Statistical Inference — Solutions Book (Version: February 21, 2012)

Mark Trede and Willi Mutschler

Winter 2011/2012

Solutions

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. If you need assistance installing those editors, please contact us. Update: One of the best editors is RStudio!

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 Installation

If you have questions regarding the installation, please contact us.

1.2 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:

```
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

1.3 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:

```
a <- 3
b <- 4
c <- a+b
pi
Pi
PI
#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
```

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.

1.4 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:

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

1.5 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:

```
?mean
??mean
apropos("mean")
example("mean")
find("mean")
```

1.6 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:

```
#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)
```

2.1 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: `x <- 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:

```
x <- read.csv(file.choose())
```

2.4 Reading excel files and other data formats

Please see the following code for additional information and useful commands for importing data.

```
#specify the symbol for the decimal point and separator
x <- read.csv(file.choose(), dec=",",sep=";")
x
x <- read.csv(file.choose()) # if you use the "English" decimal point
x <- read.csv2(file.choose()) # if you use the "German" decimal point
x
#load from the clipboard (CTRL+C)
#Hint: This command won't work in a script (since it overrides the clipboard),
#so type it in the command window of R
x <- read.table("clipboard", header=TRUE, dec=",")
x
#use the package xlsx
library(xlsx)
?read.xlsx
x <- read.xlsx(file.choose(),sheetIndex=1)
x

### 2.3 ###
x <- read.csv("http://www.wiwi.uni-muenster.de/statistik/download/studium/inference/ws1112/data/b
x

### 2.4 ###
install.packages("foreign")
library(foreign)
library(help=foreign)
x <- read.dta(file.choose())
x

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

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

`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. Here is an example:

```
#######################################################
## Additional information: Missing Values & 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 # overrides 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!
```

3.1 Head and tail

Please have a look at the following code for additional commands:

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

3.2 Attaching dataframes

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

3.3 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

```
dim(x)
n <- dim(x)[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 positiv (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 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)
x  <- read.csv(file.choose())
head(x)
tail(x)
names(x)
str(x)
attach(x)
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)
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)
indices <- read.csv2(file.choose())
head(indices)
tail(indices)
names(indices)
str(indices)
attach(indices)

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_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"))
```

5.1 Using functions

Have a look at the following code:

```
  #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

```
?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

```
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.
truehist(r)

#3)
x <- 1:10
x; cumsum(x)
plot(cumsum(r)) # Random walk!
```

5.4 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

```
#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

```
?optimize
?optim
#1)
#look at exercise 5.5 1), the minimum lies between -1 and 0
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)

#you can try to edit phi and theta to get a better view
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
```

6.1 Conditional probabilities

Define the event Z customer is satisfied” and the event A customer answers”. The following probabilities are given

$$\begin{aligned} P(Z) &= 0.7 \\ P(A|Z) &= 0.6 \\ P\left(A|\bar{Z}\right) &= 0.15. \end{aligned}$$

Obviously,

$$\begin{aligned} P\left(\bar{Z}\right) &= 0.3 \\ P\left(\bar{A}|Z\right) &= 0.4 \\ P\left(\bar{A}|\bar{Z}\right) &= 0.85. \end{aligned}$$

We want to know $P(Z|A)$. First, we calculate

$$\begin{aligned} P(A) &= P(A|Z)\,P(Z) + P\left(A|\bar{Z}\right)P\left(\bar{Z}\right) \\ &= 0.6 \cdot 0.7 + 0.15 \cdot 0.3 \\ &= 0.465. \end{aligned}$$

By Bayes' theorem

$$\begin{aligned}P(Z|A) &= \frac{P(A|Z)\,P(Z)}{P(A)}\\ &= \frac{0.6\cdot 0.7}{0.465}\\ &= 0.9032.\end{aligned}$$

6.2 The limits of diversification (I)

6.3 The limits of diversification (II)

6.4 How fads evolve

6.5 Penalty kicks and mixed strategies

1. Optimal strategy
2. Comparison of theoretical and empirical results

6.6 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

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

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.

6.7 Gifts and donations

7.1 Student teacher ratio (I)

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

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)
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 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 Student teacher ratio (II)

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

