LABORATORY FOR APPLIED STATISTICS
UNIVERSITY OF COPENHAGEN

HELLE SØRENSEN
AUGUST 2016

Introduction to R: Exercises

This document is a collection of exercises made for the one-day course *Introduction to R* given by the Laboratory for Applied Statistics, Department of Mathematical Sciences, University of Copenhagen. Please read Section 1 about the purpose of the course and the suggested use of the exercises. Suggestions for improvements are most welcome. For example, send an email to `helle@math.ku.dk`.

Contents

1 About R, RStudio, the course, and the exercises

What is R and RStudio? R is a statistical software program. It has extremely useful tools for data exploration, data analysis, and data visualization. It is flexible and also allows for advanced programming. RStudio is a user interface for R, which provides a nice environment for working with R.

About the course and the exercises The course is aimed at researchers in natural sciences, who need tools for data exploration, data analysis, and data visualization, and who have no or little experience with R. The purposes of the course are to

- get the participants started with R, so they can start using R in their own work
- give the participants an impression about the possibilities with R

Course material The course material can be downloaded from

`http://www.statlab.math.ku.dk/english/courses/workshops/r-intro/`

and consists of the following:

- This document with exercises.
- Data files. All datsets used in exercises are saved as `csv` as well as `xlsx` files, and all the files are gathered in a zip-archive.
- R programs and R markdown files used for the presentations.

We recommend that you make a new folder for the course, put all files concerning the course in that folder, and use the folder as "working directory" in RStudio. The working directory can be changed in the Session menu.

About the exercises The exercises in this document are meant for the hands-on sessions in the course. Some important comments on the exercises:

- The collection of exercises does not constitute anything like a text-book type introduction to R, and the exercises cannot stand alone in the participants' later use of R.
- The exercises do not explain much (if any at all) of the statistics involved, so the participant should not work with the exercises on analysis of variance, say, if he or she is not familiar with that type of data analysis.
- There are far too many exercises for one day's work (probably even if you were an experienced R user). The idea is that participants can choose exercises according to their interests and experience with R, so do not drive yourself to despair if you only get through a few exercises during the day.

- Many questions are of the type "Try the following commands, and explain what happens". The last part is of course important to be able to similar commands in other settings. The participants are strongly encouraged to play with other commands along the way.
- Prerequisites are given in the beginning of each exercise in terms of previous exercises. You do not necessarily have to have solved all the exercises mentioned completely, but you would find useful commands here and there. Hopefully, I have succeeded in remembering the majority of the prerequisites, but I don't give any promises...
- Some exercises are marked with a star (*). Only few commands are written directly in these exercises, so the reader should know them from other exercises (or from somewhere else).

Installation of R and RStudio. R packages R and RStudio are avaliable for all platforms (Windows, Mac and Linux). They are open-source programs, and they can be downloaded from `http://www.r-project.org` and `http://www.rstudio.org`, respectively.

Apart from many functions that are immediately avalaible after installation, there exists a huge number of R packages, *i.e.* collections of functions written by users and made available to other users. Many of those packages can be downloaded and *installed* from the CRAN repository — most easily in RStudio via the Package menu in the lower right window. After installation the package must be *loaded* before the functions in the package are available for use, again via the Package menu in RStudio. Notice that packages should be installed one time only, but must be loaded in each R session where you need it.

Material about R There are many monographs on the use of R, as well as enormous amounts of online material. Among many others, the following books could be adequate for the course participants: Ekstrøm and Sørensen (2011) and Dalgaard (2008) are textbooks in (applied) statistics with comprehensive use of R. Both are written for introductory statistics courses, and thus cover basic statistical tools. Martinussen *et al.* (2012) is about basic use of R as well as statistical analysis with R. Some of the topics are relative advanced (random effects, repeated measures, survival analysis), and the reader is supposed to be somewhat familiar with the statistical methods in advance. Ekstrøm (2012) is a handbook answering many basic as well as advanced questions about R. Venables and Ripley (2002) is a standard reference among R users (although in principle written for the language `S` rather than `R`). It contains data examples on a wealth of statistical methods.

Many of the exercises in this note are inspired from material from these books.

2 Getting started

2.1 Working with R and RStudio

Prerequisites: None

You would like to save relevant commands for later use. Therefore you should write your commands in a file — a socalled R script or R program. The main part of this exercise learns you how to work with such files in RStudio.

1. Start RStudio. Go to the console (lower left window) and write

   ```
   3+2
   ```

   at the promt. Then use R to calculate $3 \cdot 4$ and $8/2 - 3 \cdot 4$ (use $*$ for multiplication).

   Commands written directly at the prompt are not saved for later use.

2. If there is not already an editor open in the upper left window, then go to the file menu and open a new script. Type one of the commands from before in the editor, hold the Control buttom and push the Enter buttom. Then the command is transferred to the console and the command is executed (just as before).

3. Make a few more commands in the editor, and run them. Also, mark several commands with the arrow buttoms and use Control-Enter.

4. Save the file, close the file, and quit RStudio (all via the File menu). You are asked if you want to save the workspace image. If you answer "Yes", then everything is saved — both relevant and irrelevant stuff — so unless you have made time-consuming computations, you should say "No".

5. Start RStudio again, and open the file you just saved (again via the File menu), and redo some of the calculations.

6. Try the command

   ```
   # 4+7
   ```

   Nothing is computed! This is because the character # can be used for comments. Anything after # is discarded by R. It cannot be recommended enough that you make appropriate comments in your R programs. That makes it much easier to use the programs again after a while.

2.2 Built-in mathematical functions

Prerequisites: None

All standard functions (and many non-standard, too) are programmed in R. Here come some examples.

1. Try the commands `sqrt(16)`, `16^0.5`. Compute 4^3.

2. Try the commands `log10(1000)`, `log(1000)`, `exp(log(1000))`. Then try the command `log2(64)`. Make sure you understand different logarithmic functions.

3. Try the command `?log`. Then a help page opens in the lower right window of RStudio. Read the first few lines; does the text match your observations from the previous question?

4. Try the commands `pi`, `round(pi)`, `round(pi, digits=4)`, and `trunc(pi)`.

5. The sine and cosine functions are implemented in `sin` and `cos`. Calculate $\sin(\pi)$, $\cos(\pi)$, $\sin(\pi/2)$, $\cos(\pi/2)$.

2.3 Vectors

Prerequisites: None

1. Try the commands:

```
x <- c(3,6,8)
x
x/2
x^2
sqrt(x)

x[2]
x[c(1,3)]
x[-3]
y <- c(2,5,1)
y
x-y
x*y

x[y>1.5]
y[x==6]

4:10
seq(2,3,by=0.1)
rep(x,each=4)
```

   An important point is that operations are carried out elementwise!

2. Assume that we have registered the height and weight for four people: Heights in cm are 180, 165, 160, 193; weights in kg are 87, 58, 65, 100. Make two vectors, `height` and `weight`, with the data. The bodymass index (BMI) is defined as

$$\frac{\text{weight in kg}}{(\text{height in m})^2}$$

Make a vector with the BMI values for the four people, and a vector with the natural logarithm to the BMI values. Finally make a vector with the weights for those people who have a BMI larger than 25.

2.4 Simple summary statistics

Prerequisites: None

In an experiment the dry weight has been measured for 8 plants grown under certain conditions (the values are given below).

1. Try the following commands in order to make a vector with the values and compute various sample statistics:

```
dry <- c(77, 93, 92, 68, 88, 75, 100)
dry

sum(dry)
length(dry)
mean(dry)
sum(dry)/length(dry)                    ## Checking

sort(dry)
median(dry)

sd(dry)
var(dry)
sd(dry)^2
sum((dry-mean(dry))^2) / (length(dry)-1) ## Checking

min(dry)
max(dry)

summary(dry)
```

2.5 Vectors and sample statistics (more examples)*

Prerequisites: Exercises 2.3 (vectors) and 2.4 (summary statistics)

1. Assume that we have the following three observations of temperature: 23°C, 27°C, 19°C. Make a vector with these values. Recall the relation between the Celcius and Fahrenheit temperature scale:

$$\text{degress in Fahrenheit} = \text{degrees in Celcisus} \cdot \frac{9}{5} + 32$$

Make a new vector with the temperatures in Fahrenheit.

2. Assume that you are interested in cone-shaped structures, and have measured the height and radius of 6 cones. Make vectors with these values as follows:

```
R <- c(2.27, 1.98, 1.69, 1.88, 1.64, 2.14)
H <- c(8.28, 8.04, 9.06, 8.70, 7.58, 8.34)
```

   Recall that the volume of a cone with radius R and height H is given by $\frac{1}{3}\pi R^2 H$. Make a vector with the volumes of the 6 cones.

3. Compute the mean, median and standard deviation of the cone volumes. Compute also the mean of volume for the cones with a height less than 8.5.

