# Introduction to R – Exercises

Willi Mutschler & Martina Danielova Zaharieva

Spring 2018

## Contents

# 1 Introduction

1. Start R-Studio and have a look at all menu items.
2. Under “Tools - Options ...” choose your preferred setting.
3. Install the packages `MASS`, `foreign`, `xlsx`, `rgl`.
4. The current working directory (where R reads and writes files) can be found by the command `getwd()`. Find your current working directory.
5. Use the command `setwd("c:/path")` to change the working directory to drive `c:` and path `/path`. Note that the path name is structured by slashes (/), *not* backslashes (\). Change the working directory to `c:/temp` and check if the change has been successful.[1]
6. Open a new script file. Type the commands to perform the following assignments:

$$\begin{aligned} a &= \frac{3\cdot(4+9)}{8-12.5} \\ b &= (1,4,1999,2011) \\ d &= 2\pi \\ e &= a+d \end{aligned}$$

   Save the script and quit R.
7. Start R and re-open the script. Mark all lines (CTRL-A) and execute them (CTRL-R). Print $a, b, d, e$. Why is the variable name $c$ not used?

[1]The working directory can also be changed via the menu: Tools – Options ...

## 2 Logical operators

1. Use the command `c()` to define the vectors

$$\begin{aligned} x &= (-1,0,1,4,9,2,1,4.5,1.1,-0.9) \\ y &= (1,1,2,2,3,3,4,4,5,NA). \end{aligned}$$

2. Determine the lengths of the vectors using `length()`, check if `length(x)==length(y)`.
3. Perform the following logical operations:

$$\begin{aligned} x < & \quad y \\ x < & \quad 0 \\ x+3 \geq & \quad 0 \\ y < & \quad 0 \\ x < 0 \text{or} & \quad y < 0 \end{aligned}$$

4. Use `all` to check if all elements of $x+3 \geq 0$.
5. Use `all` to check if all elements of $y > 0$. Use any to check if at least one element of $y > 0$.

# 3 Arithmetic operators and mathematical functions

1. Define the vectors

$$\begin{aligned} x &= (-1, 0, 1, 4, 9, 2, 1, 4.5, 1.1, -0.9) \\ y &= (1, 1, 2, 2, 3, 3, 4, 4, 5, NA). \end{aligned}$$

   and compute $x + y$ and $xy$ and $y/x$.

2. Compute $\ln(x)$. Determine the length of the result vector.

3. Use `any` to check if the vector `x` contains elements satisfying $\sqrt{x} \geq 2$.

4. Compute

$$\begin{aligned} a &= \sum_{i=1}^{10} x_i \\ b &= \sum_{i=1}^{10} y_i^2. \end{aligned}$$

   Use the `na.rm=TRUE` option (**na**-**rem**ove) of the `sum` command to drop the `NA` in $y$.

5. Compute

$$\sum_{i=1}^{10} x_i y_i^2.$$

6. The `sum` command is a convenient way to count the number of elements satifying a certain condition. Count the number of elements of $x > 0$.

7. Predict what the following commands will return:

```
x^y
x^(1/y)
log(exp(y))
y*c(-1,1)
x+c(-1,0,1)
sum(y*c(-1,1),na.rm=TRUE)
```

# 4 Matrix functions

1. Define the matrix
$$X = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix},$$
print its transpose, its dimensions and its determinant.

2. Compute the trace of $X$ (i.e. the sum of its diagonal elements).

3. Type `diag(X) <- c(7,8,9)` to change the diagonal elements. Compute the eigenvalues of (the new) $X$.

4. Invert $X$ and compute the eigenvalues of $X^{-1}$.

5. Define the vector $a = (1,3,2)$ and compute `a*X`, `a%*%X`, and `X%*%a`.

6. Compute the quadratic form $a'Xa$.

7. Define the matrix
$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
and define
$$Y = \begin{bmatrix} 1 & 4 & 7 & 1 & 0 & 0 \\ 2 & 5 & 8 & 0 & 1 & 0 \\ 3 & 6 & 9 & 0 & 0 & 1 \end{bmatrix}$$
and
$$Z = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

8. Predict what the following commands will return:

```
cbind(1,X)
rbind(Y,c(1,2,3))
X%*%I
dim(X%*%Y)
t(Y)+Z
solve(t(Z)%*%Z)%*%(t(Z)%*%Z)
```

# 5 Set operations and special functions

1. Define the vectors

$$\begin{aligned} x &= (-1, 0, 1, 4, 9, 2, 1, 4.5, 1.1, -0.9) \\ y &= (1, 1, 2, 2, 3, 3, 4, 4, 5, NA). \end{aligned}$$

   and compute $x \cup y$. Determine the lengths of $x$, $y$ and $x \cup y$.

