

Applied Methods in Statistics

Thore Egeland and Raju Rimal

Year: 2018

Chapter 1

Practical Information

Exercises in this site is relevant for Stat340 course which discusses some of the applied topics in statistics. For the exercises, we will use open source R statistical software along with RStudio, an integrated development environment for R. We advise student to install the latest version of R and RStudio on their laptop computer. In addition, we will use few packages in R which we will discuss during the exercise period. Students are highly encouraged to complete these exercises on their own and also participate in Group Exercises. Follow the link below to install R and RStudio.

Install R and Install RStudio

See: Lecture and Exercise Plan and Reference Books

Lecture and Exercise Plan

Week	Topics	Exercises
Week 6 (Feb. 05)	Overview, R and R Studio	Getting Started
Week 7 (Feb. 12)	Regression Analysis	Exercise 1
Week 8 (Feb. 19)	Analysis of Variance	Exercise 2
Week 9 (Feb. 26)	Principal Component Analysis	Exercise 3
Week 10 (Mar. 05)	Multivariate Statistics (PCR, PLS)	Exercise 4
Week 11 (Mar. 12)	Cluster Analysis	Exercise 5
Week 12 (Mar. 19)	Classification	Exercise 6
Week 14 - 15 (Apr. 2, Apr. 9)	Generalized Linear Models	Exercise 7-8
Week 16 (Apr. 16)	Random Effects Models	Exercise 9
Week 17 (Apr. 23)	Mixed Effects Models	Exercise 10

See: Practical Information | Reference Books

Reference Books

See: Practical Information | Lecture and Exercise Plan

Chapter 2

Getting Started

In this section, we will dive into R and RStudio and get used to with it to some extent. We will continue learning about R and RStudio as we go on. R is an open source programming language basically used in data analysis. In R there are many packages that are created for specific purposes and they have made R rich and powerful. In this course, apart from default R package (that is installed and already loaded), we will use few other packages which we will install and load as we go through our exercises. We can use following command to install a package. Below, a car package is used as an example:

```
install.packages("car")
```

To load the package we use library function as,

```
library(car)
```

Following screenshot help you to install package using RStudio IDE,

Exercise 1: Create New Project

Creating a project allows us to organize the files and related materials during our study. File => New Project opens a window to create new project. It will be easier to access all the resources, if all the scripts and datasets are within a main folder, i.e. the project folder.

The examples in this exercise uses the project folder as the main folder. Although it is not necessary, but throughout the exercises we will use `_data` folder as a folder containing all

the data that we are using in the course.

Exercise 2: Importing data in R

The usual data sources can be a text file in txt or csv format or spreadsheet(excel) file in xls or.xlsx. Data and R-objects can also be imported from rds, rda or rdata. Below, we will discuss these in detail. In addition you can also find some animated images showing how we can import data in RStudio for each of these file formats.

Import txt or csv

Base R-package has `read.table` and `read.csv` for importing a text or comma separated file (csv) files. Download `bodydata.txt` and to import it in R as,

```
bodydata <- read.table("_data/bodydata.txt", header = TRUE)
```

Here the argument `header` is `TRUE` if the data has header in its first row. The argument `sep` takes `\t`, `,` or `;` based on if the columns in the text data are tab-separated, comma-separated or separated by semi-colons. If the decimal values in the data are represented by `,`, the `dec` argument takes the value `,`. For further help see: `?read.table`

Import Microsoft Excel spreadsheet

An R-package `readxl` helps to import excel file. If it is not installed, you should install it as,

```
install.packages("readxl")
```

Download `bodydata.xlsx` from canvas and load it to R as,

```
library(readxl)
bodydata <- read_excel("_data/bodydata.xlsx", sheet = 1)
```

For further help and arguments on this function load the library as `library(readxl)` and see: `?read_excel` or `?read_xlsx`.

Load rdata, rda or rds

One can save data, models and other R-objects in `rdata` or `rds` format. In order to load `rdata`, we can use `load` function. Download `bodydata.rdata` from canvas and load it to R.

```
load("_data/bodydata.rdata")
```

Reading data from clipboard (“pasting” copied data into an R object)

You can import data in clipboard in R. For example the data you copied in Excel or Word files.

```
bodydata <- read.table(file = "clipboard", header = TRUE)
```

In Mac, you need to do,

```
bodydata <- read.table(file = pipe("pbpaste"), header = TRUE)
```

This allows us to get the data from anywhere just after they are copied.

After every import in above examples, we have saved our imported data in `bodydata` variable. This an R-object that holds any kind of data-structures such as `matrix`, `data.frame`, `list` or fitted models. We can find these R-objects in “Environment” tab in RStudio.

Exercise 3: Exporting data to a file

To export an r-object to a file, `write.table` function is used. For example, we can export the `bodydata` table we just imported as a text file named `bodydata-export.txt` as,

```
write.table(bodydata, file = "_data/bodydata-export.txt",  
            row.names = FALSE, quote = FALSE)
```

We can also use `write.csv` function to export the data in `csv` format. The `txt` and `csv` file format only holds data in tabular structures. Sometimes we need to save other R-objects

such as fitted models or list for which we can use Rdata format. We can save our bodydata objects in Rdata format as,

```
save(bodydata, file = "_data/bodydata-export.rdata")
```

This will export bodydata object to a file named bodydata-export.rdata in _data folder in your project directory.

Exercise 4: Data Structure in R

The dataset we imported in Exercise 2: Importing data in R is a data frame. DataFrame is a structure that R uses to keep the data in that particular format. If you do `class(bodydata)` for the data we have imported before, we can see `data.frame` as its class. There are other data structures in R. Some basic structure that R uses are discussed below:

Vector

A vector is a one-dimensional object where you can store elements of different modes such as “logical” (TRUE or FALSE), “integer”, “numeric”, “character” etc. All elements of a vector must be of same mode. For example,

```
x <- c(TRUE, FALSE, FALSE, TRUE, TRUE)
y <- c("TRUE", "FALSE", "Not Sure")
z <- c(2, 3, 5, 6, 10)
```

Here, x, y and z are of class logical, character and numeric respectively. Although in vector y we have TRUE and FALSE they are in character format. The function `c` is used to define a vector. However functions that are used to create sequences also gives us a vector. For example,

```
(a_sequence <- seq(from = 0, to = 10, by = 2))
```

```
[1] 0 2 4 6 8 10
```

```
(b_sequence <- 1:10)
```

```
[1] 1 2 3 4 5 6 7 8 9 10
```


Here both `a_sequence` and `b_sequence` are vector. Give special attention to the way we have created the sequence of numbers. It will be useful in many situations in future exercises.

Matrix

A matrix is a two dimensional structure with row and column. As this is an extension of vector structure, matrix must have elements of same mode as in a vector. For example:

```
(a_matrix <- matrix(1:25, nrow = 5, ncol = 5))
```

	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	1	6	11	16	21
[2,]	2	7	12	17	22
[3,]	3	8	13	18	23
[4,]	4	9	14	19	24
[5,]	5	10	15	20	25

```
(b_matrix <- diag(1:5))
```

	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	1	0	0	0	0
[2,]	0	2	0	0	0
[3,]	0	0	3	0	0
[4,]	0	0	0	4	0
[5,]	0	0	0	0	5

Here, `a_matrix` is created from a vector of sequence of 1 to 25 in 5 rows and 5 columns. We can also define a diagonal matrix as `b_matrix` with numbers from 1 to 5 in its diagonal.

Array

An array is an extension of Matrix structure in three or more dimension. We can define an array as,

```
(an_array <- array(1:24, dim = c(2, 4, 3)))
```

, , 1

```

      [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8

```

```
, , 2
```

```

      [,1] [,2] [,3] [,4]
[1,]     9    11    13    15
[2,]    10    12    14    16

```

```
, , 3
```

```

      [,1] [,2] [,3] [,4]
[1,]    17    19    21    23
[2,]    18    20    22    24

```

List

All the above structure we discussed require that the the elements in them to be of same mode such as numeric, character and logical. Sometimes it is necessary to keep objects of different modes in same place. List is a structure that helps in such situation. A list can contain list, matrix, vector, numeric or any other data structure as its elements. For example:

```

a_list <- list(
  a_matrix = matrix(1:6, nrow = 2, ncol = 3),
  a_vector = 2:7,
  a_list = list(a = 1, b = 3:6),
  a_logical = c(TRUE, FALSE, TRUE, NA)
)
a_list

```

```

$a_matrix
      [,1] [,2] [,3]
[1,]    1    3    5

```

```
[2,]    2    4    6
```

```
$a_vector
```

```
[1] 2 3 4 5 6 7
```

```
$a_list
```

```
$a_list$a
```

```
[1] 1
```

```
$a_list$b
```

```
[1] 3 4 5 6
```

```
$a_logical
```

```
[1] TRUE FALSE TRUE NA
```

In above example, `a_list` contains a matrix, a numeric vector, a list and a logical vector.

Data Frame

Data Frame is a list kept in tabular structure. Every column of a data frame has a name assigned to it. The `bodydata` dataset we have imported is an example of data frame. Data frame is the most used data structure to keep data in tabular format. Lets create a data frame:

```
a_dataframe <- data.frame(
  character = c("a", "b", "c"),
  numeric = 1:3,
  logical = c(TRUE, FALSE, NA)
)
a_dataframe
```

	character	numeric	logical
1	a	1	TRUE
2	b	2	FALSE
3	c	3	NA

Every column of a `data.frame` is a vector. Different columns of a data frame can contain element of different modes. For example: the first column can be a character vector while the second column can be a numeric vector as in the example above.

Exercise 5: Exploring the data

Structure of an R-object

The first command you need to learn is `str` function in order to explore any object in R. Lets apply this to our `bodydata`,

```
str(bodydata)
```

```
'data.frame':  407 obs. of  4 variables:
 $ Weight      : num  65.6 80.7 72.6 78.8 74.8 86.4 78.4 62 81.6 76.6 ...
 $ Height      : num  174 194 186 187 182 ...
 $ Age         : num  21 28 23 22 21 26 27 23 21 23 ...
 $ Circumference: num  71.5 83.2 77.8 80 82.5 82 76.8 68.5 77.5 81.9 ...
```

This output shows us that `bodydata` is a `data.frame` with 407 rows and 4 numeric variables - Weight, Height, Age, Circumference.

Accessing elements from R-objects

Different data structure have different way of accessing elements from them.

Extracting elements from vector, matrix and array

For vector, matrix and array we can use `[` for accessing their elements. Lets create a vector, a matrix and an array as follows,

```
a_vector <- c("one", "two", "three", "four", "five")
a_matrix <- matrix(1:24, nrow = 3, ncol = 8)
an_array <- array(1:24, dim = c(2, 3, 4))
```

Extracting element at position 3 to 5 in a_vector `a_vector[3:5]` will give three, four, five, the elements at position index 3, 4, and 5. In R, position index starts from 1.

Extracting element in rows 2, 3 and columns 2, 4, 6, 8 from a_matrix This is a two dimensional structure, we give row-index and column-index inside `[]` operator separated by comma as,

```
a_matrix[c(2, 3), c(2, 4, 6, 8)]
```

```
      [,1] [,2] [,3] [,4]
[1,]     5    11    17    23
[2,]     6    12    18    24
```

We can also write this as,

```
a_matrix[2:3, seq(from = 2, to = 8, by = 2)]
```

```
      [,1] [,2] [,3] [,4]
[1,]     5    11    17    23
[2,]     6    12    18    24
```

Here `seq(from = 2, to = 8, by = 2)` create sequence even integer from 2 to 8 which is used as column index for extracting elements from `a_matrix`.

Extracting first element of an_array: Here `an_array` is an array structure of dimension three. So, we have to use three index vector inside `[]` operator in order to extract element from it. For instance `an_array[1, 1, 1]` gives 1 as its first element of the array.

In all these structures we can only supply index of one or more dimension. For example, `a_matrix[1:2,]` where we have only mentioned the row index, will give elements in *first* and *second* row from all columns. i.e.

```
a_matrix[1:2, ]
```

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]     1     4     7    10    13    16    19    22
[2,]     2     5     8    11    14    17    20    23
```

Extracting elements from data.frame and list

Lets create a `data.frame` and a `list` as,

```

a_dataframe <- data.frame(
  fertilizer = c("Low", "Low", "High", "High"),
  yield = c(12.5, 13.1, 15.3, 16.2)
)
a_list <- list(
  facebook = data.frame(
    name = c("Gareth", "Raju", "Marek", "Franchisco"),
    has_profile = c(TRUE, TRUE, FALSE, TRUE)
  ),
  twitter = c("@gareth", "@raju", "@marek", "franchisco")
)

```

Extracting third and fourth row of fertilizer from a_dataframe Same as extracting elements as matrix as discussed above we can use row and column index as `a_dataframe[3:4, 1]`. We have used 1 in place of column index since fertilizer is in first column. We can also use name instead as `a_dataframe[3:4, "fertilizer"]`.

```
a_dataframe[3:4, "fertilizer"]
```

```

[1] High High
Levels: High Low

```

Extracting first element of a_list We can use `[[` for extracting elements from a_list. For example `a_list[[1]]` will give the first element of the list. Here in our list we have two elements with names facebook and twitter. So, we can also use their names as `a_list[["facebook"]]` which is not possible if they do not have any name.

```
a_list[["facebook"]]
```

	name	has_profile
1	Gareth	TRUE
2	Raju	TRUE
3	Marek	FALSE
4	Franchisco	TRUE

We can also use `$` operator to extract elements from named list and a data frame. For example, `bodydata$Weight` extracts Weight variable from bodydata dataset.

View Data in RStudio

Newer version of RStudio support viewing data in different structures. To view bodydata we have imported in Exercise 2: Importing data in R, we can use `View(bodydata)`. If you have not imported the, you need to follow the exercise and import the data first. We can also click the data in “Environment” tab to view it.

Summary of data

We can compute basic descriptive summary statistics using `summary` function as,

```
summary(bodydata)
```

Weight	Height	Age	Circumference
Min. : 42.0	Min. :150	Min. :18.0	Min. : 57.9
1st Qu.: 58.5	1st Qu.:164	1st Qu.:23.0	1st Qu.: 68.0
Median : 68.6	Median :171	Median :27.0	Median : 75.6
Mean : 69.2	Mean :171	Mean :29.9	Mean : 76.9
3rd Qu.: 78.8	3rd Qu.:178	3rd Qu.:35.0	3rd Qu.: 84.3
Max. :108.6	Max. :198	Max. :67.0	Max. :113.2

Dimension of data

The number of elements in a data structure like vector and list we can use `length` function. For example: if we extract `Weight` variable from `bodydata` we will get a numeric vector. The length of this vector is,

```
length(bodydata$Weight)
```

A multi-dimensional data structure like matrix, array and data frame has dimension. We can use `dim` function to find the dimension.

```
dim(bodydata)
```

```
[1] 407 4
```

Here, the first and second item refers to the number of rows and number of columns of `bodydata`. Similarly, we can use `nrow(bodydata)` and `ncol(bodydata)` to obtain these

number individually.

Lets Practice

- 1) Take a look at the top 5 rows of bodydata
- 2) Take a look at the top 5 rows of Height and Circumference variables of bodydata
- 3) Apply summary function on Age variable of bodydata

Exercise 6: Subsets of data and logical operators

Logical vector and index vector

A lot of times we want to get a subset of data filtering rows or columns of a dataframe. For which we can perform logical test and get TRUE or FALSE as result. This vector of logical can then be used to subset the observations from a dataframe.

For example, Lets extract observation from bodydata with Weight greater than 80. You might be wondering why following code does not work,

```
isHeavy <- Weight > 80
```

Error in eval(expr, envir, enclos): object 'Weight' not found

But remember that, the variable Weight is a part of bodydata. We have to extract Weight from the bodydata first. In R, with and within function helps you in this respect. In the following code, with function goes inside bodydata and execute the expression Weight > 80.

```
isHeavy <- with(bodydata, Weight > 80)
```

Here the logical vector isHeavy is computed by performing a logical operation on Weight variable within bodydata. The same operation can be done as,

```
isHeavy <- bodydata$Weight > 80
```

Take a look at this variable, what is it? :


```
head(isHeavy)
```

```
[1] FALSE TRUE FALSE FALSE FALSE TRUE
```

Yes, it is a vector of TRUE and FALSE with same length as Weight. Here the condition has compared each element of Weight results TRUE if it is greater than 80 and FALSE if it is less than 80.

Identify the elements We can identify which observations that are heavy by the which() function

```
HeavyId <- which(isHeavy)
```

This will return a vector of row index for the observations that are heavy, i.e. greater than 80. So how many are heavy? To find the size of a vector we can use length function.