2.6 Help pages

Prerequisites: None

There are help pages for every function i R, but, admittedly, they are not always too easy to read for an unexprerienced R user.

1. Try the command `?which.min`. This opens a help page in the lower right window of RStudio. What does the function do?

2. Try the commands

```
which.min(c(2,5,1,7,8))
which.max(c(2,5,1,1,8))
```

   Is the output as expected?

3. You must know the name of the function in order to open the help page as above. Sometimes (often, even) you do not know the name of the R functions; then google can often help you. Try, for example, to search the text `R minimum vector`.

3 Reading data from files and working with datasets

3.1 Reading data from csv files, inspection of data

Prerequisites: None

Suppose that your data is saved as a `csv` file (csv stand for comma-separated values). We need to read the data into R. Consider an experiment where girth, height and volume has meen measured for 31 cherry trees. The data are saved in the file `cherry.csv`.

1. Open the file `cherry.csv` in Excel (or a similar application), and make sure you understand the structure of the file.

2. Go to R, use the command

   ```
   cherry <- read.csv(file.choose(), dec='.', sep=';')
   ```

   Another window pops up, and you must click your way through to the relevant `csv` file.

   Once the dataset has been constructed, the name `cherry` appears in the upper right box in RStudio. Click the name, and you can see the content in the upper left box.

   Notice how you can change the characters for decimal comma and field separator according to what is used in the `csv` file, so the file is interpreted in the correct way (more about this below).

3. Whenever you have constructed a new dataset (or made changes in an existing one) it is extremely important that you check that it has been constructed correctly. The following commands may be useful for that purpose:

   ```
   head(cherry)
   plot(cherry)
   summary(cherry)
   ```

   Try the commands!

4. Did you get lists of minimum, maximum, mean, *etc.*, for all three variables?

   If not, it is because R has coded some of the variables as categorical variables rather than numerical variables (R reads a number with decimals as a text string). Luckily this can be changed with the `dec` option in `read.csv`. Try instead

   ```
   cherry1 <- read.csv(file.choose(), dec=',', sep=';')
   summary(cherry1)
   ```

The question about coding is important, so let us say it a bit more specifically: A common problem is that one or more numerical variables have been coded as categorical variables because the decimal seperator has not been interpreted correctly. Then R reads a number with decimals as a text string. This causes all sorts of trouble, and the sooner you detect it, the better. The summary of numerical variable lists the minimum, maximum, mean, *etc.*, for the

variable. If, instead, you get information about how many times specific values occur in the dataset, then your variable is coded as a categorical value, and you have to change the `dec` option in your `read.csv` command.

5. (Optional) You can write the file name instead of `file.choose()`, such that you do not have to click your way through to the file. The path to the file must be given, starting from the current working directory. If the file is in the current working directory, then the relevant command is

   ```
   cherry2 <- read.csv('cherry.csv', dec='.', sep=';')
   ```

   You can change the working directory in the Session menu (Set working directory).

6. (Optional) Consider an experiment with tomatoes. Three different varieties and four different seed densities have been tested, and there are three replications for each of the 12 combinations. The yield has been registered for each of the 36 field plots. The data are saved in the file `tomatoes.csv`

   Create a dataset called `tomatoes`, say, in R, and inspect the data with `summary`. Which variables are quantitative and which variables are categorical?

3.2 Reading data from Excel files

Prerequisites: Exercise 3.1

You probably most often save your data in Excel (or something similar), and therefore need to "transfer" your data from Excel to R. There are several ways of doing this; here comes two if them:

(a) Read the Excel file directly into R with the function `read.xlsx`. This is the nicest solution; however problems now and then occur due to different versions of Excel and different computer systems. It also requires that you have the rights to install add-on packages on your computer (previously this was sometimes a problem on SCIENCE-PC's).

(b) Save your Excel sheet as a comma separated file (`.csv`), and use the function `read.csv` to read the file as described in Exercise 3.1. Make sure that the decimal seperator and the field seperator in `read.csv` match those used to save the `csv` file.

Let us consider the same data as in Exercise 3.1: Girth, height and volume has meen measured for 31 cherry trees. The data are saved in the file `cherry.xlsx`.

We first try method (a). Here is something that works on my computer:

1. First install the package `xlsx` via the Package menu (this requires connection to the internet). Second load the package via the Package menu. See Section 1 for more information about R packages.

2. Use the command

```
cherry3 <- read.xlsx(file.choose(), sheetIndex=1)
```

and choose the relevant file.

3. Inspect the dataset `cherry3`. In particular, make sure that all variable are coded correctly as numerial variables.

If method (a) did not work, then try method (b):

4. Open the file `cherry.xlsx` in Excel. Go to the file menu, and save the data as a `csv` file (use `Save As`). As a starting point, you can use the default options for delimiters.

5. Go to R and use the command

```
cherry4 <- read.csv(file.choose(), dec='.', sep=';')
```

and choose the `csv` file that you just created.

6. Inspect the dataset `cherry4`. In particular, make sure that all variable are coded correctly as numerial variables. If this is not the case, then try to modify the `dec` and `sep` options so they match those from the `csv` file.

3.3 Working with datasets: Inspection, using variables, attaching

Prerequisites: Exercise 3.1 (reading the cherry data)

Consider the dataset `cherry` from Exercise 3.1 or 3.2.

1. *(Inspection of data)* Click the name `cherry` in the upper right box in RStudio (as you probably already did in Exercise 3.1). Then try the following commands (one at a time) and explain what you see:

```
head(cherry)
cherry
plot(cherry)
summary(cherry)
```

2. *(The $ syntax)* Try the following commands one at a time; notice that the first two give you an error message:

```
Girth
hist(Girth)
cherry$Girth
mean(cherry$Girth)
hist(cherry$Girth)
```

3. *(The* `with` *function)* Try the following commands:

```
with(cherry, hist(Height))
with(cherry, mean(Height))
```

4. *(Attaching a dataset)* Try the following commands one at a time:

```
attach(cherry)
Girth
mean(Girth)
hist(Girth)
plot(Height, Volume)
detach(cherry)
Girth
```

The point of questions 2–4 is that you need to tell R where to look for the variables. You do so either by using the $ syntax every time you need the variable (which quickly becomes tendersome), by using the `with` function "outside the command", or by attaching the dataset. If you attach datasets, you should be very careful not to have several datasets with the same variable names attached at the same time. This can be very dangerous because it is not always obvious which dataset is then used.

5. *(Structure of datasets)* Make sure you understand the structure of a dataset. More specifically, for example: What is the difference between a variable and a dataset? What do you see in the different rows of a dataset? What do you see in the different columns in a dataset?

3.4 Transformation of variables, subsets of datasets

Prerequisites: Exercise 3.1 (reading the cherry data)

Consider the dataset `cherry` from Exercise 3.1 or 3.2.

1. *(Transformation of variables in a dataset)* Construct a new dataset, `cherry1`, with the command

```
cherry1 <- transform(cherry, logVolume=log(Volume), logGirth=log(Girth))
```

and notice that the name appears in the upper right box of RStudio. Inspect the new dataset, either by clicking on the name in the upper right box, or by one of the commands

```
cherry1
head(cherry1)
hist(cherry1$logVolume)
```

Notice that `log` is the natural logarithm. If you prefer the logarithm with base 2 or 10, you should use the functions `log2` or `log10` instead og `log`.

2. *(Subset of a dataset)* Try the following commands (one at a time). Make sure you understand the output.

```
cherry[3,]
cherry[3:5,]
cherry[-c(2,4),]
```

Construct and inspect each of the following datasets (one at at time). Make sure you understand the content of each dataset.

```
subset(cherry, Height>70)
subset(cherry, Height>=70)
subset(cherry, Height==80)
subset(cherry, Height==80, select=c(Girth,Volume))
subset(cherry, Height>80 & Girth>15)
subset(cherry, Height>80 | Girth>15)
```

3.5 Merging datasets

Prerequisites: Exercise 3.1 (reading the cherry data)

It sometimes happens that data comes from different sources (files), and should be merged before analysis. Consider the dataset `cherry` from Exercise 3.1. Always check that the datasets are as you supposed them to be!

1. *(Merging data, new datalines)* Assume that data from two more trees are made available. Try the following commands that construct a dataset with the new data and merge the two datasets:

```
newData <- data.frame(Girth=c(11.5, 17.0), Height=c(71, 75), Volume=c(22, 40))
newData
allData <- rbind(cherry, newData)
allData
```

2. *(Merging data, new variables)* Assume that the precipitation at the location of each tree has also been registered and saved in a variable called `precipitation`. The variable below contains random numbers with mean 50 and standard deviation 10, and are meaningless — yet the commands illustrate the merging of a new variable and an existing dataset.

```
precipitation <- rnorm(n=31, mean=50, sd=10)
precipitation
allData2 <- cbind(cherry, precipitation)
allData2
```

Of course the variables should be ordered the same way in the dataset and the new variable. See (Martinussen *et al.*, 2012, Section 4.2) for more advanced merging using the `merge` function.