2. Count the number of elements of $y$ that are element of $x$.

3. Determine the length of the vector of unique elements of $y$.

4. Compute the vector $z$ with elements

$$z_i = \sum_{j=1}^{i} x_j$$

   for $i = 1, \ldots, 10$.

5. Find the position of the largest element of $x$.

# 6 Sequences and replications

1. Generate the vectors

$$
\begin{aligned}
x_1 &= (1,2,3,\ldots,9)\\
x_2 &= (0,1,0,1,0,1,0,1)\\
x_3 &= (1,1,1,1,1,1,1,1)\\
x_4 &= (-1,1,-1,1,-1,1)\\
x_5 &= (1980,1985,1990,\ldots,2010)\\
x_6 &= (0,0.01,0.02,\ldots,0.99,1)
\end{aligned}
$$

2. Replications can also be generated for vectors of strings (characters). Type

```
a <- c("a","b","c")
rep(a,3)
rep(a,times=3)
rep(a,each=3)
```

3. Generate a grid of $n = 500$ equidistant points on the interval $[-\pi,\pi]$.

4. Compare `1:10+1` and `1:(10+1)`.

5. Predict what the following commands will return:

```
rep("bla",10)
rep(rep(1:3,2),each=4)
rep(c(1,6,NA,2),times=c(2,2,5,3))
```

# 7 The apply command

Define the matrices

$$X = \begin{bmatrix} 1 & 2 & 7 & 9 \\ 9 & 5 & 6 & 4 \\ 3 & 3 & 5 & 4 \end{bmatrix},$$

$$Y = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & NA \\ 3 & 6 & 9 \end{bmatrix}.$$

In the following exercises always use the `apply` command.

1. Compute the row sums of $X$.
2. Compute the column means of $X$.
3. Compute the column ranges of $X$.
4. Compute the column sums of $Y$. Add the `sum` function option `na.rm=TRUE` to the `apply` command.
5. For each column of $X$, find the position (the row number) of the largest element.
6. For each column of $X$, compute the cumulated sums (`cumsum`).
7. For each row of $X$, find the number of elements that are smaller than 5.
8. For each column of $Y$, determine the number of non-missing observations.
9. For each row of $X$, determine the unique elements. Note that in this case the `apply` command does not return a matrix.

# 8 Reading and writing text files

Download the files `bsp1.txt`, `bsp2.txt` and `bsp3.txt` from the internet site of the course and save them to your working directory. The three files contain computer generated random numbers.

1. Read the file `bsp1.txt` into a dataframe `x1`. Have a look at the file and the data format *before* you decide which reading command you use (`read.csv`, `read.csv2` or `read.table`). Print the dataframe. If the dataframe is too large for your screen, you can use the commands `head` and `tail` to print just parts of it.

2. Read the files `bsp2.txt` and `bsp3.txt` into dataframes `x2` and `x3`. Note that `bsp2.txt` contains both numeric and character entries. It is usually advisable to set the option `as.is=TRUE` when reading characters (strings).

3. Print the class of `x2`, its dimension, and its variable names (use `names`).

4. Print a summary of `x3`.

5. Use the `apply` command to compute the mean and the standard deviation of each column of `x3`.

6. Create a small dataframe `a` with two variables

| x | y |
|---|---|
| 1 | 4 |
| 2 | 5 |
| 3 | 6 |

and write it to a file in your working directory.

# 9 Reading data online

Install and activate the packages `TTR` and `fImport`.

1. Read the (large) file `lest2001.csv` directly from the following internet site into a dataframe `x`. The complete URL is

   `http://www.wiwi.uni-muenster.de/05/download/`
   `studium/R/ws1112/data/lest2001.csv`

   The file is the campus file of the German income tax records 2001 (the data are provided by the Research Data Centre of the Federal Statistical Office, they are described in `lest2001.pdf`). Take care to set the options of the `read.csv` or `read.table` command correctly. The data format is as follows:

   - All data entries are separated by semi-colons.
   - The first row contains the variables names.
   - Missing values are denoted by a dot.
   - Apart from the last column all data are integer values.
   - The decimal sign in the last column is a dot.

   Execute `y <- x$zve`. The variable `y` now contains the taxable income (**z**u **v**ersteuerndes **E**inkommen). Compute its range, its median, its mean, its variance, and the 0.01- and 0.99-quantiles. Remember to include the option `na.rm=TRUE` in the functions.

2. Read the help text of the `getYahooData` command of the `TTR` package. Read the data about the Deutsche Bank (symbol `DB`) from January 1st, 2000 to the most recent date into the object `x`. Plot the closing price by `plot(x$Close)`.