```
####################################
library(AER)
#1)
x <- read.csv(file.choose())
head(x)
attach(x)
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)
coeftest(regr, vcov=vcovHC)

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

7.3 Capital asset pricing model

7.4 Differences

7.5 Omitted variable bias

```
###############################
#### Omitted variable bias ####
###############################
x <- read.csv2(file.choose())
head(x)
attach(x)
cor(x)
```

```
#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.

7.6 Asymptotic normality

7.7 Pitfalls in the linear regression model (I)

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

#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
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"))
```

7.8 Pitfalls in the linear regression model (II)

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

7.9 Pitfalls in the linear regression model (III)

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

n <- dim(z)[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)
```

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 Bivariate exponential

8.3 Gaussianity or else?

The program might look like this:

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

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

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

#2)
cor(gaussian)

#3)
pairs(gaussian)
```

```
#4)
Y <- V1+V2+V3+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.

8.4 Gaussian and uncorrelated, but dependent

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

   $$Pr(Y \leq x) = E[Pr(Y \leq x|U)] = Pr(X \leq x) \cdot Pr(U = 1) + Pr(-X \leq x) \cdot Pr(U = -1)$$
   $$= \Phi(x) \cdot \frac{1}{2} + \Phi(x) \cdot \frac{1}{2} = \Phi(x)$$

   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(AER)
   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*).

8.5 Distribution of summands

8.6 Plotting bivariate pdfs and cdfs

8.7 Linear transformation

8.8 Delta method

1. The first order Taylor expansion of f around μ is

$$Y = f(X) \approx f(\mu) + f'(\mu)(X-\mu) + \text{rest},$$

   where the rest is negligible in the limit $X \to \mu$ (and dropped from the notation). We see that Y is just a linear transformation of X.

2. Because Y is a linear transformation of X, and X is asymptotically normal, so is Y. The asymptotic mean of Y is

$$E(Y) = E(f(\mu) + f'(\mu)(X-\mu)) \to f(\mu)$$

   and the asymptotic variance is

$$\begin{aligned} Var(Y) &= Var(f(\mu) + f'(\mu)(X-\mu)) \\ &= [f'(\mu)]^2 Var(X) \\ &\to [f'(\mu)]^2 \sigma^2. \end{aligned}$$

3. Let $X_1, \ldots, X_n$ be a random sample from X such that

$$\sqrt{n}\left(\frac{1}{n}\sum_{i=1}^{n} X_i - \mu\right) \to U \sim N(0, \sigma^2).$$

   If this condition (convergence in distribution) holds, then the approximation is valid.

4. The first order Taylor expansion of f around μ is

$$Y = f(X) \approx f(\mu) + D_f(\mu)(X-\mu)$$

5. Because Y is a linear transformation of X, and X is asymptotically normal, so is Y. The asymptotic mean of Y is

$$E(Y) = E(f(\mu) + D_f(\mu)(X-\mu)) \to f(\mu)$$

   and the asymptotic variance is

$$\begin{aligned} Var(Y) &= Var(f(\mu) + D_f(\mu)(X-\mu)) \\ &= D_f(\mu) Var(X) D_f(\mu)' \\ &\to D_f(\mu) \Sigma D_f(\mu)'. \end{aligned}$$

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

The program might look like this:

```
###############################################
#### Law of large numbers for the variance ####
###############################################
#1)
    z <- rep(NA,1000)
    for (i in 1:1000) {
      #x <- rnorm(i,mean=10,sd=2) ; variance <- 2^2
      #x <- runif(i,min=0,max=6) ; variance <- 3
      x <- rgeom(i,prob=0.2); variance <-20
      z[i] <- sum((x-mean(x))^2)/i
    }
    par(mfrow=c(1,2))

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

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

    par(mfrow=c(1,2))