3.6 Working with data: Tomato yields*

Prerequisites: Exercises 3.1, 3.3, 3.4 (reading and working with datasets)

Consider an experiment with tomatoes. Three different varieties and four different seed densities have been tested, and there are three replications for each of the 12 combinations. The yield has been registered for each of the 36 field plots. The data are saved in the file `tomatoes.xlsx`

1. Create a dataset called `tomatoes`, say, in R, and inspect the data with the functions `plot` and `summary`. Which variables are quantitative and which variables are categorical?
2. Make a histogram of the yield variable. Moreover, compute the mean, median and standard deviation of the yield variable.
3. Try the following commands, and explain the output:

```
table(variety)
table(density)
table(density, variety)
```

4. Make a new dataset with two more variables: one with the squareroot of the yield values and one with the logarithm of the yield values.
5. Make a dataset which only contains the datalines corresponding to the variety Ife (use `variety=='Ife'` in a `subset` command).
6. Make a dataset containing datalines for the variety Pusa with density less than 25000. What is the median of the corresponding yield values?
7. Make a dataset without observation numbers 5 and 24. Make a histogram of the yield values from this dataset.

4 Graphs

4.1 Basic high-level plots

Prerequisites: Exercises 3.1 and 3.3 (reading and working with the cherry data)

This exercise introduces some basic high-level plots, which can be produced with very simple commands. Here we use the default options (which are often fine), but it is an important point that graphs can easily be modified, see Exercise 4.2.

Consider first the dataset `cherry` from Exercise 3.1, and recall that it has variables `Girth`, `Height` and `Volume`.

1. *(Scatterplots)* The most important graphics function `plot` function. It does different things depending on the type of argument(s) supplied to the function.

   Try the commands and explain what you see:

   ```
   plot(cherry)
   plot(Girth, Volume)
   plot(log(Girth), log(Volume))
   plot(Height)
   ```

2. *(Histograms and boxplots)* Try the commands

   ```
   hist(Volume)
   boxplot(Volume)
   ```

3. *(Barplots)* The temperature in New York was measured daily for five month (May to September). The average temperatures in degrees Fahrenheit were 65.5, 79.1, 83.9, 84.0, and 76.9, respectively. Try the commands

   ```
   temp <- c(65.5, 79.1, 83.9, 84.0, 76.9)
   barplot(temp, names=5:9)
   ```

4.2 Modifications of scatterplots

Prerequisites: Exercises 3.1 and 3.3 (reading and working with data)

The temperature in New York was measured daily for five month (May to September). The data from July and August are saved in the files `ny-temp.xlsx` and `ny-temp.csv`. There are three variable: `Month` with values 7 and 8, `Day` with values 1–31, and `Temp` with the temperature in degrees Fahrenheit.

1. Read the data into a dataset, `NYtemp`. Then make two sub-dataset, `july` and `august` with the data from July and August, respectively.

2. *(Plotting symbols, lines, axes, title)* Try the following commands and explain what happens:

```
attach(july)
plot(Day, Temp)
plot(Day, Temp, pch=16)
plot(Day, Temp, pch=2)
plot(Day, Temp, type="l")
plot(Day, Temp, type="b")
plot(Day, Temp, type="l", lty=2)
plot(Day, Temp, type="l", lwd=2)
plot(Day, Temp, type="l", col="blue")
plot(Day, Temp, type="l", xlab="Day in month", ylab="Temperature (F)",
     main="Temperatur in New York")
plot(Day, Temp, type="l", cex.lab=1.3)
plot(Day, Temp, type="l", cex.axis=1.3)
plot(Day, Temp, type="l", xlim=c(0,40), ylim=c(50,100))
detach(july)
```

3. *(Adding data points)* Try the following commands and explain what you see:

```
range(july$Temp)
range(august$Temp)
plot(july$Day, july$Temp, type="l", ylim=c(70,100))
points(august$Day, august$Temp, col="red")
plot(july$Day, july$Temp, type="l", ylim=c(70,120))
lines(august$Day, august$Temp, col="green")
```

4. *(Legends)* Add a legend to the previous plot:

```
legend(5,120, c("July", "August"), text.col=c("Black","Green"))
```

5. *(Adding straight lines)* Try the commands:

```
plot(july$Day, july$Temp, type="l", ylim=c(70,100))
abline(h=73, col="red")
abline(v=14, col="blue")
abline(84.652, -0.0468)
```

   Notice how the final command adds a line to the plot with intercept and slope as specified in the command. The actual values are the estimates from a linear regression, see Exercise 5.1, question 1, on how to make R compute those numbers.

6. *(Points coloured according to third variable)* In the R package `MASS` there is a dataset called `cats` with variables `Sex`, `Bwt` (body weight in kg), `Hwt` (in g). Run the following commands:

```
library(MASS)
data(cats)
plot(cats)
```

   Then try the command and explain what you see:

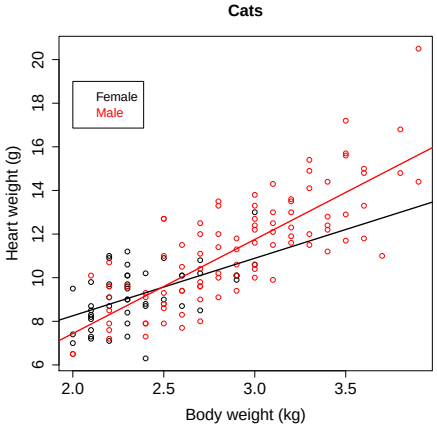

Figure 1: Scatterplot with regression lines for male and female cats.

```
plot(cats$Bwt, cats$Hwt, col=cats$Sex)
```

7. Modify the figure such that it looks similar to Figure 1. The lines are regression lines for linear regression models fitted to male and female cats, respectively, namely

$$\text{Male cats:} \quad \texttt{HWt} = -1.184 + 4.313 \cdot \texttt{Bwt}$$
$$\text{Female cats:} \quad \texttt{HWt} = 2.981 + 2.636 \cdot \texttt{Bwt}$$

   (See Exercise 5.1, question 1, on how to fit such linear regressions).

4.3 Modifications of histograms, parallel boxplots

Prerequisites: None

1. In the R package `MASS` there is a dataset called `cats`. Run the following commands:

```
library(MASS)
data(cats)
plot(cats)
```

   The variables `Bwt` and `Hwt` contain the weight of the body (kg) and the heart (g), respectively. There are both male and female cats.

2. *(Modification of histograms)* Try the following commands and explain what happens:

```
attach(cats)
hist(Hwt[Sex=="M"])
hist(Hwt[Sex=="M"], prob=T)
hist(Hwt[Sex=="M"], breaks=c(5,10,15,20,25))  ## Not nice
```

   Use the `xlab` and `main` arguments to change the x-label and the title to something more appropriate.

3. *(Parallell boxplots)* Try the following commands and explain what happens:

```
boxplot(Hwt[Sex=="F"])
boxplot(Hwt[Sex=="M"])
boxplot(Hwt ~ Sex)
```

4.4 Exporting graphics

Prerequisites: Exercise 4.2 (the temperature data)

Consider the data from Exercise 4.2.

There are (at least) two different ways to save your plots as files as `pdf` files that can be used for other documents. The plots can also be saved in other formats, and it is possible to set the size (width and height) of the graphs if the default is not satisfactory.

1. *(Save an existing plot)* Try the following commands:

```
plot(july$Day, july$Temp)
dev.print(file="plotfile1.pdf", device=pdf)
```

   The file `plotfile1.pdf` should now appear in the working directory. You can get information about the current working directory with the command `getwd()`. Find the file and open it.

2. *(Plot directly in a pdf file)* Try the following commands:

```
pdf("plotfile3.pdf")
plot(july$Day, july$Temp, col="blue")
dev.off()
```

   The commands open a file in the working directory, makes the graph in the file, and closes the file again. Find the file and open it.

5 Regression

5.1 Simple and multiple linear regression: Cherry data

Prerequisites: Exercises 3.1 and 3.3 (reading and working with the cherry data)

Consider the dataset `cherry` from Exercise 3.1. We will fit an examine a linear regression modelling the expected value of volume as a linear function of girth.