# 10 Indexing vectors

Define the following vectors

$$x = \begin{pmatrix} 1 \\ 1.1 \\ 9 \\ 8 \\ 1 \\ 4 \\ 4 \\ 1 \end{pmatrix}, \quad y = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 4 \\ 3 \\ 2 \\ NA \end{pmatrix}, \quad z = \begin{pmatrix} TRUE \\ TRUE \\ FALSE \\ FALSE \\ TRUE \\ FALSE \\ FALSE \\ FALSE \end{pmatrix}$$

1. Predict what the following commands will return (and then check if you are right):

```
x[-2]
x[2:5]
x[c(1,5,8)]
x[-c(1,5,8)]
x[y]
x[seq(2,8,by=2)]
x[rep(1:3,4)]
```

2. Predict what the following commands will return (and then check if you are right):

```
y[z]
y[!z]
y[x>2]
y[x==1]
x[!is.na(y)]
y[!is.na(y)]
```

3. Indexing is not only used to read certain elements of a vector but also to change them. Execute `x2 <- x` to make a copy of `x`. Change all elements of `x2` that have the value 4 to the value $-4$. Print `x2`.

4. Change all elements of `x2` that have the value 1 to missing value (`NA`). Print `x2`.

5. Execute `x2[z] <- 0`. Print `x2`.

# 11 Indexing matrices

Define the matrix `x <- matrix(c(1:12,12:1),4,6)`.

1. Predict what the following commands will return (and then check if you are right):

```
x[1,3]
x[,5]
x[2,]
x[,-3]
x[-4,]
x[2:3,3:4]
x[2:4,4]
```

2. Predict what the following commands will return (and then check if you are right):

```
x[x>5]
x[,x[1,]<=5]
x[x[,2]>6,]
x[x[,2]>6,4:6]
x[x[,1]<3 & x[,2]<6,]
```

3. Print all rows where column 5 is at least three times larger than column 6.
4. Count the number of elements of `x` that are larger than 7.
5. Count the number of elements in row 2 that are smaller than their neighbours in row 1.
6. Count the number of elements of `x` that are larger than their left neighbour.

# 12 Indexing dataframes

Load the dataset `bsp2.txt` as dataframe `y` and print it.

1. Use different ways to print the second column of the dataframe `y` (as a vector or a dataframe).
2. Use different ways to print columns $U$ and $V$.
3. Use the `attach` command to make the variables directly accessible. Print `X`. Now `detach` the dataframe again.
4. Print all rows of `y` where the variable $U$ has value A or B.
5. Print all rows of `y` where the variable $X$ is smaller than its median and the variable $Y$ is larger than its median.
6. One can add row names to a dataframe. Execute the following command:

   ```
   row.names(y) <- paste(rep(LETTERS[1:20],each=2),rep(1:2,20),sep="")
   ```

   and print the dataframe to have a look at the new row names.
7. Use the row name and the variable name to print the value of variable `Z` at observation `T1`.
8. Print the rows for observations `G1` and `G2`.

# 13 User-defined functions

1. Define a Cobb-Douglas production function with two inputs vectors,

$$
\begin{aligned}
x &= \begin{pmatrix} L \\ K \end{pmatrix} \\
\theta &= \begin{pmatrix} A \\ \alpha \\ \beta \end{pmatrix}
\end{aligned}
$$

   and scalar output

$$y = AL^{\alpha}K^{\beta}.$$

   Evaluate the function at

$$
\begin{aligned}
x &= \begin{pmatrix} 2 \\ 3 \end{pmatrix} \\
\theta &= \begin{pmatrix} 1 \\ 0.3 \\ 0.8 \end{pmatrix}.
\end{aligned}
$$

2. Define a function `lowdecile` with one input vector $(x_1, \ldots, x_n)$ of arbitrary length. The function should compute and return the mean of all observations in the lowest decile. Define the vector

$$x = (0,0,0,0,1,1,1,1,2,2,2,2,\ldots,9,9,9,9)$$

   and apply `lowdecile` to $x$.

# 14 Sorting and merging

1. Define $x = (-1, 0, 1, 4, 9, 2, 1, 4.5, 1.1, -0.9)$ and sort the vector ascendingly. Print the second smallest element of $x$.

2. Sort $x$ descendingly and print the third largest element of $x$.

3. Execute `x <- matrix(c(1:12,12:1),6,4,byrow=T)` to define the matrix $x$. Calculate the order vector $p$ of the last column of $x$. Sort the matrix $x$ by the last column and print the sorted matrix.

4. Load the file `bsp2.txt` into the dataframe `x`. Sort `x` by the variable `U` and on the next level by the variable `X`.

5. Activate the package `foreign` and load the Stata files `wave2000.dta` and `wave2009.dta` into two dataframes. Print the variable names of both dataframes. Merge the dataframes by the variable `pid`. Keep only persons who are present in both years (by setting the option `all=FALSE`). Print the variables names of the merged dataframe.

