1} = \omega + \alpha X_{t-1}^2$$

since ε_t is independent of X_{t-1}. Given that $\varepsilon_t \sim N(0,1)$, the conditional density of X_t given X_{t-1} is

$$X_t|X_{t-1} \sim N(0, \omega + \alpha X_{t-1}^2)$$

In general we can factorize a joint likelihood function:

$$f_{X_1,\ldots,X_T}(x_1,\ldots,x_T) = \prod_{t=1}^{T} f_{X_t|X_{t-1},\ldots,X_1}(x_t|X_{t-1,\ldots,x_1})$$

In our case:

$$f_{X_1,\ldots,X_T}(x_1,\ldots,x_T) = f_{X_1}(x_1)\prod_{t=2}^{T} \frac{1}{\sqrt{2\pi(\omega+\alpha x_{t-1}^2)}} \exp\left(-\frac{1}{2}\left(\frac{x_t}{\omega+\alpha x_{t-1}^2}\right)^2\right)$$

We can ignore $f_{X_1}(x_1)$ for large sample sizes, since it contribution is relatively small the larger the sample size. The log-likelihood is then given by

$$-\frac{T-2}{2}log(2\pi) - \frac{1}{2}\sum_{t=2}^{T} log(\omega+\alpha x_{t-1}^2) - \frac{1}{2}\sum_{t=2}^{T}\left(\frac{x_t}{\omega+\alpha x_{t-1}^2}\right)^2$$

The code might look like this

```
#####################
#### ARCH Models ####
#####################

arch1_ll <- function(theta, tseries) {
  omega <- theta[1]
  alpha <- theta[2]
  Te <- length(tseries)
  return((Te - 1) / 2 * log(2 * pi)
         + 0.5 * sum(log(omega + alpha * tseries[1:(Te - 1)]^2))
         + 0.5 * sum((tseries[2:Te]^2 /
                      (omega + alpha * tseries[1:(Te - 1)]^2))))
}

arch1bsp <- read.csv(file.choose(), header = TRUE)

estimate <- optim(par = c(0.2, 0.9), fn = arch1_ll, tseries = arch1bsp$x,
                  hessian = TRUE)
estimate$par
solve(estimate$hessian)

gitter <- seq(0.05, 0.95, by = 0.01)
value_omega <- rep(NA, length(gitter))
value_alpha <- rep(NA, length(gitter))
for (i in 1:length(gitter)) {
  value_omega[i] <- -arch1_ll(c(gitter[i], estimate$par[2]),
                              tseries = arch1bsp$x)
  value_alpha[i] <- -arch1_ll(c(estimate$par[1], gitter[i]),
                              tseries = arch1bsp$x)
}
plot(gitter, value_omega, type = "l", xlab = expression(omega),
     ylab = "",
     main = expression(paste("log-Likelihood in Abhängigkeit von ", omega,
                             ", ", alpha, " = 0.9376")), lwd = 2,
     cex.axis = 1.5, cex.lab = 2, cex.main = 2)
mtext(text = expression(paste("logL(", omega, ")")), side = 2, cex = 2,
      line = 2.2)
abline(h = -estimate$value, col = "red")
abline(v = estimate$par[1], col = "red")

plot(gitter, value_alpha, type = "l", xlab = expression(alpha),
     ylab = "",
     main = expression(paste("log-Likelihood in Abhängigkeit von ", alpha,
                             ", ", omega, " = 0.2056")), lwd = 2,
     cex.axis = 1.5, cex.lab = 2, cex.main = 2)
mtext(text = expression(paste("logL(", alpha, ")")), side = 2, cex = 2,
      line = 2.2)
abline(h = -estimate$value, col = "red")
abline(v = estimate$par[2], col = "red")

# Unbedingte Varianz
estimate$par[1] / (1 - estimate$par[2])
```

- Volatility is not observable. To measure volatility by the empirical standard deviation is only valid if volatility is relatively stable, but usually we have time-variation in volatility.
- Idea of ARCH is that volatility is dependent on past observations (conditional distribution is time-varying), however, the unconditional distribution is still stationary.

12.12 Duration models

There is a huge number of duration models, but we only consider a particularly easy case, see Davidson and MacKinnon, 2004, pp. 490ff. Suppose that how long a state endures is measured by a non-negative random variable T with density function $f(t)$ and cdf $F(t)$. Define the survival function $S(t) = 1 - F(t)$ and the hazard function

$$h(t) = \frac{f(t)}{S(t)}.$$

The survivor function measures the probability that a state which stated at time $t = 0$ is sill going on at time t. The hazard function can be interpreted as the probability that the state ends in the next instant, given it has not ended yet.

1. Let T have the cdf $F(t; \theta, \alpha) = 1 - \exp\left(-(\theta t)^{\alpha}\right)$ with parameters θ and α (note that this is the Weibull distribution). Derive the density $f(t)$, the survival function $S(t)$ and the hazard function $h(t)$. Interpret the parameter α.
2. Assume that n completed (independent) durations t_1,..,t_n have been observed. Derive the log-likelihood function under the assumption that parameter θ depends on some exogenous vector X_i in the following way,
$$\theta_i = \exp(X_i'\beta).$$

Use $f(t) = h(t)S(t)$ to split the log-likelihood function into two sums and rewrite the log-likelihood accordingly. Note that, if some spells are incomplete (i.e. they have not ended yet) the log-likelihood can be adapted easily by simply dropping their contributions to the hazard part of the log-likelihood.

3. Load the artificial dataset `spells.csv`. The first variable is the duration, the other three variables are exogenous (one is the intercept). Spells with duration 0.5 are incomplete. Estimate the parameters $\beta_1, \beta_2, \beta_3$, and α and their standard errors by maximum likelihood.

12.12 Duration models

1. Obtaining the survivor function is very easy:

$$S(t) \equiv 1 - F(t) = exp(-(\theta t)^\alpha)$$

For the pdf

$$f(t) = \frac{\partial F(t)}{\partial t} = \alpha\theta^\alpha t^{\alpha-1} exp(-(\theta t)^\alpha)$$

The hazard function is then given by

$$h(t) \equiv \frac{f(t)}{S(t)} = \frac{\alpha\theta^\alpha t^{\alpha-1} exp(-(\theta t)^\alpha)}{exp(-(\theta t)^\alpha)} = \alpha\theta^\alpha t^{\alpha-1}$$

When $\alpha = 1$, the Weibull distribution collapses to the exponential and the hazard is just a constant (duration independent). For $\alpha < 1$, the hazard is decreasing over time (negative duration dependence) and for $\alpha > 1$ it is increasing (positive duration dependence).

2. Taking the log of $f(t_i)$ yields

$$log(f(t_i)) = log\,(h(t_i)) + log\,(S(t_i)) = log(\alpha_i) + \alpha_i log(\theta_i) + (\alpha_i - 1)log(t_i) - (\theta_i t_i)_i^\alpha$$

Since $\theta_i = exp(X_i\beta)$ and $\alpha_i = \alpha$ for all i, we have

$$log(f(t_i, X_i, \alpha, \beta)) = log(\alpha) + \alpha X_i\beta + (\alpha - 1)log(t_i) - t_i^\alpha \cdot exp(\alpha X_i\beta)$$

Summing over all n independent observations gives the log-likelihood

$$log(f(t, X, \alpha, \beta)) = nlog(\alpha) + \alpha \sum_{i=1}^{n} X_i\beta + (\alpha - 1)\sum_{i=1}^{n} log(t_i) - \sum_{i=1}^{n} t_i^\alpha \cdot exp(\alpha X_i\beta)$$

Since data sets contain observations for which t_i is not actually observed (e.g. sample of people who entered unemployment at various points in time, then it is extremely likely that some peaple in the sample were still unemployed when data collection ended). We can deal with this censoring, i.e. the observed t_i is the duration of an incomplete spell, it is the logarithm of the probability of censoring, which is the probability that the duration exceed t_i, that is, the log of the survivor function. Denote U as the set of n_u uncensored observations, the loglikelihood function for the entire sample is then

$$log(f(t, X, \alpha, \beta)) = n_u log(\alpha) + \alpha \sum_{i \in U} X_i\beta + (\alpha - 1)\sum_{i \in U} log(t_i) - \sum_{i=1}^{n} t_i^\alpha \cdot exp(\alpha X_i\beta)$$

Uncensored observations contribute to both terms, while censored observations contribute only to the Survivor function.

3. The code might look like this

```
########################
#### Duration models ####
########################

## Load data
spells <- read.csv(file.choose())
View(spells)

##Negative log-likelihood
neg_log_likeli <- function(param, dat) {
  beta <- as.matrix(param[1:3])
  alpha <- param[4]

  t <- dat$duration
  const <- dat$const
  x1 <- dat$X1
  x2 <- dat$X2
  X <- as.matrix(cbind(const,x1,x2))
  idx_uncens <- which(t<0.5)
  nu <- length(idx_uncens)
  log_likeli <- nu*log(alpha) + alpha*sum(X[idx_uncens,]%*%beta)
            + (alpha-1)*sum(log(t[idx_uncens])) - sum((t^alpha)*exp(alpha*X%*%beta))
  return(-log_likeli)
}

##Schätzung der Parameter
par_anf <- c(1,0.7,0.3, 1.3)
neg_log_likeli(par_anf,spells)
opt <- optim(par=par_anf, neg_log_likeli, dat=spells, hessian=T)
print(opt$par)

##Standardfehler
diag(solve(opt$hessian))
```

- The terminology like survival or hazard is due to evolutionary concepts, however, duration models in economics are quite often found in the labor market literature. Also: strike duration, state of being single (until marriage), etc.

12.13 Ultra-high-frequency data

A model of the duration between individual transactions on stock exchanges has been suggested by Engle and Russell, "Autoregressive Conditional Duration: A New Model for Irregularly Spaced Transaction Data", *Econometrica*, 66 (1998) 1127-1162. The article can be downloaded (password protected pdf) from the internet site of this course. Let X_i denote the duration between transaction $i-1$ and transaction i. The model assumes that

$$X_i \sim \psi_i \varepsilon_i$$

where ε_i is i.i.d. standard exponentially distributed with density function e^{-x}. The scale parameter depends on previous observations in a way similar to ARCH models

$$\psi_i = \omega + \sum_{j=1}^{p} \alpha_j X_{i-j}.$$

For simplicity, we set $p = 1$.

1. Factorize the joint density function of $X_1, \ldots, X_T$.
2. Ignore the marginal density of X_1 and write an R function to compute the log-likelihood of $X_2, \ldots, X_T$.
3. Load the (artificial) dataset `acd1bsp.csv` and estimate ω and α_1 by maximizing the log-likelihood numerically.
4. Compute the covariance matrix of $\hat{\omega}, \hat{\alpha}_1$.

12.13 Ultra-high-frequency data

12.14 Spatial dependence

Observations may not only be dependent over time, but also over space. For instance, real estate prices can be influenced by prices in neighboring regions. A simple case of spatial dependence is the spatial autoregressive model,

$$y = \rho W y + \alpha + \delta z + u \tag{2}$$

where y is an $(n \times 1)$-vector of endogenous variables, W is a symmetric $(n \times n)$-weight matrix, Z is an $(n \times 1)$-vector of a (single) exogenous variable, $u \sim N\left(0, \sigma^2 I\right)$ is an $(n \times 1)$-vector of disturbances. The unknown parameters of the model are α, δ, ρ, and σ, the spatial autocorrelation is driven by the parameter ρ. The weight matrix W can be specified in a number of ways. Often, element W_{ij} simply indicates if regions i and j are direct neighbors, $W_{ij} > 0$, or not, $W_{ij} = 0$. If m_i is the number of direct neighbors of i, then $W_{ij} = 1/m_i$, such that $\sum_j W_{ij} = 1$. Since the model (2) cannot be estimated consistently by OLS, we perform a maximum likelihood estimation of the parameters.

1. Solve (2) for y and derive its multivariate normal distribution (ie. its expectation vector and its covariance matrix).
2. Use the multivariate distribution of y to show that the log-likelihood function is
$$-\frac{n}{2} \ln\left(2\pi\sigma^2\right) + \ln\left(\det\left(I_n - \rho W\right)\right) - \frac{\left(y - \rho W y - \alpha - \delta z\right)' \left(y - \rho W y - \alpha - \delta z\right)}{2\sigma^2}$$
Hints: If $X \sim N\left(\mu, \Sigma\right)$ is multivariate normal with K dimensions, then its density at $x = (x_1, \ldots, x_K)'$ is $f_X\left(x\right) = \left(2\pi\right)^{-K/2}\left[\det(\mathbf{\Sigma})\right]^{-1/2} \cdot \exp\left(-\frac{1}{2}\left(\mathbf{x}-\mu\right)' \mathbf{\Sigma}^{-1}\left(\mathbf{x}-\mu\right)\right)$. The following results for determinants (of suitable matrices) may also help: $\det\left(AB\right) = \det A \det B$, $\det(aA) = a^n \det A$, where A is $n \times n$ and a is a real scalar, and $\det(A^{-1}) = \det(A)^{-1}$.
3. Load the datasets `spatialdata.csv` and `neighbourhood.csv`. The first one contains data on house prices (column 1) and disposable household income (column 2) in the 413 German “Kreise”; the second one is the 413×413 neighborhood matrix.

4. Calculate the normalized neighborhood matrix W such that each row of W sums to unity.
5. Write an R program to compute the log-likelihood function and estimate the parameters α, δ, ρ and σ of the model.
6. Compute the standard errors for $\hat{\alpha}$, $\hat{\delta}$, $\hat{\rho}$, and $\hat{\sigma}$. Test if there is significant spatial autocorrelation.

12.14 Spatial dependence

1. The model can be written as

$$\begin{aligned}(I-\rho W)\,y &= \alpha+\delta z+u\\ y &= (I-\rho W)^{-1}(\alpha+\delta z)+(I-\rho W)^{-1}u.\end{aligned}$$

To keep the notation short, define $D=(I-\rho W)^{-1}$. Since u is multivariate normal, $u\sim N(0,\sigma^2 I_n)$, we find that

$$y\sim N(\mu,\Sigma)$$

with

$$\begin{aligned}\mu &= D(\alpha+\delta z)\\ \Sigma &= \sigma^2 DD'\end{aligned}$$

2. The joint density of y, i.e. the likelihood function, is

$$\begin{aligned}L(\alpha,\delta,\rho,\sigma) &= (2\pi)^{-n/2}\left[\det(\mathbf{\Sigma})\right]^{-1/2}\cdot\exp\left(-\frac{1}{2}(\mathbf{x}-\mu)'\mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)\right)\\ &= (2\pi)^{-n/2}\left[\det(\sigma^2DD')\right]^{-1/2}\\ &\quad\times\exp\left(-\frac{1}{2}(y-D(\alpha+\delta z))'\left(\sigma^2DD'\right)^{-1}(y-D(\alpha+\delta z))\right).\end{aligned}$$

Consider the terms in turn,

$$\begin{aligned}\left[\det(\sigma^2DD')\right]^{-1/2} &= \left[\sigma^{2n}\det(DD')\right]^{-1/2}\\ &= \left(\sigma^2\right)^{-n/2}\det(D)^{-1}\\ &= \left(\sigma^2\right)^{-n/2}\det(I-\rho W)\end{aligned}$$

since $\det(D)=\det(D')$ and $\det(D)^{-1}=\det(I-\rho W)$. Next, the term inside the exponential function is

$$\begin{aligned}&-\frac{1}{2}(y-D(\alpha+\delta z))'\left(\sigma^2DD'\right)^{-1}(y-D(\alpha+\delta z))\\ =\ &-\frac{1}{2\sigma^2}(y-D(\alpha+\delta z))'D^{-1\prime}D^{-1}(y-D(\alpha+\delta z))\\ =\ &-\frac{1}{2\sigma^2}\left[D^{-1}(y-D(\alpha+\delta z))\right]'\left[D^{-1}(y-D(\alpha+\delta z))\right]\\ =\ &-\frac{1}{2\sigma^2}\left[D^{-1}y-\alpha-\delta z\right]'\left[D^{-1}y-\alpha-\delta z\right]\\ =\ &-\frac{(y-\rho Wy-\alpha-\delta z)'(y-\rho Wy-\alpha-\delta z)}{2\sigma^2}.\end{aligned}$$

Hence, the log-likelihood function is

$$\begin{aligned}\ln L(\alpha,\delta,\rho,\sigma) &= -\frac{n}{2}\ln\left(2\pi\sigma^2\right)+\ln\left(\det\left(I-\rho W\right)\right)\\ &\quad -\frac{\left(y-\rho Wy-\alpha-\delta z\right)'\left(y-\rho Wy-\alpha-\delta z\right)}{2\sigma^2}.\end{aligned}$$

3. (...)