1. (*Fit of model and plots*) Use the following commands:

   ```
   plot(cherry$Girth, cherry$Volume)
   lm(Volume ~ Girth, data=cherry)
   ```

   This gives you the estimated intercept and slope. Notice how we specify the dataset; then it is not necessary to attach the data or use the $ syntax (cf. Exercise 3.3.) It is often convenient to give the object with the fit a name. Try

   ```
   linreg1 <- lm(Volume ~ Girth, data=cherry)
   linreg1
   abline(linreg1)
   ```

2. (*Estimates, standard errors, tests*) We need more information than just the estimated coefficients. Try

   ```
   summary(linreg1)
   ```

   Some explanation of the output is probably needed. The "Coefficients" part is the most important, and it is important to understand its structure. It has two lines — one for each parameter in the model — and four columns. The first line is about the intercept, the second line is about the slope. The columns are

   - `Estimate`: The estimated value of the parameter (intercept or slope)
   - `Std. Error`: The standard error (estimated standard deviation) associated with the estimate in the first column
   - `t value`: The t-test statistic for the hypothesis that the corresponding parameter is zero. Computed as the estimate divided by the standard error.
   - `Pr(>|t|)`: The p-value associated with the hypothesis just mentioned. In particular the p-value in the second line is for the hypothesis that there is no effect of girth on volume.

   Below the "Coefficients" part you find, among others, the "Residual standard error", *i.e.* the estimated standard deviation for the observations.

   Identify the estimates and corresponding standard errors in the output. Is there a significant effect of girth on volume? What is the estimated standard deviation for the observations?

3. (*Confidence intervals*) Try the command `confint(linreg1)`, which gives you 95% confidence intervals for the intercept and slope.

4. (*Model validation*) The easiest way to make model validation plots for the model fit is as follows:

   ```
   par(mfrow=c(2,2))
   plot(linreg1)
   ```

   Notice how the command `par(mfrow=c(2,2))` splits the plot window into 4 subplots.

   Does the model seem to be appropriate for the data?

   Fitted values, raw residuals and standardized residuals from the fit are extracted with the functions `fitted`, `residuals` and `rstandard`, respectively, so the classical model validation plots can also be obtained as follows (the commands with `abline` add relevant lines to the plots):

   ```
   plot(fitted(linreg1), residuals(linreg1))
   abline(h=0)
   plot(fitted(linreg1), rstandard(linreg1))
   abline(h=0)
   qqnorm(rstandard(linreg1))
   abline(0,1)
   ```

5. (*Transformation*) Fit a new linear regression model where you use `log(Volume)` as the response variable and `log(Girth)` as covariate.

   Is the model appropriate for the data? Is the effect of `log(Girth)` significant? What is the estimated relation between `log(Girth)` and `log(Volume)`? Which relation between `Girth` and `Volume` does this correspond to?

6. (*Multiple linear regression*) Fit a multiple linear regresison model where `log(Girth)` as well as `log(Height)` are used as covariate as follows:

   ```
   linreg3 <- lm(log(Volume) ~ log(Girth) + log(Height), data=cherry)
   ```

   Are both covariates significant (use `summary`)? Finally try the command

   ```
   anova(linreg3, linreg2)
   ```

   which carries out the F-test for comparison of the two models. Did you see the p-value before (explain where and why)?

5.2 Simple linear regression, prediction: Heart and body weights*

Prerequisites: Exercises 3.3 (working with datasets) and 5.1 (linear regression)

1. In the R package `MASS` there is a dataset called `cats`. Run the following commands:

```
library(MASS)
data(cats)
```

Have a look at the dataset. The variables `Bwt` and `Hwt` give the weight of the body (kg) and the heart (g), respectively. There are both male and female cats. Make a dataset with the data from males only.

2. Make a scatterplot of the data for the male cats (`Bwt` on x-axis, `Hwt` on y-axis). Does it look reasonable to use a linear regression model for the data?

3. Fit a linear regresison model for the male cats, that allows for prediction of the heart weight given the body weight. Add the fitted regression line to the scatterplot from the previous question.

4. Find the coefficients of the fitted line. How large is the expected difference in heart weight for two cats with a difference of 1 kg in bodyweight? Find a confidence interval for this difference? How large is the expected difference in heart weight for two cats with a difference of 100 g in bodyweight?

5. Use model validation plot to examine if the model is appropriate for the data.

6. Use the estimates to find the expected heart weight for a male cat that weighs 3 kg. Then try the commands (where you replace the name `regModel` with whatever name you gave the the model fit in question 2).

```
newObs <- data.frame(Bwt=3)
newObs
predict(regModel, newObs)
predict(regModel, newObs, interval="predict")
```

5.3 Multiple linear regression: Toxicity of dissolutions*

Prerequisites: Exercises 3.1, 3.3 (reading and working with data) and 5.1 (linear regression)

Data from 24 chemical dissolutions have been collected in order to examine the association between the toxicity of the dissolution on the one side and three explanatory variables on the other side. The data are saved in the files `lser.xlsx` and `lser.csv` with the following variables:

- `tox`: Toxicity of the dissolution
- `base`: Ablility to accept hydrogen ions
- `acid`: Ablility to liberate hydrogen ions
- `colour`: Ability to change colour

1. Make an R dataset called `lser` with the data, and use the command `plot(lser)` to get an overview of the data.

2. Fit a multiple linear regression model with `tox` as response and `base`, `acid` and `colour` as explanatory variables. Is the model appropriate for the data? What is the interpretation of the parameter estimates?

3. Are all three explanatory variables significant? Remove insignificant variables (one at a time) until all terms are significant.

4. Calculate the expected toxicity for a solvent which has `base=0.60`, `acid=0.95`, and `colour=0.52`.

5.4 Nonlinear regression: Enzyme kinetics

Prerequisites: Exercises 3.1, 3.3 (reading and working with data), and 4.1 (scatter plot)

In a chemical experiment the enzyme activity has been measured for different concentrations of the substrate and different concentrations of an inhibitor. The enzyme activity is measured as a reaction rate. There are three measurement series corresponding to three concentrations of the inhibitor (no inhibitor, 50 μM, 100 μM). For each series the reaction rate has been measured for 6 different substrate concentrations ranging from 10 μM to 600 μM. There are two replications and therefore 36 observations in total.

Data are available in the files `inhib.xlsx` and `inhib.csv` with the following variables:

- `S`: Concentration of substrate
- `I`: Concentration of inhibitor
- `R`: Reaction rate

1. Make a scatterplot of the data with `S` on the x-axis and `R` on the y-axis. Why is this plot not very illustrative? Try the commands

```
grp <- c(rep(1,times=12), rep(2,times=12), rep(3,times=12))
plot(S, R, col=grp)
plot(S, R, pch=grp)
```

When there is no inhibitor, the association between substrate concentration and reaction rate is most often described by the so-called Michaelis-Menten relation:

$$R \approx \frac{V_{\max} \cdot S}{K + S} \tag{1}$$

where $V_{\max}$ and K are parameters that should be estimated from the data. This is typically done with least squares. The function `nls` can do this, but needs starting values for the parameters.

2. Make a dataset, `dat0`, which only contains the data with no inhibitor (`I`=0). Then try the commands

```
mm0 <- nls(R ~ Vmax * S / (K + S), start = list(Vmax=3, K=100), data=dat0)
summary(mm0)
```

Make sure you understand the output.

3. It is not possible to extract standardized residuals from a `nls` fit, so we have to evaluate the raw residuals instead. The residualplot with raw residuals against fitted values is made in the usual way:

   ```
   plot(fitted(mm0), residuals(mm0))
   ```

   Does the model seem to be appropriare for the data?

4. Make a scatterplot for the data with no inhibitor. Then try the following commands:

   ```
   f <- function(S) 2.9811 * S / (35.802 + S)
   plot(f, from=0, to=620, add=T)
   ```

   The first command defines the fitted function, and the second adds a graph to the scatterplot.

Now we want to make fit a model to the complete dataset. Consider the association

$$R \approx \frac{V_{\max} \cdot S}{K_1 \cdot (1 + I/K_2) + S} \tag{2}$$

between inhihitor concentration, substrate concentration and reaction rate. Here $V_{\max}$, K_1, and K_2 are parameters to be estimated from the data.

5. Fit the model with `nls`. You can for example use $V_{\max} = 3$, $K_1 = 100$, and $K_2 = 25$ as starting values. What are the estimated coefficients?

6. Make the scatterplot from question 1 again, with different colours for different inhihitor concentrations. Add three fitted curves to the plot, one for each inhibitor concentration. Use the same colours for the graphs as you did for the datapoints.

7. Alternatively, we could use separate Michaelis-Menten models for each inhibitor concentration, *i.e.* use (1), but with three different values of $V_{\max}$ and K. This model can be fitted with the command

   ```
   mm2 <- nls(R ~ Vmax[grp] * S / (K[grp] + S),
              start = list(Vmax=c(3,3,3), K=c(100,100,100)), data=inhib)
   ```

   Notice that starting values are needed for each parameter in the model. Fit the model.

8. The model given by (2) is nested in the model just fitted (explain why), and the models can be compared with an F-test using `anova`. If `mm1` is the model from question 5, then try

   ```
   anova(mm1, mm2)
   ```

   Is the model given by (2) appropriate for the data?