6. Calculate the mean life satisfaction in 2000 and 2009.

7. Calculate the mean change in life satisfaction for persons who were single in 2000 and married in 2009.

# 15 Frequency tables

1. Read the Stata file `wave2009.dta` into a dataframe. Tabulate the variables `gender`, `marital` and `children`.
2. Read the file `bsp2.txt` into a dataframe. Compute and plot the frequency tables of the variables $U$ and $V$.
3. Define the vector $x = (1,1,1,1,1,2,2,2,2,3,3,3,4,4,5,9)$. Compute the absolute and relative frequency tables and determine the number of different values of $x$.
4. Read the file `lest2001.csv` (wage and income tax data) into a dataframe using the option `as.is=TRUE`. The variable `ef8` gives the gender (male=0, female=1). Tabulate `ef8`.
5. The `lest2001.csv` dataset contains a weighting variable (`samplingweight`). Each row counts as `samplingweight` observations. Re-compute the absolute frequencies of `ef8` taking into account the weights.
6. Load the Stata file `mikrozensus2002cf.dta` into a dataframe (the data are described in the file `mikrozensus2002cf.pdf`). Consider the variable `ef455` (age of flat or house) which is coded into nine classes. Tabulate the age structure.

# 16 Cumulative distribution function and quantile function

1. Load the dataset `bsp1.txt` and calculate the value of the empirical distribution function of `Variable1` at the point $x = 10$.

2. Plot the empirical distribution function of `Variable2`. Improve the readability of the plot by adding an appropriate heading and axes labels (use the options `main` and `xlab,ylab`).

3. Compute the $p$-quantiles of `Variable1` for $p = 0.01, 0.05, 0.1, 0.5, 0.9, 0.95, 0.99$.

4. Load the dataset `lest2001.csv` and compute the value of the empirical distribution function of `zve` (taxable income) at the points 0, 12000, and 60000.

5. Use the `quantile` command to compute the 0.99-quantile, the 0.999-quantile, and the 0.9999-quantile of `zve`.

# 17 Mean, variance and standard deviation

1. Read the file `lest2001.csv` (wage and income tax data) into a dataframe. The variable `zve` reports taxable income. Compute the mean taxable income without weighting the observations. Then re-compute the mean using the weights given in the `samplingweight`. Hint: Use `weighted.mean`.

2. Compute the unweighted and weighted variance of `zve`.

3. Load the Stata file `mikrozensus2002cf.dta` into a dataframe and consider the variables `ef455` (age of flat or house) and `ef466` (cost of heating and warm water in April 2002). Compute the mean cost of heating and warm water for each age class. Hint: You may consider using the command `by`.

# 18 Histograms

In this section, please always use the command `truehist` (which is included in the `MASS` package) to generate histograms.

1. Load the file `gemeinden2006.csv` into a dataframe. Delete all observations where the number of inhabitants (`Einwohner`) is smaller than 5. Plot the histogram of the logarithm of the variable `Einwohner`.
2. Add the density function of a fitted normal distribution to the
3. Load the Stata file `mikrozensus2002cf.dta` into a dataframe. Consider the variable `ef462` (rent in April 2002). Drop all observations where the rent exceeds 2000 Euro. Plot the histogram
4. Load the Stata file `mikrozensus2002cf.dta` into a dataframe.
   (a) Plot the histogram of the variable `ef453` (size of flat in square meters).
   (b) Drop all observations with more than 300 $m^2$ and plot the histogram again.
   (c) Set the number of bins in the histogram to 15.

# 19 Contingency tables, correlation and covariance

1. Execute `data(Titanic)` to load the object `Titanic` of class `table`. Print it as an ordinary table and as a flat table. Plot it as well. Compute the univariate marginal distributions using the `apply` command. Compute the bivariate marginal distribution of survival and social class (again using `apply`).

2. Load the file `covmat.csv` into a dataframe.

   (a) Compute the covariance matrix using the option `use="complete"`. Check if the covariance matrix is positive definite.

   (b) Now compute the covariance using the option `"pairwise"` and check again, if the covariance matrix is positive definite.

3. Load the Stata file `mikrozensus2002cf.dta` into a dataframe. Consider the two variables `ef141` (normal hours worked) and `ef372` (net income per capita in April 2002). The variable `ef372` is coded into 24 income brackets (see data description `mikrozensus2002cf.pdf`). Drop all observations where `ef372` is larger than 24 (i.e. missing values etc.) and recode the remaining observations to the midpoints of the income brackets. Compute the correlation coefficient of hours worked and net income.

# 20 Programming

1. This exercise illustrates that loops are often not very efficient.

   - Create the vector $x = (1, 2, \ldots, 1\,000\,000)$ and convert it from `integer` to `numeric` using the conversion command `as.numeric`.
   - Set $S = 0$.
   - Write a `for`-loop to compute the sum of all vector elements without using the `sum` command.
   - Put the command `p0 <- proc.time()[3]` in front of the loop and the command `print(proc.time()[3]-p0` at the end. These commands allow to measure the execution time of the loop.
   - Compare your result with the execution time of the `sum` command.