The code might look like this:

```
############################
#### Spatial dependence ####
############################
# 3)
# Einlesen der Datensätze
spatial = read.csv(file.choose(), header = T, sep = "")
neighbour = read.csv(file.choose(), header = T, sep = "")

# 4)
# Normalisieren der Nachbarschaftsmatrix
W.norm = neighbour / rowSums(neighbour)

# 5)
# Log-Likelihood-Funktion bestimmen und optimieren
ll_Spatial = function(y, z, W, theta){
  W = as.matrix(W)
  n = length(y)
  rho = theta[1]
  alpha = theta[2]
  delta = theta[3]
  sigma = theta[4] # hier sigma^2
  In = diag(n)
  A = t(y - rho*W%*%y - alpha - delta*z)%*%(y - rho*W%*%y - alpha - delta*z)
  loglik = -n/2*log(2*pi*sigma) + log(det(In - rho*W)) - A/(2*sigma)
  return(loglik)
}

optModel = optim(c(0.5 ,1 ,2 ,3) , ll_Spatial , y = spatial$HPrice07 ,
                 z =spatial$HHIncome07 , W = W.norm, hessian = T,
                 method = "L-BFGS-B",lower = c(-1,-Inf ,-Inf ,0),
                 upper = c(1, Inf , Inf , Inf), control = list(fnscale = -1))

# optimale Parameter des Modells
(param <- optModel$par)

# 6)
# Test auf signifikante spatiale Autokorrelation
# Hessematrix
(H = optModel$hessian)

# Geschätzte Kovarianzmatrix
```

```
(covmat <- -solve(H))

# Berechnen der Standardfehler
(errors = sqrt(diag(covmat)))

# t-Test auf signifikante Autokorrelation
param[1]/errors[1]
# Eindeutig größer als das 1-alpha/2 Quantil der N(0,1)-Verteilung (1.96)
# --> zum Niveau 5% ein von 0 verschiedene räumliche Autokorrelation
```

13 Instrumental variables

13.1 The miracle of the instruments

Since instruments are elements of information sets, one can construct an arbitrary number of additional instruments by (nonlinear) transformations of instruments. The following example shows that creating instruments "out of nothing" is possible but does not work very well in practice. Consider the following simple linear model,

$$y_t = \alpha + \beta_1 x_{1t} + \beta_2 x_{2t} + u_t$$

for $t = 1, \ldots, T$. The error term u_t is correlated with both exogenous variables x_{1t}, x_{2t} but uncorrelated with an instrument variable w_t,

$$\begin{pmatrix} x_{1t} \\ x_{2t} \\ u_t \\ w_t \end{pmatrix} \sim N\left(\begin{bmatrix} 5 \\ 5 \\ 0 \\ 5 \end{bmatrix}, \begin{bmatrix} 2 & 0.3 & 0.5 & 0.7 \\ 0.3 & 1 & 0.5 & 0.7 \\ 0.5 & 0.5 & 1 & 0 \\ 0.7 & 0.7 & 0 & 1 \end{bmatrix} \right).$$

1. Activate the packages `MASS` and `AER`. Generate a sample of size $n = 1000$ from the multivariate normal distribution using the `mvrnorm` command of the `MASS` package. Show that the command `ivreg` (of the `AER` package) does not work as there is only one instrument but two endogenous regressors.
2. Write a program that performs the following steps.

- Create an empty matrix Z with $R = 1000$ rows and 3 columns.
- Start a `for`-loop over $r = 1, \ldots, R$.
- Inside the loop, generate a sample of size $n = 1000$ from the multivariate normal distribution using the `mvrnorm` command of the `MASS` package.
- Use the columns for x_1, x_2 and u to compute the values of the endogenous variable

$$y_t = 1 + 2x_{1t} + 3x_{2t} + u_t.$$

- Use the column for w to create *two* instruments w_1 and w_2,

$$\begin{aligned} w_{1t} &= w_t^2 \\ w_{2t} &= w_t^3. \end{aligned}$$

- Use the command `ivreg` of the `AER` package to compute the IV estimation. Save the coefficient estimates $\hat{\alpha}, \hat{\beta}_1, \hat{\beta}_2$ in row r of the matrix Z.
- End the loop.
- Compute the median of the three estimates $\hat{\alpha}, \hat{\beta}_1, \hat{\beta}_2$.
- Compute the standard errors of the estimates.
- Split the screen using the command `par(mfrow=c(3,1))` and plot the three histograms.

13.1 The miracle of the instruments The code might look like this:

```
########################################
#### The miracle of the instruments ####
########################################
#1)
library(MASS)
library(AER)

# Generate data: both exogenous variables are correlated with u
Omega <- matrix(c(
  2,0.3,0.5,0.7,
  0.3,1,0.5,0.7,
  0.5,0.5,1,0,
  0.7,0.7,0,1),nrow=4,ncol=4)

n <- 100
dat <- mvrnorm(n,c(5,5,0,5),Omega)
x1 <- dat[,1]
x2 <- dat[,2]
u <- dat[,3]
w <- dat[,4] # instrument
y <- 1+2*x1+3*x2+u
obj <- ivreg(y~x1+x2|w) #does not work

#2)
R <- 1000
Z <- matrix(NA,nrow=R,ncol=3)
ZZ <- matrix(NA,nrow=R,ncol=3)
for(r in 1:R) {
  dat <- mvrnorm(n,c(5,5,0,5),Omega)
  x1 <- dat[,1]
  x2 <- dat[,2]
  u <- dat[,3]
  w <- dat[,4] # instrument
  y <- 1+2*x1+3*x2+u

  #OLS is inconsistent
  ols <- lm(y~x1+x2)

  # IV from AER-package
  w2 <- w^2
  w3 <- w^3
  obj <- ivreg(y~x1+x2|w2+w3)

  #store estimates
  Z[r,] <- coefficients(obj)
  ZZ[r,] <- coefficients(ols)
}

print(apply(Z,2,median))
print(apply(Z,2,mean))
print(apply(Z,2,sd))
print(apply(ZZ,2,median))
print(apply(ZZ,2,mean))
print(apply(ZZ,2,sd))

par(mfrow=c(3,1))
truehist(Z[,1])
truehist(Z[,2])
truehist(Z[,3])
```

13.2 Linear combinations of instruments

This exercise is close to exercise 8.2 of Davidson and MacKinnon (2004). Consider the simple IV estimator $\hat{\beta}_{IV}$, computed first with an $T \times K$ matrix W of instruments, and then with another $T \times K$ matrix WJ, where J is a $K \times K$ nonsingular matrix. Show that the two estimators coincide. Hence, if the model is just identified, linear combinations of the K instruments have no effect.

13.2 Linear combinations of instruments

The two estimators are

$$(W'X)^{-1}W'y \text{ and } (J'W'X)^{-1}J'W'y$$

Since J and $(W'X)$ are both $k \times k$ nonsingular matrices, we get

$$(J'W'X)^{-1}J'W'y = (W'X)^{-1}(J')^{-1}J'W'y = (W'X)^{-1}W'y$$

13.3 Compulsory School Attendance

This exercise is a replication of some parts of the article "Does Compulsory School Attendance Affect Schooling and Earnings?" by Angrist and Krueger, *Quarterly Journal of Economics* 106 (1991) 979-1014.

1. Load the Stata dataset `AngristKrueger1991Data.dta`. For persons born between 1930Q1 and 1939Q4, plot the years of education against the year of birth (see Figure I in the article). Do the same for persons born between 1940Q1 and 1949Q4 (see Figure II).
2. For 1930Q1 until 1949Q4, plot the mean log weekly earnings against the year of birth (see Figure V).
3. From now on, we only consider persons born between 1920Q1 and 1929Q4. Drop all other observations.[8] Regress the log weekly earnings on the years of education and a set of nine dummies for the year of birth[9] using the OLS command `lm` (see column (1) in Table IV of the article).
4. Compute age as the difference 1970 minus date-of-birth, e.g. a person born in 1925Q3 has age $1970 - 1925.75 = 44.25$. Add age and age-squared to the OLS regression (see column (3) in Table IV).

[8]If you have `attach`ed the dataframe, please first delete all variables from your workspace by `rm(list=ls())`. Then re-load the dataset.

[9]The easiest way to deal with the dummy variable is as follows: Create a new variable in the following way: `Dyear <- factor(yob)`. If this variable is included as a regressor in the `lm` command, R will automatically generate the necessary dummy variables.

5. Activate the `AER` package. The `ivreg` command can be used for instrumental variables estimation; its syntax is close to the syntax of the `lm` command, see `?ivreg`.

   The instrumental variables used by Angrist and Krueger are the year of birth, and the year of birth interacting with the quarter of birth. To avoid multicollinearity, one quarter per year has to be dropped from the list of instruments. To economize on time and computer resources, define the instrument variable as a `factor`.[10]

   Estimate an IV regression of log weekly wage on education and year dummies using the instruments of Angrist and Krueger (see column (2) in Table IV).

6. Add age and age-squared to the IV regression (see column (4) in Table IV).

13.3 Compulsory School Attendance

The code might look like this:

```
######################################
#### Compulsory School Attendance ####
######################################
library(foreign)
library(AER)
graphics.off()

#1) Replicate Figure I and II
x <- read.dta(file.choose())
View(x)
dob <- x$yob+(x$qob-1)*0.25

# Figure I
Z <- matrix(NA,40,2)
Z[,1] <- seq(1930,1939.75,by=0.25)
for(i in 1:dim(Z)[1]) {
  Z[i,2] <- mean(x$educ[dob==Z[i,1]])
}
plot(Z,t="o",main="Figure I",xlab="Year of Birth",
     ylab="Years of Completed Education",ylim=c(12.2,13.2))

# Figure II
Z <- matrix(NA,40,2)
Z[,1] <- seq(1940,1949.75,by=0.25)
for(i in 1:dim(Z)[1]) {
  Z[i,2] <- mean(x$educ[dob==Z[i,1]])
}
plot(Z,t="o",main="Figure II",xlab="Year of Birth"
     ,ylab="Years of Completed Education",ylim=c(13,13.9))

#2) Replicate Figure V
Z <- matrix(NA,80,2)
```

[10]Suppose the date of birth (`dob`) is given as 1920, 1920.25, 1920.5, 1920.75, ... Then execute the following commands to create a `factor` of instruments:

```
Dq <- dob
Dq[Dq-floor(Dq)==0.75] <- 0
Dq <- factor(Dq)
```

The `factor` `Dq` can now be used as an instrument, representing all required dummy instruments.

```
Z[,1] <- seq(1930,1949.75,by=0.25)
for(i in 1:dim(Z)[1]) {
  Z[i,2] <- mean(x$lwklywge[dob==Z[i,1]])
}
plot(Z,t="o",main="Figure V",xlab="Year of Birth",ylab="Log Weekly Earnings")

#3) Replicate column 1 of table IV
# Backup data
backup <- x
# Drop all persons born after 1929Q4
x <- x[x$yob<1930,]
dob <- x$yob+(x$qob-1)*0.25

# Create yob dummies
Dyear <- factor(x$yob)

# Column (1)
regr <- lm(x$lwklywge ~ x$educ + Dyear)
summary(regr)

#4) Replicate column 3 of table IV
age <- 1970-dob
# Column (3)
regr <- lm(x$lwklywge~x$educ+Dyear+age+I(age^2))
summary(regr)

#5) Replicate column 2 of table IV
Dq <- dob
Dq[Dq-floor(Dq)==0.75] <- 0
Dq <- factor(Dq)
# Column (2)
regr <- ivreg(x$lwklywge~x$educ+Dyear|Dq+Dyear)
summary(regr)

#6) Replicate column 4 of table IV
# Column (4)
regr <- ivreg(x$lwklywge~x$educ+age+I(age^2)+Dyear|Dq+Dyear)
summary(regr)
```

13.4 A simple example

This "simple example" is close to exercise 8.10 of Davidson and MacKinnon (2004). Consider the model

$$\begin{aligned} y_t &= \beta_0 x_t + \sigma_u u_t \\ x_t &= \pi_0 w_t + \sigma_v v_t \end{aligned}$$

with

$$\begin{pmatrix} u_t \\ v_t \end{pmatrix} \sim N\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix} \right).$$

and $t = 1, \ldots, T$. Write an R program to generate at least $R = 1000$ samples for x and y with sample size $T = 10$ using the parameters $\sigma_u = \sigma_v = 1$, $\pi_0 = 1$, $\beta_0 = 0$, and $\rho = 0.5$. For the exogenous instrument $w = (w_1, \ldots, w_T)'$, use independent drawings from the standard normal distribution, and then rescale w so that $w'w$ is equal to T.

For each simulated sample, compute the simple IV estimator (if you use the `ivreg` command of the `AER` package, make sure to drop the intercept by including "`-1`"as a regressor). Then draw the empirical distribution function[11] of the realizations of the estimator on the same plot as the cdf of the normal distribution with mean zero and variance $\sigma_u^2/(T\pi_0^2)$.

In addition, for each simulated sample, compute the OLS estimator, and plot the empirical distribution function of the realizations of this estimator on the same axes as the empirical distribution function of the realizations of the IV estimator.

Redo the exercise for sample size $T = 100$, and – if your computer is fast enough – also for $T = 1000$.