```
length(HeavyId)
```

```
[1] 94
```

Here, 94 observations have Weight larger than 80.

Exercise

- 1) Identify who are taller than 180 and save this logical vector as an object called isTall.

```
isTall <- with(bodydata, Height > 180)
```

- 2) How many observations have height taller than 180?

```
TallId <- which(isTall)  
length(TallId)
```

```
[1] 76
```

- 3) How many observations are both tall and heavy? Here, you can use length function as above to find how many person are taller than 180.

```
isBoth <- isHeavy * isTall
```

How is this computation done? Here `isHeavy` and `isTall` contains `TRUE` and `FALSE`. The multiplication of logical operator results a logical vector with `TRUE` only if both the vectors are `TRUE` else `FALSE`.

Alternatively :

```
isBoth <- which(isHeavy & isTall)
```

The `&` operator result `TRUE` if both `isHeavy` and `isTall` are `TRUE` else, `FALSE` which is same as previous.

Subsetting data frame

Example 1

Lets create a subset of the data called `bodydataTallAndHeavy` containing only the observations for tall and heavy persons as defined by `isBoth`.

```
bodydataTallAndHeavy <- bodydata[isBoth, ]
```

For other logical tests see help file `?Comparison`

Example 2

Lets create a random subset of 50 observations. For this we first sample 50 row index randomly from all rows in `bodydata`. The `sample` function is used for the purpose. In the following code, `nrow(bodydata)` return the number of rows in `bodydata`. The `sample` function takes two argument `x` which can be a vector or a integer and `size` which is the size of the sample to be drawn.

```
idx <- sample(x = nrow(bodydata), size = 50)
```

Here, 50 rows are sampled from the total number of rows and the index of the selected rows are saved on vector `idx`.

Using this vector we can select the observations in `bodydata` to create a new data set called `bodydataRandom` as,

```
bodydataRandom <- bodydata[idx, ]
```

Here is the first five rows of bodydataRandom dataset.

```
head(bodydataRandom, n = 5)
```

	Weight	Height	Age	Circumference
50	59.1	167	29	73.1
253	59.0	170	23	66.5
15	71.0	174	23	78.0
339	58.2	169	28	67.2
78	56.8	170	23	72.0

Exercise

Create a subset of dataset bodydata including the observation with Age larger an 55 and Circumference larger than 80. Save this dataset named subdata.

```
idx <- with(bodydata, Age > 55 & Circumference > 80)
subdata <- bodydata[idx, ]
subdata
```

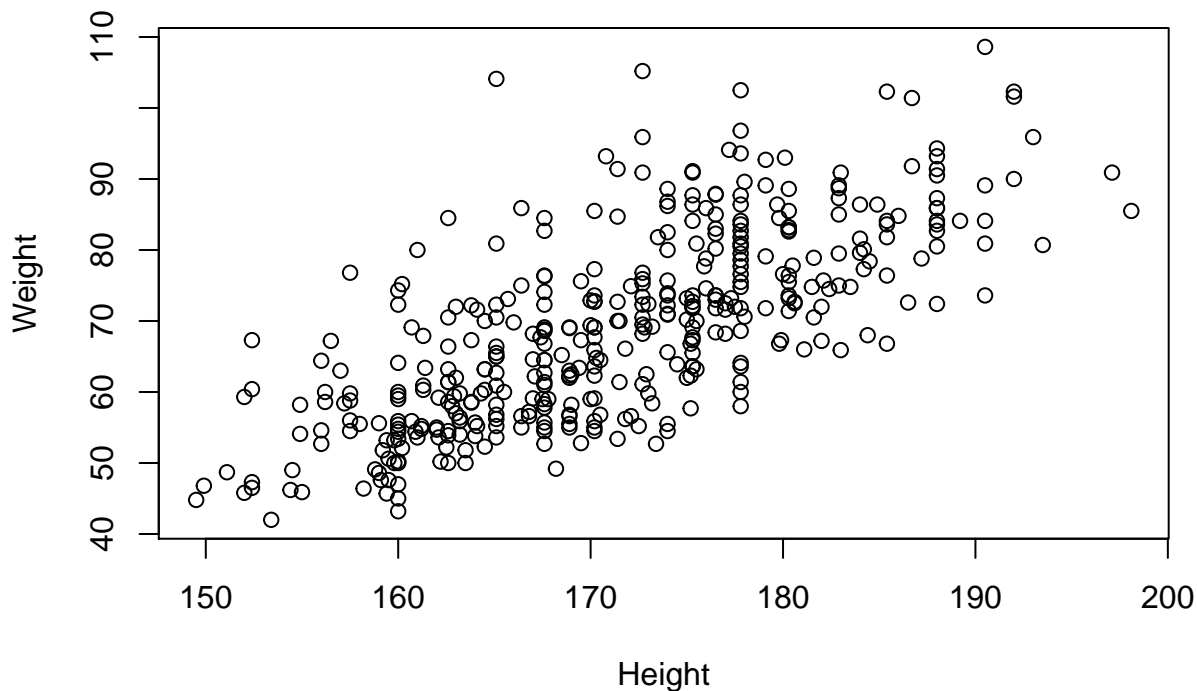
	Weight	Height	Age	Circumference
136	76.4	185	62	94.8
189	73.6	175	60	90.5
207	66.8	168	62	81.5
231	80.0	174	65	98.6

For those who are interested in playing more with data, have a look at <http://r4ds.had.co.nz/transform.html>

Exercise 7: Graphics

Plot the heights versus the weights for all observations in `bodydata`.

```
with(bodydata, plot(x = Height, y = Weight))
```



Spice up the plot

Check out the presentation for lesson 1 to see how to spice up the plot

Explore the `?par` help file

Use the `isBoth` variable to create a color vector

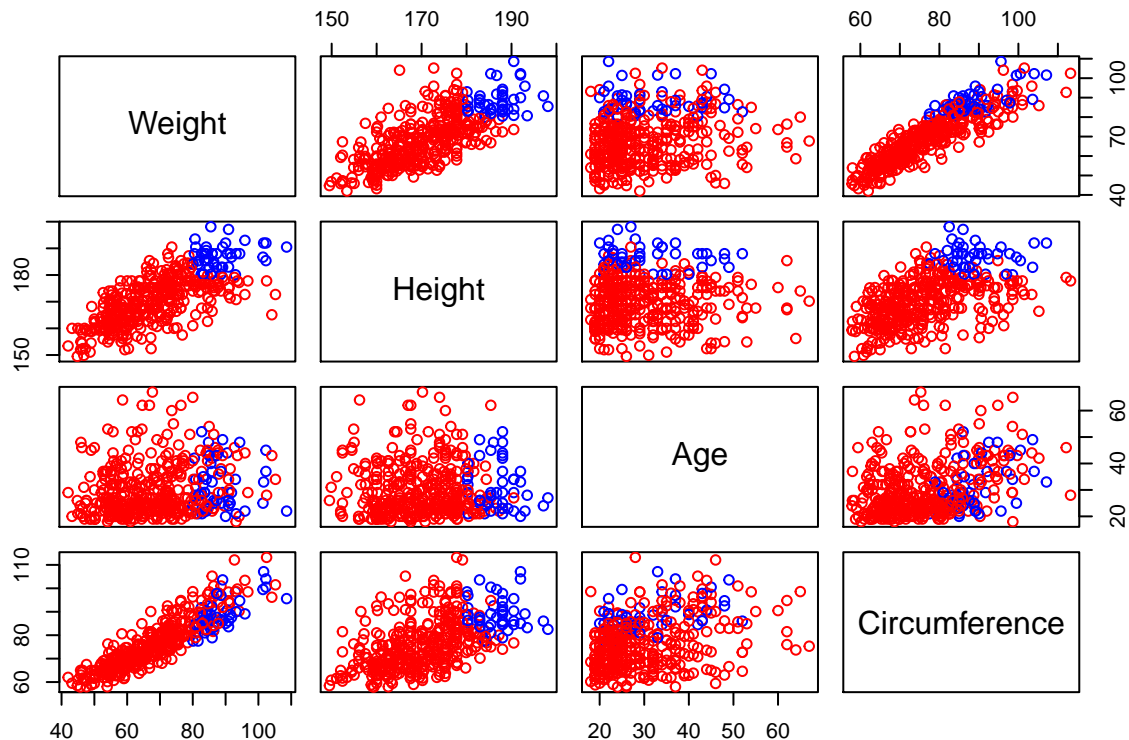
```
mycolors <- ifelse((isHeavy & isTall), "blue", "red")
```

Here, `isHeavy & isTall` returns a logical vector. The `ifelse` function returns blue if TRUE and red if FALSE for each element of the logical vector. The colors are then used in the plot so that all the Heavy and Tall person will be colored "blue" and rest as "red".

Use “mycolors” in the `col` argument of the `plot` function to mark the tall and heavy individuals

Plot all variables against each other

```
pairs(bodydata, col = mycolors)
```



Which variables seem to be most correlated to each other?

Here, Weight and Circumference seems to have highest correlation.

Which variables are least correlated to each other?

Age and Height variables seems to have least correlation.

Check by,

```
cor(bodydata)
```

	Weight	Height	Age	Circumference
Weight	1.000	0.7208	0.1870	0.899
Height	0.721	1.0000	0.0482	0.545
Age	0.187	0.0482	1.0000	0.355
Circumference	0.899	0.5448	0.3547	1.000

This returns the correlation matrix for the variables, and the guess made earlier true. Further, check out the help file for `pairs`, and the examples at the end. Try to make a pairs plot with scatter plots with smoothed lines in the lower left triangle, histograms on the diagonal, and correlation numbers in the upper right triangle.

Lets first create a function which create histogram. The function will later be used in the `pairs` function to create its diagonal plots.

```
panel.hist <- function(x, ...) {
  usr <- par("usr"); on.exit(par(usr))
  par(usr = c(usr[1:2], 0, 1.5) )
  h <- hist(x, plot = FALSE)
  breaks <- h$breaks; nB <- length(breaks)
  y <- h$counts; y <- y/max(y)
  rect(breaks[-nB], 0, breaks[-1], y, col = "cyan", ...)
}
```

Now, create a function that will display correlation on pairs plot.

```
panel.cor <- function(x, y, digits = 2, prefix = "", cex.cor, ...) {
  usr <- par("usr"); on.exit(par(usr))
  par(usr = c(0, 1, 0, 1))
  r <- abs(cor(x, y))
  txt <- format(c(r, 0.123456789), digits = digits)[1]
  txt <- paste0(prefix, txt)
  if (missing(cex.cor)) cex.cor <- 0.8 / strwidth(txt)
  text(0.5, 0.5, txt, cex = cex.cor * r)
}
```

Now, the above functions are implemented on the pairs plot,

```
pairs(bodydata,
      lower.panel = panel.smooth,
```

```
upper.panel = panel.cor,
diag.panel = panel.hist)
```



Here the `panel.smooth` deals with the smooth line on the lower panel of pairs plot.

Note:: Chapter 5 of the R book contains numerous examples of graphics. **Note::** For those interested in playing around with plots in R checkout: <http://r4ds.had.co.nz/data-visualisation.html>

Chapter 3

Regression Analysis

In this exercise we will use `birth.rdata` and `bodydata.rdata` datasets. We can load those data as below:

```
load("_data/birth.rdata")
load("_data/bodydata.rdata")
load("_data/mtcars.rdata")
```

Least Squares App

Play around with the least squares app on <http://solve.shinyapps.io/LeastSquaresApp>

1. Use $N=10$
2. Try to adjust manually the intercept and the slope to minimize the sum of squared errors, K .
3. Display the least square estimate to see how close you were.
4. Display the true model. Were you close?

Once Again,

1. Increase to $N=100$
2. Repeat the procedure.
3. Are you closer to the true model now?

Dataset: birth

The dataset `birth` records 189 birth weights from Massachusetts, USA, and some additional variables. The variables are,

Var	Description
LOW	Equals YES if birth weight is below 2500g and NO otherwise.
AGE	Age of the mother in years.
LWT	Weight in pounds of mother.
SMK	YES if mother smokes and NO otherwise.
BWT	Birth weight in g (RESPONSE).

The data appears in frontier as `birth.rdata`. Download the data into your STAT340 course folder and load the data set in RStudio.

Overview of data

- Take a look at the top 10 rows of the data using the `head()` function

```
head(birth, n = 10)
```

```

      LOW AGE LWT SMK  BWT
1  YES  28 120 YES  709
2  YES  29 130  NO 1021
3  YES  34 187 YES 1135
4  YES  25 105  NO 1330
5  YES  25  85  NO 1474
6  YES  27 150  NO 1588
7  YES  23  97  NO 1588
8  YES  24 128  NO 1701
9  YES  24 132  NO 1729
10 YES  21 165 YES 1790

```

- Use the `summary()` function to get a short summary of the variables.

```
summary(birth)
```

LOW	AGE	LWT	SMK	BWT
NO :130	Min. :14.0	Min. : 80	NO :115	Min. : 709
YES: 59	1st Qu.:19.0	1st Qu.:110	YES: 74	1st Qu.:2414
	Median :23.0	Median :121		Median :2977
	Mean :23.2	Mean :130		Mean :2945
	3rd Qu.:26.0	3rd Qu.:140		3rd Qu.:3475
	Max. :45.0	Max. :250		Max. :4990

- What is the proportion of smoking mothers in the data set?

The proportion of smoking mother is 0.39

- What is the average age of a mother giving birth?

The average age of mother giving birth is 23.2 years.

- What is the average birth weight of children from non-smoking and smoking mothers?

```
tapply(birth$BWT, INDEX = birth$SMK, FUN = mean)
```

```
NO YES
3055 2773
```

The function returns the mean birth weight for children of non-smoking and smoking mothers.

- What is the standard deviation of birth weight of children from non-smoking and smoking mothers?