2. Create a grid vector $x$ of 60 equidistant points $x_1, \ldots, x_{60}$ on the interval $[-10, 10]$, and another grid vector $y$ of 70 points $y_1, \ldots, y_{70}$ on $[-10, 10]$. Create an empty matrix $Z$ of dimension $60 \times 70$.

   Write a double loop to compute the matrix elements

   $$Z_{ij} = \frac{10}{r_{ij}} \cdot \sin(r_{ij})$$

   where $r_{ij} = \sqrt{x_i^2 + y_j^2}$. Execute `persp(x,y,Z)`.

3. Load the dataset `fussballdaten.csv`. It contains all "1. Bundesliga" results between the saisons 1996/1997 and 2008/2009.

   - Create an alphabetically ordered vector of all clubs in the dataset.
   - Write a loop over all clubs. For each club compute the proportion of games won.
   - Order the clubs descendingly according to the proportion of games won and plot a `barplot` of the proportion.

4. Load the dataset `bsp1.txt`. Use the command `ifelse` to change all zero entries of the first variable to ones (and leave all other entries unchanged).

# 21 Random numbers

This section is not only about random number generation but also includes exercises about the R-functions for standard distributions in statistics.

1. Let $X \sim N(0,1)$. Compute the probability $P(|X| > 3.5)$. Generate $n = 10000$ random draws $X_1,\dots,X_n$ from $X$ and count the number of observations $|X_i| > 3.5$. Repeat drawing random samples $R = 5000$ times and write the counts into a vector $Z_1,\dots,Z_{5000}$ of length 5000. Tabulate $Z$ and compare the frequencies with the probability function of a suitably fitted Poisson distribution.

2. Generate $n = 10000$ draws from a lognormal distribution $X \sim e^Y$ where $Y \sim N(1, 0.5^2)$ (the parameters in the R function are `meanlog=1` and `sdlog=0.5`). Split the screen into two plotting areas using the commnd `par(mfrow=c(2,1))`. Plot the histograms of $X$ and $\ln X$.

3. Generate $n = 10000$ draws from $X \sim N(0,1)$. Compute the cumulated means, i.e.

$$\bar{X}_j = \frac{1}{j}\sum_{i=1}^{j} X_i$$

for $j = 1,\dots,n$ and plot them. Hint: Use the command `cumsum`.

# 22 Simulations

1. This exercise illustrates the one-sample $t$-test.

   (a) Generate $n = 10$ obserations from $X \sim N(10, 3^2)$. Compute the mean and the standard deviation of $X_1, \dots, X_{10}$.

   (b) The $t$-statistics of the hypothesis test $H_0 : \mu = 10$ against $H_1 : \mu \neq 10$ is

   $$t = \sqrt{10}\frac{\bar{X} - 10}{sd}$$

   where $sd$ is the standard deviation (as computed by `sd`). Compute the $t$-statistic.

   (c) Create an empty vector $Z$ of length $R = 5000$. Write a loop over $r = 1, \dots, R$ and repeat steps (a) and (b) for each $r$. Save the $t$-statistic at $Z_r$.

   (d) Plot the histogram of $Z_1, \dots, Z_R$ and add the density function of the $t_9$-distribution.

2. The classical central limit theorem states that the standardized sum of i.i.d. random variables with finite variance converges in distribution to the standard normal distribution $N(0, 1)$. This exercise illustrates the central limit theorem.

   (a) Write a simulation that performs the following steps:

   - Generate a random sample $X_1, \dots, X_5$ of size $n = 5$ from the standard exponential distribution $Exp(1)$.
   - Compute the sample sum.
   - Repeat the steps $R = 10\,000$ times. For each replication, store the sum, e.g. into a vector $Z$.
   - Plot the histogram of the sum and add the density function of $N(m, s^2)$ where $m$ is the mean of $Z$ and $s$ is the standard deviation of $Z$.

   (b) Increase the sample size $n$ in (a) to $n = 50, 500, 5000$.

   (c) Redo (a) and (b) with other distributions than the exponential. Use the uniform distribution, the $t$-distribution with 3 degrees of freedom, the Bernoulli distribution (i.e. binomial with parameter `size=1`), the Poisson distribution.

   (d) The central limit theorem breaks down if the variance of the summands is infinite. Redo (a) and (b) using a $t$-distribution with only 1.5 degrees of freedom.

# 23 Linear regressions

1. Load the Stata dataset `wages.dta`. The variables are `earnings` (in Euro, 2009), `age`, `gender` (male=1, female=2), `education` (year of education), `hours` (hours worked during 2009), and `weight`.

   (a) Compute the (unweighted) wage equation

   $$\ln \text{earnings}_i = \alpha + \beta_1 \text{age}_i + \beta_2 \text{age}_i^2 + \beta_3 \text{education}_i + \beta_4 \text{gender}_i + u_i,$$

   print the summary of the `lm`-object, and interpret the output.

   (b) Add an interaction term for `education` and `gender` to the regression.