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

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(AER)
N <- 10000
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
   #standardization: subtract the expectation and
   #divide by the standard deviation, times the
   #square root of the numbers used to calculate the mean
  Zn <- (Yn/n - expect)*sqrt(n/vari)
  #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")
  # this 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(AER)
N <- 10000
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) AR process
N <- 10000
mu <- 0.5
rho=0.8
sigma <- 2 #this is the standard deviation of the random term in the AR(1) process
vari <- sigma^2/(1-rho^2) #analytical variance of an AR(1)-process
par(mfrow = c(1,2),pty = "s")
for(n in 1:20) {
  X <- filter((1-rho)*mu+rnorm(n*N,sd=sigma),rho,method="recursive",init=(1-rho)*mu)
  M <- matrix(X, N, n)
  Yn <- rowSums(M)
  Zn <- (Yn/n - mu)*sqrt(n/vari) #standardization
  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)
}

#4)
library(AER)
N <- 10000
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 Another limit distribution

9.5 Limits of maxima (I)

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

```
################################
#### Limits of maxima (II) ####
################################
  library(AER)
  I <- 1000
  n <- 100
  M <- rep(NA,I)
  R <- rep(NA,I)
  for (i in 1:I) {
    x <- rt(n,df=1.5)
    M[i] <- max(x)
    R[i] <- M[i]/qt(1-1/i,1.5)
  }
  truehist(R)
  coord <- par("usr")
  x <- seq(coord[1],coord[2],length=500)
  lines(x,1.5*x^(-2.5)*exp(-x^(-1.5)))
```

9.7 Limits of maxima (III)

10.1 Desirable properties

10.2 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.

10.3 Unbiasedness

10.4 Relative efficiency

10.5 Consistency

10.6 The standard OLS estimator

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]
  d <- dat$y-exp(alpha+beta*dat$x)
  return(sum(d^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]
  d <- dat$y-(beta1+beta2*dat$x1+1/beta2*dat$x2)
  return(sum(d^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 Autoregressions

11.3 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

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

11.4 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,
$$\operatorname{plim} \hat{\lambda} = \operatorname{plim} \frac{1}{\hat{\mu}_1} = \frac{1}{\operatorname{plim} \hat{\mu}_1} = \frac{1}{\mu_1} = \lambda.$$

11.5 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:

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

11.6 Method of moments for the Pareto distribution

1. For a given K the moment estimator of α is given by
$$\mu_1 = \frac{\alpha K}{\alpha - 1}$$
$$\alpha = \frac{\mu_1}{\mu_1 - K}$$
$$\hat{\alpha} = \frac{\hat{\mu_1}}{\hat{\mu_1} - K}$$

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).

11.7 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)
```

11.8 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).

12.1 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 loglikelihood.

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

  daxreturns <- read.csv(file.choose(), sep=";", dec=",")
  View(daxreturns)
  attach(daxreturns)
  plot(daxret)
  z <- daxret*(-1)
  daxlosses <- z[z>=2]
  alphahat <- length(daxlosses)/(sum(log(daxlosses) - log(2)))

  plot(daxlosses)
  truehist(daxlosses)
  hist(daxlosses,prob=T)
  coord <- par("usr")
  x <- seq(coord[1], coord[2], length.out = length(daxlosses))
  lines(x,dpareto(x,location=2,shape=alphahat),col="red",lwd=2)

  #Likelihood of the observations as a function of alpha
  likelihood <- function(alpha,K,x) {
    z <- prod(dpareto(x,K,alpha))
    return(z)
  }
  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)
f <- function(x) -likelihood(x,2,daxlosses)
optimize(f,lower=0,upper=3)
```

12.2 Parameters of the uniform distribution

1. Consider a sample $x_1, \dots, x_n$ 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)
```

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
  logl <- 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),logl,dat=x,cens=12,hessian=T)
  print(obj$par)  # Point estimates
  print(solve(obj$hessian)) # Numerical covariance matrix
```