13.4 A simple example

The code might look like this

```
##########################
#### A simple example ####
##########################
library(AER);library(MASS);
R <- 1000
TT <- 10
sigu <- 1
sigv <- 1
pi0 <-1
bet0 <- 0
rho <- 0.5
w <- rnorm(TT)
w <- sqrt(TT/(t(w)%*%w))*w
(t(w)%*%w==TT)
#Rescaling w would not affect the results if x remained unchanged.
#If x changes, any change in the scaling of w would have to be offset
#by a compensating change in pi0. With the speicfied rescaling there
#is no need to change pi0 when we change the way in which the instrument
#is generated, because the variance of w is equal to 1 in every set
#of simulated data

Z <- matrix(NA,R,2)
fb <- txtProgressBar(min=0, max=R, style=3)
for (r in 1:R){
  uv <- mvrnorm(TT,c(0,0),matrix(c(1,rho,rho,1),2,2))
  w <- rnorm(TT)
  w <- sqrt(TT/(t(w)%*%w))*w
  x <- pi0*w+sigv*uv[,1]
  y <- bet0*x + sigu*uv[,2]
  Z[r,1] <- coefficients(ivreg(y~x-1|w))
  Z[r,2] <- coefficients(lm(y~x-1))
  setTxtProgressBar(fb,r)
}
```

[11]The easiest way to do so is to use the R function `ecdf`, e.g. `plot(ecdf(...))`.

```
close(fb)

plot(ecdf(Z[,1]),col="blue",main="IV and OLS Estimators",xlim=c(-2,2))
lines(ecdf(Z[,2]),col="red")
curve(pnorm(x,mean=0,sd=(sigu^2/(TT*pi0^2))),add=T)
legend(locator(1),c("TRUE","IV","OLS"),fill=c("black","blue","red"))
#Distributions differ, especially in the left-hand tail, which is very much
#longer for the edf of the iv estimator than asymptotic theory would suggest
```

13.5 Money demand

Load the dataset `money.csv`. The data is taken from the web site of Davidson and MacKinnon (2004). The file contains seasonally adjusted quarterly data for the logarithm of real money supply (m_t), real GDP (y_t), and the 3-month treasury bill rate (r_t) for Canada.

1. This is exercise 8.25 of Davidson and MacKinnon (2004). Estimate the model

$$m_t = \beta_1 + \beta_2 r_t + \beta_3 y_t + \beta_4 m_{t-1} + \beta_5 m_{t-2} + u_t$$

   by OLS for the period 1968:1 to 1998:4. Then perform a Durbin-Wu-Hausman test for the hypothesis that the interest rate, r_t, can be treated as exogenous, using r_{t-1} and r_{t-2} as additional instruments.

2. This is exercise 8.26 of Davidson and MacKinnon (2004). Estimate the model by generalized instrumental variables, treating r_t as endogenous and using r_{t-1} and r_{t-2} as additional instruments. Are the estimates much different from the OLS ones?

3. For the IV estimation, perform a test of over-identifying restrictions.

13.5 Money demand

1. Durbin-Wu-Hausman test:
   H_0 : r_t can be treated as exogenous (OLS is better $E(X'u) = 0$) vs. H_1 : r_t cannot be treated as exogenous (OLS is not consistent, IV model is better $E(W'u) = 0$), with r_{t-1} and r_{t-2} as additional instruments.
   Idea: compare $\widehat{\beta_{IV}} - \widehat{\beta_{OLS}}$. To test if this difference is significantly different from zero, perform a Wald test of $\delta = 0$ in the Wu-regression: $y = X\beta + P_W\widetilde{X}\delta$ with $\widetilde{X}$ including all possible endogenous regressors (here r_t) and $P_W = W(W'W)^{-1}W'$

2. $\widehat{\beta_{GIV}} = (X'P_WX)^{-1}X'P_Wy$

3. Test of overidentifying restrictions:
   Idea: Test if IV residuals can be explained by the full set of instruments W.
   H_0 : Instruments are valid and uncorrelated with the residuals.
   Testregression: $u_i = W_i'\gamma + \varepsilon_i$ with $i = 1, \dots, n$.
   Teststatistics:$n\ R^2 \sim \chi^2(m)$ with m : degrees of overidentification.
   If instruments pass the test (that is H_0 is not rejected), they are valid by this criterion.

The code might look like this:

```
######################
#### Money demand ####
######################
library(AER)
# read data, define vectors and matrices
money <- read.csv(file.choose())
View(money)
m <- money$m
r <- money$r
y <- money$y
TT <- length(m)
# Matrix of regressors
X <- cbind(1,r[5:TT],y[5:TT],m[4:(TT-1)],m[3:(TT-2)])
# Matrix of Instruments
W <- cbind(1,r[4:(TT-1)], r[3:(TT-2)],y[5:TT], m[4:(TT-1)], m[3:(TT-2)])
# Projection-matrix
Pw <- W %*% solve( t(W)%*%W ) %*% t(W)

#1)
OLS <- lm(m[5:TT] ~ r[5:TT] + y[5:TT] + m[4:(TT-1)] + m[3:(TT-2)])
summary(OLS)

# Durbin-Wu-Hausman test
# If r_t-1 and r_t-2 are appropriate instruments for r_t,
# then the OLS estimator is not consistent, but the IV-estimator is!
WuRegr <- lm(m[5:TT] ~ Pw%*%r[5:TT] + r[5:TT] + y[5:TT] + m[4:(TT-1)] + m[3:(TT-2)])
summary(WuRegr)
# H0 cannot be rejected, meaning r_t can be treated as exogenous

#2)
IV <- ivreg(m[5:TT] ~ r[5:TT] + y[5:TT] + m[4:(TT-1)] + m[3:(TT-2)]
            |r[4:(TT-1)]+ r[3:(TT-2)] + y[5:TT] + m[4:(TT-1)] + m[3:(TT-2)])
summary(IV)
#You can get the same coefficients, if you simply use the Formula
betaGIV <- solve(t(X)%*%Pw%*%X)%*%t(X)%*%Pw%*%m[5:TT]
betaGIV

# compare to OLS coefficients:
OLS$coefficients
IV$coefficients

# Test if the coefficient is the same
covmat <- IV$sigma^2*IV$cov
linearHypothesis(IV,paste("r[5:TT]=",OLS$coefficients[2]),V=covmat) # H0 can not be rejected
# Not surprising as we have seen in the DWU test

#3)
u <- residuals(IV)
n <- length(u)
#perform OLS and store the results
obj <- summary(lm(u ~ r[4:(TT-1)] + r[3:(TT-2)] + y[5:TT] + m[4:(TT-1)] + m[3:(TT-2)]))
teststat <- n*obj$r.squared
```

```
1-pchisq(teststat,df=1) # p-value
#Nullhypothesis can be rejected
#This strongly suggests that either at least one of rt-1 and rt-2 is not a valid instrument,
#or that the model is misspecified. Here the latter is more sensible, in particular,
#we could simply add rt-1 and rt-2 to the regressors of the model
```

13.6 Tests for the IV model

Load the dataset `fertility.csv` and its description (`fertility.pdf`) from the internet site of the course. The dataset is provided on the internet site of the textbook by Stock and Watson. It is a subset of the data used by Angrist and Evans, "Children and their parents' labor supply: Evidence from exogenous variation in family size", *American Economic Review*, 88 (1998) 450-77. Since some variables included in Angrist and Evans are missing, we cannot reproduce their results exactly.

1. The variable `morekids` indicates if there are more than two children. The variable `samesex` indicates if the first two children are both boys or both girls. Compute the fraction of families that had another child if the first two children were of the same sex, and the fraction if the first two children were of different sex (see Table 3, married women, 1980 data, lower half of the table, in Angrist and Evans).
2. We would like to estimate the causal effect of `morekids` on the number of weeks worked by the mother. Perform an OLS regression of `weeksm1` on `morekids` plus all other variables except `samesex`. Explain why OLS is inappropriate for estimating the causal effect.
3. Explain why the variable `samesex` is a valid instrument for the regression of `weeksm1` on `morekids`.
4. Perform an IV regression of `weeksm1` on `morekids` using `samesex` as instrument.
5. Perform an asymptotic t-test of the null hypothesis that the coefficients of `hispan` and `othrace` are equal. Hint: The estimated covariance matrix of $\hat{\beta}_{IV}$ can be computed by `a$sigma^2*a$cov` where `a` is the object returned by the command `ivreg`.
6. Perform a Wald test of the null hypothesis that the three coefficients of `boy1st`, `boy2nd`, and `hispan` are all equal to zero.

13.6 Tests for the IV model

The code might look like this:

```
################################
#### Tests for the IV model ####
################################
library(AER)
fertility <- read.csv2(file.choose())
View(fertility)
morekids <- fertility$morekids
samesex <- fertility$samesex
weeksm1 <- fertility$weeksm1
boy1st <- fertility$boy1st
boy2nd <- fertility$boy2nd
agem1 <- fertility$agem1
```

```
black <- fertility$black
hispan <- fertility$hispan
othrace <- fertility$othrace
#1)
n <- dim(fertility)[1] #number of families
x <- fertility[morekids==1,] # only families that had another child
m <- dim(x)[1] #number of families that had another child
dim(x[x$samesex==1,])[1]/m
dim(x[x$samesex==0,])[1]/m

#2)
regr <- lm(weeksm1 ~ morekids + boy1st + boy2nd + agem1 + black + hispan + othrace)
summary(regr)
#The coefficient is -6.23209. This indicates that women with
#more than 2 children work 6.23209 fewer weeks per year than women
#with 2 or fewer children. However, both fertility (morekids) and
#laborsupply (weeks worked) are choice variables. A woman who works
#more than average (positive regression error) may also be a woman
#who is less likely to have an additional child. This would imply
#that morekids is positively correlated with the regression error.
#OLS estimator is thus positively biased.

#3)
#Relevance of instrument
#Samesex is random and unrelated to any of the other variables
#including the error term in the labor supply equation. Thus, the
#instrument is exogenous. The instrument is also relevant (see part 1).
#You can also test for relevance: compute the F-Statistic in the
#regression: $morekids_i = \beta_0+\beta_1 samesex_i +\varepsilon_i$.
relevInst <- lm(morekids ~ samesex)
summary(relevInst) # F-Statistic is high!

#4)
ivmod <- ivreg(weeksm1 ~ morekids+boy1st + boy2nd + agem1 + black
               + hispan + othrace|samesex + boy1st + boy2nd
               + agem1 + black + hispan + othrace)
summary(ivmod)

#5) t-test that two coefficients are equal
covmat <- ivmod$sigma^2*ivmod$cov
betaHisp <- ivmod$coefficients[7]
betaOthr <- ivmod$coefficients[6]
varHisp <- covmat[7,7]
varOthr <- covmat[6,6]
ttest <- (betaHisp -betaOthr)/sqrt(varHisp + varOthr)
abs(ttest) > 2.56 # reject H0 at least on a 1% level!
1-pt(abs(ttest),n-7) #p-value is zero!


#6)
betahat <- ivmod$coefficients[c(3,4,7)]
beta0 <- c(0,0,0)
```

```
ivmod0 <- ivreg(weeksm1 ~ morekids + agem1 + black + hispan
                + othrace|samesex + boy1st + boy2nd
                + agem1 + black + hispan + othrace)
# Wald test formula and p-value
Wald <- (betahat-beta0)%*%solve(covmat[c(3,4,7),c(3,4,7)])%*%(betahat - beta0)
1-pchisq(Wald,df=3)
#or simply use the linearHypothesis command
linearHypothesis(ivmod,c("boy1st=0","boy2nd=0","hispan=0"), V=covmat)
```

14 GMM

14.1 The R package gmm

Install and activate the R package `gmm`.

1. Read (at least) section 2 of the R vignette "Computing Generalized Empirical Likelihood and Generalized Method of Moments with R" (`gmm_with_R.pdf`) which can be found on the internet site of the course or in the documentation of the package.
2. Explain the relationship between the elementary zero functions f and the functions g used in the `gmm` package.
3. This is the example given in section 3.1 of the R vignette. Suppose you want to estimate the parameters μ and σ of a normal distribution X by GMM using the three moment conditions

$$\begin{aligned} E(X) &= \mu \\ E((X-\mu)^2) &= \sigma^2 \\ E(X^3) &= \mu\left(\mu^2+3\sigma^2\right). \end{aligned}$$

   Write an R function with arguments $\theta = (\mu, \sigma)$ and data X that computes and returns the moment conditions g.
4. Set the random number seed, `set.seed(123)`. Generate $n = 100$ random numbers from the normal distribution $N(4, 2^2)$. Using the starting values $(\mu_0, \sigma_0) = (0, 0)$ run the `gmm` command and save the estimation results in the object `res`. Print `summary(res)` and interpret the output.

14.1 The R packagae gmm

Basic scheme to use gmm:

1. Set up matices with data Y and/or instruments W.
2. Specify the moment conditions

```
g <- function(param,dat) {moment conditions}
```

   For the gmm package this means that you have to program the formula inbetween the expectation $E[g(\theta, Y)] = 0$. Here:

$$\begin{aligned} X-\mu \\ (X-\mu)^2-\sigma^2 \\ X^3-\mu(\mu^2-3\sigma^2) \end{aligned}$$

3. Call $gmm(\underbrace{g=g}_{\text{function}}, \underbrace{x=Y}_{\text{data}}, \underbrace{t0=c(0,\dots,0)}_{\text{starting values}}, \dots)$. Use appropriate arguments for numerical optimization, weights, gradient (to improve precision), etc. Most of the times just use the standard setting.

The code might look like this:

```
############################
#### The R packagae gmm ####
############################
install.packages("gmm")
library(gmm)
##generate data
set.seed(123)
n <- 200
dat <- rnorm(n,mean=4,sd=2)

##moment conditions
g <- function(param,x) {
  m1 <- param[1]-x
  m2 <- param[2]^2-(x-param[1])^2
  m3 <- x^3 - param[1]*(param[1]^2+3*param[2]^2)
  f <- cbind(m1,m2,m3)
  return(f)
}
g(c(3,1),dat) #gives you a n-by-L matrix with n observations and L moments

##gmm estimation with standard settings
t0 <- c(mu=3, sigm=1)
res1 <- gmm(g,dat,t0)
print(res1)
summary(res1)
coef(res1)
vcov(res1)
confint(res1)
specTest(res1)

##gmm estimation with gradient
#gradient of gbar=1/n*sum(g)
Dgbar <- function(param,x){
  xbar <- mean(x)
  Dgbar <- matrix(c(1, 2*(xbar-param[1]),
    -3*(param[1]^2+param[2]^2), 0, 2*param[2], -6*param[1]*param[2]),nrow=3)
  return(Dgbar)
}
Dgbar(c(3,1),dat)
t0 <- c(mu=3, sigm=1)
res2 <- gmm(g,dat,t0,grad=Dgbar)
print(res2);summary(res2);coef(res2);vcov(res2);confint(res2);specTest(res2)

##gmm estimation with restrictions on parameter space (with nlminb)
t0 <- c(mu=3, sigm=1)
res3 <- gmm(g,dat,t0,optfct="nlminb",lower=c(-5,0),upper=c(5,5))
print(res3);summary(res3);coef(res3);vcov(res3);confint(res3);specTest(res3)

##gmm estimation: Iterative GMM (ITGMM) and continuous updated GMM (CUE)
t0 <- c(mu=3, sigm=1)
res4 <- gmm(g,dat,t0,type="iterative",crit=1e-5,itermax=200)
print(res4);summary(res4);coef(res4);vcov(res4);confint(res4);specTest(res4)
```

```
res5 <- gmm(g,dat,res4$coef,type="cue")#use ITGMM as starting values
print(res5);summary(res5);coef(res5);vcov(res5);confint(res5);specTest(res5)
```

14.2 Nonlinear least squares estimation and GMM

Nonlinear least squares estimation is a special case of GMM. Consider the nonlinear regression model

$$y_t = x_t(\beta) + u_t$$

where $x_t(\beta)$ is nonlinear function of the parameters and the data. Assume that the u_t are i.i.d. with $E(u_t) = 0$ and $Var(u_t) = \sigma^2$ and independent of the x_t.