   (c) Compute the weighted hourly wage equation

   $$\ln \frac{\text{earnings}_i}{\text{hours}_i} = \alpha + \beta_1 \text{age}_i + \beta_2 \text{age}_i^2 + \beta_3 \text{education}_i + \beta_4 \text{gender}_i + u_i,$$

   print the summary of the `lm`-object, and interpret the output.

   (d) Activate the package `sandwich` (or the package `AER`). Use the function `coeftest` to compute the heteroskedasticity robust standard errors for the estimated coefficients.

   (e) Predict the hourly wage of a male person aged 60 years as a function of education (vary the years of education between 9 and 18). Set the option `se.fit=TRUE`. Inspect the object returned by the `predict` command. Plot the predicted values and add the $\pm 2$ standard deviations confidence intervals.

2. Load the dataset `bsp4.txt`.

   (a) Plot the scatterplot of $y$ against $x$.

   (b) Perform a simple linear regression of $y$ on $x$ and save the results as an `lm`-object `obj`. Add the regression line of $y$ on $x$ to the plot.

   (c) Extract the fitted values from `obj` and add them as red points to the plot (use the command `points`).

   (d) Extract the residuals of the regression and calculate the sum of the squared residuals,

   $$SSR = \sum_{i=1}^{100} \hat{u}_i^2$$

   (e) Compute the total sum of squares and the explained sum of squares,

   $$\begin{aligned} TSS &= \sum_{i=1}^{100} (y_i - \bar{y})^2 \\ ESS &= \sum_{i=1}^{100} (\hat{y}_i - \bar{y})^2 \end{aligned}$$

   and show that $ESS + SSR = TSS$.

# 24 Numerical optimization

1. Define the polynomial `function`

$$f(x) = x(x-2)(x-4)(x-5) - 10$$

and plot it over the interval $[0,5]$.

(a) Use `optimize` to find the minimum of the function. Set the interval to $[0,5]$ and then to $[0,6]$.

(b) Even though one-dimensional optimization problems should be solved by `optimize`, use the `optim`-command with option `method="BFGS"` to find the minimum of the function $f$. Set the starting values to $0,1,2,3,-1$ and check if the algorithm finds the global minimum.

2. Consider the Cobb-Douglas production function with two inputs $x_1$ and $x_2$,

$$y = f(x_1, x_2) = x_1^{0.3} x_2^{0.6}.$$

Suppose the output price is $p_y = 1$ and the input prices are $p_1 = 2$ and $p_2 = 0.5$. Define the profit function as a `function` and numerically maximize it.

3. Consider the nonlinear regression model

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

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

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

numerically with respect to $a$ and $b$.

# 25 Maximum likelihood

1. Load the dataset `round86.csv` into a dataframe. The first column is the actual earnings in 1986 of $n = 443$ persons as derived from the employers files, the second column is the earnings reported by the persons.

   (a) Compute the reporting errors and plot their histogram.

   (b) Assume that the reporting error $x$ follows a Laplace distribution with density
   $$f(x) = \frac{1}{2b} \exp\left(-\frac{|x-\mu|}{b}\right).$$
   Write a function of $\theta = (\mu, b)$ and $x = (x_1, \dots, x_n)$ to compute the log-likelihood function
   $$\ln L(\theta, x) = \sum_{i=1}^{n} \ln f(x).$$
   Hint: The absolut value function $|\cdot|$ is `abs(·)`.

   (c) Numerically maximize the log-likelihood function with respect to $\mu$ and $b$. Set the starting values to $\mu_0 = 0$ and $b_0 = 3000$.

2. Load the dataset `fussballdaten.csv` into a dataframe.

   (a) Use indexing to obtain the vector of goals scored in home matches by the team `dortmund`. Tabulate and plot the vector.

   (b) Fit a Poisson distribution to the number of goals. The maximum likelihood estimator of the parameter $\lambda$ of the Poisson distribution is $\hat{\lambda} = \bar{X}$ where $\bar{X}$ is the mean number of goals. Add the probabilities of the fitted Poisson distribution to the plot (as red points).

3. Let $X \sim LN(\mu, \sigma^2)$ and let $X_1, \dots, X_n$ be a sample drawn from $X$. The $X_i$ are not observable. Instead one can only observe
   $$Y_i = \begin{cases} X_i & \text{if } X_i < c \\ c & \text{if } X_i \geq c \end{cases}$$
   where $c$ is a known constant. The likelihood of $Y_1, \dots, Y_n$ is the product of all densities $f_X(y_i)$, for observations with $Y_i < c$, times the product of all probabilities that $Y_i = c$ for observations with $Y_i = c$.

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

(a) Write an R function that computes the likelihood of $\mu$ and $\sigma^2$ given the observations $Y_1,\ldots,Y_n$ (and given $c$).

(b) Load the dataset `censoredln.csv`.

(c) Numerically maximize the likelihood function. The censoring value is $c = 12$.