```
tapply(birth$BWT, INDEX = birth$SMK, FUN = sd)
```

```
NO YES
752 660
```

The `sd()` function computes the sample standard deviation of a vector of observations.

Linear Regression

Run a simple linear regression model with BWT as response and LWT as predictor, like this,

```
birth1 <- lm(BWT ~ LWT, data = birth)
summary(birth1)
```

Call:

```
lm(formula = BWT ~ LWT, data = birth)
```

Residuals:

Min	1Q	Median	3Q	Max
-2192.2	-503.6	-3.9	508.3	2075.5

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	2369.67	228.43	10.37	<2e-16 ***
LWT	4.43	1.71	2.59	0.01 *

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

s: 718 on 187 degrees of freedom

Multiple R-squared: 0.0345,

Adjusted R-squared: 0.0294

F-statistic: 6.69 on 1 and 187 DF, p-value: 0.0105

Here, the regression model is,

$$\text{BWT} = \beta_0 + \beta_1 \text{LWT} + \epsilon$$

where $\epsilon \sim N(0, \sigma^2)$

Test the significance of LWT on BWT with a 5% test level and at a 1% level

What is the hypothesis you are testing?

The hypothesis for testing the significance of LWT on BWT is,

$$H_0 : \beta_1 = 0 \text{ vs } H_1 : \beta_1 \neq 0$$

- What is the conclusion?

The p -value corresponding to β_1 is less than 0.05 but greater than 0.01. So, at 5% test level, LWT is significant while at 1% test level, it is not significant.

- Find the R-squared. Do you think the model fits the data well?

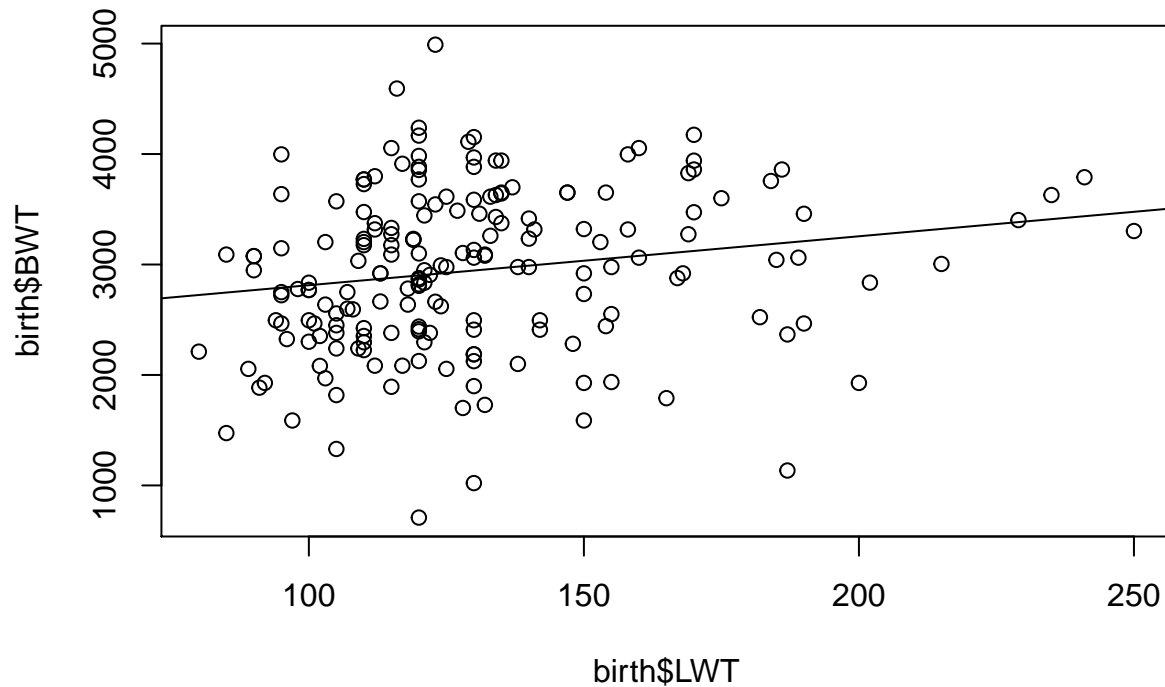
The r-squared (R^2) for the model is 0.03, this shows that only 3% of variation present in birth weight (BWT) is explained by weight of mother (LWT). Here, the model fits the data poorly.

Scatter Plot

- Make a scatter plot of LWT vs BWT

The scatter plot of LWT and BWT is,

```
plot(x = birth$LWT, y = birth$BWT)
abline(birth1)
```



- Make a comment to the plot in light of the output of your analyses.
1. The intercept for the regression line is 2369.67 and the slope is 4.43.
 2. The data-points are scattered around the regression line where BWT vary most
 3. Since the data-points are scattered much, the model could only explain small variation present in BWT with LWT.

Confidence Intervals

- Find 95% confidence intervals for the regression coefficients of the birth1 model

```
confint(birth1)
```

```

          2.5 %   97.5 %
(Intercept) 1919.04 2820.30
LWT          1.05    7.81

```

- Also find 99% confidence intervals

```
confint(birth1, level = 0.99)
```

```

          0.5 %   99.5 %
(Intercept) 1775.2097 2964.13

```

LWT -0.0287 8.89

- Comment on the intervals

1. It is 95% certain that the interval (1.05, 7.809) covers the true β_1 . Similarly, it is 99% certain that the interval (-0.029, 8.887) covers the true β_1 .
2. The 99% confidence is larger than 95% confidence. In other words, being more certain about the true value needs larger confidence interval.
3. Moreover, the 95% does not include zero while 99% interval includes zero. This is equivalent with the result that β_1 coefficient is significant at a 5% test level, but not significant at a 1% test level.

Regression with categories

Here we will fit a separate regression for smoking and non-smoking groups. You can identify the observation numbers of the smokers by:

```
smokeYes <- which(birth$SMK == "YES")
```

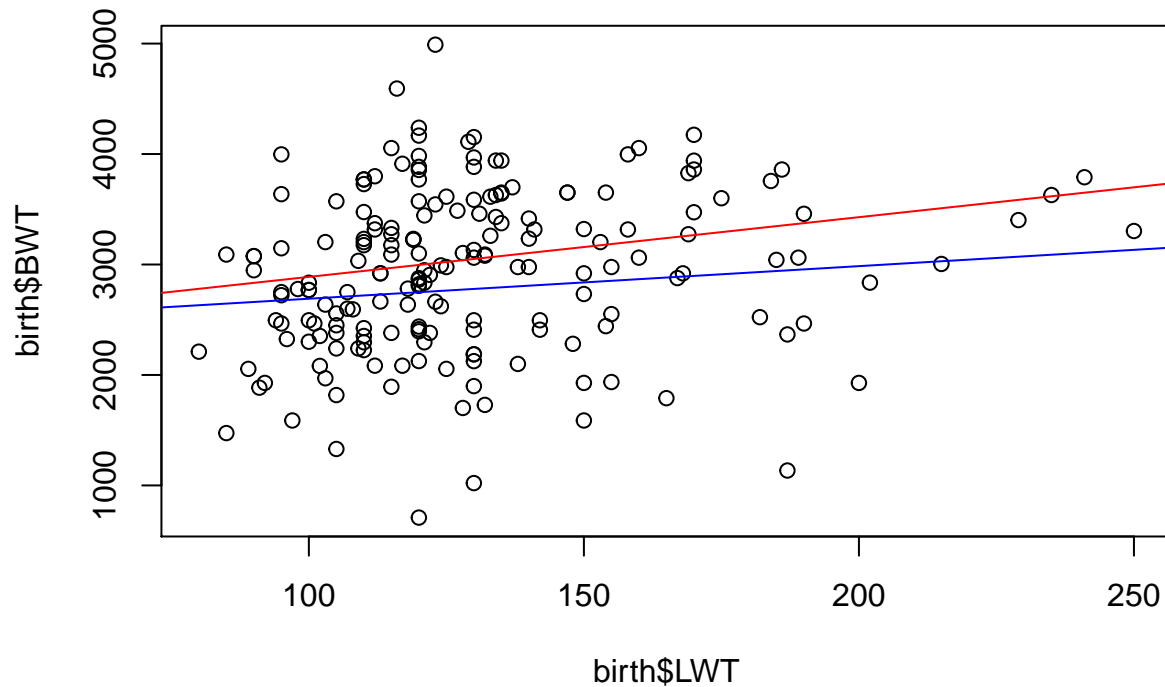
Fit the same model as birth1, but separate models for non-smokers and smokers, and call the models birth2 and birth3. (Hint: select observations by the subset argument in the lm-function using the smokeYes variable.)

```
birth2 <- lm(BWT ~ LWT, data = birth, subset = -smokeYes)
birth3 <- lm(BWT ~ LWT, data = birth, subset = smokeYes)
```

Interprete these models

- Make a scatter plot of LWT vs BWT and add two fitted lines from the model fitted above.

```
plot(x = birth$LWT, y = birth$BWT)
abline(birth2, col = "red")
abline(birth3, col = "blue")
```



- Comment on the plot

Fitted lines for both non-smokers and smokers seems very similar, but it is difficult to tell whether they are significantly different. We will later see how we can model both mother-groups simultaneously and be able to test this difference.

- Is LWT significant at a 5% level on BWT for the smokers?

```
summary(birth3)
```

Call:

```
lm(formula = BWT ~ LWT, data = birth, subset = smokeYes)
```

Residuals:

Min	1Q	Median	3Q	Max
-2040.3	-416.3	33.9	472.2	1488.7

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	2395.37	301.47	7.95	0.000000000019 ***
LWT	2.95	2.28	1.30	0.2

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

s: 657 on 72 degrees of freedom

Multiple R-squared: 0.0228,

Adjusted R-squared: 0.00921

F-statistic: 1.68 on 1 and 72 DF, p-value: 0.199

The hypothesis for testing the significance of LWT is,

$$H_0 : \beta_1 = 0 \text{ vs } H_1 : \beta_1 \neq 0$$

From the summary of model birth3 above, p-value corresponding to LWT is higher than 0.05 and we fail to reject H_0 , which suggests that LWT is not significant for smokers group. In other words, LWT does not have any linear relationship with BWT at 95% confidence level for smokers group.

Assume a model with both LWT and AGE as predictors for BWT using all observations.

- Write up the model and the model assumptions.

$$\text{BWT} = \beta_0 + \beta_1 \text{LWT} + \beta_2 \text{AGE} + \epsilon$$

Assumptions:

The error term ϵ follows $N(0, \sigma^2)$ iid, i.e error terms are independently normally distributed with mean 0 and constant variance σ^2 .

- What is the interpretation of the regression coefficients?
1. β_1 gives the expected amount of change in BWT for unit change in LWT when AGE is held constant, i.e. if LWT increases by 1 pound, BWT will increase by β_1 grams for people of the same AGE.
 2. β_2 gives the expected amount of change in BWT (in grams) if AGE increase by 1 year and LWT is held constant.

Fit the model in RStudio, call it birth4 and comment on the results.

```
birth4 <- lm(BWT ~ LWT + AGE, data = birth)
summary(birth4)
```

Call:

```
lm(formula = BWT ~ LWT + AGE, data = birth)
```

Residuals:

Min	1Q	Median	3Q	Max
-2232.8	-500.5	32.1	520.3	1899.3

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	2215.76	299.24	7.4	0.00000000000044 ***
LWT	4.18	1.74	2.4	0.018 *
AGE	8.02	10.06	0.8	0.426

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

s: 719 on 186 degrees of freedom

Multiple R-squared: 0.0378,

Adjusted R-squared: 0.0275

F-statistic: 3.65 on 2 and 186 DF, p-value: 0.0278

The summary output shows that LWT is significant at 5% level of significance but not at 1%. AGE has very high p-value and thus is not significant, i.e. there is not any linear relationship of AGE with BWT. The explained variation is still very low with an $R^2 = 0.038$.

Optional:

Look at the presentation file Regression.Rmd from lecture 2 and produce for the birth4-model a similar 3D-plot as on page 15. You may need to install the R-packages: rgl, nlme, mgcv and car first. Use the figure to get an understanding of the effects of LWT and AGE on BWT.

A 3D plot

```
library(scatterplot3d)
with(birth, {
```

```

# Start Plot
plot3d <- scatterplot3d(LWT, AGE, BWT, type = "p", highlight.3d = TRUE,
                        mar = c(3, 3, 2, 3), pch = 19, cex.symbols = 0.5,
                        main = "Residuals and fitted plane for model: birth4",
                        angle = 45, box = FALSE)

# Add fitted plane for model birth4
plot3d$plane3d(birth4, col = "grey50", lty.box = "solid", polygon_args = list(bg = "li

# True Values
true <- plot3d$xyz.convert(LWT, AGE, BWT)

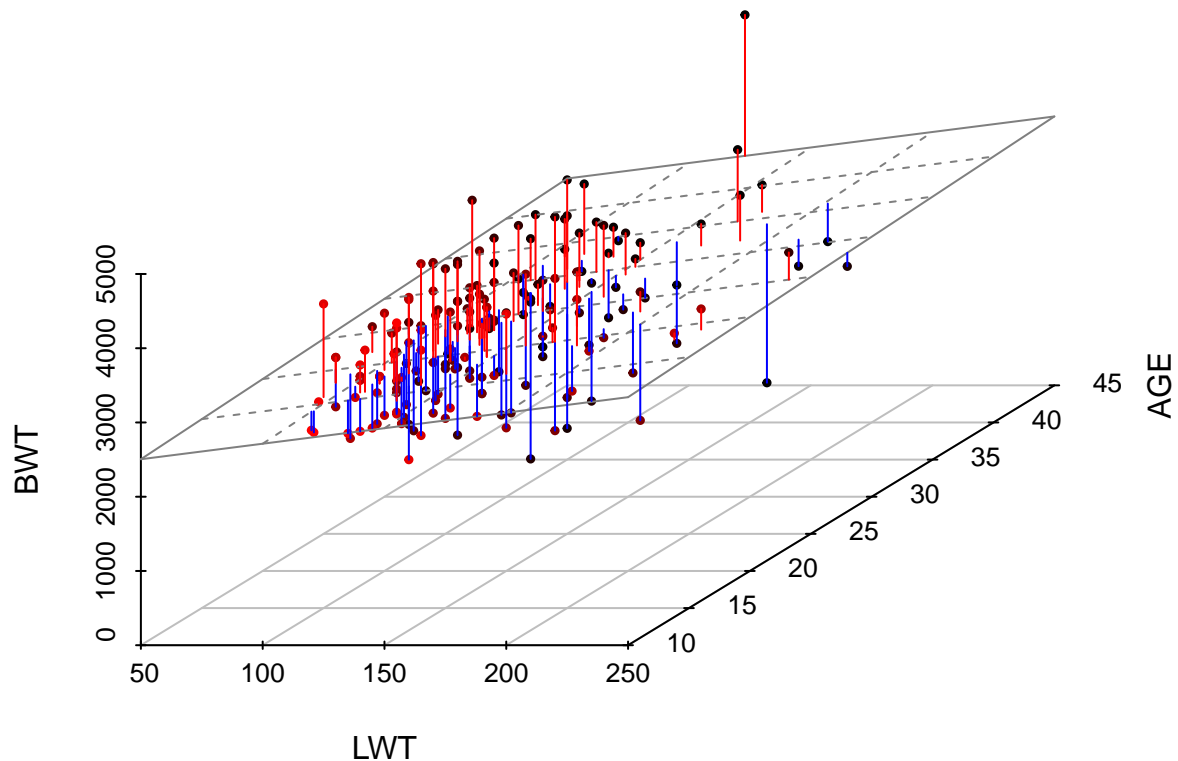
# Predicted Values
fitted <- plot3d$xyz.convert(LWT, AGE, fitted(birth4))

# Is the residuals negative?
neg_res <- 1 + (resid(birth4) > 0)

# Add segment for the residuals
segments(true$x, true$y, fitted$x, fitted$y, col = c("blue", "red")[neg_res])
})

```

Residuals and fitted plane for model: birth4



An interactive 3D plot

```
library(car)
scatter3d(BWT ~ LWT + AGE, data = birth, axis.ticks = TRUE, revolutions = 1)
```

For grouped: Smoking vs Non-Smoking:

```
car::scatter3d(BWT ~ LWT + AGE, data = birth, axis.ticks = TRUE,
               revolutions = 1, groups = birth$SMK)
```

Interpretation

- What is the interpretation of the estimated regression coefficients for LWT and AGE in this model?

From the summary output of `birth4` model, the β coefficient for LWT is 4.179 and AGE is 8.021. This shows that, if weight of mother (LWT) increases by 1 pound, the birth weight (BWT) is estimated to increase by 4.179 grams if AGE is held constant. Similarly, if the age of a mother (AGE) increases by 1 year, the birth weight (BWT) is estimated to increase by 8.021 grams, if LWT is held constant. The regression coefficients are therefore equal to the slopes of the gridlines of the surface in the figure.

Dataset: bodydata

Training Samples

Create a training data set called `bodytrain` containing the first 20 observations only, by:

```
bodytrain <- bodydata[1:20,]
```

Fitting Model

Fit one at a time three simple regression models with Weight as response and each of Height, Age and Circumference as predictors, name the models `Model1`, `Model2` and `Model3`, respectively. Use the `summary()` function on each model to print out a summary of the fitted models. Use the first 20 observations as your training data.