5.5 Logistic regression: Pneumoconiosis among coalminers

Prerequisites: None (although Exercise 5.1 is perhaps useful for understanding the output)

Binary variables are variables with two possible outcomes, for example dead or alive, healthy or sick, germinated or not. Logistic regression is relevant when the response variable is binary. The aim is to relate the probabilities of the two outcomes to one or more explanatory variables. In the simplest case with one continuous covariate, x, and binary response, y, the logistic regression model assumes that log-odds is a linear function of x. Let $p = P(y = 1)$, and recall that the odds is defined as $P(y = 1)/P(y = 0) = p/(1-p)$. Then the assumption is

$$\log\left(\frac{p}{1-p}\right) = \alpha + \beta x. \tag{3}$$

Just as in ordinary linear regression (with continuous response) the first aim is to estimate the parameters α and β.

This exercise is about pneumoconiosis among coalface workers. Data were collected in order to examine the relationship between exposure time (years) and risk of disease. Severity of disease was originally rated into three categories, but here we will use only two (normal and diseased):

Exposure time	Normal	Diseased
5.8	98	0
15	51	3
21.5	34	9
27.5	35	13
33.5	32	19
39.5	23	15
46	12	16
51.5	4	7

The data are saved in the files `coalworker1.xlsx` and `coalworker1.csv`.

Notice how, for each person, it has been observed whether he was diseased or not. This corresponds to a binary variable. In the following we will consider a model with log-exposure as covariate, *i.e.* with `log(exposure)` as the x-variable in (3).

1. Read the data into R; call the dataset `coalworker1`. Then try the following commands and make sure you understand what happens:

   ```
   total <- normal + diseased
   relativeFreq <- diseased/total
   logOdds <- log(relativeFreq / (1-relativeFreq))
   plot(log(exposure), logOdds)
   ```

   What does the plot tell you about the appropriateness of the logistic regression model with log-exposure as explanatory variable?

2. Fit the logistic regression model with the following commands:

```
status <- matrix(c(diseased,normal), ncol=2)
status
logreg1 <- glm(status~log(exposure), family=binomial)
```

Notice the option `family=binomial`. It tells R to interpret values in the matrix `status` as outcomes from a binomial distribution.

3. Try the commands

```
summary(logreg1)
abline(logreg1)
```

What are the estimated values of α and β? Does the model seem to fit the data reasonably well?

Above, the data was represented by the numbers of diseased and normal for each level of exposure with 8 observations in total. The data could also be represented with one observation per person (371 observations in total). The files `coalworker2.xlsx` and `coalworker2.csv` contain the data in this representation with the variable `y` telling whether the person was diseased (`y=1`) or normal (`y=0`). Apart from this there is as variable `exposure2` with the exposure time.

4. Read the data into R in the new form; call the dataset `coalworker2`. Make sure you understand the structure of the new dataset, and make sure you understand that `coalworker1` and `coalworker2` contain the same information.

5. When the data is represented as in `coalworker2`, the logistic regression model is fitted as follows:

```
logreg2 <- glm(y ~ log(exposure2), family=binomial, data=coalworker2)
summary(logreg2)
```

Fit the model and make sure that you get the same estimates as you did with `logreg1` before.

The last questions are about the interpretation of the estimates and the models.

6. Consider a person with exposure time equal to 30 years. What is the estimated log-odds of this person being diseased? What is the estimated probability that the person is diseased?

Hints: For the estimation of log-odds, remember that the explanatory variable is the log-transformed exposure time. For the probability you should solve expression (3) for p.

7. How many years of exposure gives a 50% risk of having developed the disease?

Hint: We are looking for the exposure time corresponding to $p = 0.5$. What is the corresponding value of log-odds? Use the estimates to compute the (logarithmic) exposure time.

8. What happens with the odds if the exposure time is doubled?

   Hint: What happens to log-exposure if exposure time is doubled? What is the effect on log-odds? Which effect on odds does that correpsond to?

6 Analysis of variance (ANOVA)

6.1 Oneway analysis of variance: Psoriasis

Prerequisites: Exercises 3.1 and 3.3 (reading and working with datasets)

Psoriasis is an immune-mediated disease that affects the skin. Researchers carried out an microarray experiment with skin from 37 people in order to examine a potential association between the disease and a certain gene (IGFL4). For each of the 37 samples the gene expression was measured as an intensity.

There were three different types of skin samples: 15 skin samples were from psoriasis patients and from a part of the body affected by the disease (`psor`); 15 samples were from psoriasis patients but from a part of the body not affected by the disease (`psne`); and 7 skin samples were from healthy people (`control`). The data is saved in the files `psoriasis.xlsx` and `psoriasis.csv` with variables `intensity` and `type`. There is also a variable `typeNum`, which is not used until the very last question.

The scientific question is whether the gene expression level differs between the three types/groups, and the natural type of analysis is thus a oneway analysis of variance (ANOVA).

1. *(Stripchart, boxplots)* Try the following commands to get an overview of the data, and explain what you learn from the plots:

   ```
   stripchart(intensity ~ type)
   boxplot(intensity ~ type)
   ```

2. *(Group means)* It is well-known that the group means are essential ingredients in the analysis. Find the groups means with the commands

   ```
   mean(intensity[type=="healthy"])
   mean(intensity[type=="psne"])
   mean(intensity[type=="psor"])
   ```

   Notice that you would rarely do this as part of the analysis, but the values are useful to understand what happens in the next questions.

3. *(Fit of oneway ANOVA, reference group)* The model for oneway analysis of variance is fitted with

   ```
   oneway1 <- lm(intensity ~ type, data=psoriasis)
   ```

   Notice that this is analogous to fitting a simple linear regression: `intensity` is the response variable; `type` is an explanatory variable.

   Then make a summary of the model:

   ```
   summary(oneway1)
   ```

Some explanation is most likely useful at this point. The "Coefficients" part of the output has three lines — one per group or parameter in the model.

The lines are denoted `(Intercept)`, denoted `typepsne` and `typepsor`. No line is denoted `typehealthy`. This is because R has selected `healthy` as the reference group, and compares the other groups to this reference group. More specifically the `(Intercept)` line concerns the expected value of the reference group, so the estimate is simple the mean of the 7 observations from healthy people, whereas the estimate in the `typepsne` line is the difference between the mean of the `psne` observations and the `healthy` observations. Similarly for the `typepsor` line.

All four values in the same line contain information about the estimation of the same parameter:

- `Estimate`: The estimated value of the parameter (mean for reference group or difference between group mean and reference group mean for the other groups)
- `Std. Error`: The standard error associated with the estimate in the first column
- `t value`: The t-test statistic for the hypothesis that the corresponding parameter is zero. Computed as the estimate divided by the standard error.
- `Pr(>|t|)`: The p-value associated with the hypothesis just mentioned.

In the `typepsne` line, for example, you find the estimate of the difference between the `psor` and `healthy` group, the correponding standard error, as well as information about the test for the hypothesis that this difference is zero, *i.e.* that the expected value is the same for `psne` observations and `healthy` observations.

At first glance, the parameterization may seem may seem annoying, but there is a point: Differences between groups means — not the groups means themselves — are the primary matters of interest in a oneway ANOVA, and the above version of the model gives directly the interesting quantities regarding comparison to the reference group.

Below the "Coefficients" part you find, among others, the "Residual standard error", *i.e.* the estimated standard deviation for the observations (not the parametere estimates).

4. *(Parameterization in terms of group means)* Try

```
oneway2 <- lm(intensity ~ type -1, data=psoriasis)
summary(oneway2)
```

and find the three group means as the estimates.

Notice that the residual standard error in `oneway1` and `oneway2` are the same. This is because they fit the same model! We say that we have different versions, or parameterizations, of the model. The one fit is not more or less correct than the other, but they are useful for different purposes: `oneway1` gives you all the useful information about comparison to a reference group, whereas `oneway2` gives you all the useful information about the group means themselves.

5. *(Interpretation of output)* Let us make sure that we understand the output correctly:

- Is there a significant difference between the `healthy` group and the `psne` group?

- Is there a significant difference between the `healthy` group and the `psor` group?
- Is the p-value for the comparison between the `psne` and `psor` groups available in the current output?
- Which hypothesis is tested in the `typepsor` line in `oneway2`? Is this a relevant hypothesis?
- Why are the standard errors for `psne` (and `psor`) not the same in `oneway1` and `oneway2`?
- Why are the standard errors for `healthy` and `psne` not the same in `oneway2` even though they both concern group means?