(d) Compute the asymptotic covariance matrix of $\hat{\mu}$ and $\hat{\sigma}^2$.

## 26 Dates and times

1. Use the command `as.Date` to generate the following dates or vectors of dates:
   (a) January 1st, 2012
   (b) November 9th, 1989
   (c) All Wednesdays in 2012
   (d) The first days of all months from January 2005 to December 2011.
   (e) The first days of all quarters from April 2000 to January 2012.
   (f) Use `difftime` to compute the number of days between your date of birth and today.

2. Use the command `strptime` to generate the following dates and times or vectors of dates and times of class `POSIXlt`:
   (a) January 19th, 2012, 08:45:00
   (b) January 19th, 2012, 12:00:00
   (c) All minutes between the times in (a) and (b)
   (d) What happens if you add 1 to a `POSIXlt` object?

3. Load the dataset `indices.csv` into a dataframe.
   (a) Use the command `strptime` to convert the first column of the dataframe into a vector of class `POSIXlt`.
   (b) Plot the DAX index as a black line against the date vector.
   (c) Compute the daily returns of the DAX index,

   $$r_t = \ln \frac{DAX_t}{DAX_{t-1}}$$

   and calculate the mean return for each weekday. Plot the mean returns for each weekday as a bar chart.
   (d) Use the function `holiday` of the `timeDate` package to identify the holidays at Deutsche Börse. Remove the holiday returns and redo the computations for the weekday returns.

# 27 Time series: Basics

Activate the `zoo` package.

1. Load the dataset `BIP.csv` into a dataframe. The first column is the GDP in current prices, the second column is the price deflated (chain) index of GDP with 2005=100.

   (a) The first observation is the first quarter of 2000, the last observation is the third quarter of 2011. Create a `ts` time series object of the GDP in current prices.

   (b) Print the time series.

   (c) Plot the time series.

   (d) Use the command `rollmean` to add a moving average of length $k = 4$ to the plot.

   (e) Plot the time series of the differences between GDP and the rolling mean.

   (f) Use the function `aggregate` with option `nfrequency=1` to compute the time series of the annual GDP values.

   (g) Plot the differences of the logarithm of the annual GDP values (i.e. the annual growth rates).

2. Load the dataset `m3.csv` into a dataframe. It contains the money aggregate M3 for the Euro area.[2] Remember to set the option `as.is=TRUE` in the `read.csv` command.

   (a) Create a `zoo` object of the M3 time series.

   (b) Print the time series, setting the option `style` to each of its three possible values, i.e. `horizontal`, `vertical`, and `plain`.

   (c) Plot the time series and add the rolling mean (with window width 12 months) in red.

   (d) Plot the deviation of the time series from its rolling mean.

   (e) Use the command `lag` to define the time series $M3_{t-12}$ (i.e. M3 in the same month of the previous year).

   (f) Plot $M3_t - M3_{t-12}$ and $\ln M3_t - \ln M3_{t-12}$.

3. Convert the M3 time series to class `ts`.

   (a) Read `?decompose`. Decompose the time series into a trend, a seasonal component, and a remainder term using the function `decompose`. Assign the function value to an object (which will be of class `decomposed.ts`), and plot this object.

   (b) Try both `type` options (i.e. `additive` and `multiplicative`). Which one would you prefer?

4. Continue to use the M3 time series.

   (a) Execute `acf(diff(M3))` and interpret the plot.

   (b) Returns from the `acf` command are of class `acf`, see `?acf`. Extract the autocorrelations of `diff(M3)` as a vector.

5. Consider the dataset `electricity.csv`. It contains hourly electricity prices (in EUR/MWh) at the EEX.

[2] https://stats.ecb.europa.eu/stats/download/bsi_ma_historical_nsa_dp/bsi_ma_historical_nsa_dp/bsi_hist_nsa_u2_2.pdf

(a) The time format is YYYY-MM-DD, HH:00. Read the dataset into a dataframe (using the option `as.is=TRUE`) and create a vector for the date-time information.

(b) Plot the time series of the electricity prices.

(c) Use the `acf` command to compute and display the autocorrelation function up to lag order 750.

(d) Convert the price time series into an `ts` object with start date 1 and end date 365 (and, of course, frequency 24). Use the command `aggregate` to calculate the average daily prices.

## 28 Time series: model estimation and unit roots

Activate the packages `tseries`, `fGarch`, and `vars`.

1. Load the artificial dataset `ar1.csv`. It contains two variables, the endogenous variable $Y$ and the exogenous variable $X$. The linear relationship $Y_t = \alpha + \beta X_t$ is disturbed by a first order autoregressive error term $u_t$ with

$$u_t = \rho u_{t-1} + \varepsilon_t$$

   where $\varepsilon_t \sim N(0, \sigma^2)$ and $|\rho| < 1$.