```
model1 <- lm(Weight ~ Height, data = bodytrain)
model2 <- lm(Weight ~ Age, data = bodytrain)
model3 <- lm(Weight ~ Circumference, data = bodytrain)
```

Understanding the fitted Model

- Test whether the three predictors are significant. Use a 5% test level.

The summary result for model1 is,

```
summary(model1)
```

Call:

```
lm(formula = Weight ~ Height, data = bodytrain)
```

Residuals:

Min	1Q	Median	3Q	Max
-11.516	-2.596	0.026	2.914	11.745

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-38.231	38.360	-1.00	0.3322
Height	0.639	0.212	3.01	0.0076 **

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

s: 5.84 on 18 degrees of freedom

Multiple R-squared: 0.334,

Adjusted R-squared: 0.297

F-statistic: 9.04 on 1 and 18 DF, p-value: 0.00757

Here, at 5% test level, Height is significant (p-value for Height is less than 0.05). The summary result for model2 is,

```
summary(model2)
```

Call:

```
lm(formula = Weight ~ Age, data = bodytrain)
```

Residuals:

Min	1Q	Median	3Q	Max
-15.084	-3.918	0.898	4.251	12.439

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	80.745	14.563	5.54	0.000029 ***
Age	-0.159	0.625	-0.25	0.8

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

s: 7.14 on 18 degrees of freedom

Multiple R-squared: 0.00359,

Adjusted R-squared: -0.0518

F-statistic: 0.0648 on 1 and 18 DF, p-value: 0.802

Here, at 5% test level, Age is not significant (p-value for age is greater than 0.05). Finally, the summary result for model3 is,

```
summary(model3)
```

Call:

```
lm(formula = Weight ~ Circumference, data = bodytrain)
```

Residuals:

Min	1Q	Median	3Q	Max
-8.937	-3.540	0.038	2.956	8.659

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	3.674	17.104	0.21	0.83234
Circumference	0.914	0.212	4.30	0.00043 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

s: 5.03 on 18 degrees of freedom

Multiple R-squared: 0.507,

Adjusted R-squared: 0.479

F-statistic: 18.5 on 1 and 18 DF, p-value: 0.000431

Here, at 5% test level, Circumference is significant (p-value for circumference is less than 0.05).

- Which model gives a better linear fit in terms of R-squared?

The model with Circumference as predictor has highest R^2 among the models. This model explains 50.67 percent of variation present in the response Weight.

Compute the correlation matrix of the bodydtrain data by:

```
cormat <- cor(bodytrain)
```

You can square the correlations by:

```
cormat ^ 2
```

	Weight	Height	Age	Circumference
Weight	1.00000	0.33440	0.00358748	0.50671271
Height	0.33440	1.00000	0.00167276	0.06116116
Age	0.00359	0.00167	1.00000000	0.00000422
Circumference	0.50671	0.06116	0.00000422	1.00000000

- Compare the squared correlations under the Weight column with the R-squared values from the three models.

The square of correlation between each predictor variable with response is equals to the R^2 obtained in model1, model2 and model3. However, this only applies to simple regression with one predictor.

If we “predict” the same observations as was used to fit the model, we get the so-called fitted values. These can be retrieved from the model1 by model1\$fitted.values

- Compute the squared correlations between the weights of bodytrain and the fitted values of model1.

```
cors <- cor(bodytrain[ , 1], model1$fitted.values)
cors ^ 2
```

```
[1] 0.334
```

- Compare with the R-squared of model1

The square of correlation between the fitted values from a model with the response is equal to the R^2 obtained from the model. This is a result which extends to multiple regression.

- For each model locate and compare the estimates for the error variance, σ^2 .

By applying the `anova()` function to each model object we obtain the analysis of variance tables containing the MSE, i.e. the Mean Sum of Squares of the Error (Residuals), which is the estimator for the error variance.

```
anova(model1)
```

Analysis of Variance Table

Response: Weight

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Height	1	308	308.3	9.04	0.0076 **
Residuals	18	614	34.1		

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```
anova(model2)
```

Analysis of Variance Table

Response: Weight

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Age	1	3	3.3	0.06	0.8
Residuals	18	919	51.0		

```
anova(model3)
```

Analysis of Variance Table

Response: Weight


```

              Df Sum Sq Mean Sq F value    Pr(>F)
Circumference  1     467      467    18.5 0.00043 ***
Residuals     18     455        25
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

- Which model has the smallest error variance estimate?

Model 3 has the smallest error variance estimate. We can also obtain the error variance estimate using the “Residual standard error” from the summary output since, $MSE = s^2$.

Multiple Linear Regression and Prediction

- Fit a model 4 with both Height and Circumference as predictors.

```

model4 <- lm(Weight ~ Height + Circumference, data = bodytrain)
summary(model4)

```

Call:

```
lm(formula = Weight ~ Height + Circumference, data = bodytrain)
```

Residuals:

```

      Min       1Q   Median       3Q      Max
-5.319 -3.536 -0.782  2.803  6.397

```

Coefficients:

```

              Estimate Std. Error t value Pr(>|t|)
(Intercept)   -70.820     28.452   -2.49  0.02346 *
Height          0.473      0.157    3.02  0.00770 **
Circumference   0.778      0.182    4.27  0.00051 ***

```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

s: 4.17 on 17 degrees of freedom

Multiple R-squared: 0.679,

Adjusted R-squared: 0.641

F-statistic: 18 on 2 and 17 DF, p-value: 0.0000638

- Get test observations for prediction: (Make a test data set called *bodytest* containing observations 21:40 (Hint: Check how we made *bodytrain* above))

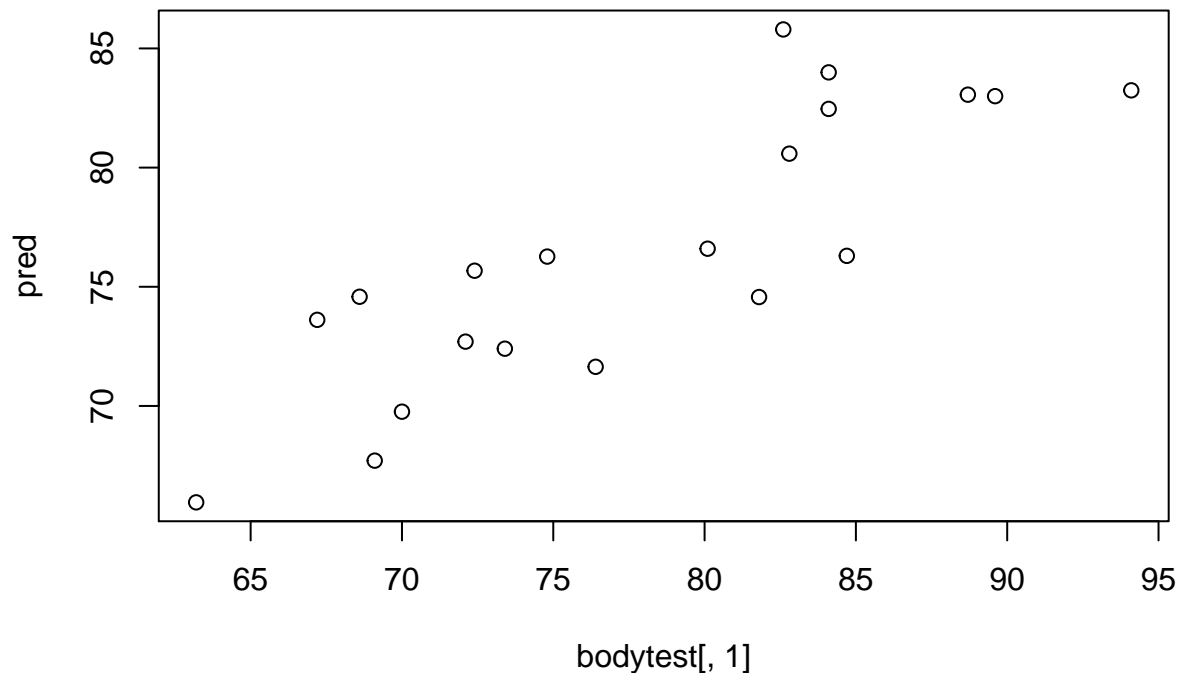
```
bodytest <- bodydata[21:40, ]
```

- Use *model4* to predict the Weights of the testdata.

```
pred <- predict(model4, newdata = bodytest)
```

- Make a scatter plot of the actually observed weights of the test data and the predicted weights.

```
plot(bodytest[, 1], pred)
```



- Compute the squared correlation between the actually observed Weights and the predicted weights.

```
cor(pred, bodytest[, 1]) ^ 2
```

```
[1] 0.714
```

What you get here is a so-called “prediction R-squared” of this model.

- Compare with the R-squared of *model4*

The prediction R-squared is close to the R-squared of model4 (0.679) which indicates that the results from model4 generalize well to new observations.

Extra on R-squared

In statistics we aim at finding models which fit the data well. However, the R-squared may easily lead to overfitting of models, that is by including too many variables.

- Fit a model with all three predictors to the bodytrain data:

```
model5 <- lm(Weight ~ Height + Circumference + Age, data = bodytrain)
summary(model5)
```

Call:

```
lm(formula = Weight ~ Height + Circumference + Age, data = bodytrain)
```

Residuals:

Min	1Q	Median	3Q	Max
-5.409	-3.390	-0.912	3.040	6.983

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-66.579	30.010	-2.22	0.04133	*
Height	0.477	0.160	2.98	0.00883	**
Circumference	0.777	0.186	4.18	0.00071	***
Age	-0.209	0.373	-0.56	0.58235	

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

s: 4.26 on 16 degrees of freedom

Multiple R-squared: 0.685,

Adjusted R-squared: 0.626

F-statistic: 11.6 on 3 and 16 DF, p-value: 0.000273

- Lets add some randomly generated junk from a normal distribution

```
Junk1 <- rnorm(n = 20, mean = 0, sd = 10)
Junk2 <- rnorm(20, 0, 10)
Junk3 <- rnorm(20, 0, 10)
model6 <- lm(Weight ~ Height + Circumference + Age +
             Junk1 + Junk2 + Junk3, data = bodytrain)
summary(model6)
```

Call:

```
lm(formula = Weight ~ Height + Circumference + Age + Junk1 +
    Junk2 + Junk3, data = bodytrain)
```

Residuals:

Min	1Q	Median	3Q	Max
-4.90	-2.94	-1.18	2.96	6.47

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-45.2110	34.4211	-1.31	0.2117
Height	0.3716	0.1850	2.01	0.0659 .
Circumference	0.7785	0.2384	3.27	0.0061 **
Age	-0.3277	0.4102	-0.80	0.4387
Junk1	-0.1569	0.1394	-1.13	0.2808
Junk2	0.1641	0.1367	1.20	0.2512
Junk3	-0.0771	0.1361	-0.57	0.5807

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

s: 4.34 on 13 degrees of freedom

Multiple R-squared: 0.734,

Adjusted R-squared: 0.611

F-statistic: 5.98 on 6 and 13 DF, p-value: 0.00346

Exercises

- Compare models 5 and 6. What happens to the R-squared? (Compare also the adjusted

R-squared values for models 5 and 6.)

The results will vary from time to time since we sample random junk, but in general we will observe that R-squared increase, whereas the adjusted R-squared decrease as we add more junk variables.

- Try to add 3 more junk variables, Junk4, Junk5 and Junk6.

```
Junk4 <- rnorm(n = 20, mean = 0, sd = 10)
Junk5 <- rnorm(20, 0, 10)
Junk6 <- rnorm(20, 0, 10)
model6 <- lm(Weight ~ Height + Circumference + Age + Junk1 + Junk2 + Junk3 +
             Junk4 + Junk5 + Junk6, data = bodytrain)
summary(model6)
```

Call:

```
lm(formula = Weight ~ Height + Circumference + Age + Junk1 +
    Junk2 + Junk3 + Junk4 + Junk5 + Junk6, data = bodytrain)
```

Residuals:

Min	1Q	Median	3Q	Max
-4.47	-2.07	1.04	1.92	4.34

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-94.5392	34.6101	-2.73	0.0211 *
Height	0.6117	0.1901	3.22	0.0092 **
Circumference	0.7818	0.2005	3.90	0.0030 **
Age	-0.0701	0.3488	-0.20	0.8448
Junk1	0.0822	0.1653	0.50	0.6295
Junk2	-0.0209	0.1469	-0.14	0.8897
Junk3	0.0488	0.1323	0.37	0.7197
Junk4	-0.2964	0.1181	-2.51	0.0309 *
Junk5	0.1626	0.1280	1.27	0.2326
Junk6	-0.0469	0.0826	-0.57	0.5822

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

s: 3.55 on 10 degrees of freedom

Multiple R-squared: 0.863,

Adjusted R-squared: 0.74

F-statistic: 7.01 on 9 and 10 DF, p-value: 0.00269

- Observe the R-squared values (*What is the lesson to be learned here?*)

Adding variables and only observing R-squared may be misleading. We should at least also keep in mind that a simple model is better. Hence, if adding more variables does not increase R-squared very much, we should keep the simpler model. If in addition the difference between the R-squared and the adjusted R-squared starts to get large, it is a clear indicator of overfitting.

Dataset: mtcars

We will use an old data set from 1974 on gasoline consumption for various cars which is part of the datasets package in R.

```
head(mtcars)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.62	16.5	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.88	17.0	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.32	18.6	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.21	19.4	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.44	17.0	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.46	20.2	1	0	3	1

As you can see there are multiple variables. If you have the datasets package you may look at the help file for the data by:

```
?datasets::mtcars
```

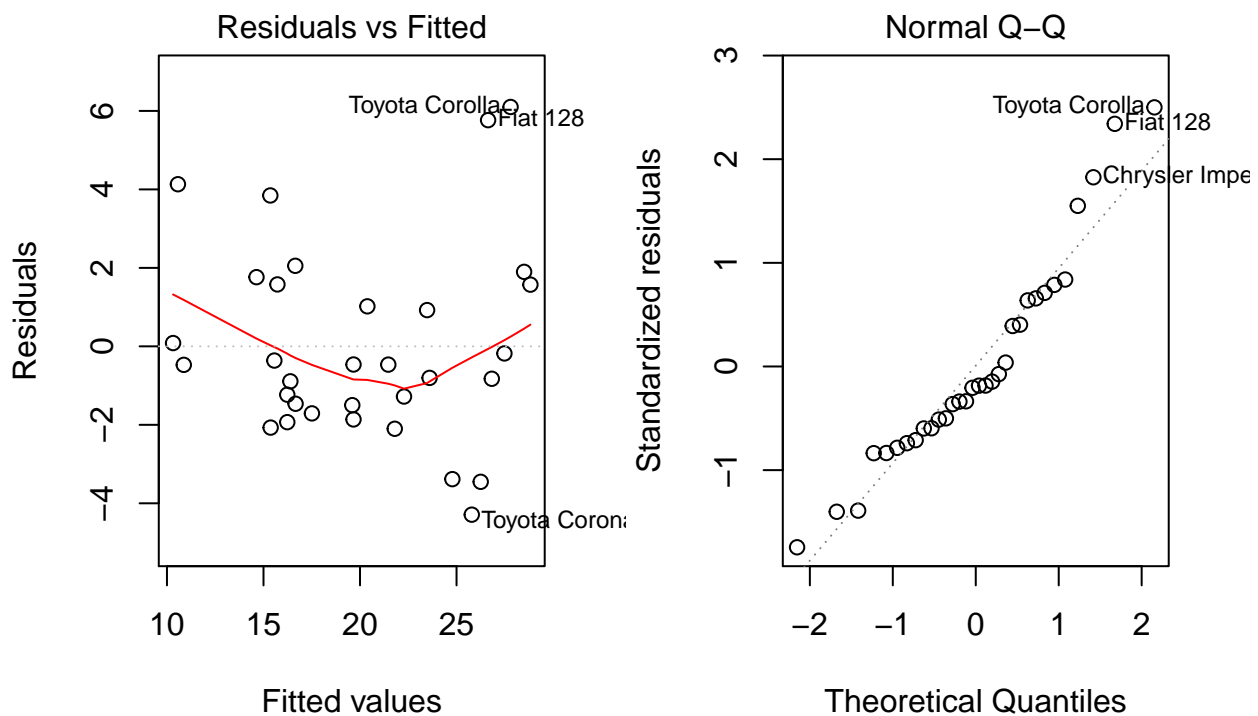
Ex-1: Model Fitting

- Fit a multiple linear regression model with mpg (Miles/Gallon) as response variable and wt (Weight) and cyl (Number of cylinders) as predictors. Call the model object "cars1".

```
cars1 <- lm(mpg ~ cyl + wt, data = mtcars)
```

- Check the model assumptions by residual analysis.

```
par(mfrow = c(1,2)) #This creates a layout for plots, one row and two columns
plot(cars1, which = c(1,2)) #Only the two first residual plots
```



- Give a summary of the results and compute the ANOVA-table.

```
summary(cars1)
```

Call:

```
lm(formula = mpg ~ cyl + wt, data = mtcars)
```

Residuals:

Min	1Q	Median	3Q	Max
-4.289	-1.551	-0.468	1.574	6.100

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	39.686	1.715	23.14	< 2e-16 ***
cyl	-1.508	0.415	-3.64	0.00106 **
wt	-3.191	0.757	-4.22	0.00022 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

s: 2.57 on 29 degrees of freedom

Multiple R-squared: 0.83,

Adjusted R-squared: 0.819

F-statistic: 70.9 on 2 and 29 DF, p-value: 0.00000000000681

```
anova(cars1)
```

Analysis of Variance Table

Response: mpg

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
cyl	1	818	818	124.0	0.0000000000054 ***
wt	1	117	117	17.8	0.00022 ***
Residuals	29	191	7		

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

- Give a short report on the results.