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
  logl <- 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),logl,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
  logl2 <- 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),logl2,dat=expgrowth,hessian=T)
  print(obj2$par)  # Point estimates
```

12.5 Multiple linear regression model

12.6 Tobit model

12.7 Probit model

12.8 Logit model

12.9 Heckman regression

12.10 Count data

12.11 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 ??.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

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

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

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

12.12 Duration models

12.13 ARCH models

12.14 Ultra-high-frequency data

12.15 Panel model with random effects

12.16 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

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

Consider the terms in turn,

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

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

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

Hence, the log-likelihood function is

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

3. (...)

13.1 Asymptotically pathological matrices

1. Linear trend:

$$\frac{1}{n}\left(X_n' X_n\right) = \frac{1}{n}\begin{pmatrix} 1 & 1 & \dots & 1 \\ 1 & 2 & \dots & n \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ \vdots & \vdots \\ 1 & n \end{pmatrix} = \begin{pmatrix} 1 & \frac{n+1}{2} \\ \frac{n+1}{2} & \frac{(n+1)(2n+1)}{6} \end{pmatrix}$$

does not converge, since $\frac{n+1}{2} \to \infty$. A linear trend does not work!

2. Stochastic trend

$$\begin{aligned}
\frac{1}{n}\left(X_n' X_n\right) &= \frac{1}{n}\begin{pmatrix} 1 & 1 & \dots & 1 \\ z_1 & z_1 + z_2 & \dots & \sum_{i=1}^n z_i \end{pmatrix} \cdot \begin{pmatrix} 1 & z_1 \\ 1 & z_1 + z_2 \\ \vdots & \vdots \\ 1 & \sum_{i=1}^n z_i \end{pmatrix} \\
&= \begin{pmatrix} 1 & \frac{1}{n}\sum_{i=1}^n (n+1-i) z_i \\ \frac{1}{n}\sum_{i=1}^n (n+1-i) z_i & \frac{1}{n}\sum_{i=1}^n \sum_{j=1}^i z_j^2 \end{pmatrix}
\end{aligned}$$

Convergence in probability:

$$\lim_{n\to\infty} P\left(|X_n - X| \geq \epsilon\right) = 0 \quad \text{for all } \epsilon > 0$$

When the sample gets large, it becomes increasingly unlikely for X_n to differ from X by even an arbitrarily small constant ϵ. Sufficient conditions are:

$$\lim_{n\to\infty} E\left(X_n\right) = \mu$$
$$\lim_{n\to\infty} E\left(X_n - \mu\right)^2 = \lim_{n\to\infty} Var(X_n) = 0$$

Here

$$\frac{1}{n}\sum_{i=1}^{n}(n+1-i)E(z_i) = 0$$

$$\frac{1}{n^2}\sum_{i=1}^{n}(n+1-i)^2 Var(z_i) = \frac{1}{n^2}\sum_{i=1}^{n}(n+1-i)^2 = \frac{2n^3+3n^2+n}{6n^2}$$

this does not converge to 0 as $n \to \infty$. Stochastic trend does not work as well.

3. Little information

$$\frac{1}{n}\left(X_n' X_n\right) = \frac{1}{n}\begin{pmatrix} 1 & 0 & 1 & \dots & 1 \\ 0 & 1 & 1 & \dots & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ \vdots & \vdots \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} \frac{n-1}{n} & \frac{n-2}{n} \\ \frac{n-2}{n} & \frac{n-1}{n} \end{pmatrix} \quad \text{is invertible.}$$

As $n \to \infty$ this matrix becomes the identity matrix, which of course is not invertible.

13.2 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])
```

13.3 Linear combinations of instruments

13.4 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)
  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)
```

13.5 A simple example

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)
  attach(fertility)
  #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)

  #3)
    #Relevance of instrument
    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)
```

Additional remarks:

2) 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) 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: $morekids_i = \beta_0 + \beta_1 samesex_i + \varepsilon_i$.