   (a) Run an OLS regression of $Y$ on $X$ and put the residuals into a vector `uhat`.

   (b) Use the command `ar` to estimate the autoregression coeffcient $\rho$ and the variance $\sigma^2$.

   (c) Extract the standard error of $\hat{\rho}$. Is $\hat{\rho}$ significantly positive?

2. Load the dataset `BIP.csv` and convert the first column (i.e. GDP in current prices) to class `ts`.[3]

   (a) Use the command `decompose` to calculate the trend, the saisonal pattern and the remainder term of the time series.

   (b) Fit an $AR(1)$ process to the remainder term (`$random`). In order to remove the missing values set the option `na.action=na.omit`.

   (c) Test if the first order autocorrelation coefficient is significantly different from zero.

   (d) Now use the command `arima` to fit an $AR(1)$ model to the remainder term.

   (e) Fit an $MA(1)$ model to the remainder term.

3. Load the dataset `investment.csv`. It contains the gross machinery investment (Bruttoausrüstungsinvestitionen) in current prices (in billion Euros).

   (a) Convert the investment time series to class `ts` and plot it in levels and percentage changes (differences of the logs). Ignore the structural break in 1990 due to Germany's unification.

   (b) Fit $ARIMA(p,1,q)$ models to the time series with $p, q \in \{0,1,2\}$. Find the lowest $AIC$ value of the 9 models.

   (c) Perform an $ADF$ test. Does investment exhibit a unit root?

   (d) Perform an $ADF$ test for the differenced time series of investment. Do the differences have a unit root?

4. Load the dataset `indices.csv` and compute the returns of the DAX index as

$$r_t^{DAX} = \ln \frac{DAX_t}{DAX_{t-1}}.$$

   (a) Fit a $GARCH(1,1)$ model to the returns by `garch`.

   (b) Fit a $GARCH(1,1)$ model by `garchFit` and compare your results with (a).

   (c) Add an $AR(2)$ mean equation.

[3] See exercise 27.1.

5. Consider the bivariate $VAR(1)$ process

$$y_t = \begin{bmatrix} 0.9 & 0 \\ 0.5 & 0.5 \end{bmatrix} y_{t-1} + \varepsilon_t, \qquad \varepsilon_t \sim N\left( \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0.3 \\ 0.3 & 1 \end{bmatrix} \right)$$

where $y_t = (y_{1t}, y_{2t})'$, see A. Staszewska-Bystrova (2011), Bootstrap Prediction Bands for Forecast Paths from Vector Autoregressive Models, *Journal of Forecasting*, 30: 721-735.

(a) Write an R program that performs the following steps.

- Set $R = 1000$.
- Initiate an empty matrix $Z$ of dimension $R \times 4$.
- For $r = 1,\ldots,R$: Set $y_1 = (0,0)$; simulate $y_t$ for $t = 2,\ldots,200$; drop the first 100 observations; use the `VAR` command of the `vars` package to estimate the four coefficients of the autoregression matrix; save the coefficients in row $r$ of matrix $Z$.

(b) Use the `apply` command to compute the mean estimates of the four coefficients. Are the VAR estimators biased?

(c) Plot the histograms of the four coefficients in a $2 \times 2$ plot.

# 29 Graphics

The following exercises rely on a Harvard University tutorial called "Introduction to R Graphics with ggplot2". The exercises show you how to investigate a dataset about housing prices visually using `ggplot2`.

1. Load the dataset `landdata-states.csv` and create a histogramm of Home.Value using `aes` and `geom_histogram`.
2. Redo the first exercise using an appropriate binwidth using `binwidth`.
3. Plot two countries of your choice over time with two different colours using `geom_point`, `subset` and `color`.
4. Load the package ggThemeAssist and polish your last graphic using `ggThemeAssistGadget(ggplot2-object)`.
5. Make a scatterplot of Land.Value in logs on the x-axis and Strcuture.Cost on the y-axis for the first quarter of 2001 using `geom_point` and `subset`. (We refer to this as the basis plot in the following.)
6. Add a fitted linear regression line to the basis plot and color the points with respect to Home.Value using `geom_smooth` and `color`.
7. Use a fitted polynomial instead of the regression line in the last plot.
8. Use the unique variable names in State as y-values in the basis plot using `geom_text`
9. Use different shapes due to the variable region with `shape` and different colours due to the variable Home.Value with `color` for the last plot.
10. Calculate the mean of Home.Value for every state using `aggregate` and plot your new data.frame using `geom_bar` and `stat = "identity"`.
11. Plot State on the x-axis and Home.Price.Index on the y-axis using `geom_point`. Use Date as color to distinguish the plot with respect to the time.
12. Use `alpha` and `size` to make the last plot clearer.
13. Use `scale_color_gradient` to change the color of the last plot.
14. Plot the Home.Value over time for every state in a different colour.
15. Divide the last plot in multiple plots each for every state using `facet_wrap`.