Both cyl and wt appears to be highly significant predictors for mpg. The estimated effects are negative implying that the mileage decreases as both weight and cylinder numbers increase, which is a reasonable result. The R^2 is 0.83, hence, about 83% of the variability in mileage is explained by the linear relationship with cyl and wt. The residual plot of fitted values versus residuals gives an indication of a non-linear relationship, which may be a result of non-linear dependencies or missing explanatory variable(s). The normal

probability plot is more or less OK.

Ex-2: Indicator variable

The am variable is an indicator variable for transmission system of the cars, 0=automatic, 1=manual. Run the following model in R:

```
cars2 <- lm(mpg ~ cyl + wt*am, data = mtcars)
```

- Write up the assumed model which has been run here. Also write up the estimated models for automatic and manual transmission, respectively.

The model is:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_2 \cdot x_3 + \epsilon$$

where $y = \text{mpg}$, $x_1 = \text{cyl}$, $x_2 = \text{wt}$, $x_3 = \text{am}$ and $\epsilon \sim N(0, \sigma^2)$.

The fitted model from R is:

```
summary(cars2)
```

Call:

```
lm(formula = mpg ~ cyl + wt * am, data = mtcars)
```

Residuals:

Min	1Q	Median	3Q	Max
-3.462	-1.491	-0.788	1.396	5.350

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	34.283	2.796	12.26	0.00000000000015 ***
cyl	-1.181	0.380	-3.11	0.0044 **
wt	-2.369	0.824	-2.87	0.0078 **
am	11.939	3.845	3.10	0.0044 **
wt:am	-4.197	1.312	-3.20	0.0035 **

```

---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

s: 2.26 on 27 degrees of freedom
Multiple R-squared:  0.877,
Adjusted R-squared:  0.859
F-statistic: 48.1 on 4 and 27 DF,  p-value: 0.000000000000664

```

For automatic transmission ($am=x_3=0$) we have the estimated model:

$$\hat{y} = 34.28 - 1.18x_1 - 2.37x_2$$

For manual transmission ($am=x_3=1$) we have

$$\begin{aligned}\hat{y} &= 34.28 - 1.18x_1 - 2.37x_2 + 11.94 \cdot 1 - 4.20x_2 \cdot 1 \\ &= 46.22 - 1.18x_1 - 6.57x_2\end{aligned}$$

We observe that the negative effect of weight on mileage is larger for manual transmission than for automatic.

Ex-3: Comparing models - Partial F-test

From the p-values we observe that transmission gives a significant addition to the intercept and to the effect of weight, respectively. These p-values correspond to testing each effect GIVEN that all other variables are included in the model. Sometimes we would rather like to test several effects jointly. For instance, should we add both transmission (am) AND the interaction between transmission and weight ($wt:am$) to the model? This is a joint test of the significance of transmission in the model. To accomplish this we may compare the fits of `cars2` (a full model) with `cars1` (a reduced model) since the difference between these models are exactly the transmission effects. This is called a partial F-test (Fisher test) where we test whether the SSE has decreased significantly as we go from the reduced model to the full model. The partial F-test may be run in RStudio by:

```
anova(cars1, cars2)
```

Analysis of Variance Table

Model 1: mpg ~ cyl + wt

Model 2: mpg ~ cyl + wt * am

	Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
1	29	191				
2	27	138	2	52.7	5.13	0.013 *

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

- From the output we see that the test statistic is an F-statistic. Is there a significant effect of transmission do you think?

A lengthy answer:

We are here really testing the hypotheses:

$$H_0 : \beta_3 = \beta_4 = 0$$

versus the alternative that at least one of them is different from zero.

We reject the null-hypothesis at test-level α if

$$F = \frac{(SSE_{\text{red.mod}} - SSE_{\text{full.mod}})/r}{MSE_{\text{full.mod}}}$$

is larger than $F_{\alpha, r, n-p}$, where r is the difference in the degrees of freedom for SSE for the two models (here $r = 2$), and $n - p$ are the degrees of freedom for SSE of the full model (here $n - p = 27$).

From the output we have (note RSS = SSE):

$$F = \frac{(191.17 - 138.51)/2}{138.51/27} = 5.13$$

We reject at level $\alpha = 0.05$ if this observed F is larger than $F_{0.05, 2, 27}$. At this point we could look this up in a Fisher-table, or alternatively compute this quantile of the Fisher distribution by:

```
qf(0.05, 2, 27, lower.tail = FALSE)
```

```
[1] 3.35
```

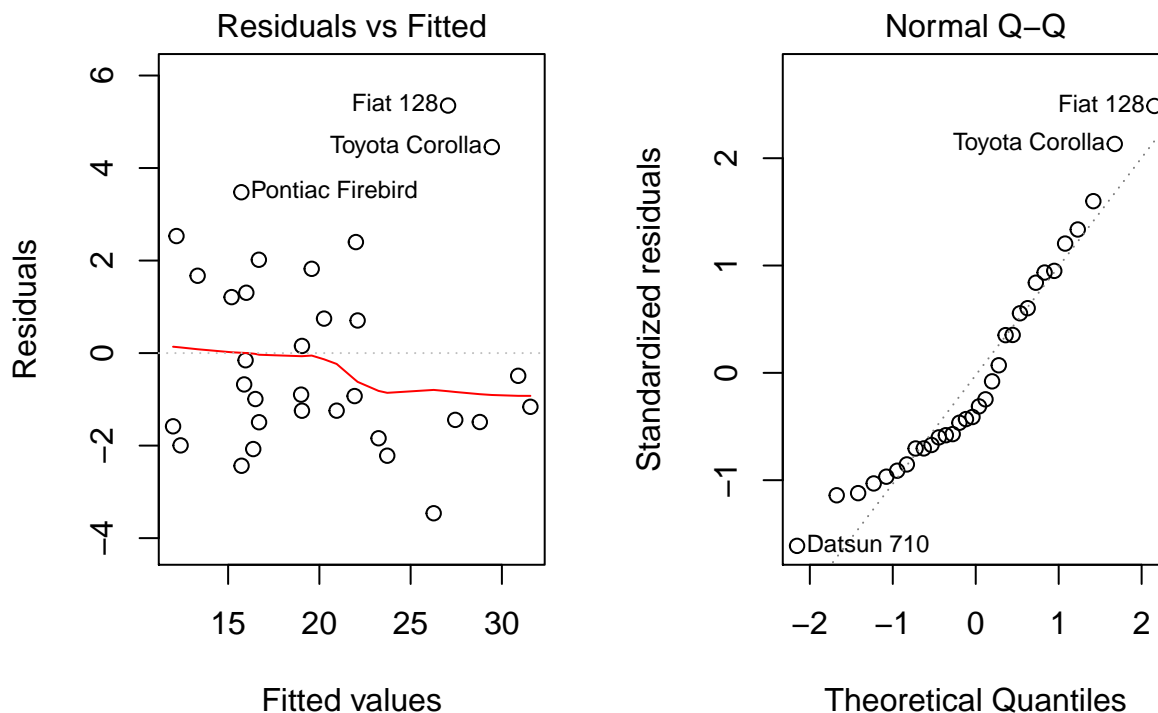
See ?FDist for help-file for the Fisher distribution.

We reject the null-hypothesis.

Alternatively we reject since the p-value from the output is smaller than 0.05.

- Perform a residual analysis of the cars2 model.

```
par(mfrow = c(1,2))
plot(cars2, which = c(1,2))
```



The linearity has improved, but maybe there is an increasing variance with increasing fitted value (estimated mileage). The normality looks good.

Ex-4: Influential measurements

- Use the `influence.measures()` function to compute the Cook's distances and the leverage (hat) values for all observations according to the cars2 model. Are there any influential observations according to these measures?

```
summary(influence.measures(cars2))
```

Potentially influential observations of

```
lm(formula = mpg ~ cyl + wt * am, data = mtcars) :
```

	dfb.1_	dfb.cyl	dfb.wt	dfb.am	dfb.wt:m	dffit	cov.r
Lincoln Continental	0.38	0.16	-0.52	-0.28	0.23	-0.59	1.57_*
Fiat 128	0.10	-0.31	0.17	0.25	-0.15	0.91	0.36_*
Ford Pantera L	0.01	-0.02	0.01	0.01	-0.02	-0.04	1.61_*
Maserati Bora	-0.03	0.09	-0.05	-0.35	0.49	0.73	1.64_*

	cook.d	hat
Lincoln Continental	0.07	0.33
Fiat 128	0.13	0.10
Ford Pantera L	0.00	0.25
Maserati Bora	0.11	0.38

Four observations are flagged by R, but none according to Cook's distance or leverage (hat).

Ex-5: Model selection

- Fit a third multiple linear regression model with mpg (Miles/Gallon) as response variable and cyl, disp, hp, drat, wt and qsec as predictor variables. Call the model object "cars3". Report a summary of the analysis.

```
cars3 <- lm(mpg ~ cyl + disp + hp + drat + wt + qsec, data = mtcars)
summary(cars3)
```

Call:

```
lm(formula = mpg ~ cyl + disp + hp + drat + wt + qsec, data = mtcars)
```

Residuals:

Min	1Q	Median	3Q	Max
-3.968	-1.580	-0.435	1.166	5.527

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	26.3074	14.6299	1.80	0.0842 .
cyl	-0.8186	0.8116	-1.01	0.3228
disp	0.0132	0.0120	1.10	0.2831
hp	-0.0179	0.0155	-1.16	0.2585
drat	1.3204	1.4795	0.89	0.3806
wt	-4.1908	1.2579	-3.33	0.0027 **
qsec	0.4015	0.5166	0.78	0.4444

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

s: 2.56 on 25 degrees of freedom

Multiple R-squared: 0.855,

Adjusted R-squared: 0.82

F-statistic: 24.5 on 6 and 25 DF, p-value: 0.00000000245

Apparently only wt is significant, but having many variables in a model may lead to inflated Std. Errors of the estimates due to correlation between predictors, and problems finding truly significant variables.

We would like to check various sub-models of this model by combining different variables. Install and load the 'mixlm' package. The package contains a function called `best.subsets()` which can help us find a good model.

```
best.subsets(cars3)
```

	cyl	disp	hp	drat	wt	qsec	RSS	R2	R2adj	Cp
1 (1)					*		278	0.753	0.745	14.56
1 (2)	*						308	0.726	0.717	19.15
1 (3)		*					317	0.718	0.709	20.50
1 (4)			*				448	0.602	0.589	40.46
1 (5)				*			604	0.464	0.446	64.30
2 (1)	*				*		191	0.830	0.819	3.24
2 (2)			*		*		195	0.827	0.815	3.83
2 (3)					*	*	195	0.826	0.814	3.89
2 (4)		*			*		247	0.781	0.766	11.72
2 (5)				*	*		269	0.761	0.744	15.17

3	(1)	*		*		*	177	0.843	0.826	3.01
3	(2)	*				*	181	0.840	0.822	3.62
3	(3)				*	*	184	0.837	0.820	4.07
3	(4)			*	*	*	184	0.837	0.819	4.09
3	(5)			*		*	186	0.835	0.817	4.45
4	(1)	*	*	*		*	170	0.849	0.826	4.07
4	(2)			*	*	*	174	0.845	0.822	4.63
4	(3)	*		*	*	*	174	0.845	0.822	4.67
4	(4)	*		*		*	175	0.844	0.821	4.80
4	(5)	*	*			*	176	0.844	0.821	4.86
5	(1)	*	*	*	*	*	167	0.851	0.823	5.60
5	(2)	*	*	*		*	169	0.850	0.821	5.80
5	(3)		*	*	*	*	170	0.849	0.820	6.02
5	(4)	*		*	*	*	171	0.848	0.819	6.20
5	(5)	*	*		*	*	172	0.847	0.818	6.34
6	(1)	*	*	*	*	*	163	0.855	0.820	7.00

The function reports by default the 5 best models for each model size (number of predictors). The model size is given in the first column. Column two is the rank within model size, then comes a column for each variable with a star indicating that a given variable is part of the model. Finally comes the residual sum of squares (RSS or SSE), R^2 , R^2 -adjusted and finally a diagnostic called Mallow's Cp.

- Which sub-model would you say is the best fitting model according to the R^2 -adjusted?

A couple of models are quite similar, but the largest R^2 -adjusted is obtained with a model with predictors `cyl`, `hp` and `wt`. This is also a quite simple model with few predictors. We should always strive for simple models and choose the simpler model in cases where the fit appears to be more or less equal for several models.

Ex-6: Model validation

On canvas you find a file called "CV.R" containing two functions `CV()` and `Kfold()`. Download this file and open it in RStudio and press the "source" button up to the right in the script window. This will run the file and create these functions. You can also scour the file as,

```
source('_functions/CV.R')
```

A fitted model should ideally be validated on a test set of new and un-touched data. We could predict the new samples using our best choice model and evaluate the prediction performance. If the model predicts well, we probably have a good model!

If we don't have a test set, we may perform Cross-validation. The most common version is the Leave-One-Out Cross-validation where we successively remove one observation from the data and fit the model to the remaining observations. The fitted model is used to predict the left out observation. After fitting a model, the left out observation is put back, and another is left out. In total we then fit n models, and perform n predictions.

- Use the `CV()` function to perform a Leave-One-Out CV using the `cars2` model fit by:

```
res <- CV(cars2)
```

```
print(res)
```

```
$pred
```

```
[1] 22.0 20.1 26.6 19.4 16.4 19.1 16.6 21.3 21.9 19.0 19.1 15.1 15.9 15.9
[15] 13.1 12.8 11.1 26.5 31.0 28.7 24.5 16.6 16.9 15.9 15.4 29.0 27.6 32.0
[29] 16.0 21.1 12.3 23.7
```

```
$mse
```

```
[1] 6.09
```

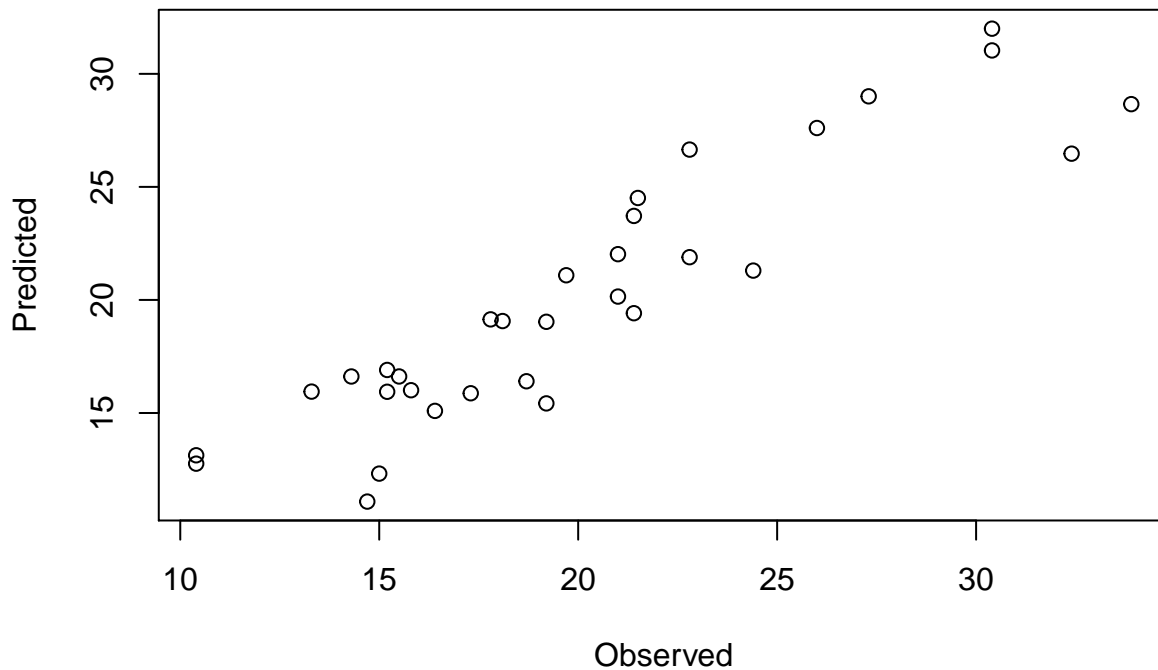
```
$r.squared.pred
```

```
[1] 0.828
```

The `CV()` returns a list with three elements, the predictions, the Mean Square Error of Prediction and R^2 -predicted.