1. Formulate the general model and its least squares estimation in the GMM framework.
2. As a special case, consider the exponential model (see exercise 11.1),

$$y_t = \exp(\alpha + \beta x_t) + u_t$$

   where $u_t \sim N(0, \sigma^2)$. Load the dataset `expgrowth.csv` from the course site and estimate the parameters α and β and their standard errors by GMM using the command `gmm`. Compare your results with the maximum likelihood estimates computed in exercise 11.1.

14.2 Nonlinear least squares estimation and GMM

1. As elementary zero functions we can use $E(u_t) = 0$ and $E(u_t^2 - \sigma^2) = 0$. As Instruments we can use the elementary zero function itself and $x_t(\beta)$, since $E(x_t(\beta) \cdot u_t) = E(x_t(\beta) \cdot (y_t - x_t(\beta))) = 0$. That is we have three moment conditions:

$$\begin{aligned} E(u_t) &= E(y_t - x_t(\beta)) = 0 \\ E(x_t(\beta)u_t) &= E(x_t(\beta) \cdot (y_t - x_t(\beta))) = 0 \\ E(u_t^2 - \sigma^2) &= E((y_t - x_t(\beta)^2 - \sigma^2) = 0 \end{aligned}$$

2. Here $x_t(\beta) = exp(\alpha + \beta x_t)$. This yields the same results as in exercise 11.1.

The code might look like this:

```
####################################
#### Nonlinear regression model ####
####################################
expgrowth <- read.csv(file.choose())
View(expgrowth)
x <- expgrowth$x
y <- expgrowth$y

g <- function(param,dat){
  y <- dat[,1]
  x <- dat[,2]
  u <- y - exp(param[1]+param[2]*x)
  f1 <- u
  f2 <- u^2
  m1 <- 1*f1
  m2 <- x*f1
  m3 <- f2-param[3]^2
  return(cbind(m1,m2,m3))
}
g(c(1,0.1,1),cbind(y,x)) # g gives you a nxq matrix with n observations and q moments

gmmmod <- gmm(g,x=cbind(y,x),c(alpha=0,beta=0.01,sigma=1))
print(gmmmod)
gmmmod <- gmm(g,x=cbind(y,x),c(alpha=2,beta=0.01,sigma=2))
print(gmmmod$coefficients)
#depends on the start-values, in order to be more precise one could specify the gradient
summary(gmmmod)
```

14.3 Ordinary least squares estimation and GMM

Ordinary least squares estimation is a special case of GMM. This exercise compares the two uses of the gmm package, i.e. explicitly taking into account linearity or not. Consider the linear regression model

$$y_t = \alpha + \beta_1 x_{1,t} + \beta_2 x_{2,t} + u_t$$

for $t = 1, \ldots, n$. Assume that the standard assumptions are satisfied.

1. Load the dataset `olsgmm.csv`. It contains $n = 100$ (artificial) observations (y_1, x_{11}, x_{21}),. . . , (y_n, x_{1n}, x_{2n}).
2. Estimate the linear regression model using the ordinary least squares command `lm`.
3. Estimate the linear regression model with GMM explicitly taking into account linearity. Save the results and compare it to the OLS estimator.
4. Program a function `g` with two arguments. The first argument is the vector of deep parameters $\theta = (\alpha, \beta_1, \beta_2)$. The second argument is the data matrix. The function `g1` should return the moment conditions as a matrix,

$$\begin{bmatrix} f_{11} & f_{21} & f_{31} \\ f_{12} & f_{22} & f_{32} \\ \vdots & \vdots & \vdots \\ f_{1n} & f_{2n} & f_{3n} \end{bmatrix}$$

   where

$$\begin{aligned} f_{1t} &= (y_t - \alpha - \beta_1 x_{1t} - \beta_2 x_{2t}) \\ f_{2t} &= (y_t - \alpha - \beta_1 x_{1t} - \beta_2 x_{2t}) x_{1t} \\ f_{3t} &= (y_t - \alpha - \beta_1 x_{1t} - \beta_2 x_{2t}) x_{2t} \end{aligned}$$

   for $t = 1, \ldots, n$.
5. Now re-estimate the linear regression model using the gmm syntax for general nonlinear models. Set the starting values to (1,0.5,1.2) (these are the true values used for simulating the data). Compare the result with the result obtained when taking into account linearity.

14.3 Ordinary least squares estimation and GMM

The code might look like this

```
####################################################
#### Ordinary least squares estimation and GMM ####
####################################################
#Get data
olsgmm <- read.csv(file.choose())
View(olsgmm)
y <- olsgmm$y
x1 <- olsgmm$x1
x2 <- olsgmm$x2

#OLS
ols <- lm(y~x1+x2)
summary(ols)

#GMM estimation explicitly taking into account linearity
gmm_lin <- gmm(y~x1+x2,cbind(x1,x2))
gmm_lin$coef - ols$coef #numerically the same

#GMM estimation without linearity
g <- function(param,dat){
  y <- dat$y
  x1 <- dat$x1
  x2 <- dat$x2
  u <- y -param[1] -param[2]*x1 -param[3]*x2
  f1 <- u
  f2 <- u*x1
  f3 <- u*x2
  return(cbind(f1,f2,f3))
}
g(c(1,0.5,1.2),olsgmm) # g gives you a nxq matrix with n observations and q moments

t0<-c(alpha=1,beta1=0.5,beta2=1.2)
gmm_nonlin <- gmm(g,olsgmm,t0)
print(gmm_nonlin$coefficients)
print(gmm_lin$coefficients)
```

14.4 Maximum likelihood estimation and GMM

Maximum likelihood estimation is a special case of GMM. Let $X_1, \ldots, X_n$ be a random sample from the random variable X. We know the distributional family of X (e.g. normal distribution) but we do not know the parameters. Denote the density function by f and the parameters by θ.

1. Show that the maximum likelihood estimation can be formulated in the GMM framework.
2. As a special case, consider the censored lognormal distribution (see exercise 12.3). Let $X \sim LN(\mu, \sigma^2)$ and let $X_1, \ldots, X_n$ be an unobserved sample from X. The observations are
$$Y_i = \begin{cases} X_i & \text{if } X_i < c \\ c & \text{if } X_i \geq c \end{cases}$$
where $c = 12$ is a known constant. Load the dataset `censoredln.csv` and estimate the parameters μ and σ and their standard errors by GMM using the command `gmm`. Compare your results with the maximum likelihood estimates computed in exercise 12.3.

14.4 Maximum likelihood estimation and GMM

1. The likelihood function is given by $L(\theta; x_1, \dots, x_n) = \prod_{i=1}^{n} f_X(x_i;\theta)$ and the log Likelihood

$$logL(\theta; x_1, \dots, x_n) = \sum_{i=1}^{n} logf_X(x_i;\theta)$$

We gradient vector $g(\theta) = \partial logL(\theta)/\partial\theta$ is called the score vector. Define the gradient contributions

$$G_{ij}(\theta, x_i) = \frac{\partial logf_X(x_i;\theta)}{\partial\theta_j}$$

We have shown in the lecture that

$$E(G_{ij}(\theta, x_i) = \frac{\partial logf_X(x_i;\theta)}{\partial\theta_j}) = 0$$

So the gradient contributions are our moment conditions.

2. Let's first derive the gradient contributions of a normally distributed variable X with density function $f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} exp\{-(x-\mu)/(2\sigma^2)\}$:

$$\frac{\partial logf_X(x)}{\partial\mu} = \frac{x-\mu}{\sigma^2}$$
$$\frac{\partial logf_X(x)}{\partial\sigma} = \frac{-1}{\sigma} + \frac{(x-\mu)^2}{\sigma^3}$$

Furthermore we can use the implicit function theorem to derive the derivatives of the cdf of a normally distributed variable x with mean μ and standard deviation σ:

$$\frac{\partial\Phi\left(\frac{x-\mu}{\sigma}\right)}{\partial\mu} = -\phi\left(\frac{x-\mu}{\sigma}\right)$$
$$\frac{\partial\Phi\left(\frac{x-\mu}{\sigma}\right)}{\partial\sigma} = -\phi\left(\frac{x-\mu}{\sigma}\right)\frac{x-\mu}{\sigma^2}$$

with Φ the cdf and ϕ as the pdf of a standard normally distributed variable. We have shown in exercise 12.3 that the log-likelihood function for censored data is equal to

$$\log L(\mu,\sigma;x) = \sum_{i=1}^{n_1} \log f_X(log(x_i);\mu,\sigma) + \sum_{i=1}^{n_2} \log\left(1-\Phi\left(\frac{log(c)-\mu}{\sigma}\right)\right)$$

where we used the fact that if a random vector X is log-normally distributed, $log(X)$ is normally distributed. Furthermore, n_1 is the number of non-censored observations and n_2 the number of censored observations, $n_1 + n_2 = n$). The function g for the gmm package should therefore distinguish between censored and uncensored observations. A typical row for an uncensored observation of g should have the following structure

$$\left[\frac{log(x_i)-\mu}{\sigma^2}, \quad \frac{(log(x_i)-\mu)^2-\sigma^2}{\sigma^3}\right]$$

whereas for a censored observation we require

$$\frac{\phi\left(\frac{log(c)-\mu}{\sigma}\right)}{1-\Phi\left(\frac{log(c)-\mu}{\sigma}\right)} \cdot \left[1, \quad \frac{log(c)-\mu}{\sigma^2}\right]$$

The code might look like this

```
################################################
#### Maximum likelihood estimation and GMM ####
################################################
library(gmm)
# Get data
censoredln <- read.table(file.choose(), header=T, quote="\"")

g <- function(param,dat){
  cens <- 12
  x <- dat
  mue <- param[1]
  sigm <- param[2]
  muncens_i1 <- (log(x[x<12])-mue)/(sigm^2)
  muncens_i2 <- ((log(x[x<12])-mue)^2-sigm^2)/(sigm^3)
  muncens <- cbind(muncens_i1,muncens_i2)
  fact_cens <- 1/pnorm((mue-log(cens))/sigm)*dnorm((mue-log(cens))/sigm)
  num_cens <- length(x[x==12])
  mcens_i1 <- rep(fact_cens,num_cens)
  mcens_i2 <- rep(fact_cens*(log(cens)-mue)/sigm,num_cens)
  mcens <- cbind(mcens_i1,mcens_i2)
  return(rbind(muncens,mcens))
}

# Optimization
t0 <- c(mue = 2, sigm=0.5)
gmm(g,censoredln,t0=t0)
```

14.5 Instrumental variables estimation and GMM

Instrumental variable estimation is a special case of GMM. Consider the linear regression model

$$y = X\beta + u$$

with $u \sim N(0, \sigma^2 I)$. The error term and the regressor matrix X may be correlated but there is a set of instrumental variables W such that $E(u_t|W_t) = 0$.

1. Formulate the general model and the IV estimation in the GMM framework.
2. As a special case, consider the money demand model of exercise 13.5,

$$m_t = \beta_1 + \beta_2 r_t + \beta_3 y_t + \beta_4 m_{t-1} + \beta_5 m_{t-2} + u_t$$

   with the logarithm of real money supply (m_t), real GDP (y_t), and the 3-month treasury bill rate (r_t) for Canada. Load the dataset `money.csv` and estimate the parameters $\beta_1, \ldots, \beta_5$ by GMM using r_{t-1} and r_{t-2} as instruments for the endogenous regressor r_t.

14.5 Instrumental variables estimation and GMM

The gmm framework is given by

$$\begin{aligned} E[m_t - X_t\beta] &= 0 \\ E[y_t(m_t - X_t\beta)] &= 0 \\ E[m_{t-1}(m_t - X_t\beta)] &= 0 \\ E[m_{t-2}(m_t - X_t\beta)] &= 0 \\ E[r_{t-1}(m_t - X_t\beta)] &= 0 \\ E[r_{t-2}(m_t - X_t\beta)] &= 0 \end{aligned}$$

The code might look like this:

```
####################################################
#### Instrumental variables estimation and GMM ####
####################################################
library(gmm)
library(AER)
# read data, define vectors and matrices
money <- read.csv(file.choose())
View(money)
m <- money$m
r <- money$r
y <- money$y
TT <- length(m)

yt <- y[5:TT]
mt <- m[5:TT]; mt1 <- m[4:(TT-1)]; mt2 <- m[3:(TT-2)]
rt <- r[5:TT]; rt1 <- r[4:(TT-1)]; rt2 <- r[3:(TT-2)]

# Gmm estimation in linear model notation
obj <- gmm(mt~rt+yt+mt1+mt2, ~rt1+rt2+yt+mt1+mt2)
obj$coefficients

# Generalized IV estimation
IV <- ivreg(mt~rt+yt+mt1+mt2|rt1+rt2+yt+mt1+mt2)
IV$coefficients
```

14.6 Moment conditions and moment existence

Consider the simple linear model without an intercept

$$y_t = \beta x_t + u_t.$$

Assume that x_t has a t-distribution with 3 degrees of freedom and variance 1. The unit variance can be attained by dividing the t-distribution by $\sqrt{3}$, i.e. `rt(n,df=3)/sqrt(3)`. The error terms u_t are independent of x_t; they have a t_3-distribution with variance σ^2. Set $\beta = 0.9$ and $\sigma^2 = 1$.

1. Generate a sample $(x_1, y_1), \ldots, (x_n, y_n)$ of size $n = 100$.
2. Compute the GMM estimates $\hat{\beta}$ and $\hat{\sigma}$ using the moment conditions

$$\begin{aligned} g_{t1} &= y_t - \beta x_t \\ g_{t2} &= (y_t - \beta x_t)\, x_t \\ g_{t3} &= (y_t - \beta x_t)^2 - \sigma^2. \end{aligned}$$

3. Within a loop $r = 1, \dots, R$, repeat steps 1. and 2. a large number of times and plot the histogram of $\hat{\sigma}$. Is the distribution of $\hat{\sigma}$ well approximated by a normal distribution?
4. Check if the weighting scheme (`wmatrix="ident"` or `wmatrix="optimal"`) influences the distribution of $\hat{\sigma}$.
5. Change the distribution of x_t and u_t from the t_3-distribution with variance 1 to the standard normal distribution.
6. If your computer is fast enough (or you are willing to wait longer), increase the sample size n and redo this exercise.