5) $H_0 : \beta_4 - \beta_5 = 0$, $t = \frac{\widehat{\beta_4} - \widehat{\beta_5}}{\sqrt{\widehat{var(\beta_4)} + \widehat{var(\beta_5)}}}$ under the assumption that β_4 and β_5 are independent.

13.7 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. $\widehat{\beta_{GIV}} = (X'P_W X)^{-1} X' P_W y$

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 $i = 1, \ldots, n$.
   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
WuRegr <- lm(m[5:TT] ~ Pw%*%r[5:TT] + y[5:TT] + m[4:(TT-1)] + m[3:(TT-2)])
summary(WuRegr)
# H0 can be rejected.
# 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!
# Hence, r_t should not 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

#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, the model is thus not overidentified.
```

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)

  #three moment conditions
  g1 <- function(param,x) {
    m1 <- param[1]-x
    m2 <- param[2]^2-(x-param[1])^2
    m3 <- x^3 - param[1]*(param[2]^2+3*param[2]^2)
    f <- cbind(m1,m2,m3)
    return(f)
  }

  #generate data
  set.seed(123)
  n <- 200
  dat <- rnorm(n,mean=4,sd=2)

  #gmm estimation
  res1 <- gmm(g1,dat,c(mu=0,sig=0))
  print(res1)
  summary(res1)
```

14.2 Nonlinear least squares estimation and GMM

1. $E(u_t|x_t(\beta) = 0 \Rightarrow E(x_t(\beta) \cdot u_t) = E(x_t(\beta) \cdot (y_t - x_t(\beta))) = 0$.
2. Here $x_t(\beta) = exp(\alpha + \beta x_t)$. With that we can estimate α and β, but not σ^2. So we need another moment condition: $E(u_t^2 - \sigma^2) = E((y_t - x_t(\beta))^2 - \sigma^2) = 0$. This yields the same results as in exercise **??**.

The code might look like this:

```
####################################
#### Nonlinear regression model ####
####################################
  expgrowth <- read.csv(file.choose())
  View(expgrowth)
  x <- expgrowth$x
  y <- expgrowth$y

  g4 <- function(param,dat){
    y <- dat[,1]
    x <- dat[,2]
    f1 <- y - exp(param[1]+param[2]*x)
    m1 <- cbind(1,x)*f1
    f2 <- (y-exp(param[1]+param[2]*x))^2-param[3]^2
    m2 <- f2
    return(cbind(m1,m2))
  }

  g4(c(1,0.1,1),cbind(y,x)) # g gives you a nxq matrix with n observations and q moments
  gmmmod <- gmm(g4,x=cbind(y,x),c(alpha=0,beta=0.01,sigma=1))
  print(gmmmod)
  gmmmod <- gmm(g4,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)
```

14.4 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)]