- Make a plot of the observed mpg versus the cross-validation predictions.

```
plot(mtcars$mpg, res$pred, xlab = "Observed", ylab = "Predicted")
```

- Is the best model from the previous exercise, identified by the `best.subsets()`, a better model in terms of prediction error (MSEP)? The MSEP is defined by:

$$\text{MSEP} = \frac{1}{n} \sum_{i=1}^n \left(y_i - \hat{y}_{(i)} \right)^2$$

where $\hat{y}_{(i)}$ is the prediction of y_i using a model where observation i was left out from the model estimation. A small value implies better prediction.

```
cars4 <- lm(mpg ~ cyl + hp + wt, data = mtcars)
CV(cars4)
```

```
$pred
```

```
[1] 22.97 22.08 26.26 20.91 16.98 20.41 15.65 23.65 23.38 20.03 20.10
[12] 14.97 16.05 16.07 11.02 10.09  8.74 26.32 28.71 27.35 25.83 17.69
[23] 18.09 14.79 15.57 27.70 26.62 27.72 16.58 21.28 12.89 24.61
```

```
$mse
```

```
[1] 7.19
```

```
$r.squared.pred
```

```
[1] 0.797
```

No, this model does not predict better than cars2.

- Leave-One-out CV is known to have large uncertainty, and using a K-fold CV is an alternative. Then the data are divided into K subsets (folds) of approximately equal sizes, and a leave-one-fold-out CV is performed instead. A K=10 is often recommended. Here, since n=32 a K=8 is better, since this gives subsets of equal sizes. Create K=8 random folds by

```
myfolds <- Kfold(n = 32, K = 8, random = TRUE)
```

```
print(myfolds)
```

```
[[1]]
```

```
[1] 11 22 14 10
```

```
[[2]]
```

```
[1] 8 6 4 21
```

```
[[3]]
```

```
[1] 17 25 9 16
```

```
[[4]]
```

```
[1] 29 26 5 3
```

```
[[5]]
```

```
[1] 1 24 19 30
```

```
[[6]]
```

```
[1] 32 15 28 31
```

```
[[7]]
```

```
[1] 2 27 20 23
```

```
[[8]]
```

```
[1] 13 18 7 12
```

Since the folds are sampled randomly, we get r different folds each time we run `Kfold()`.

As you see, `myfolds` is a list of 8 random subsets of observation numbers. Re-run the

validation of cars2 by

```
CV(cars2, folds = myfolds)
```

```
$pred
```

```
[1] 22.3 20.1 27.0 19.5 16.2 18.8 16.6 21.7 21.9 19.2 19.2 15.3 16.1 16.1  
[15] 13.2 10.8 11.0 26.5 31.2 28.8 23.5 16.8 16.8 16.0 15.3 29.4 26.9 32.4  
[29] 16.0 21.3 12.3 23.2
```

```
$mse
```

```
[1] 5.83
```

```
$r.squared.pred
```

```
[1] 0.837
```

Note that the result now will vary if you repeat the creation of random K-folds for the cross-validation. Try to make a new myfolds and run the CV again.

Chapter 4

Analysis of variance

We will use following datasets for this exercise:

```
load("_data/city.rdata")
load("_data/nsr.rdata")
load("_data/barley.rdata")
```

R-package we will use in this exercises

```
# load the library
library(mixlm)
library(effects)
```

Chlorine levels in cities

Independent measurements of chlorine (ppm parts per million) were taken from 3 large cities:

```
city
```

	City1	City2	City3
1	0.46	1.21	0.44
2	0.92	10.86	0.71
3	0.86	2.09	2.87
4	0.33	6.76	0.81

5	0.53	3.20	7.34
6	5.13	2.02	2.21
7	1.51	2.16	9.00
8	1.00	0.44	17.20
9	0.70	7.80	2.76
10	1.89	1.20	1.07
11	0.68	1.39	2.07
12	0.71	2.42	1.70
13	0.41	1.62	7.89
14	0.78	10.78	0.79
15	0.11	3.25	0.24
16	2.70	0.41	4.44
17	0.41	0.52	15.40
18	0.75	3.71	4.21
19	0.81	5.20	8.63
20	0.35	13.89	3.90

- Load the `city.rdata` which is available on Canvas.

Before analyzing the data it is best to “stack” the data into two columns, a response column `y` and a city factor column `city`. By using the `stack()` function in R you can restructure the city data by

```
citydata <- stack(city)
```

- Use the `colnames()` function to rename the variables as `y` and `city`.

```
colnames(citydata) <- c("y", "city")
```

- Assume an ANOVA-model with chlorine as response and city as a factor. What assumptions do you make?

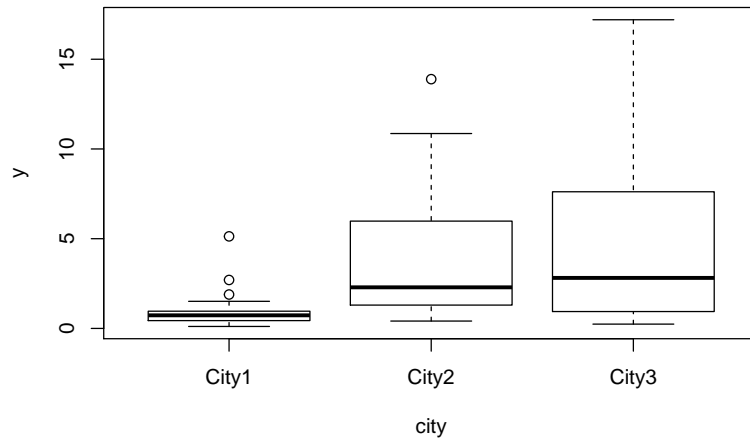
The model assumptions are:

$$y_{ij} = \mu_i + \epsilon_{ij}$$

where the error terms are assumed independent and identically distributed $\epsilon_{ij} \sim N(0, \sigma^2)$. Further μ_i is the expected chlorine level in city i .

- Make a box-plot of the data. Describe the variabilities between cities and within cities.

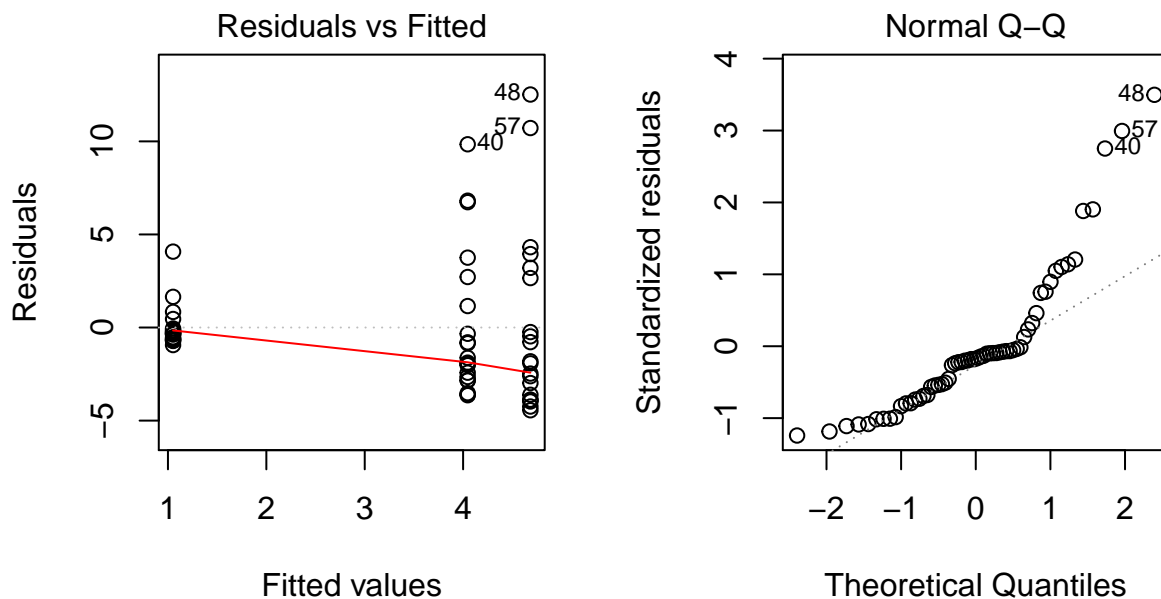
```
plot(y ~ city, data = citydata)
```



City 1 seems to have both the lowest average chlorine levels and the smallest variability in the measurements. Cities 2 and 3 are quite similar with both higher means and variability.

- Fit the model and perform a residual analysis. Comment on the model fit.

```
options(contrasts=c("contr.treatment", "contr.poly"))
citymod1 <- lm(y ~ city, data = citydata)
par(mfrow = c(1, 2))
plot(citymod1, which = c(1, 2))
```



The residual analysis reveals two problems: 1) Non-constant variability, which was also

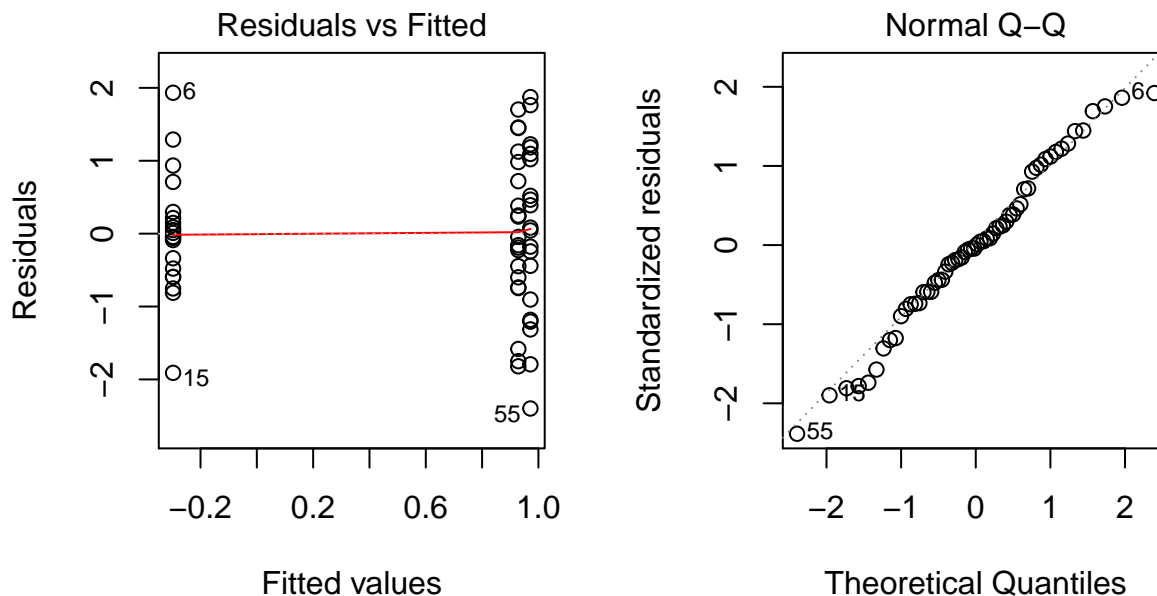
observed in the boxplot, and 2) A skewed and non-normal distribution for the residuals with a heavy tail to the right.

- Make a log-transformation of y to create the variable `logy`, and add the new variable to your data set by:

```
citydata$logy <- log(citydata$y)
```

- Fit a new model with `logy` as response and check the model fit again.

```
citymod2 <- lm(logy ~ city, data = citydata)
par(mfrow = c(1,2))
plot(citymod2, which = c(1,2))
```



Both problems from the previous model appear to be corrected. We continue with this model.

- Estimate all model parameters from the latter model. Explain what the parameters measure given the parameterization of your choice (See lecture notes for parameterization).

The ANOVA analysis using `contr.treatment` parametrization:

```
summary(citymod2)
```

Call:


```
lm(formula = logy ~ city, data = citydata)
```

Residuals:

	Min	1Q	Median	3Q	Max
	-2.3987	-0.5954	-0.0175	0.7122	1.9326

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-0.297	0.231	-1.29	0.20266
city(City2)	1.226	0.326	3.76	0.00041 ***
city(City3)	1.269	0.326	3.89	0.00027 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

s: 1.03 on 57 degrees of freedom

Multiple R-squared: 0.255,

Adjusted R-squared: 0.229

F-statistic: 9.75 on 2 and 57 DF, p-value: 0.000228

```
Anova(citymod2)
```

Anova Table (Type II tests)

Response: logy

	Sum Sq	Df	F value	Pr(>F)
city	20.8	2	9.75	0.00023 ***
Residuals	60.7	57		

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```
coefs <- citymod2$coefficients
```

We observe that $R^2 = 0.25$ indicating that there is still a lot of unexplained variability in the response. We get the following estimates for the city means of $\log(\text{chlorine})$ -levels using this parametrization (city 1 is reference city):

$$\mu_1 = \mu = -0.3$$

$$\mu_2 = \mu + \alpha_2 = 0.93$$

$$\mu_3 = \mu + \alpha_3 = 0.97$$

The estimate of the error variance is $\hat{\sigma}^2 = MSE = SSE/(N - 3) = 60.724/57 = 1.065$.

As we see from above, the “intercept” μ is the expected log-chlorine level for city 1. The α_2 is the difference between city 1 and city 2, and α_3 is the difference between city 1 and city 3. The σ^2 is the variance for log-chlorine levels measured in the same city.

- State the hypotheses for testing whether the expected chlorine levels differ between the cities, choose a test level α and perform the test. What is the conclusion?

The hypotheses are:

$$H_0 : \mu_1 = \mu_2 = \mu_3$$

versus

$$H_1 : \mu_i \neq \mu_{i'}$$

for at least two different cities i and i' .

Note: This is by the chosen parametrization equivalent to testing

$$H_0 : \alpha_2 = \alpha_3 = 0$$

versus

$$H_1 : \alpha_i \neq 0$$

for at least one $i \in \{2, 3\}$.

For test-level $\alpha = 0.05$ we reject the null-hypothesis if $F > F_{\alpha, 2, 57}$ or if the p-value is smaller than 0.05. Here we observe a very small p-value and reject the null-hypothesis. We claim that the expected log-chlorine levels differ between at least two of the cities, and from the observed means we know that cities 1 and 3 are significantly different since they have the largest observed difference in means. There may also be a significant difference between cities 1 and 2, and 2 and 3, but this should be checked by a pair-wise contrast.

Data from the NSR education test

The Norwegian Centre for Science Recruitment (NSR) has an online “education test” where youths may answer a questionnaire to check their so-called cognitive types, their science interest, their preferred learning methods and their interest to various science subjects. The test suggests different areas within the STEM (Science, Technology, Engineering and Mathematics) within which the youth may find suitable work.

We have an excerpt of these data which can be downloaded from Canvas as the `nsr.rdata` file. The data.frame `NSRdata` contains two variables, Science and Age:

```
head(NSRdata, 5)
```

	Science	Age
101	4.2	16
102	4.2	16
103	4.2	16
104	3.0	19
105	2.8	16

Science is an average liking score (scale 1-6) to various STEM-subjects, and Age is a factor indicating different age-groups:

- a) 1: 1-12 yrs
- b) 13: 13-15 yrs
- c) 16: 16-19 yrs
- d) 19: 19-29 yrs
- e) 30: 30 + yrs

- Perform an analysis of the NSR data to check whether Age influences the liking to STEM subjects. State the model, fit the model, check model assumptions, test hypotheses, and give model critique. Write a short summary of the results.

Answer will come later.

- Follow up the previous exercise by performing a Tukey test for all pair wise comparisons with overall (family-wise) error rate 5%. Give a summary of the results.