14.6 Moment conditions and moment existence

The code might look like this:

```
################################################
#### Moment conditions and moment existence ####
################################################
library(gmm)
library(MASS)

n <- 100
x <- rt(n,df=3)/sqrt(3)
u <- rt(n,df=3)/sqrt(3)
y <- 0.9*x+u

g <- function(theta,dat) {
  y <- dat[,1]
  x <- dat[,2]
  bet <- theta[1]
  sigm <- theta[2]
  u <- y-bet*x
  m1 <- u
  m2 <- u*x
  m3 <- u^2-sigm^2
  return(cbind(m1,m2,m3))
}
# GMM estimates for beta and sigma
obj <- gmm(g,cbind(y,x),t0=c(0.9,1),wmatrix="optimal")
obj1 <- gmm(g,cbind(y,x),t0=c(0.9,1),wmatrix="ident")
obj
obj1

# Whole thing within a loop
R <- 1000
n <- 100 # or try n=1000
nu <- 3
Z1 <- matrix(NA,R,2)
Z2 <- matrix(NA,R,2)
for(r in 1:R) {
  x <- rt(n,df=nu)/sqrt(nu/(nu-2))
  u <- rt(n,df=nu)/sqrt(nu/(nu-2))
  y <- 0.9*x+u
  obj1 <- gmm(g,x=cbind(y,x),t0=c(0.9,1),wmatrix="optimal")
  Z1[r,] <- (coefficients(obj1)-c(0.9,1))/sqrt(diag(vcov(obj1)))
  obj2 <- gmm(g,x=cbind(y,x),t0=c(0.9,1),wmatrix="ident")
  Z2[r,] <- (coefficients(obj2)-c(0.9,1))/sqrt(diag(vcov(obj2)))
}
# mean of the estimates for beta and sigma
apply(Z1,2,mean)
apply(Z2,2,mean)
#histogram of sigma compared to a normal distribution
truehist(Z1[,2])
xx <- seq(min(Z1[,2]),max(Z1[,2]),length=500)
lines(xx,dnorm(xx,mean(Z1[,2]),sd(Z1[,2])))

truehist(Z2[,2])
xx <- seq(min(Z2[,2]),max(Z2[,2]),length=500)
lines(xx,dnorm(xx,mean(Z2[,2]),sd(Z2[,2])))

###Now with the normal distribution
R <- 1000 # or try R=1000
n <- 100 # or try n=1000
Z1 <- matrix(NA,R,2)
Z2 <- matrix(NA,R,2)
for(r in 1:R) {
  x <- rnorm(n)
  u <- rnorm(n)
  y <- 0.9*x+u
  obj1 <- gmm(g,cbind(y,x),t0=c(0.9,1),wmatrix="optimal")
  obj2 <- gmm(g,cbind(y,x),t0=c(0.9,1),wmatrix="ident")
  Z1[r,] <- (coefficients(obj1)-c(0.9,1))/sqrt(diag(vcov(obj1)))
  Z2[r,] <- (coefficients(obj2)-c(0.9,1))/sqrt(diag(vcov(obj2)))
}
# mean of the estimates for beta and sigma
apply(Z1,2,mean)
apply(Z2,2,mean)

#histogram of sigma compared to a normal distribution
truehist(Z1[,2])
xx <- seq(min(Z1[,2]),max(Z1[,2]),length=500)
lines(xx,dnorm(xx,mean(Z1[,2]),sd(Z1[,2])))

truehist(Z2[,2])
xx <- seq(min(Z2[,2]),max(Z2[,2]),length=500)
lines(xx,dnorm(xx,mean(Z2[,2]),sd(Z2[,2])))
```

- The GMM estimator is asymptotically normally distributed. However, the estimator for the t-distribution with 3 degrees of freedom is not asymptotically normally distributed. This is because for convergence higher moments have to exist, which they don't for the t-distribution. If you use the normal distribution or another distribution, then the distribution is well approximated by a normal distribution.

- The weighting scheme does not influence the approximate distribution.

14.7 Standard CAPM

In their overview article about GMM applications in finance,[12] Jagannathan et al. (2002) consider the stochastic discount factor representation of the standard capital asset pricing model,

$$\begin{aligned} E\left(m_t R_{it}\right) &= 1 \\ m_t &= \theta_1 + \theta_2 R_{mt} \end{aligned}$$

where $R_{it} = 1 + r_{it}$ is the gross return (and r_{it} the return) of asset i and R_{mt} the market portfolio gross return.

1. Rewrite the standard CAPM such that it fits into the GMM framework, i.e. formulate the moment conditions.
2. Type `data(Finance)` to load the dataset that is included in the `gmm` package. Read `?Finance` to learn about the data structure.
3. Estimate the standard CAPM for the first five companies (i.e. `WMK`, `UIS`, `ORB`, `MAT`, `ABAX`) using the variable `rm` as the (net) market return.
4. Use the function `specTest` to test the overidentifying restrictions.

14.7 Standard CAPM

The moment conditions are given by

$$E\left[(\theta_1 + \theta_2 R_{mt}) R_{it} - 1\right] = 0$$

The code might look like this:

```
#######################
#### Standard CAPM ####
#######################
library(gmm)
#Get data
data(Finance)
?Finance
View(Finance)
#let's only take the first 500 observations
r <- Finance[,c("WMK","UIS","ORB","MAT","ABAX")]
rm <- Finance[,"rm"]

#Define moment conditions
g <- function(param,dat){
  R <- 1+dat[,1:5]
  Rm <- 1+ dat[,6]
  m <- (param[1]+param[2]*Rm)*R-1
  return(m)
}

#estimate GMM
```

[12]Jagannathan, R., Skoulakis, G. and Wang, Z. (2002), Generalized Method of Moments: Applications in Finance, *Journal of Business and Economic Statistics*, 20: 470-481. The password protected article is downloadable from the course site.

```
obj <- gmm(g,x=cbind(r,rm),c(0,0)) #this gives you an error
mode(r) # problem is in the data: r is a list and not numeric. gmm needs numeric data
mode(rm) # rm is numeric
mode(cbind(r,rm)) #if we combine the data, we still get a list
X <- as.matrix(cbind(r,rm)) #so let's convert it into a numeric structure
mode(X)
obj <- gmm(g,x=X,c(0,0))
obj
summary(obj)

# Test of overidentifying restrictions
specTest(obj) #confirms the non-rejection of the theory
```

We can think of a return as a payoff with price one. If you pay one dollar today, the return is how many dollars or units of consumption you get tomorrow.

The asset pricing model says that, although expected returns can vary across time and assets, expected discounted returns should always be the same, 1.

14.8 Consumption-based CAPM

In their overview article about GMM applications in finance,[13] Jagannathan et al. (2002) consider the moment equations

$$E\left(\left(\beta\left(\frac{c_{t+1}}{c_t}\right)^{-\gamma}R_{i,t+1}-1\right)z_t\right)=0$$

for $i=1,\dots,N$. Here, c_t is consumption in period t, $R_{i,t}$ is the gross return of asset i from $t-1$ to t, z_t is a vector of variables known at time t, the parameter β is the time-preference parameter, and the parameter γ is the coefficient of relative risk aversion in the utility function $u(c)=c^{1-\gamma}/(1-\gamma)$.

1. Load the datasets `consumptiondata.csv` and `dax30ann.csv`. The consumption dataset contains information about aggregate consumption levels in current prices from 1970 to 1991 (West Germany) and 1991 to 2010 (Germany). We will only consider variable `V7` (see `LangeReihenKonsum2011Q3.pdf`, page 8). The second dataset contains the start-of-year levels of the DAX30 performance index from 1969 to 2011.
2. Compute the consumption growth rates for West Germany from 1971 to 1991 and for Germany from 1992 to 2010 and concatenate them.
3. Let $z_t=(1,c_t/c_{t-1},R_{DAX,t})$. Set up the model in the GMM framework and estimate β and γ using the `gmm` package.

14.8 Consumption-based CAPM

The code might look like this:

```
#####################################
###### Consumption-based CAPM ######
```

[13]Jagannathan, R., Skoulakis, G. and Wang, Z. (2002), Generalized Method of Moments: Applications in Finance, *Journal of Business and Economic Statistics*, 20: 470-481. The password protected article is downloadable from the course site.

```
####################################
#Load Data
consumptiondata <- read.csv(file.choose(), sep=";", dec=",")
dax30ann <- read.csv(file.choose())
View(consumptiondata)
View(dax30ann)
V7 <- consumptiondata$V7
dax30 <- dax30ann$dax30

#growth rates for consumption
gr1 <- V7[2:22]/V7[1:21]
gr2 <- V7[24:42]/V7[23:41]
gr <- c(gr1,gr2)
#gross return of DAX
R <- dax30[3:42]/dax30[2:41]
#Matrix of data
X <- cbind(gr,R)

g <- function(param,dat){
  TT <- dim(dat)[1]
  gr <- dat[,1]
  R <- dat[,2]
  f <- param[1]*(gr[-1]^(-param[2]))*R[-1]-1
  z <- cbind(1,gr[-TT],R[-TT])
  m <- f*z
  return(m)
}
g(c(1,1),X)
obj <- gmm(g,x=X,c(1,1))
obj
```

14.9 Minimum distance estimation

Consider the following, highly simplified, model of earnings dynamics (F. Guvenen, "An empirical investigation of labor income processes", *Review of Economic Dynamics*, 12 (2009) 58-79),

$$\begin{aligned} y_t^i &= \beta_i t + u_t^i \\ u_t^i &= \rho u_{t-1}^i + \eta_t^i \end{aligned}$$

where y_t^i is log-earnings of person i with t periods of Labour market experience, β_i is an individual specific random effect with variance σ_β^2, ρ is the persistence parameter, and η_t^i are i.i.d. innovations with variance σ_η^2. It can be shown that for $h \geq 2$ the covariance between Δy_t^i and Δy_{t+h}^i is

$$Cov\left(\Delta y_t^i, \Delta y_{t+h}^i\right) = \sigma_\beta^2 - \left[\rho^{h-1}\left(\frac{1-\rho}{1+\rho}\right)\sigma_\eta^2\right]. \tag{3}$$

1. The theoretical $(H+1)\times(H+1)$-covariance matrix of $\left[\Delta y_t^i, \Delta y_{t+1}^i, \ldots, \Delta y_{t+H}^i\right]$ depends on the three unknown parameters σ_β^2, ρ and σ_η^2. The GMM approach to estimation requires that the differences between the elements of the theoretical covariance matrix[14] and its

[14]Due to the symmetry of the covariance matrix, the elements below (or above) the diagonal are omitted.

empirical counterparts should be minimized with respect to the parameter vector $\theta = (\sigma_\beta^2, \rho, \sigma_\eta^2)$. Set $H = 10$ and write an R program that can estimate the parameters by GMM using the command `gmm`. Note that the first order covariances must not enter the estimation since (3) is only valid for $h \geq 2$.

2. Load the artificial dataset `logearnings.csv`. The rows are individuals $i = 1, \ldots, N$, the column are the periods $t = 1, \ldots, T$ with $N = 2000$ and $T = 15$. Compute the parameter estimates and their standard errors.

3. Perform a test of the overidentifying restrictions.

14.9 Minimum distance estimation

15 Indirect inference

15.1 AR(1) processes

The seemingly simple autoregressive process

$$x_t = \rho x_{t-1} + \varepsilon_t$$

with $\varepsilon_t \sim N(0, \sigma^2)$ is sometimes surprisingly hard to estimate. In the following, always use

```
x <- filter(rnorm(n),rho,method="r",init=rnorm(1))
```

to generate a path of length n.

1. Simulate the distribution of the estimator $\hat{\rho}$ for $\rho = 0.8$ and $n = 100$. Use the command `ar` with options `order=1` and `aic=F` to estimate ρ (if you like, try different estimation methods, e.g. `ols` or `mle`).
2. Simulate the distribution of $\hat{\rho}$ for the unit root process with $\rho = 1$.
3. Simulate the distribution of $\hat{\rho}$ for the explosive process with $\rho = 1.01$.
4. Write an R program to estimate the $AR(1)$ parameter ρ by indirect inference. The auxiliary model is, of course, itself an $AR(1)$ process. The number of auxiliary paths should be $H = 10$.
5. Determine the distribution of the indirect inference estimator $\hat{\rho}$ by simulation for values of $\rho = 0.8, 1, 1.01$.

15.1 AR(1) processes

The code might look like this:

```
############################################
#### Indirect Inference - AR(1) process ####
############################################
library(MASS)
#Define function that estimates the ar process for different values of rho,n and R
simestim <- function(rho,n,R) {
  Z <- matrix(NA,R,3)
  for (r in 1:R) {
    x <- filter(rnorm(n),rho,method="r",init=rnorm(1))
    Z[r,1]<- ar(x,aic=F,order = 1,method = "yw")$ar # method: yule-walker
    Z[r,2]<- ar(x,aic=F,order = 1, method = "ols")$ar #method: ols
    Z[r,3]<- ar(x,aic=F,order = 1, method = "mle")$ar #method: mle
  }
  return(Z)
}

#1) Stationary AR(1)
stationary <- simestim(rho=0.9,n=100,R=1000)
truehist(stationary[,1],main="Yule-Walker")
truehist(stationary[,2],main="OLS")
truehist(stationary[,3],main="ML")
```

```
#2) Random Walk, methods: yule-walker, ols, mle
randomwalk <- simestim(rho=1,n=100,R=1000)
truehist(randomwalk[,1],main="Yule-Walker")
truehist(randomwalk[,2],main="OLS")
truehist(randomwalk[,3],main="ML")

#3) Explosive AR(1)
explosive <- simestim(rho=1.01,n=100,R=1000)
truehist(explosive[,1],main="Yule-Walker")
truehist(explosive[,2],main="OLS")
truehist(explosive[,3],main="ML")


#4) Estimation by indirect inference, auxiliary model is an AR(1)-process
H <- 100; n <- 100; W <- diag(1)

rhohat <- function(truedata,H,n,W){
  thetahat <- ar(truedata,aic=F,order = 1)$ar #auxiliary model with true data
  thetahat <- as.matrix(thetahat)             #as matrix so you can compute Q

  f <- function(rho){
    thetahatsim <- rep(NA,H)
    set.seed(123)
    for (h in 1:H){
      simdata <- filter(rnorm(n),rho,method="r",init=rnorm(1)) #simulated data depending on beta
      thetahatsim[h] <- ar(simdata,aic=F,order = 1)$ar          #store estimator
    }
    thetatilde <- mean(thetahatsim)
    Q <- t(thetahat - thetatilde) %*% W %*% (thetahat - thetatilde)
    return(Q)
  }
  # indirect inference estimator for rho
  rhohat <- optimize(f,lower=0.7,upper=1.5)
  return(rhohat$minimum)
}

truedata <- filter(rnorm(n),0.9,method="r",init=rnorm(1)) #true model with rho=0.8

rhohat(truedata,H=10,n=100,W=diag(1))
ar(truedata,aic=F,order = 1,method="ols")$ar #method: ols
ar(truedata,aic=F,order = 1,method="yw")$ar #method: yw
ar(truedata,aic=F,order = 1,method="mle")$ar #method: mle
```

15.2 Filter models using the Kalman filter

Consider the univariate dynamic linear model

$$\begin{aligned} y_t &= \theta_t + v_t, \qquad v_t \sim N(0, V) \\ \theta_t &= \theta_{t-1} + w_t, \qquad w_t \sim N(0, W) \\ \theta_0 &\sim N(m_0, C_0) \end{aligned}$$

where only y_t is observed, but we are interested in the unobservable state variable θ_t. This model is sometimes called random walk plus noise. In R, the package `dlm` provides commands to deal with dynamic linear models. The notation in this exercise is adapted to the `dlm` package.

1. Install and activate the package `dlm`.
2. Define a `dlm`-object `mod <- dlm(FF=1,GG=1,V=9,W=1,m0=0,C0=100)`. This object represents the random walk plus noise model with known parameters V and W (and m_0 and C_0).
3. Load the dataset `rwnoise.csv` and plot the variable `y`.
4. Add the Kalman filtered estimated state variable $\hat{\theta}_t$ to the plot. The Kalman filtered series can very easily be computed by the command `dlmFilter(y,mod)$m[-1]`.
5. In general, the model parameters V and W are unknown. Estimate V and W by indirect inference. The auxiliary parameters are the $MA(1)$ parameter and the error term variance of an $ARIMA(0,1,1)$ model[15] (hence, the model is exactly identified).