6. *(Test for homogeneity between groups)* It is standard to carry out an F-test for the overall effect of the explanatory variable. To be precise, the hypothesis is that the expected values are the same in all groups. One way to carry out the test is to fit two models — the model with as well as the model without the variable in question — and compare them with the `anova` function:

```
noEffect <- lm(intensity ~ 1, data=psoriasis)
anova(noEffect, oneway1)
```

Notice the way the model with no `type` effect is fitted: with `1` on the right-hand side of the `~`, meaning that there are no explanatory variables in the model, such that all observations are assumed to have the same distribution (this is true under the hypothesis).

Is there a significant difference between the three type of skin samples?

7. *(Model validation)* Try the commands

```
par(mfrow=c(2,2))
plot(oneway1)
```

Does the model seem to be appropriate for the data? Check also the stripchart and the boxplot from question 1 again, and discuss how they can be used to assess the validity of (some of) the assumptions behind the analysis.

8. *(Change of reference group)* As default R sorts the groups in alphabetical order and chooses the first one as the reference group. In our case this happened to be the `healthy` group, but this was a coincidence. Luckily, it is easy to change reference group. Try the following commands and explain what you see:

```
type
newType <- relevel(type, ref="psor")
newType
newType
oneway3 <- lm(intensity ~ newType, data=psoriasis)
summary(oneway3)
```

9. *(Categorical variables coded with numeric values)* Finally, we are going to consider the variable `typeNum`. Try the following commands:

```
typeNum
table(type, typeNum)
reg <- lm(intensity ~ typeNum, data=psoriasis)
summary(reg)
```

Notice how `typeNum` gives the same group structure as `type`, but with numbers instead of letter names.

Which model is fitted here in `reg`? Why does the model not make sense in the current set-up? *Hint:* Look at the name of the model.

So what should you do if a categorical variable is coded with numeric values? Try the following commands:

```
newVariable <- factor(typeNum)
newVariable
newModel <- lm(intensity ~ newVariable -1, data=psoriasis)
summary(newModel)
```

Explain the difference between the variables `typeNum` and `newVariable`. Compare the estimates from `newModel` and `oneway2`.

6.2 Oneway ANOVA: Pillbugs*

Prerequisites: Exercises 3.1, 3.3 (reading and working with datasets), and 6.1 (ANOVA)

An experiment on the effect of different stimuli was carried out with 60 pillbugs. The bugs were split into three groups: 20 bugs were exposed to strong light, 20 bugs were exposed to moisture, and 20 bugs were used as controls. For each bug it was registered how many seconds it used to move six inches. The data are saved in the files `pillbugs.xlsx` and `pillbugs.csv` with variables `time` and `group`.

1. Make stripcharts and/or parallell boxplots where you use `time` as response. Do the same where you use `log(time)` as response. Explain why it is more reasonable to use `log(time)` than `time` as the response in a oneway ANOVA.
2. Fit a oneway ANOVA model with `log(time)` as response, and carry out model validation. Is the model appropriate?
3. Is the expected value for log-time the same for all three treatment groups?
4. What is the estimated expected log-time for the control group? For the group with light exposure? For the group with moisture exposure?
5. What is the estimated difference in expected log-time between the group with light exposure and the control group? What is the interpretation of the exponential of this value?
6. Does the light exposure have a significant effect on log-time? How about the moisture exposure?

7. Finally, fit also the oneway ANOVA with `time` (not `log(time)`) as response, and carry out model validation. What do you see?

6.3 Twoway ANOVA: Growth of soybean plants

Prerequisites: Exercises 3.1, 3.3 (reading and working with datasets), and 6.1 (ANOVA)

An experiment with 52 soybean plants was carried out in order to examine the effect of light and stress on plant growth. There were two different levels of light exposure (`low` and `moderate`), and two different levels of stres (`no` or `yes`, where `yes` means that the plant has been shaken daily for 20 minutes). The 52 plants were divided into four groups corresponsing to the combinations of the light and stress treatments. After a periode the leaf areas was measured for each plant. The data is saved in the files `soybean.xlsx` and `soybean.csv`.

1. It is natural to start with a twoway ANOVA with interaction between stress and light. This model can be fitted in several different ways, for example:

   ```
   twowayWithInt <- lm(leafarea ~ stress * light, data=soybean)
   ```

   Fit the above model, and carry out model validation.

2. Make a summary of the model, and make sure you understand the estimates. In particular: Find, for each of the four stimuli combinations, the expected value of leafarea.

   Hint: Notice how reference levels are selected for each of the two factors (`low` and `no`, respectively), such that the intercept is to be interpreted as the expected value for this combination of stimuli. This estimate should be "corrected" for the other stimuli groups.

3. Make an interaction plot as follows:

   ```
   interaction.plot(stress, light, leafarea)
   ```

   Make sure you understand what has been plotted. Does the graph indicate an interaction effect between light and stress stimuli or not?

   A test of the hypothesis that there is no interaction is carried out by fitting the model without interaction (with main effects only), and comparing the two models with `anova`:

   ```
   twowayWithoutInt <- lm(leafarea ~ stress + light, data=soybean)
   anova(twowayWithoutInt, twowayWithInt)
   ```

   What is your conclusion regarding interaction? Make a summary of the model without interaction, and make sure you understand the estimates.

4. Is there a significant difference between the two light exposure levels? Is there a significant effect of stress?

   Hint: For the light exposure, say, make a model with `stress` as the only explanatory variable, and compare to the model without interaction. Alternatively, look at the summary from the model without interaction.

6.4 An analysis with categorical as well as quantitative variables: FEV

Prerequisites: Exercises 3.1, 3.3 (reading and working with datasets), 5.1 (linear regression), and 6.1 (ANOVA)

In Chapter 5 we considered models with only quantitative explanatory variables, whereas in this chapter we have so far considered categorical explanatory variables only. In this exercise we would like to use both types.

The primary objective of the analysis is to examine if children exposed to smoking have lower respiratory function than children who are not exposed, but we will also account for other variables that may influence respiratory function. The dataset contains information on more than 600 children. The measured outcome of interest is forced expiratory volume (FEV), which is, essentially, the amount of air an individual can exhale in the first second of a forceful breath. The data is saved in the files `fev.xlsx` and `fev.csv` and include the following variables: `FEV` (liters), `Age` (years), `Ht` (height, measured in inches), `Gender`, and `Smoke` (exposure to smoking, 0 = no, 1 = yes).

1. Read the data into R, and make a scatterplot matrix of the data: If you have called the dataset `fevdata`, then use the command `plot(fevdata)`.

2. Which of the variables in the dataset are quantitative and which variables are categorical? Make factor-type versions of the categorical variables, *i.e.* define

   ```
   GenFac <- factor(Gender)
   SmokeFac <- factor(Smoke)
   ```

3. Fit a model as follows:

   ```
   fevdata <- transform(fevdata, HtSqr=Ht^2)
   fevModel0 <- lm(FEV ~ Age + Ht + HtSqr + GenFac + SmokeFac +
                   SmokeFac*GenFac + GenFac*Age, data=fevdata)
   ```

   Make sure that you understand all the terms in the model (including the interactions and the term `HtSqr`).

4. Is the model appropriate for the data, or is some transformation of the response needed?

5. Simplify the model (possibly after transformation) as much as possible, *i.e.*, test the significance of the terms in the model and remove non-significant terms. Remember that you use should only remove one term (variable or interaction) from the model at a time.

6. What is your conclusion regarding smoking: Does smoking status influence respiratory function? If yes, how much? Discuss also he effect of the other variables.

7 Principal component analysis

There are several functions in R that can be used for principal component analysis. Below we will use the `princomp` function as working horse. Other possibilities include the `rda` function from the `vegan` package, which has advanced options for scaling, among others.

7.1 PCA: Physical measurements of crabs

Prerequisites: Exercise 3.3 (working with datasets)

The data for this exercise come from 200 specimens of a certain type of crabs. The crabs come in two colours (blue and orange). In the experiment 100 of each type were collected, 50 males and 50 females, and for each of the 200 crabs, five quantities were measured: The carapace/shell length (`CL`), carapace/shell width (`CW`), size of frontal lobe (`FL`), rear width (`RW`), and body depth (`BD`). The experimenters are interested in characterization of the colour types (and sexes) in terms of the variables.

The data are available as the dataset `crab` in the `MASS` package.

1. We are going to work on the log-measurements. Try the following commands and explain what happens:

   ```
   library(MASS)                  # Load package
   head(crabs)                    # Just looking at the data
   logcrabs <- log(crabs[,4:8])   # Dataset with log-values
   head(logcrabs)                 # The log-dataset

   group <- crabs$sex : crabs$sp  # Group variable
   group
   plot(logcrabs, col=group)
   ```

   In particular, does the raw data make it possible to easily distinguish between the four groups? Notice that black/red/green/blue corresponds to Female-Blue/Female-Orange/Male-Blue/Male-Orange.