```
NSRmod <- lm(Science ~ Age, data = NSRdata)
Anova(NSRmod, type = "III")
```

Anova Table (Type III tests)

Response: Science

	Sum Sq	Df	F value	Pr(>F)
(Intercept)	1601	1	1557.6	<2e-16 ***
Age	225	4	54.7	<2e-16 ***
Residuals	10270	9995		

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```
pt <- simple.glht(NSRmod, "Age", corr = c("Tukey"), level = 0.95)
print(pt)
```

Simultaneous Confidence Intervals and Tests for General Linear Hypotheses

Multiple Comparisons of Means: Tukey Contrasts

Fit: `lm(formula = Science ~ Age, data = NSRdata)`

Quantile = 2.73

Minimum significant difference = 0.157

95% confidence level

Linear Hypotheses:

	Lower	Center	Upper	Std.Err	t value	P(>t)
1-13	0.0455	0.2021	0.3588	0.0574	3.52	0.0040 **
1-16	-0.0183	0.1384	0.2951	0.0574	2.41	0.1126
1-19	0.1079	0.2646	0.4212	0.0574	4.61	0.00004063 ***
1-30	-0.3466	-0.1899	-0.0333	0.0574	-3.31	0.0084 **
13-16	-0.2204	-0.0637	0.0929	0.0574	-1.11	0.8015
13-19	-0.0942	0.0624	0.2191	0.0574	1.09	0.8132
13-30	-0.5487	-0.3921	-0.2354	0.0574	-6.83	< 2e-16 ***
16-19	-0.0305	0.1262	0.2828	0.0574	2.20	0.1808
16-30	-0.4850	-0.3283	-0.1717	0.0574	-5.72	0.00000011 ***
19-30	-0.6112	-0.4545	-0.2978	0.0574	-7.91	< 2e-16 ***

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(Adjusted p values reported -- single-step method)
```

WARNING: Unbalanced data may lead to poor estimates

The anova table indicates a clear significance of Age-group with regard to the interest level to Science subjects. hence, at least two age-groups are have significantly different expected means. We can use the Tukey test to identify pairwise differences.

The Tukey output provides differences in mean (column “center”) between all pairs of groups, and at the top of output the minimum difference yielding a significant difference in means is given to be 0.157. That is, all age-groups with a difference in averages of more than 0.157 are significantly different, according to Tukey. From the p-values we observe that 6 out of 10 pairs are significantly different, and the largest difference is found between age groups 30+ and 19-29.

```
cld(pt)
```

Tukey's HSD

Alpha: 0.05

	Mean	G1	G2	G3
30	3.33	A		
1	3.14		B	
16	3.00		B	C
13	2.94			C
19	2.88			C

The compact letter display gives a grouping of the similar levels, and there are three groups of levels that are internally non-significantly different. 30+ is different from all other levels, whereas 1-12 and 16-19 are similar and 16-19, 13-15 and 19-29 are also similar.

Barley Data

In `barley.rdata` are results from an experiment where the response is yield of barley per 1000 square meter, and the factors sorts of barley, soil types, types of fertilizers. In addition was the experiment done in two different geographical areas (sites).

- Assume a two factor model including the main effects of sort and soil and their interaction. State the model and explain all parameters under a sum-to-zero parametrization.

The model can be written as:

$$y_{ijk} = \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij} + \epsilon_{ijk}$$

where, $\epsilon \sim N(0, \sigma^2)$

The parametrization restrictions for “sum-to-zero” are:

$$\sum_i \alpha_i = 0, \sum_j \beta_j = 0, \sum_i (\alpha\beta)_{ij} = 0 \text{ and } \sum_j (\alpha\beta)_{ij} = 0$$

- Fit the model in R and estimate all parameters

```
options(contrasts = c("contr.sum", "contr.poly"))
mod1 <- lm(Yield ~ sort * soil, data = Barley)
summod1 <- summary(mod1)
summod1$coef
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	380.9	10.1	37.55	1.70e-25
sort(1)	-23.6	10.1	-2.33	2.73e-02
soil(1)	-43.1	10.1	-4.25	2.14e-04
sort(1):soil(1)	13.8	10.1	1.36	1.84e-01

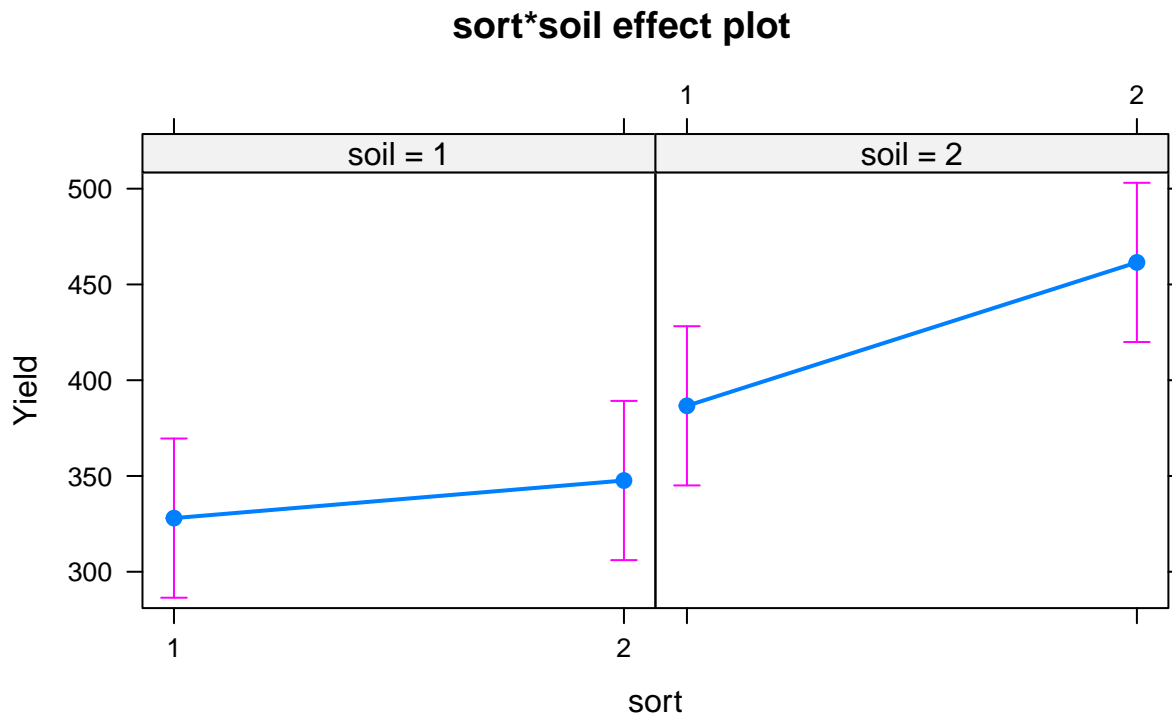
```
summod1$sigma ^ 2
```

```
[1] 3293
```

From the model summaries we find the estimated parameters to be:

1. The estimated overall mean yield: $\hat{\mu} = 380.94$
 2. The estimated effects of sort 1 and 2: $\hat{\alpha}_1 = -23.625, \hat{\alpha}_2 = -\hat{\alpha}_1 = 23.625$
 3. The estimated effects of soil 1 and 2: $\hat{\beta}_1 = -43.125, \hat{\beta}_2 = -\hat{\beta}_1 = 43.125$
 4. The estimated interaction effects of sort and soil: $\hat{\alpha}\hat{\beta}_{11} = 13.812, \hat{\alpha}\hat{\beta}_{12} = -\hat{\alpha}\hat{\beta}_{11} -13.812, \hat{\alpha}\hat{\beta}_{21} = -\hat{\alpha}\hat{\beta}_{11} -13.812, \hat{\alpha}\hat{\beta}_{22} = -\hat{\alpha}\hat{\beta}_{21} = -\hat{\alpha}\hat{\beta}_{12} = 13.812$
 5. The estimated within sort:soil levels unexplained variability: $\hat{\sigma}^2 = MSE = 3293.21$
- Make an interaction plot and try to conclude about the presence of interaction from the plot. How would you explain interaction effect in this example for a person with experience in agriculture, but minimal statistical experience?

```
plot(allEffects(mod1, confidence.level = .95))
```



From the plot, it seems that a farmer can expect an increase in yield when using sort 2 instead of sort 1 for both types of soil, and there may be a higher gain of sort 2 over sort 1 for soil-type 2 than for soil type 1. If this is so, there is a so-called interaction effect of sort and soil on yield.

- Perform a hypothesis test for the interaction effect. Conclusion?

```
Anova(mod1, type = "III")
```

Anova Table (Type III tests)

Response: Yield

	Sum Sq	Df	F value	Pr(>F)	
(Intercept)	4643628	1	1410.06	< 2e-16	***
sort	17861	1	5.42	0.02731	*
soil	59513	1	18.07	0.00021	***
sort:soil	6105	1	1.85	0.18419	
Residuals	92210	28			

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

The interaction between soil and sort is insignificant (high p-value). This result indicates that the slight non-parallel tendency of the lines in the interaction plot may be due to random errors. More data would be needed to verify any interaction effect.

- Fit a four factor model with sort, soil, fert and site as factors and with all possible 2-factor, 3-factor and 4-factor interactions. Use `Anova()` from the `car`-package with argument `type="III"` to produce an ANOVA table. Type III means that all factors are tested as if they were the last factor to be added to the model. Are there any higher order significant effects? Compare the R-squared and the adjusted R-squared values. What can you conclude from these?

```
mod2 <- lm(Yield ~ sort * soil * fert * site, data = Barley)
Anova(mod2, type = "III")
```

Anova Table (Type III tests)

Response: Yield

	Sum Sq	Df	F value	Pr(>F)	
(Intercept)	4643628	1	4503.19	< 2e-16	***
sort	17861	1	17.32	0.00073	***
soil	59512	1	57.71	0.0000011	***
fert	2813	1	2.73	0.11812	
site	1458	1	1.41	0.25176	
sort:soil	6105	1	5.92	0.02707	*
sort:fert	2080	1	2.02	0.17472	
soil:fert	33930	1	32.90	0.0000306	***
sort:site	23220	1	22.52	0.00022	***

soil:site	595	1	0.58	0.45849
fert:site	105	1	0.10	0.75364
sort:soil:fert	2245	1	2.18	0.15953
sort:soil:site	1405	1	1.36	0.26029
sort:fert:site	7442	1	7.22	0.01622 *
soil:fert:site	12	1	0.01	0.91370
sort:soil:fert:site	406	1	0.39	0.53914
Residuals	16499	16		

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```
summary(mod2)$r.squared
```

```
[1] 0.906
```

```
summary(mod2)$adj.r.squared
```

```
[1] 0.818
```

There is only one interaction effect significant at 5% levels, namely the `sort:fert:site` interaction. However, the least squares estimator is prone to inflated variance estimates and difficulties in finding significance when the number of variables k rises compared to the number of observations N and if variables are highly correlated. Too many variables to estimate from a limited number of observations leaves few degrees of freedom to SSE, and tests with low power. Lack of significance may be an over-fitting problem. By removing some of the least significant variables, the problem is reduced, and a reduced model with significant effects may be identified.

The large difference between R^2 and R_{adj}^2 is also an indication of an over-fitted model.

- Fit also a reduced model without site, but all other effects up to 3rd order interactions. Perform a partial F-test with the `anova()` function to test whether site (and all its interactions with the others) should be excluded from the model (See also Exercise set 3 and Ex-3).

```
mod3 <- lm(Yield ~ sort * soil * fert, data = Barley)
anova(mod2, mod3)
```

Analysis of Variance Table

```
Model 1: Yield ~ sort * soil * fert * site
```

```
Model 2: Yield ~ sort * soil * fert
```

```
  Res.Df  RSS Df Sum of Sq  F Pr(>F)
1      16 16499
2      24 51142 -8    -34644 4.2 0.0071 **
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The anova result for partial F-test shows that site is needed to a certain degree in the model. The low p-value reject the hypothesis that there is no effect of site on Yield.

- In the `mixlm`-package there are convenient functions for performing automatic model selection by either `backward()` elimination of factors from a “full-model”, by `forward()` addition of factors from a minimal model with only intercept, or a combined `stepwise()` function which combines both forward and backward addition/elimination.

For the `backward` function the least significant factor is removed from the model in each step if the p-value is larger than testlevel `alpha`. If all factors are significant at any step, the procedure stops. The elimination obeys the so-called marginality principle (hierarchy of factors) which states that any lower order effect or interaction should not be removed from the model if it is part of a higher order significant interaction. This is a good principle for practical data analysis.

Try to run the `backward()` function on the model object you created above with all four factors and interactions. Use `alpha=0.05` as test level. Use the Step-information from the output to explain which factors being excluded at each step. What is the final reduced model?

```
mod4 <- backward(mod2, alpha = 0.05)
```

```
Backward elimination, alpha-to-remove: 0.05
```

```
Full model: Yield ~ sort * soil * fert * site
```

```
<environment: 0x7faeec326438>
```

	Step	RSS	AIC	R2pred	Cp	F value	Pr(>F)
sort:soil:fert:site	1	16905	231	0.659	14.4	0.39	0.54
soil:fert:site	2	16918	229	0.696	12.4	0.01	0.91

sort:soil:site	3	18322	229	0.704	11.8	1.49	0.24
soil:site	4	18917	228	0.724	10.3	0.62	0.44
sort:soil:fert	5	21162	230	0.720	10.5	2.37	0.14

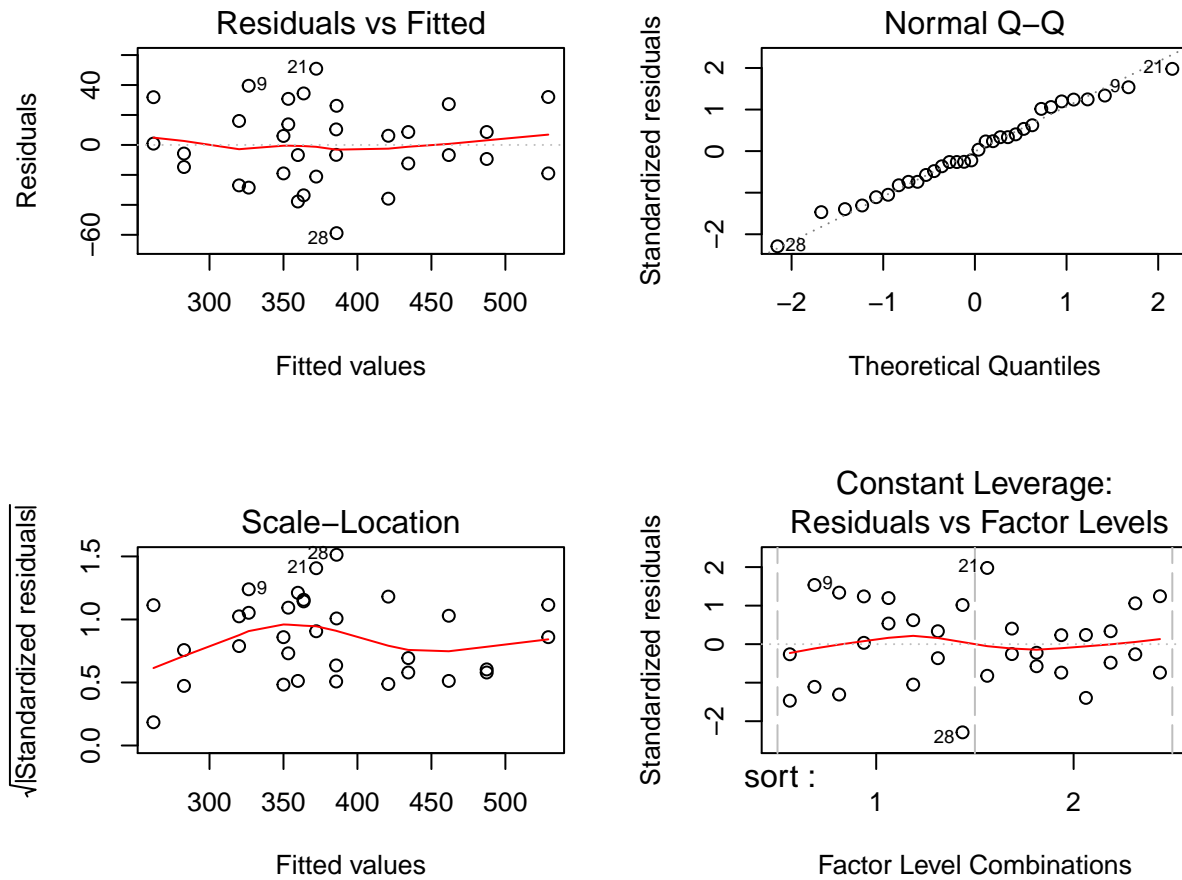
In the first step, the 4th order interaction is removed. In 2nd, 3rd and 4th step some 3rd order and a 2nd order interactions are removed. After removing interaction between sort,soil and site in the 5th step, the resulting model is,

```
Yield ~ sort + soil + fert + site + sort:soil + sort:fert + soil:fert +  
      sort:site + fert:site + sort:fert:site
```

- Fit the reduced model and perform a model check using residual analysis.