15.2 Filter models

15.3 Estimation of the Cox-Ingersoll-Ross model

Cox, Ingersoll and Ross, "A Theory of the Term Structure of Interest Rates", *Econometrica* 53 (1985) 385-407, suggest a continuous-time model for the short-term interest rate. The stochastic process is described by the stochastic differential equation

$$dX_t = (\theta_1 - \theta_2 X_t)\, dt + \theta_3 \sqrt{X_t} dW_t \tag{4}$$

where W_t is a standard Wiener process and $X_0 > 0$.

1. Activate the R package `sde`. Generate and plot a single path on the time interval $[0, 200]$ of an Cox-Ingersoll-Ross process with parameters $\theta_1 = 0.03$, $\theta_2 = 0.5$, and $\theta_3 = 0.08$ and starting value $X_0 = 0.06$ using the command `sde.sim`. Set the number of steps to $N = 200$.
2. Continuous-time models are sometimes estimated by discretizing them in a crude way. The discretized version of (4) is, of course,

$$X_t = X_{t-1} + (\theta_1 - \theta_2 X_{t-1}) + \theta_3 \sqrt{X_{t-1}} \varepsilon_t \tag{5}$$

   with $\varepsilon_t \sim N(0, 1)$ and starting value $X_0 = \theta_1/\theta_2$. Find estimators for the parameters $\theta_1, \theta_2, \theta_3$ in (5) that can be computed fast (e.g. least squares estimators).
3. Load the dataset `cirpath.csv`. The process is not observed continuously. The dataset only contains observations of X_t at discrete time points $t = 1, \ldots, 200$. Estimate the parameters θ_1, θ_2, and θ_3 by indirect inference with the auxiliary model (5). Assume that the (unobserved) starting value is $X_0 = \theta_1/\theta_2$.

15.3 Estimation of the Cox-Ingersoll-Ross model

Estimators:

[15] To estimate an $ARIMA(p, d, q)$ model in R, use the command `a <- arima(x,order=c(p,d,q))`. The coefficients can be extracted by `a$coef` and error term variance is `a$sigma2`.

- **OLS**: We first need to transform the model variables in (5) to:

$$Y_t := \frac{X_t}{\sqrt{X_{t-1}}}, \qquad Z_t := \sqrt{X_{t-1}}, \qquad V_t := \frac{1}{\sqrt{X_{t-1}}}$$

  Then

$$Y_t = \theta_1 V_t + (1-\theta_2)Z_t + \theta_3\varepsilon_t$$

  This model is linear, the parameters are constant and $\theta_3\varepsilon_t \overset{iid}{\sim} N(0,\theta_3^2)$, the exogenous variables and the error terms are uncorrelated and there is no multicollinearity. Hence the OLS estimator is an unbiased and consistent estimator.

- **GMM**: As there are three parameters to estimate, at least three moment conditions are needed. For instance:

$$\begin{aligned} E(X_t - X_{t-1} - \theta_1 + \theta_2 X_{t-1}) &= 0 \\ E(X_t(X_t - X_{t-1} - \theta_1 + \theta_2 X_{t-1})) &= 0 \\ E\left(\left[\frac{X_t - X_{t-1} - \theta_1 + \theta_2 X_{t-1}}{\theta_3\sqrt{X_{t-1}}}\right]^2 - 1\right) &= 0 \end{aligned}$$

  since $\theta_3\sqrt{X_{t-1}}\varepsilon_t$ has expectation zero, X_t and ε_t are independent, and ε_t has variance of one.

```
####################################################
#### Estimation of the Cox-Ingersoll-Ross modell####
####################################################
library(gmm)
#since the package sde has its own gmm command,
#save the gmm-command from the gmm package
gmm1 <- gmm
library(sde)

#1) single path of an Cox-Ingersoll-Ross process

tet1 <- 0.03;  tet2 <- 0.5; tet3 <- 0.08
simdat <- sde.sim(t0=0,T=200,X0=tet1/tet2,N=200,theta=c(tet1,tet2,tet3),model="CIR")
plot(simdat,main="Cox-Ingersoll-Ross")  # process seems to be stationary
#add the longrun mean
abline(a=(tet1/tet2), b=0, col="red")
if (2*tet1 > tet3^2)  print("stationary") else print("not stationary")
#the condition 2*theta1 > theta3^2 is fulfilled, process is stationary


#2) Estimation of auxiliary AR(1)-model: Define Functions for OLS-model and GMM-model

# Auxiliary model with OLS
AuxOLS <- function(dat,startval){
  #transform the data to get a linear model
  z <- sqrt(dat[1:(length(dat)-1)])
  v <- 1/z
  y <- dat[2:length(dat)]/sqrt(dat[1:(length(dat)-1)])
  #calculate the estimate:
  ols <- lm(y~v+z -1)
  theta1hat<-coefficients(ols)[1]
  theta2hat<-(1-coefficients(ols)[2])
  theta3hat<-sqrt(var(residuals(ols)))
  thetahat <- rbind(theta1hat,theta2hat,theta3hat)
  colnames(thetahat) <- "Discrete OLS Estimate"
  rownames(thetahat) <- c("theta1","theta2","theta3")
  return(thetahat) #returns a 3x1 matrix
}

# Auxiliary model with GMM
AuxGMM <- function(dat,startval){
  # define moment functions
  g <- function(theta, X){
    TT <- length(X)
    u <- X[-TT] -theta[1] - (1-theta[2])*X[-1]
    m1 <- u
    m2 <- u*X[-1]
    m3 <- u^2 -theta[3]^2*X[-1]
    m4 <- (u^2 -theta[3]^2*X[-1])*X[-1]
    f <- cbind(m1,m2,m3,m4)
    return(f)
  }
  # estimate using the gmm package
  estim <- gmm1(g,dat,startval)
  thetahat <- cbind(estim$coefficients)
  colnames(thetahat) <- "Discrete GMM Estimate"
  rownames(thetahat) <- c("theta1","theta2","theta3")
  return(thetahat) #returns a 3x1 matrix
}

# Compare estimates
AuxOLS(dat=simdat,startval=c(0.03,0.5,0.08)) #ols
AuxGMM(dat=simdat,startval=c(0.03,0.5,0.08)) #gmm
AuxGMM(dat=simdat,startval=c(0.01,0.1,0.01)) #start values don't matter much


#3) Indirect inference estimation

# Function that computes the Indirect Inference estimator given
#the data, steps, matrix of weights, starting values, auxiliary model and optim or constrOptim
CIRIndirect <- function(truedata,H,W,startval,GMM=F,fast=F){
  if (GMM==F) AuxMod = AuxOLS else AuxMod = AuxGMM
  thetahat <- AuxMod(dat=truedata,startval=startval) #auxiliary model with true data

  f <- function(theta){
    thetahatsim <- matrix(NA,H,3)
    set.seed(123)
    for (h in 1:H){
      theta1 <- theta[1]; theta2 <- theta[2]; theta3 <- theta[3]
      simdata <- sde.sim(t0=0,T=200,X0=theta1/theta2,N=200,theta=c(theta1,theta2,theta3),
                              model="CIR") #simulate data depending on theta
      thetahatsim[h,] <- t(AuxMod(simdata,startval))
      #AuxMod returns a 3x1 matrix, thus transpose it to store in the matrix
    }
    thetatilde <- colMeans(thetahatsim)
    Q <- t(thetahat - thetatilde) %*% W %*% (thetahat - thetatilde)
    return(Q)
  }

  # Find the optimal theta
  if (fast==F) {
    res <- constrOptim(startval,f,ui=rbind(c(1,0,0),c(0,-1,0),c(0,0,1)),ci=c(0,-2,0),
                                                                method="Nelder-Mead")
  }else {
    # optim command using bounds, (fast)
    res <- optim(startval,f,lower=c(0.01,0.01,0.01),upper=c(Inf,2,Inf),method="L-BFGS-B")
  }

  #print the output depending on which model was used
  if (GMM==F) {
    result <- rbind(res$par); rownames(result) <- "Indirect Inference (OLS)";
    colnames(result) <- c("theta1","theta2","theta3")
  }else {
    result <- rbind(res$par); rownames(result) <- "Indirect Inference (GMM)";
    colnames(result) <- c("theta1","theta2","theta3")
  }

  return(result)
}

# Estimation for dataset cirpath.csv
cirpath <- read.csv(file.choose(),header=T)
plot(cirpath$x,type="l") #seems to be stationary

#fast - using bounded optim
resultOLS <- CIRIndirect(truedata=cirpath$x,H=10,W=diag(3),startval=c(0.03,0.5,0.08),
                                                                     GMM=F,fast=T)
print(resultOLS)
resultOLS1 <- CIRIndirect(truedata=cirpath$x,H=10,W=diag(3),startval=c(0.02,0.4,0.07),
                                                                     GMM=F,fast=T)
print(resultOLS1)
resultGMM <- CIRIndirect(truedata=cirpath$x,H=10,W=diag(3),startval=c(0.03,0.5,0.08),
                                                                     GMM=T,fast=T)
print(resultGMM)
resultGMM1 <- CIRIndirect(truedata=cirpath$x,H=10,W=diag(3),startval=c(0.02,0.4,0.07),
                                                                     GMM=T,fast=T)
print(resultGMM1)

#slow - using constrOptim
resultOLSslow <- CIRIndirect(truedata=cirpath$x,H=10,W=diag(3),startval=c(0.03,0.5,0.08),
                                                                     GMM=F,fast=F) #slow
print(resultOLSslow)
```

```
resultGMMslow <- CIRIndirect(truedata=cirpath$x,H=10,W=diag(3),startval=c(0.03,0.5,0.08),
                                                                 GMM=T,fast=F) #very slow
print(resultGMMslow)


# Test the estimation procedure with the simulated dataset simdat
tet1 <- 0.03;  tet2 <- 0.5; tet3 <- 0.08 #stationary process
simdat <- sde.sim(t0=0,T=200,X0=tet1/tet2,N=200,theta=c(tet1,tet2,tet3),model="CIR")

r1 <- CIRIndirect(truedata=simdat,H=10,W=diag(3),startval=c(0.03,0.5,0.08),GMM=F,fast=T)
r2 <- CIRIndirect(truedata=simdat,H=10,W=diag(3),startval=c(0.03,0.5,0.08),GMM=F,fast=F)
r3 <- CIRIndirect(truedata=simdat,H=10,W=diag(3),startval=c(0.03,0.5,0.08),GMM=T,fast=T)
r4 <- CIRIndirect(truedata=simdat,H=10,W=diag(3),startval=c(0.03,0.5,0.08),GMM=T,fast=F)
r5 <- CIRIndirect(truedata=simdat,H=10,W=diag(3),startval=c(0.02,0.4,0.07),GMM=T,fast=T)
```

15.4 Ornstein-Uhlenbeck process

Consider the continuous-time stochastic process described by the stochastic differential equation

$$dX_t = \lambda\left(\mu - X_t\right)dt + \sigma dW_t \tag{6}$$

where W_t is a standard Wiener process, $\lambda > 0$ is a parameter of the strength of mean-reversion, μ is the long-run mean, and $\sigma > 0$ is a volatility parameter.

1. Install and activate the R package `sde`. It provides commands to simulate paths of stochastic processes described by stochastic differential equations. Generate and plot a single path on the time interval $[0, 100]$ of an Ornstein-Uhlenbeck process with parameters $\lambda = 0.9$, $\mu = 0$, and $\sigma = 1$ and starting value $X_0 = 2$ using the command `sde.sim`. Note that the parametrization of the `sde` command differs from (6) with $\theta_1 = \lambda\mu$, $\theta_2 = \lambda$, and $\theta_3 = \sigma$. Set the number of steps to $N = 100$.
2. Continuous-time models are sometimes estimated by discretizing them in a rough way. The discretized version of (6) is, of course, $X_t - X_{t-1} = \lambda\left(\mu - X_{t-1}\right) + \sigma\varepsilon_t$ with $\varepsilon_t \sim N(0,1)$ and starting value $X_0 = \mu$. Rewriting gives

   $$X_t = \lambda\mu + (1-\lambda)X_{t-1} + u_t$$

   with $u_t \sim N(0, \sigma^2)$. This exercise shows that simply estimating the discrete model can be severely misleading!

   For this create an empty vector `Z` of length $R = 1000$. Write a loop over $r = 1, \ldots, R$ performing the following steps for each replication.

   - Generate a path of the Ornstein-Uhlenbeck process `x` given in exercise 1.
   - Fit an $AR(1)$ process to the path using the command `ar(x,order=1,aic=F)` or, alternatively, the command `arima(x,order=c(1,0,0))`. Both commands estimate (5), but only `arima` reports the estimated intercept.
   - Save the AR coefficient in `Z[r]`. The AR coefficient is the estimate of $1 - \lambda$.
   - After the loop, plot the histogram of `Z`. Comment on the distribution of $1 - \hat{\lambda}$.
3. Load the dataset `oupath.csv`. The process is not observed continuously. The dataset only contains observations of X_t at discrete time points $t = 1, \ldots, 100$. Estimate the parameters λ, μ, and σ by indirect inference with the auxiliary model (5). Assume that the (unobserved) starting value is $X_0 = \mu$.

15.4 Ornstein-Uhlenbeck process

The code might look like this

```
########################################################
#### Indirect Inference - Ornstein-Uhlenbeck process ####
########################################################

#1) Single path of an Ornstein-Uhlenbeck process
install.packages("sde")
library(sde)
?sde.sim #look at the example
# Ornstein-Uhlenbeck
drift <- expression(0.9*0 - 0.9 * x)
sigma <- expression(1)
X <- sde.sim(X0=2,drift=drift, sigma=sigma,N=100,T=100)
plot(X,main="Ornstein-Uhlenbeck")
# or specify model to generate data
mu <- 0;  lambda <- 0.9; sigma <- 1
X <- sde.sim(t0=0,T=100,X0=2, N=100,theta=c(lambda*mu,lambda,sigma),model="OU")
plot(X,main="Ornstein-Uhlenbeck")

#2) Estimation of discrete model
library(AER)
R <- 1000
Z <- rep(NA,R)
mu <- 0;  lambda <- 0.9; sigma <- 1

for (r in 1:R){
  X <- sde.sim(t0=0,T=100,X0=2, N=100,theta=c(lambda*mu,lambda,sigma),model="OU")
  estim <- arima(X,order=c(1,0,0))
  Z[r] <- estim$coef[1]
}

truehist(Z) #histogram of 1-lambda, totally wrong
truehist(1-Z) #histogram of lambda, totally wrong

#3) Indirect inference estimation
oupath <- read.table(file.choose(), header=T)
truedata <- oupath$x
plot(truedata,type="l")
#Hint: The data generating process uses lambda=1.3; mu=9; sigma=3; T=N<-100

OUIndirect <- function(truedata,H,W,startval){
  estim <- arima(truedata,order=c(1,0,0)) #auxiliary model with true data
  lambdahat <- 1-estim$coef[1]
  muhat <- estim$coef[2]/lambdahat
  sigmahat <- sqrt(estim$sigma2)
  thetahat <- rbind(lambdahat,muhat,sigmahat)

  f <- function(theta){
    thetahatsim <- matrix(NA,H,3)
    set.seed(123)
```

```
      for (h in 1:H){
        lambda <- theta[1]; mu <- theta[2]; sigma <- theta[3]
        #simulated data depending on theta
        simdata <- sde.sim(t0=0,T=100,X0=mu, N=100,theta=c(lambda*mu,lambda,sigma),model="OU")
        estimsim <- arima(simdata,order=c(1,0,0))         #store estimator
        lambdahatsim <- 1-estimsim$coef[1]
        muhatsim <- estimsim$coef[2]/lambdahatsim
        sigmahatsim <- estimsim$sigma2
        thetahatsim[h,1] <- lambdahatsim
        thetahatsim[h,2] <- muhatsim
        thetahatsim[h,3] <- sigmahatsim
      }
      thetatilde <- colMeans(thetahatsim)
      Q <- t(thetahat - thetatilde) %*% W %*% (thetahat - thetatilde)
      return(Q)
    }
    # optimize ove all thetas
    res <- optim(startval,f)
    return(res$par)
  }

  OUIndirect(truedata=truedata,H=10,W=diag(3),startval=c(1.3,9,3))
```

15.5 Time-aggregated observations

Consider the geometric Brownian motion described by the stochastic differential equation

$$dX_t = \mu X_t dt + \sigma X_t dW_t$$

where W_t is a standard Wiener process, μ is the drift parameter and $\sigma > 0$ is the volatility parameter, and the starting value is $X_0 = 100$. Suppose the process X_t is not observed continuously. The only observations are the time-aggregates

$$Y_t = \int_{t-1}^{t} X_t dt \qquad (7)$$

for $t = 1, \ldots, T$. The Y_t could be interpreted as average stock prices over time intervals $[t-1, t]$, that are relevant for Asian option pricing. This exercise explores how to estimate μ and σ from observations $Y_1, \ldots, Y_T$ by indirect inference.