2. A PCA can be carried out with the `princomp` function. If we use the option `cor=T`, then the correlation matrix (rather than the covariance function) is used. This correspods to a scaling of the variables. Try the following commands and discuss the output:

   ```
   pca <- princomp(logcrabs, cor=T)
   pca
   summary(pca)
   plot(pca)
   loadings(pca)
   pca$scores
   ```

3. Try the following commands and discuss the graphs. I particular, is it possible to use the principal components (the scores) to distinguish between the four groups? Which

aspects of crab characteristics relate to the three first components (recall the association between colours in the graphs, and colour/sex of the crabs from question 1)?

```
scorData <- data.frame(pca$scores)
plot(scorData)
plot(scorData, col=group)
plot(scorData[,1:3], col=group)
plot(logcrabs[,1], scorData[,1], col=group)
```

7.2 PCA: Ecological zones along the Doubs River

Prerequisites: Exercise 7.1 (PCA)

As part of a large project on characterization of ecological zones, 11 environmental variables were measures at 30 sites along the Doubs River. The variables were distance from the source, *i.e.* from the start location (`das`), altitude (`alt`), slope (`pen`), mean minimum discharge (`deb`), pH of water (`pH`), concentration of calcium, phosphate, nitrate, ammonium, respectively (`dur`, `pho`, `nit`, `amm`), dissolved oxygene (`oxy`), biological oxygen demand (`dbo`). See Borcard *et al.* (2011) for more details.

The data is saved in the files `DoubsEnv.xlsx` and `DoubsEnv.csv`.

1. Read the data into an R dataset called `doubs`.
2. The first variable in the dataset, `das`, is not an environmental variable, so we will only use the remaining variables. Run a PCA on the those data, *i.e.* on `doubs[,-1]`.
3. How many components are needed to explain 75% and 90%, respectively, of the total variation? Make a plot of the first two principal components.
4. One of the aims of the study was to identify ecological zones, *i.e.* groups of sites that are similar in certain senses. Based on the abundance of different fish species, the 30 sites were allocated to four clusters. This allocation is given by the variable `clus4` below.

   ```
   clus4 <- c(rep(1,10), rep(2,9), rep(3,3), rep(4,3), rep(3,5))
   clus4
   ```

   For example,the first site in included in cluster 1, whereas the last site is included in cluster 3.

   Make the plot of the two first principal components again, this time with the points coloured according to the clusters. Do the first two principal components contain information about the clusters?

8 Matrices

8.1 Matrix operations

Prerequisites: None

1. *(Construction of matrices)* Try the following commands, and explain what you see:

   ```
   A <- matrix(c(1,4,2,7,5,3), nrow=2, ncol=3)
   A
   dim(A)
   ```

   Notice how the matrix as default is filled column-wise. Try instead

   ```
   B <- matrix(c(1,4,2,7,5,3), nrow=2, ncol=3, byrow=T)
   B
   dim(B)
   ```

   Make the matrices C and D in R, and check that you get the right thing:

   $$C = \begin{pmatrix} 1 & -1 \\ -1 & 3 \end{pmatrix} \quad \text{and} \quad D = \begin{pmatrix} 6 & 4 \\ 3 & 2 \end{pmatrix}$$

2. *(Indices)* Try the following commands and explain what you see:

   ```
   B[1,2]
   B[2,2] <- 6
   B
   B[1,]
   B[,3]
   B[,3] <- c(8,9)
   B
   ```

3. *(Matrix computations)* Try the following command and explain what you see:

   ```
   C+D
   C-D
   C*D
   ```

   In particular, notice how `C*D` is *not* the usual matrix product, but element-wise multiplication, which is only rarely relevant. Matrix multiplication goes like this (check that the result is correct):

   ```
   C %*% D
   ```

4. *(Transpose)* Try the command `t(A)`. What happens? For the above matrices, calculate the matrix products $A'A$ and $B'B$, where a prime means transpose.

5. *(Diagonal elements and matrices)* Try the following commands and explain what you see:

```
diag(C)
diag(D)
diag(c(2,6,7))
diag(4)
```

6. *(Determinant)* The determinant of a square matrix is computed with the `det` function. Find the determinants of C and D. Are the matrices regular or singular? What happens if you write `det(A)`?

7. *(Inverse matrix)* The inverse of a regular matrix is computed with `solve`. Try the commands and explain what you see:

```
solve(C)
solve(C) %*% C
C %*% solve(C)
solve(D)
```

8. *(Solve matrix equations)* Say that you want to solve the equation $Cx = y$ for a known square matrix C and a known vector y. Then write `solve(C,y)`.

   Try the command `solve(C, c(1,1))`, and check that the answer is correct. Then try the following commands and compare to the output of `solve(C)`:

```
solve(C, c(1,0))
solve(C, c(0,1))
```

9. *(Eigen decomposition)* Eigen values and eigen vectors for for a square matrix are computed with `eigen`. Try

```
eigen(C)
lambda <- eigen(C)$values
lambda
Q <- eigen(C)$vectors
Q
```

   Then `lambda` contains the eigen values and the columns of `Q` are the corresponding eigen vectors. Check that this is correct, *i.e.* check that $CQ_1 = \lambda_1 Q_1$ and $CQ_2 = \lambda_2 Q_2$ where λ_j is the jth eigen value and Q_j is the jth column in Q.

   Also compute `Q %*% Delta %*% t(Q)`. What do you get?

   Finally, try the following commands and explain why you get complex numbers:

```
F <- matrix(c(-1,-1,-1,1,2,0,0,1,2), nrow=3, ncol=3)
eigen(F)
```

8.2 Least squares estimates in linear regression*

Prerequisites: Exercises 8.1 (matrices) and 5.1 (linear regression, only for question 3)

Consider data consisting of pairs $(x_1, y_1), \ldots, (x_n, y_n)$, and assume that the pairs are independent and that the conditional variance of y_i given x_i is the same for all i. The linear regression on y of x assumes that the expected value of y_i is a linear function of x_i:

$$\mathrm{E}y_i = \beta_0 + \beta_1 x_i$$

If $y_1, \ldots, y_n$ are collected in a vector Y, the so-called the design matrix X is defined by

$$X = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix},$$

and the unknown parameters, β_0 and β_1, are collected in a vector β, then the assumption on the expected values can be written on matrix form as $\mathrm{E}Y = X\beta$.

The least squares estimate of β is given by

$$\hat{\beta} = (X'X)^{-1}X'Y, \tag{4}$$

and

$$\tilde{\sigma}^2 = \frac{1}{n-1}\left(Y'Y - (X\hat{\beta})'(X\hat{\beta})\right) \tag{5}$$

is an unbiased estimate of the conditional variance of y_i given x_i. Furthermore, the estimated variance matrix of $\hat{\beta}$ is

$$\mathrm{Var}(\hat{\beta}) = \hat{\beta}\tilde{\sigma}^2(X'X)^{-1} \tag{6}$$

Consider the cherry data from Exercise 3.1 and 5.1. In the following you should use volume as response variable, y, and girth as explanatory variable, x.

1. Make the designmatrix X the cherry data.
2. Use formulas (4)–(6) to calculate $\hat{\beta}$, $\tilde{\sigma}^2$ and $\mathrm{Var}(\hat{\beta})$.
3. Fit the linear regression with `lm`, and recognize that you get the same results. More specifically:
   - Compare the estimates of the intercept/slope obtained from your matrix computatations and from `lm`.
   - Compare the variance estimate from your matrix computations to the "Residual standard error" from `lm`. What is the relation?
   - The diagonal elements of $\mathrm{Var}(\hat{\beta})$ are the estimated variances of β_0 and β_1, respectively. How do they relate to the standard errors listed in the summary of the `lm` fit?

9 Functions

9.1 Mathematical functions of a single argument

Prerequisites: None

There are many pre-programmed mathematical functions, for example `log` and `sqrt`, but sometimes you need to examine functions that are not already implemented.

1. *(Definition)* Consider the function $f(x) = 2x^2 - 0.9x - 1$, and calculate $f(0)$ and $f(1.5)$. Then try the following commands:

   ```
   f <- function(x) return(2*x^2 - 0.9*x - 1)
   f(0)
   f(1.5)
   ```

   The first line defines the function: It has a name, `f`, takes an argument, `x`, and returns a value that depends on the argument.

2. *(Graphs)* Try the following commands and see what happens:

   ```
   plot(f)
   plot(f, from=-1, to=2)
   ```

   Next, define the function $g(x) = -4x^2 + 0.2x + 4$ in R, and try the commands:

   ```
   plot(g, from=-1, to=2, add=T, col="red")
   ```

3. *(Minima and maxima)* From the graphs we see that f has a mimimum around 0.25. We can find the actual minimum point with `optimize` as follows.

   ```
   optimize(f, interval=c(-1,2))
   ```

   Notice that we should specify an interval in which R should look for a minimum.