  # Generalized IV estimation
  IV <- ivreg(mt~rt+yt+mt1+mt2|rt1+rt2+yt+mt1+mt2)
  IV

  # Gmm estimation in linear model notation
  obj <- gmm(mt~rt+yt+mt1+mt2, ~ rt1+rt2+yt+mt1+mt2)
  obj
```

14.5 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,x) {
    beta <- theta[1]
    sigma <- theta[2]
    u <- x[,2]-beta*x[,1]
    m1 <- u
    m2 <- u*x[,1]
    m3 <- u^2-sigma^2
    return(cbind(m1,m2,m3))
  }
  # GMM estimates for beta and sigma
  obj <- gmm(g,cbind(x,y),t0=c(0.9,1),wmatrix="optimal")
  obj1 <- gmm(g,cbind(x,y),t0=c(0.9,1),wmatrix="ident")
  obj
  obj1

  # Whole thing within a loop
  R <- 1000
  n <- 100 # or try n=1000
  nu <- 3
  Z <- 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
    obj <- gmm(g,x=cbind(x,y),t0=c(0.9,1),wmatrix="optimal")
    Z[r,] <- (coefficients(obj)-c(0.9,1))/sqrt(diag(vcov(obj)))
    }
  # mean of the estimates for beta and sigma
  apply(Z,2,mean)
  #histogram of sigma compared to a normal distribution
  truehist(Z[,2])
  xx <- seq(min(Z[,2]),max(Z[,2]),length=500)
  lines(xx,dnorm(xx,mean(Z[,2]),sd(Z[,2])))

  ###Now with the normal distribution
  R <- 1000 # or try R=1000
  n <- 100 # or try n=1000
  Z <- matrix(NA,R,2)
  for(r in 1:R) {
    x <- rnorm(n)
    u <- rnorm(n)
    y <- 0.9*x+u
    obj <- gmm(g,cbind(x,y),t0=c(0.9,1),wmatrix="optimal")
    Z[r,] <- (coefficients(obj)-c(0.9,1))/sqrt(diag(vcov(obj)))
    }
  # mean of the estimates for beta and sigma
  apply(Z,2,mean)

  #histogram of sigma compared to a normal distribution
  truehist(Z[,2])
  xx <- seq(min(Z[,2]),max(Z[,2]),length=500)
  lines(xx,dnorm(xx,mean(Z[,2]),sd(Z[,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.

14.6 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 ####
#######################
  #Get data
  data(Finance)
  ?Finance
```

```
View(Finance)
#let's only take the first 500 observations
r <- Finance[1:500,1:5]
rm <- Finance[1:500,"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
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
```

14.7 Consumption-based CAPM

The code might look like this:

```
####################################
###### Consumption-based CAPM ######
####################################
  #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){
    gr <- dat[,1]
    R <- dat[,2]
    m <- param[1]*(gr^(-param[2]))*R-1 #the result is sensible to the second parameter
    m <- m[-length(m)] #since m has one more entry than z, get rid of the last one
    z <- cbind(1,gr[-1],R[-1])
    return(m*z)
  }
  obj <- gmm(g,x=X,c(1,1))
  obj  #very bad results for gamma
```

14.8 Small sample properties of GMM estimators

14.9 Minimum distance estimation

15.1 AR(1) processes

The code might look like this:

```
############################################
#### Indirect Inference - AR(1) process ####
############################################
  library(AER)
  #Define function that estimates the ar process for different values of rho,n and R
  simestim <- function(rho,n,R) {
    Z <- matrix(NA,R,2)
    for (r in 1:R) {
      x <- filter(rnorm(n),rho,method="r",init=rnorm(1))
      Z[r,1]<- ar(x,aic=F,order = 1)$ar # method: yule-walker
      Z[r,2]<- ar(x,aic=F,order = 1, method = "ols")$ar #method: ols
    }
    return(Z)
  }

  #1) Stationary AR(1)
  stationary <- simestim(rho=0.99,n=100,R=1000)
  truehist(stationary[,1])
  truehist(stationary[,2])

  #2) Random Walk, methods: yule-walker, ols, mle
  randomwalk <- simestim(rho=1,n=100,R=1000)
  truehist(randomwalk[,1])
  truehist(randomwalk[,2])

  #3) Explosive AR(1)
  explosive <- simestim(rho=1.01,n=100,R=1000)
  truehist(explosive[,1])
  truehist(explosive[,2])

  #4) Estimation by indirect inference, auxiliary model is an AR(1)-process
  H <- 10; n <- 100; W <- diag(1)
  truedata <- filter(rnorm(n),0.9,method="r",init=rnorm(1)) #true model with rho=0.8
```

```
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)$ar          #store estimator
  }
  thetatilde <- mean(thetahatsim)
  Q <- t(thetahat - thetatilde) %*% W %*% (thetahat - thetatilde)
  return(Q)
}
# indirect inference estimator for rho
rhohat <- optimize(f,lower=0.7,upper=1.1)
return(rhohat$minimum)
}

rhohat(truedata,H=10,n=100,W=diag(1))
```

15.2 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))
```

15.3 Time-aggregated observations

15.4 Filter models

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

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 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(AER)
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 noch fit

#3)
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 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")
```

16.5 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)

#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(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!
```

16.6 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
```