1. Install and activate the R package `sde`. It provides commands to simulate paths of Brownian motion (`BM`) and geometric Brownian motions (`GBM`). Generate and plot a single path of the geometric Brownian motion X_t with $\mu = 0.00025$ and $\sigma = 0.015$ (these values are more or less realistic for daily stock returns), and starting values $X_0 = 1$, on the time interval $[0, T]$ with $T = 30$. Let the number of steps be $N = 3000$, i.e. 100 steps per period.

2. The time integrals in (7) can be approximated by the sums

$$Y_t \approx \sum_i X_i \cdot \Delta$$

where the sum is over all i between $t-1$ and t, and $\Delta = T/N = 0.01$ is the interval length. Create an empty matrix `Z` of dimensions $R \times 4$ with $R = 1000$. Write a loop over $r = 1, \ldots, R$ performing the following steps for each replication.

- Generate a path of the geometric Brownian motion X_t given in exercise 1.
- Calculate the four integrals Y_1, Y_2, Y_{15}, Y_{30} and save them in row r of the matrix `Z`.
- After the loop, calculate the means for each column and the variance-covariance matrix of `Z` (using the command `cov`).

3. Load the dataset `timeaggr.csv`. It contains 30 time-aggregated observations $Y_1, \ldots, Y_{30}$. Estimate μ and σ by indirect inference. As auxiliary model use an $ARIMA(1,0,1)$ process with intercept. Also include the error term variance of the $ARIMA$ model in your estimation.[16] Assume that the starting value is $X_0 = 1$.

15.5 Time-aggregated observations

[16] To estimate an $ARIMA(p,d,q)$ model in R, use the command `a <- arima(x,order=c(p,d,q))`. The coefficients can be extracted by `a$coef` and error term variance is `a$sigma2`.

16 Bootstrap

16.1 Omitted variables bias does not go away

This exercise shows that the bootstrap does not help to eliminate omitted variable bias. Reconsider exercise 7.4.

1. Load the dataset `omitted.csv` and estimate the model without the relevant variable X_4 by OLS.
2. Bootstrap the bias and standard error of the coefficients of X_2 and X_3. Set the number of bootstrap replications to $B = 5000$.

16.1 Omitted variables bias does not go away

The code might look like this:

```
###################################################
#### Omitted variables bias does not go away ####
###################################################
#1)
#Input data and make it accessible
omitted <- read.csv(file.choose(), sep=";", dec=",")
x1 <- omitted$x1
x2 <- omitted$x2
x3 <- omitted$x3
x4 <- omitted$x4
y <- omitted$y
n <- length(y)
#Estimate by OLS without x4
#Reminder: the true model is Y = 1 + 2X1 + 3X2 + 4X3 + 5X4
# X1 is uncorrelated, X2&X3 are correlated and X3&X4, X2&X4 are uncorrelated
mod <- lm(y~x1+x2+x3)
summary(mod) #only the uncorrelated variable x1 is estimated well

#2)
### 1.Alternative: Bootstrap the residuals
uhat <- mod$residuals
ahat <- coefficients(mod)[1]
beta1hat <- coefficients(mod)[2]
beta2hat <- coefficients(mod)[3]
beta3hat <- coefficients(mod)[4]

B <- 1000
betastar <- matrix(NA,B,2)
SEbetastar <- matrix(NA,B,2)
for(b in 1:B) {
  ustar <- sample(uhat,n,replace=TRUE)
  ystar <- ahat+beta1hat*x1+beta2hat*x2+beta3hat*x3+ustar
  bootmod <- lm(ystar~x1+x2+x3)
  betastar[b,1] <- coefficients(bootmod)[3] #coef for x2
  betastar[b,2] <- coefficients(bootmod)[4] #coef for x3
  SEbetastar[b,1] <- sqrt(vcov(bootmod)[3,3]) #SE for x2
  SEbetastar[b,2] <- sqrt(vcov(bootmod)[4,4]) #SE for x3
}

# Calculate the bias
print(colMeans(betastar)-c(beta2hat,beta3hat)) #omitted variable bias does not go away
colMeans(SEbetastar)

### 2. Alternative: Bootstrap of the observations
B <- 1000
betastar <- matrix(NA,B,2)
SEbetastar <- matrix(NA,B,2)
for(b in 1:B) {
  indices <- sample(1:n,n,replace=TRUE)
  x1star <- x1[indices]
  x2star <- x2[indices]
  x3star <- x3[indices]
  ystar <- y[indices]
  bootmod <- lm(ystar~x1star+x2star+x3star)
  betastar[b,1] <- coefficients(bootmod)[3] #coef for x2
  betastar[b,2] <- coefficients(bootmod)[4] #coef for x3
  SEbetastar[b,1] <- sqrt(vcov(bootmod)[3,3]) #SE for x2
  SEbetastar[b,2] <- sqrt(vcov(bootmod)[4,4]) #SE for x3
}

# Calculate the bias
print(colMeans(betastar)-c(beta2hat,beta3hat)) #omitted variable bias does not go away
colMeans(SEbetastar)
```

16.2 Confidence intervals for the Gini index

Install and activate the package ineq. It provides functions for inequality measures and concentration measures as well as Lorenz curves.

1. Load the dataset earnings.csv. It contains earnings of 11648 individuals. Compute the Gini coefficient of earnings.
2. Bootstrap the standard error of the Gini coefficient.
3. Compute the bootstrap 0.95-confidence interval for the Gini coefficient using the percentile method.

16.2 Confidence intervals for the Gini index

The program could look like this:

```
################################################
#### Confidence intervals for the Gini index ####
################################################
#1)
install.packages("ineq")
library(ineq)
```

```
earnings <- read.csv(file.choose(), header=T)
x <- earnings$x
?ineq
ineq(x,type="Gini")
Ginihat <- Gini(x)
print(Ginihat)
#2)
n <- length(x)
B <- 1000
Ginistar <- rep(NA,B)

for(b in 1:B) {
  # Draw a resample
  xx <- sample(x,n,replace=TRUE)
  # Compute and save the Gini-coefficient for the resample
  Ginistar[b] <- Gini(xx)
}

# Compute the standard error
print(sd(Ginistar))

#3)
# Sort Ginistar
Ginistar <- sort(Ginistar)

print(paste("Lower limit = ",2*Ginihat-Ginistar[0.975*B]))
print(paste("Upper limit = ",2*Ginihat-Ginistar[0.025*B]))
```

16.3 Confidence intervals for correlation coefficients

The distribution of the empirical correlation coefficient

$$\hat{\rho} = \frac{\sum_{i=1}^{n}\left(X_i - \bar{X}\right)\left(Y_i - \bar{Y}\right)}{\sqrt{\sum_{i=1}^{n}\left(X_i - \bar{X}\right)^2}\sqrt{\sum_{i=1}^{n}\left(Y_i - \bar{Y}\right)^2}}$$

is rather complicated except for some special cases. Of course, being based on moments, $\hat{\rho}$ is asymptotically normally distributed (if the relevant moments exist). However, in small samples, confidence intervals for ρ are not trivial.

1. Install and activate the package `copula`. It provides functions and commands to deal with copulas. Generate a single sample of size $n = 1000$ from (X, Y) by executing

   `x <- qexp(rcopula(gumbelCopula(1.3),1000)).`

   Row i of the $(n \times 2)$-matrix `x` is the pair (X_i, Y_i). Plot the sample and compute the correlation coefficient.

2. Simulate the distribution of $\hat{\rho}$ for sample size $n = 50$ and show that the distribution is not normal.

3. Draw a single sample of size $n = 50$. Compute the bootstrap 0.95-confidence interval for ρ using the percentile method with $B = 1000$ bootstrap replications. Check if the true value (about 0.43) is covered by the interval.

16.3 Confidence intervals for correlation coefficients

The code might look like this:

```
############################################################
#### Confidence intervals for correlation coefficients ####
############################################################
#1)
install.packages("copula")
library(copula)
x <- qexp(rcopula(gumbelCopula(1.3),1000))
plot(x)
cor(x)

#2)
library(MASS)
corvec <- rep(NA,1000)
for (k in 1:1000){
  x <- qexp(rcopula(gumbelCopula(1.3),50))
  corvec[k] <- cor(x)[1,2]
}

qqnorm(corvec)
qqline(corvec)
truehist(corvec)
curve(dnorm(x,mean=mean(corvec),sd=sd(corvec)),add=T)

#3)
library(AER)
library(MASS)
n <- 50
x <- qexp(rcopula(gumbelCopula(1.3),n))
corhat <- cor(x)[1,2]
B <- 5000
corstar <- rep(NA,B)

for(b in 1:B) {
  #Draw a resample
  indices <- sample(1:n,n,replace=T)
  xx <- x[indices,]
  corstar[b] <- cor(xx)[1,2]
}
# truehist(corstar)
# curve(dnorm(x,mean=mean(corstar),sd=sd(corstar)),add=T)
# the normal distribution does not fit

corstar <- sort(corstar)

print(paste("Lower limit = ",2*corhat-corstar[0.975*B]))
print(paste("Upper limit = ",2*corhat-corstar[0.025*B]))
#poor confidence intervals!
```

16.4 The t-test

This exercise shows that the ordinary t-test is a special case of the parametric bootstrap. Consider the simple linear regression model

$$y_t = \alpha + \beta x_t + u_t, \quad u_t \sim N(0, \sigma^2)$$

with $\alpha = 1$, $\beta = 0$ and $\sigma = 2$. Load the dataset `ttestboot.csv`. It contains the exogenous variable x and the endogenous variable y. The number of observations is $n = 9$. We want to test $H_0 : \beta = \beta_0 = 0$ against $H_1 : \beta \neq 0$.

1. Compute the ordinary OLS test statistic
$$\frac{\hat{\beta} - \beta_0}{SE(\hat{\beta})} \tag{8}$$
and the p-value of the test (both are printed by `summary` of the `lm` object).

2. The parametric bootstrap makes use of the fact that the distribution of u_t is known apart from the variance σ^2. Resamples can be generated by drawing new error terms from the normal distribution $N(0, \hat{\sigma}^2)$ where $\hat{\sigma}^2$ is an unbiased estimate of σ^2. Compute the estimates $\hat{\alpha}$, $\hat{\beta}$ and
$$\hat{\sigma}^2 = \frac{1}{7} \sum_{t=1}^{9} \hat{u}_t^2.$$
Use the function `residuals` to extract the residuals from the `lm` object, and the function `coefficients` to extract the parameter estimates $\hat{\alpha}$ and $\hat{\beta}$.

3. Prepare an empty vector `Z` of length $R = 5000$. Write a loop over $r = 1, \ldots, R$ performing the following steps:

   - Draw a random sample $u_1^*, \ldots, u_9^*$ from $N(0, \hat{\sigma}^2)$.
   - Compute
   $$y_t^* = \hat{\alpha} + \hat{\beta} x_t + u_t^*, \quad t = 1, \ldots, 9.$$
   - For the resample $(x_1, y_1^*), \ldots, (x_9, y_9^*)$, calculate the bootstrap test statistic
   $$\frac{\hat{\beta}_r^* - \hat{\beta}}{SE(\hat{\beta}_r^*)}$$
   and save it as `Z[r]`.

4. Plot the histogram of `Z` and add the density function of the t-distribution with 7 degrees of freedom.

5. Calculate the proportion of `abs(Z)` that is larger than the absolute value of the original test statistic (8). Compare your result with the p-value computed above.

16.4 The t-test

```
#####################
#### The t-test ####
####################
ttestboot <- read.csv(file.choose())
```

```
x <- ttestboot$x
y <- ttestboot$y
n <- length(y)
#Reminder: the data is generated by a=1, b=0, sigma=2

#1)
ols <- lm(y~x)
obj <- summary(ols)
tols <- obj$coefficients[2,3]
pols <- obj$coefficients[2,4]

#2)
alphahat <- coefficients(ols)[1]
betahat <- coefficients(ols)[2]
uhat <- residuals(ols)
sigma2hat <- 1/(n-2)*sum(uhat^2)
# alternativ:
# sigma2hat <- obj$sigma^2

#3)
R <- 5000
Z <- rep(NA,R)

for(r in 1:R){
  ustar <- rnorm(n,sd=sqrt(sigma2hat))
  ystar <- alphahat + betahat*x + ustar
  olsstar <- lm(ystar~x)
  betahatstar <- coefficients(olsstar)[2]
  SEbetahatstar <- summary(olsstar)$coefficients[2,2]
  Z[r] <- (betahatstar - betahat)/(SEbetahatstar)
}

#4)
library(MASS)
library(AER)
truehist(Z)
curve(dt(x,df=7),add=T) #the approximation fits perfectly

#5)
length(Z[abs(Z)>abs(tols)])/length(abs(Z))
pols
#p-value is almost the same!
```

16.5 The percentile-t-method

In the lecture, we considered the distribution of $\hat{\theta} - \theta$ and its bootstrap approximation $\theta^* - \hat{\theta}$ to determine confidence intervals. An asymptotically more efficient method is to consider the distribution of

$$\frac{\hat{\theta} - \theta}{SE(\hat{\theta})}$$

and its bootstrap approximation

$$\frac{\theta^* - \hat{\theta}}{SE(\theta^*)}.$$

Since these quantities look like t-statistics, this method is often called the percentile-t-method.

1. Start with
$$P\left(c_1 \leq \frac{\hat{\theta} - \theta}{SE(\hat{\theta})} \leq c_2\right) = 1 - \alpha$$
and derive an expression for the $(1 - \alpha)$-confidence interval.
2. Start with
$$P\left(c_1 \leq \frac{\theta^* - \hat{\theta}}{SE(\theta^*)} \leq c_2\right) = 1 - \alpha$$
and derive an expression for the bootstrap $(1 - \alpha)$-confidence interval.
3. Explain the algorithm to compute the bootstrap confidence interval.
4. Reconsider exercise 13.5. Write a program that computes the bootstrap 0.95-percentile-t-confidence interval for the coefficient of the interest rate r_t estimated with instruments r_{t-1} and r_{t-2} (as done in 13.5.2).