   From the graph we also see that g has a maximum around 0. Use `optimize` to find this maximum. *Hint:* You now want to find a maximum rather than minimum. Which function has its minimum the same place as g has its maximum?

4. *(Roots)* The function f is zero around -0.5 and 1. The function `uniroot` can give us the exact values. Try

   ```
   uniroot(f, interval=c(-1,2))   ## Gives error message
   uniroot(f, interval=c(-2,0))
   ```

   This gives you the root around -0.5. Find the root which is close to 1.

   Then find the roots of g. Finally, find the values of x for which $f(x) = g(x)$. *Hint: Which function is zero if and only if $f(x) = g(x)$?*

9.2 The mean as a least squares estimate*

Prerequisites: Exercise 9.1 (functions)

Consider an experiment where the biomass has been measured for 8 random plants grown under certain conditions. The sample values are denoted $y_1, \ldots, y_n$. Assume that we are interested in an estimate of the population average, denoted μ.

It is well known that the sample mean $\bar{y}$ is an estimate of the population average. The sample mean is also the least squares estimate: Consider the sum of squared deviations from μ, regarded as a function of μ:

$$f(\mu) = (y_1 - \mu)^2 + \cdots + (y_n - \mu)^2$$

This function has its minimum for $\mu = \bar{y}$.

Consider in the following the sample consisting of

24.7 32.5 22.6 23.9 19.6 21.6 19.9 20.9

1. Make a vector with the sample values, denoted `y`. Then make a function that takes μ (`mu`) as argument and calculates the $f(\mu)$. Call the function `f`.

2. Find the minimum of f with `optimize`, and compare with `mean(y)`.

3. Recall that the sample standard variance is defined as

$$s^2 = \frac{1}{n-1}\Big((y_1 - \bar{y})^2 + \ldots + (y_1 - \bar{y})^2\Big)$$

   Use the output from `optimize` to compute this value, and compare to the output from `var(y)`.

4. Make a plot of f with the following commands, and check it the results from `optimize` seems to be correct:

```
muVal<- 10:40
fVal <- sapply(muVal, f)
plot(muVal, fVal)
plot(muVal, fVal, type="l")
```

9.3 Mathematical functions of several arguments

Prerequisites: Exercise 9.1 (preferably, on functions)

Consider the function

$$f(x_1, x_2) = -5 - 3x_2 + 4x_2 + x_1^2 - x_1x_2 + x_2^2$$

1. *(Definition)* We can define the function in two ways. For the first one the arguments x_1 and x_2 are considered as two numbers whereas the second one considers $x = (x_1, x_2)$ as a vector. Try the following commands and make sure you understand the difference between `f1` and `f2`.

```
f1 <- function(x1,x2) return(-5-3*x1+4*x2+x1^2-x1*x2+x2^2)
f1(0,0)
f1(1,2)

f2 <- function(x) return(-5-3*x[1]+4*x[2]+x[1]^2-x[1]*x[2]+x[2]^2)
f2(c(0,0))
f2(c(1,2))
```

2. *(Contour plot)* For functions of two arguments, it is often convenient to plot the contours of the functions. A contour plot is a plot with the arguments (x_1 and x_2) at the axes and with lines/curves corresponding to different values of the function. All points of a given contour plot give rise to the same value of the function.

   Try the following commands and explain what is going on:

   ```
   x1 <- seq(0,2,length=51)
   x1
   x2 <- seq(-3,1, length=51)
   x2
   fVals <- outer(x1,x2,f1)     ## Computes f1 in all grid points
   dim(fVals)
   contour(x1,x2,fVals)
   ```

   It is not obvious how to illustrate functions that take three or more arguments.

3. *(Minimum)* The R function `optim` can be used to find the minimum (or maximum) of a function. The simplest possible `optim` command goes as follows:

   ```
   optim(par=c(0,0), fn=f2)
   ```

   Compare to the contour plot from the previous question.

   Notice that `optim` requires that the function is defined as `f2` above, *i.e.* that the argument is considered as a vector rather than as several values. Moreover, initial values should be provided where the optimization algorithm starts its search.

   In general multi-dimensional optimization is a difficult numerical problems, and one should be careful and examine the properties of the function as much as possible before applying `optim` (is there a minimum at all; is it unique; what is a reasonable place to start the algorithm?). It is often necessary to make use of the extra arguments which can be supplied to `optim`, see the help page. An alternative to `optim` is `nlminb`. It is generally slower than `optim`, but also more reliable for functions with many parameters (more than 6–8, say).

9.4 Non-linear least squares*

Prerequisites: Exercise 9.3 (functions)

In a plant experiment duckweed plants were treated with different amounts of the herbicide glyphosate, and the relative growth rate was measured. Eight different non-zero doses were

used with two replicates, and there were five control plants (no glyphosate treatment). In total, this gives 21 observations. The data are saved in the files `glyphosate.xlsx` and `glyphosate.csv`.

1. Make a R dataset with the data. Make two scatterplots: Relative growth rate (y-axis) against dose (x-axis), and relative growth rate against logarithmic dose.

The logistic function is often used to describe dose-response relations. If y is the relative growth rate and d is the dose, then

$$y \approx \frac{M}{1+(d/d_{50})^a}$$

where $M > 0$, $d_{50} > 0$ and $a > 0$ are parameters that should be estimated from the data. This is done with least squares: Write the sum of squared deviations between data and function values as a function of the uknown parameters:

$$f(M, c_{50}, a) = \left(y_1 - \frac{M}{1+(d_1/d_{50})^a}\right)^2 + \cdots + \left(y_n - \frac{M}{1+(d_n/d_{50})^a}\right)^2$$

The estimates $\hat{M}$, $\hat{c}_{50}$ and $\hat{a}$ are the values that make f as small as possible.

2. Convince yourself that the parameter M can be interpreted as the largest possible response (when the dose is extremely large), and that d_{50} can be interpreted as the dose where the expected response is half of the largest possible, *i.e.* $M/2$. Use graphs to find some loose guesses for M and d_{50}.

3. Make a function `f` that takes the parameter vector as arguments. Use `optim` to find the estimates. You can use the guesses from the previous question as initial values for M and d_{50}, and the value 1, say, for a.

4. Non-linear least squares has been implemented in the `nls` function. Try the following command and compare with the minimum you just found:

```
nls(rgr ~ M/(1+(dose/d50)^a), start=list(M=1.7, d50=exp(12), a=1))
```

5. Add a graph of the estimated relation onto the scatterplot with log-dose on the x-axis. *Hint:* Make a long vector with dose-values, a vector with the corresponding estimated values of the rekative growth rate, and use the `lines` function.

9.5 Non-mathematical functions in R*

Prerequisites: Exercises 3.1 (reading data), 3.4 (subsets of datasets), 5.1 (linear regression)

In the previous exercises we have implemented and examined mathematical functions, that is, functions that take one or more arguments and return a value. However, output of of an R function could be a graph (just think of the `plot` function), a model object (think of the `lm` function), or something else.

In the following we consider the cherry data from Exercise 3.1 and 5.1 again and use volume as response variable, y, and girth as explanatory variable, x in a linear regression.

1. Make an R function that takes an observation number, `obsNo`, as arguments and does the following (all in the same function):

   - Makes a sub-dataset that consists of all observations except the one given by the argument value `obsNo`
   - Fits the linear regression for the sub-dataset
   - Makes s scatterplot for the sub-dataset and adds the fitted regression line
   - Returns the estimates coefficients (use `coef()` on the `lm`-object)

   Your code could look something like this (where the ... should be filled by you):

```
cherryFct1 <- function(obsNo)
  {
   myData <- ...
   myReg <- lm(...)
   ...                  ## Make scatterplot
   ...                  ## Add the regression line
   return(...)
  }
```

2. Make a function that returns a matrix of dimension 31 times 2. Row *i* should contain the estimated coefficients from the regression where observation *i* is omitted. The function should take no arguments.

   Your code could look something like this:

```
cherryFct2 <- function()
{
  estMat <- matrix(...)              # Initialize the matrix
  for (i in 1:31) estMat[i,] <- ...  # Use cherryFct1
  return(estMat)
}
cherryFct2()
```

References

Borcard, D., Gillet, F., and Legendre, P. (2011). *Numerical Ecology with R*. Springer, New York.

Dalgaard, P. (2008). *Introductory Statistics with R (2nd ed.)*. Springer, New York.

Ekstrøm, C. (2012). *The R Primer*. Taylor & Francis group/CRC Press.

Ekstrøm, C. and Sørensen, H. (2011). *Introduction to Statistical Data Analysis for the Life Sciences*. Taylor & Francis group/CRC Press.

Martinussen, T., Skovgaard, I. M., and Sørensen, H. (2012). *A First Guide to Statistical Computations in R*. Biofolia.

Venables, W. and Ripley, B. (2002). *Modern Applied Statistics with S*. Springer, fourth edition.