16.5 The percentile-t-method

The code might look like this:

```
##################################
#### The percentile-t-method ####
##################################
library(AER)
# read data
money <- read.csv(file.choose())
m <- money$m
r <- money$r
y <- money$y
TT <- length(m)

yt <- y[5:TT]
mt <- m[5:TT]; mt1 <- m[4:(TT-1)]; mt2 <- m[3:(TT-2)]
rt <- r[5:TT]; rt1 <- r[4:(TT-1)]; rt2 <- r[3:(TT-2)]

# Generalized IV estimation with confidence interval
IV <- ivreg(mt~rt+yt+mt1+mt2|rt1+rt2+yt+mt1+mt2)
summary(IV)
confint(IV,parm="rt",level=.95)

# Bootstrap 0.95-percentile-t condfidence intervals
# estimate for original data
rhat <- coefficients(IV)[2]
SErhat <- sqrt(vcov(IV)[2,2])

# Draw resamples
```

```
B <- 1000
Taustar <- rep(NA,B)
for(b in 1:B) {
  indices <- sample(1:TT,TT,replace=TRUE)
  mtstar <- mt[indices]
  rtstar <- rt[indices]
  ytstar <- yt[indices]
  mt1star <- mt1[indices]
  mt2star <- mt2[indices]
  rt1star <- rt1[indices]
  rt2star <- rt2[indices]
  bootmod <- ivreg(mtstar~rtstar+ytstar+mt1star+mt2star|rt1star+rt2star+ytstar+mt1star+mt2star)
  SErstar <- sqrt(vcov(bootmod)[2,2])
  rstar <- bootmod$coefficients[2]
  Taustar[b] <- (rstar-rhat)/SErstar
}

# Sort
Taustar <- sort(Taustar)

#Compute the interval
low <- rhat-Taustar[0.975*B]*SErhat
high <- rhat-Taustar[0.025*B]*SErhat
names(low)="2.5%"; names(high)="97.5%"
CI <- c(low,high)

print(CI) #CI using bootstrap
confint(IV,parm="rt",level=.95) #CI using GIV estimation
```

16.6 Heavy tails and variance testing

Let X and Y be independent random variables both having a t-distribution with 3 degrees of freedom (a plausible model for returns). Let $(X_1,Y_1),\dots,(X_n,Y_n)$ be a random sample of size $n=200$.

1. Use simulations to show that the F-test of the hypotheses

$$\begin{aligned} H_0 &: Var(X)=Var(Y) \\ H_1 &: Var(X)\neq Var(Y) \end{aligned}$$

   rejects the null hypothesis far too often at significance level $\alpha=0.05$. Hint: Tests for equal variances can be performed by the R command `var.test(x,y)$p.value`.

2. Implement a nonparametric Wald-type bootstrap test for equality of variances. Use the ordinary F-statistic as test statistic (`var.test(x,y)$statistic`).

3. Implement a nonparametric LM-type bootstrap test for equality of variances. Use the ordinary F-statistic as test statistic (`var.test(x,y)$statistic`).

4. If your time and computer power allow, do a Monte-Carlo simulation to show that the bootstrap tests keeps the prescribed level α more closely than the ordinary F-test, i.e. that the proportion of rejections of the true null hypothesis is closer to 5% of the replications. Note, however, that the rejection probability is still substantially too high for both variants of the bootstrap tests.

16.6
```
##########################################
#### Heavy Tails and Variance Testing ####
##########################################
#1)
simnum <- 1000
rej <- rep(NA,simnum)
alpha <- 0.05
n <- 200

for (k in 1:simnum){
  x <- rt(n,3)
  y <- rt(n,3)
  rej[k] <- as.numeric(var.test(x,y)$p.value<alpha)
}
mean(rej)

#2) Wald type Bootstrap
WaldBootstrap <- function(n,x,y,Fstat,mes){
  R <- 1000
  Fstar <- rep(NA,R)
  H0 <- 1 # The F-Statistic equals 1 under H0
  for(r in 1:R) {
    # Draw resample
    xx <- sample(x,n,replace=TRUE)
    yy <- sample(y,n,replace=TRUE)
    Fstar[r] <- var.test(xx,yy)$statistic/Fstat
  }

  # Sort Fstar and get confidence intervalls
  Fstar <- sort(Fstar)
  critlow <- Fstar[0.025*R]
  crithigh <- Fstar[0.975*R]
  if (mes == 1){
    if(Fstat/H0<critlow | Fstat/H0>crithigh)  print("Reject") else print("Do not reject")
  }
  if(Fstat/H0<critlow | Fstat/H0>crithigh) out<-1 else out<-0
  return(out)
}

#3) LM type Bootstrap:
LMBootstrap <- function(n,x,y,Fstat,mes){
  R <- 1000
  Fsharp <- rep(NA,R)
  H0 <- 1 # The F-Statistic equals 1 under H0
  # Standardize data to equal variances
  x2 <- x/sd(x)
  y2 <- y/sd(y)
  for(r in 1:R) {
    # Draw resample
    xx <- sample(x2,n,replace=TRUE)
    yy <- sample(y2,n,replace=TRUE)
    Fsharp[r] <- var.test(xx,yy)$statistic/H0
  }

  # Sort Tsharp and get confidence intervals
  Fsharp <- sort(Fsharp)
  critlow <- Fsharp[0.025*R]
  crithigh <- Fsharp[0.975*R]
  if (mes == 1){
    if(Fstat/H0<critlow | Fstat/H0>crithigh) print("Reject H0") else print("Do not reject H0")
  }
  if(Fstat/H0<critlow | Fstat/H0>crithigh) out<-1 else out<-0
  return(out)
}

#4)
reps <- 1000
rej <- rep(NA,reps)
rejWald <- rep(NA,reps)
rejLM <- rep(NA,reps)
n <- 200

pb <- txtProgressBar(min=0,max=reps,style=3)
for (k in 1:reps){
  x <- rt(n,3)
  y <- rt(n,3)
  Fstat <- var.test(x,y)$statistic
  rej[k] <- as.numeric(var.test(x,y)$p.value<0.05)
  rejWald[k] <- WaldBootstrap(n,x,y,Fstat,0)
  rejLM[k] <- LMBootstrap(n,x,y,Fstat,0)
  setTxtProgressBar(pb,k)
}
close(pb)
print(mean(rejWald))
print(mean(rejLM))
print(mean(rej))
```

16.7 Time series

This exercise shows in a simple setting how to bootstrap time series. In general, this approach works well if there is a parametric time series model based on an underlying white noise process (e.g. GARCH, ARIMA, VAR). Consider the simple $AR(1)$ model with intercept

$$(x_t - \mu) = \rho (x_{t-1} - \mu) + \varepsilon_t \tag{9}$$

with $\varepsilon_t \sim N(0, \sigma^2)$.

1. Load the dataset `ar1bsp.csv`. Estimate the parameters μ and ρ using the `arima` command. The standard errors that are reported are asymptotically valid.
2. Show that bootstrapping the standard errors of $\hat{\mu}$ and $\hat{\rho}$ does not work under the ordinary bootstrap resampling scheme, i.e. drawing $x_1^*, \ldots, x_T^*$ from $x_1, \ldots, x_T$ with replacement.
3. Program the following time series bootstrap approach. Estimate the model (9) for the original sample $x_1, \ldots, x_T$ and save the parameter estimates $\hat{\mu}$ and $\hat{\rho}$ and the residuals $\hat{\varepsilon}_1, \ldots, \hat{\varepsilon}_T$. Initialize an empty $(R \times 2)$-matrix `Z` for, say, $R = 1000$. For $r = 1, \ldots, R$:

- Draw a resample $\varepsilon_1^*, \dots, \varepsilon_T^*$ from $\hat{\varepsilon}_1, \dots, \hat{\varepsilon}_T$.
- Set $x_1^* = x_1$ and compute $x_t^* = \hat{\mu} + \hat{\rho}\left(x_{t-1}^* - \hat{\mu}\right) + \varepsilon_t^*$ for $t = 2, \dots, T$.
- Estimate (9) for $x_1^*, \dots, x_T^*$ and save the estimates μ^* and ρ^* in row r of `Z`.

Compute the standard errors for both columns of `Z` and compare them to exercise 1.

16.7 Time Series

The code might look like this:

```
#####################
#### Time Series ####
#####################
#1)
ar1bsp <- read.csv(file.choose())
x <- ar1bsp$x
n <- length(x)
estimate <- arima(x,order=c(1,0,0),include.mean=TRUE)
rhohat <- estimate$coef[1]
muhat <- estimate$coef[2]

#2) iid bootstrap
R <- 1000
rhostar1 <- rep(NA,R)
mustar1 <- rep(NA,R)
for(r in 1:R)  {
  xstar1 <- sample(x,n,replace=TRUE)
  estimate1 <- arima(xstar1,order=c(1,0,0),include.mean=TRUE)
  rhostar1[r] <- estimate1$coef[1]
  mustar1[r] <- estimate1$coef[2]
}

rhostar1 <- sort(rhostar1)
mustar1 <- sort(mustar1)

print("iid-Bootstrap")
print(paste("Wahrer Wert rho = 0.9"))
print(paste("rhohat       = ",rhohat))
print(paste("Untergrenze = ",2*rhohat-rhostar1[0.975*R]))
print(paste("Obergrenze  = ",2*rhohat-rhostar1[0.025*R]))

print(paste("Wahrer Wert mu = 5"))
print(paste("muhat        = ",muhat))
print(paste("Untergrenze = ",2*muhat-mustar1[0.975*R]))
print(paste("Obergrenze  = ",2*muhat-mustar1[0.025*R]))

# Wahre Werte liegen nicht im Konfidenzintervall


#3) Zeitreihen Bootstrap
eps_hat <- estimate$residuals #original residuals
R <- 1000
```

```
rhostar2 <- rep(NA,R)
mustar2 <- rep(NA,R)

for (r in 1:R){
  indices <- sample(1:n,n,replace=TRUE)
  eps_star <- eps_hat[indices]
  xstar2 <- rep(NA,n)
  xstar2[1] <- x[1]
  for (k in 2:n){
    xstar2[k] <- muhat + rhohat*(xstar2[k-1] - muhat) + eps_star[k]
  }
  estimate2 <- arima(xstar2,order=c(1,0,0),include.mean=TRUE)
  rhostar2[r] <- estimate2$coef[1]
  mustar2[r] <- estimate2$coef[2]
}

rhostar2 <- sort(rhostar2)
mustar2 <- sort(mustar2)

print("Zeitreihen-Bootstrap")
print(paste("Wahrer Wert rho = 0.9"))
print(paste("rhohat       = ",rhohat))
print(paste("Untergrenze = ",2*rhohat-rhostar2[0.975*R]))
print(paste("Obergrenze  = ",2*rhohat-rhostar2[0.025*R]))

print(paste("Wahrer Wert mu = 5"))
print(paste("muhat       = ",muhat))
print(paste("Untergrenze = ",2*muhat-mustar2[0.975*R]))
print(paste("Obergrenze  = ",2*muhat-mustar2[0.025*R]))

# Wahre Werte liegen im Konfidenzintervall
```

16.8 Bootstrap test for the Zipf index of city size distributions

It is well known that the population size distribution of large cities can be approximated by the Zipf distribution which is a special case of the Pareto distribution with tail index $\alpha = 1$. Suppose, $X_1 \geq X_2 \geq \ldots \geq X_n$ is a descendingly ordered sample of city sizes. In regional economics, the tail index α is often estimated from the regression

$$\ln(i) = c - \alpha \ln X_i + u_i$$

where i is the rank of the city and X_i its size, the intercept parameter c is of no interest. Since the sample is ordered, the observations are no longer independent and the optimality properties of OLS vanish. In particular, the ordinary t-test does not work correctly anymore. In this exercise, ordered samples from the Zipf distribution (i.e. from the Pareto distribution with true tail index $\alpha = 1$) are generated by the command

```
x <- sort(exp(rexp(n)),decreasing=TRUE)
```

where `n` is the sample size.

1. Simulate and plot the distribution of $\hat{\alpha}$ for sample size $n = 20$. The regression can be performed by `obj <- lm(log(1:n)~log(x))`. The estimates can then be extracted by the function `coefficients(obj)`.

2. An important hypothesis is $H_0 : \alpha = 1$ against $H_1 : \alpha \neq 1$. Simulate and plot the distribution of the test statistic
$$T = \frac{\hat{\alpha} - 1}{SE(\hat{\alpha})}$$
and show that it is not t_{n-2}-distributed even though H_0 is true.
3. Explain why the simulations done in 1. and 2. can be used to find the critical values of a parametric LM-type bootstrap test of $H_0 : \alpha = 1$.

16.8 Bootstrap test for the Zipf index of city size distributions

The code might look like this:

```
######################################################################
#### Bootstrap test for the Zipf index of city size distributions ####
######################################################################
#1)
library(AER)
n <- 20
R <- 1000
y <- 1:n
Z <- rep(NA,R)

for (r in 1:R) {
  x <- sort(exp(rexp(n)),decreasing=TRUE)
  obj <- lm(log(y) ~ log(x))
  Z[r] <- coefficients(obj)[2]
}
truehist(Z)

#2)
n <- 20
R <- 1000
y <- 1:n
TT <- rep(NA,R)
for (r in 1:R) {
  x <- sort(exp(rexp(n)),decreasing=TRUE)
  obj <- lm(log(y) ~ log(x))
 TT[r] <- (coefficients(obj)[2]-1)/ sqrt(vcov(obj)[2,2])
}
truehist(TT)
curve(dt(x,df=n-2),add=T)

#3)
# True data
n <- 20
alpha <- 1 #true value
x <- sort(exp(rexp(n,rate=alpha)),decreasing=TRUE)
y <- 1:n

# Hypothetical value (here it is also the true value), Nullhypothesis
alpha0 <- 1
```

```
# Compute the test statistics for the true data
obj <- lm(log(y)~log(x))
alphahat <- coefficients(obj)[2]
SEalphahat <- sqrt(vcov(obj)[2,2])
Tstat <- (alphahat-alpha0)/SEalphahat #test statistic for true data

# Approximate distribution through bootstrap
B <- 1000
Tsharp <- rep(NA,B)

for(b in 1:B) {
  #draw a resample taking into account the nullhypothesis, alpha=alpha0=1
  #here it is also the true value
  xx <- sort(exp(rexp(n,rate=alpha0)),decreasing=TRUE)
  obj <- lm(log(y)~log(xx))
  alphasharp <- coefficients(obj)[2]
  SEalphasharp <- sqrt(vcov(obj)[2,2])
  Tsharp[b] <- (alphasharp-alpha0)/SEalphasharp
  }

# Sort the Tsharp values
Tsharp <- sort(Tsharp)
truehist(Tsharp)
critlow <- Tsharp[0.025*B]
crithigh <- Tsharp[0.975*B]

if(Tstat<critlow | Tstat>crithigh)
 print("Reject H0") else
 print("Don't reject H0")
```

Please note that there are better ways to remedy the failure of standard OLS than using the bootstrap. In particular, extreme value theory provides many useful tools for estimation and testing.