The backward function has fitted the reduced model (her named mod4):

```
mod4 <- lm(Yield ~ sort + soil + fert + site + sort:soil +  
          sort:fert + soil:fert + sort:site + fert:site + sort:fert:site,  
          data = Barley)  
  
par(mfrow = c(2,2))  
plot(mod4)
```

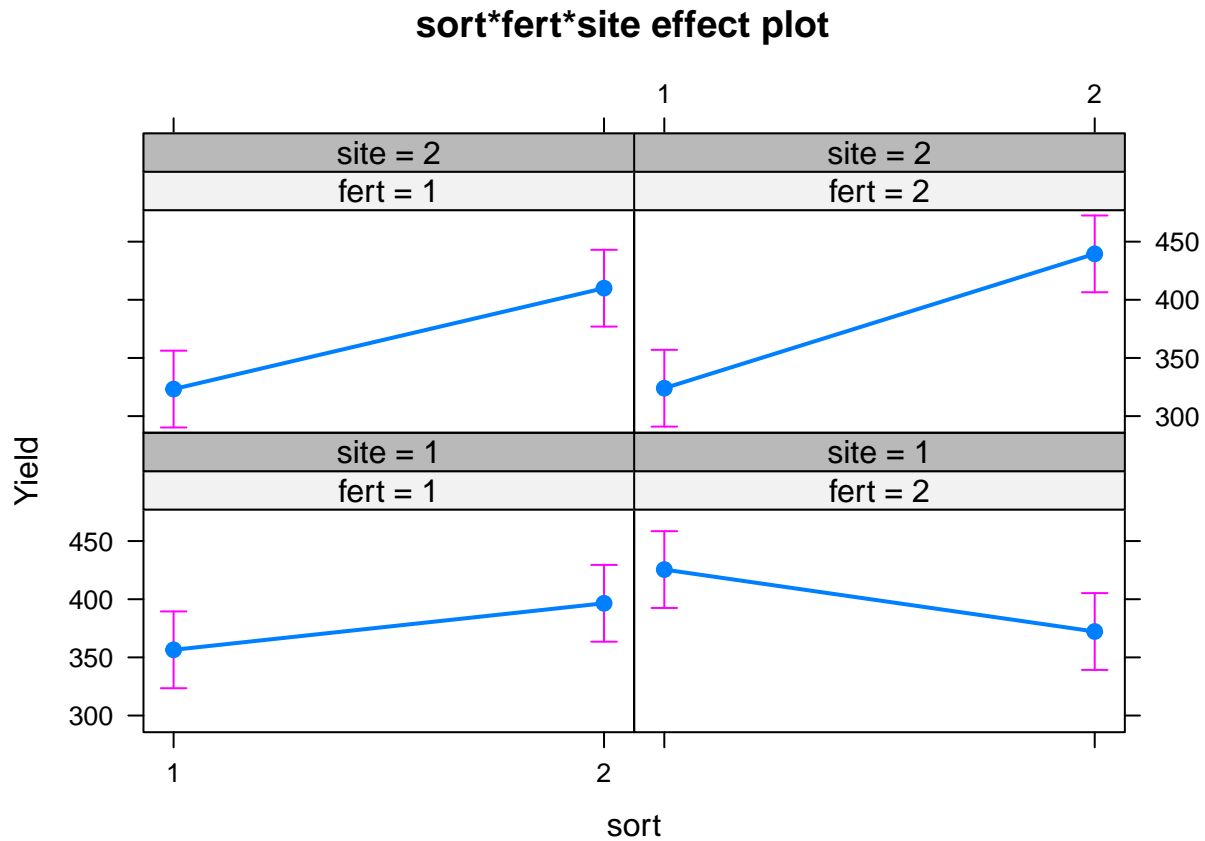


The residual plots give no evidence of any problems with the model assumptions.

- Make an interaction effects plot using the following code:

```
eff <- allEffects(mod4, confidence.level = .95)
plot(eff[3])
```

```
eff <- allEffects(mod4, confidence.level = .95)
plot(eff[3])
```



Try to explain the plot. Why do you think the interaction between sort, fertilizer and site was significant?

The plot shows that, when changing sort from 1 to 2, a farmer can expect different change in average Yield for site:1 and site:2 and the difference is not same when using fertilizer:1 and fertilizer:2.

Chapter 5

Multivariate Analysis (PCA)

We will use `trackfieldrecord.rdata` in this exercise

```
load("_data/trackfieldrecords.rdata")
```

We need to load following R package

```
library(mixlm)
```

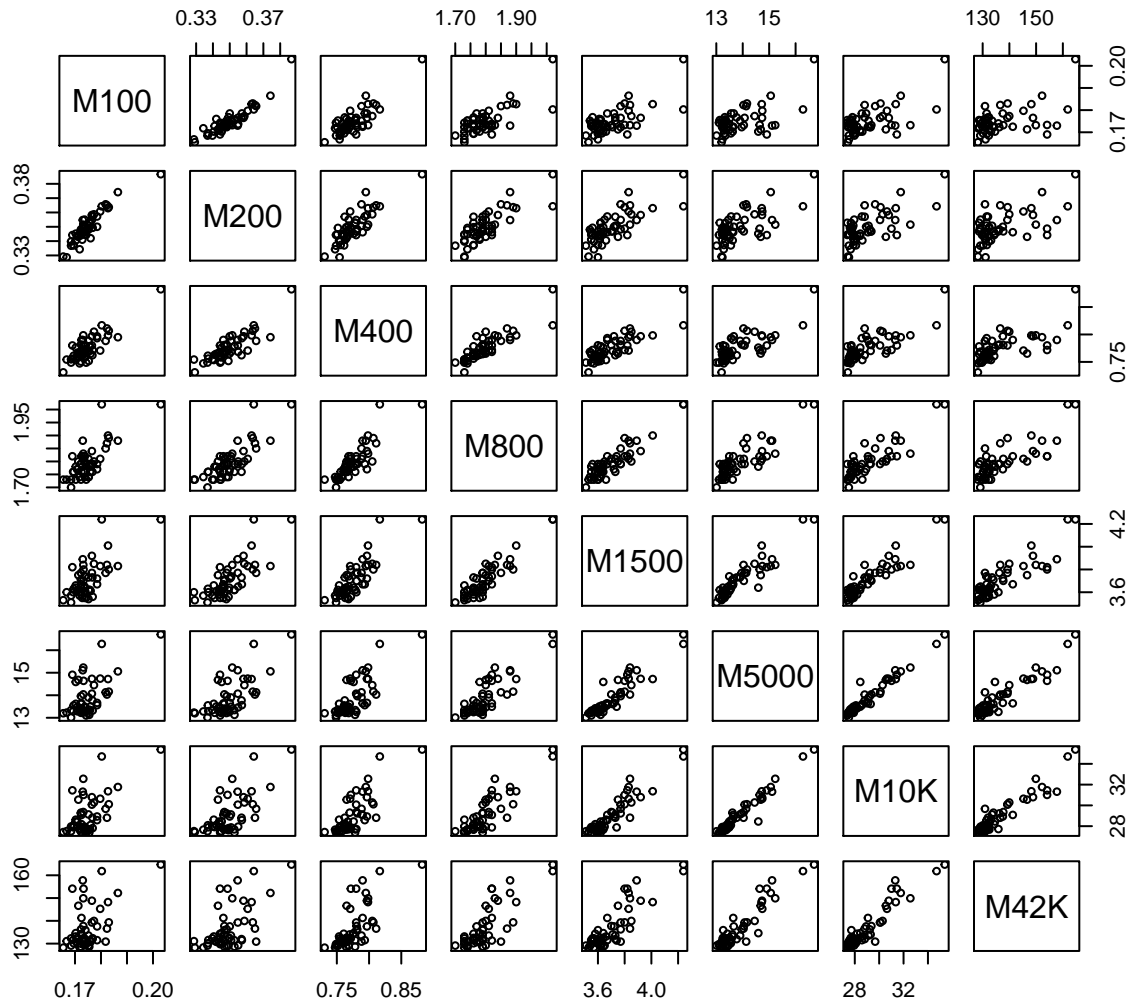
Track and Field data

Load the `trackfieldrecords.rdata` file with the objects `runMen` and `runWomen` containing national records (a few years ago...) for several track and field events like 100m, 200m and so on up to marathon.

- Explore the data for both men and women using the `pairs()` - plotting function (you must exclude the `Nation` variable since this is non-numeric). Which events appear to be most correlated with each other? Check by using the `cor()` - function.

For dataset: `runMen`

```
pairs(runMen[, -9], cex = 0.7)
```



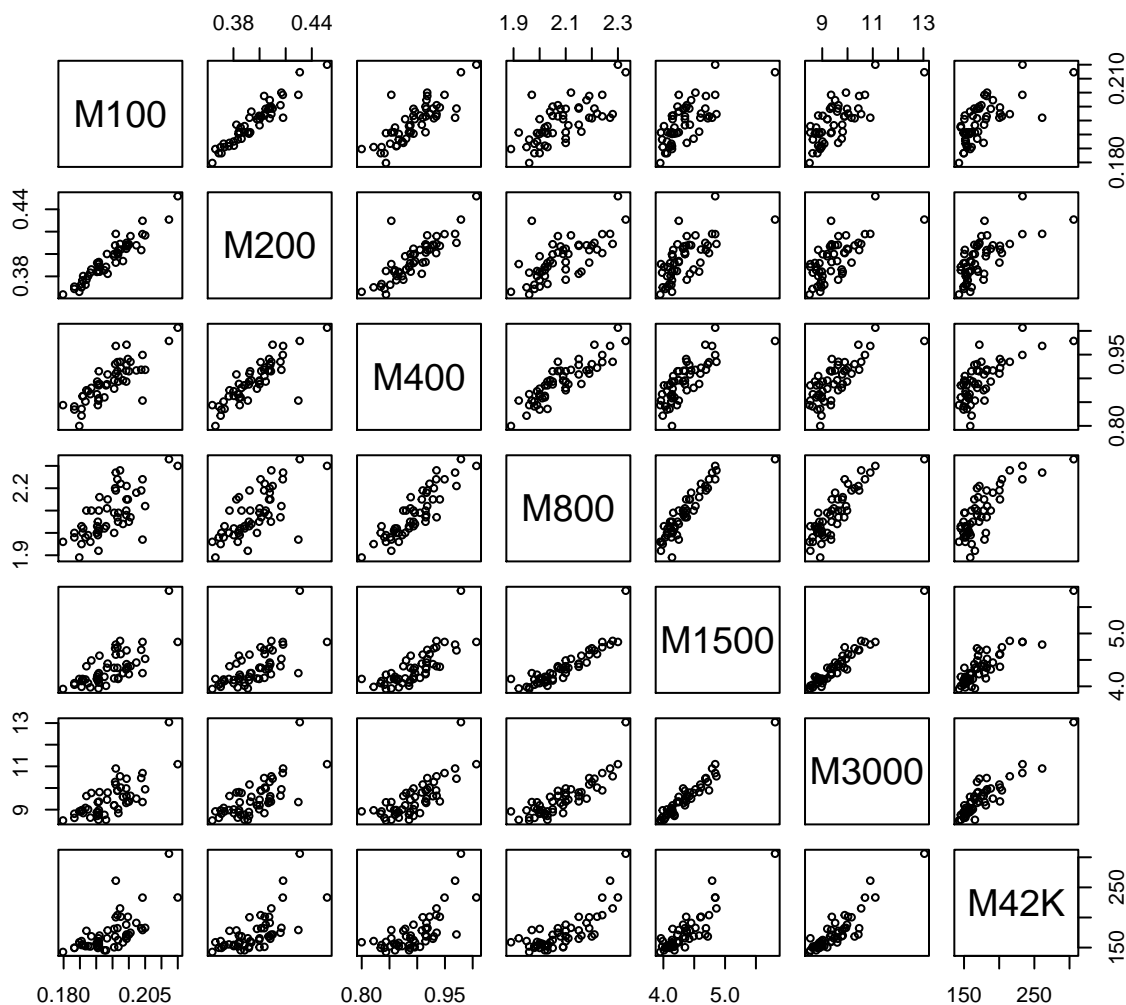
```
cor(runMen[, -9])
```

	M100	M200	M400	M800	M1500	M5000	M10K	M42K
M100	1.000	0.918	0.831	0.744	0.686	0.601	0.615	0.504
M200	0.918	1.000	0.840	0.800	0.767	0.682	0.683	0.585
M400	0.831	0.840	1.000	0.866	0.828	0.770	0.779	0.701
M800	0.744	0.800	0.866	1.000	0.914	0.858	0.864	0.803
M1500	0.686	0.767	0.828	0.914	1.000	0.926	0.933	0.865
M5000	0.601	0.682	0.770	0.858	0.926	1.000	0.974	0.931
M10K	0.615	0.683	0.779	0.864	0.933	0.974	1.000	0.943
M42K	0.504	0.585	0.701	0.803	0.865	0.931	0.943	1.000

Mens running 5000m have the highest correlation (0.974) with the mens running 10000m.

For dataset: runWomen


```
pairs(runWomen[, -8], cex = 0.7)
```



```
cor(runWomen[, -8])
```

	M100	M200	M400	M800	M1500	M3000	M42K
M100	1.000	0.947	0.808	0.693	0.704	0.724	0.683
M200	0.947	1.000	0.836	0.690	0.671	0.690	0.682
M400	0.808	0.836	1.000	0.887	0.771	0.767	0.710
M800	0.693	0.690	0.887	1.000	0.894	0.856	0.782
M1500	0.704	0.671	0.771	0.894	1.000	0.968	0.879
M3000	0.724	0.690	0.767	0.856	0.968	1.000	0.899
M42K	0.683	0.682	0.710	0.782	0.879	0.899	1.000

Womens running 3000m and 1500m have highest correlation (0.968)

- Run the following command and inspect the results:

```
cor(runWomen[, -8], runMen[, -9])
```

If you were going to “predict” a nation’s record for women’s M3000, which record among men would you use (rows in the table are women variables, columns are men)?

```
cor(runWomen[, -8], runMen[, -9])
```

	M100	M200	M400	M800	M1500	M5000	M10K	M42K
M100	0.650	0.760	0.777	0.804	0.784	0.718	0.713	0.651
M200	0.716	0.799	0.811	0.817	0.766	0.703	0.703	0.621
M400	0.645	0.715	0.792	0.778	0.770	0.738	0.731	0.691
M800	0.595	0.697	0.740	0.785	0.834	0.817	0.820	0.785
M1500	0.517	0.636	0.676	0.848	0.876	0.858	0.868	0.822
M3000	0.571	0.681	0.684	0.856	0.882	0.865	0.864	0.814
M42K	0.612	0.701	0.647	0.811	0.830	0.803	0.817	0.762

Men’s M1500 shows the highest correlation to women’s M3000 and appears to be the best indicator, but in order to really check this we should run a regression analysis with cross-validation to see which predicts best.

- Run a PCA (with `scale=FALSE`) on the men’s data and print out a summary and the weights (loadings) for the two first PC’s. How many components is needed to explain at least 99% of the variation in the data? Try to give an interpretation of PC1, and explain why so few components explains so much of the total variability.

```
pr <- prcomp(runMen[, 1:8], scale = FALSE)
summary(pr)
```

Importance of components:

	PC1	PC2	PC3	PC4	PC5	PC6
Standard deviation	9.571	0.64469	0.16890	0.05837	0.02504	0.0117
Proportion of Variance	0.995	0.00451	0.00031	0.00004	0.00001	0.0000
Cumulative Proportion	0.995	0.99964	0.99995	0.99999	1.00000	1.0000

	PC7	PC8
Standard deviation	0.00553	0.00191
Proportion of Variance	0.00000	0.00000
Cumulative Proportion	1.00000	1.00000

pr

Standard deviations (1, ..., p=8):

```
[1] 9.57139 0.64469 0.16890 0.05837 0.02504 0.01174 0.00553 0.00191
```

Rotation (n x k) = (8 x 8):

	PC1	PC2	PC3	PC4	PC5	PC6	PC7
M100	0.000311	0.00379	-0.000446	0.03426	0.0692868	0.185797	-0.387429
M200	0.000656	0.00660	-0.004506	0.06800	0.0994247	0.274399	-0.850073
M400	0.001759	0.01308	-0.005035	0.13044	0.2647439	0.885091	0.355339
M800	0.005393	0.03175	-0.020469	0.38876	0.8602795	-0.326436	0.027660
M1500	0.014222	0.08536	-0.050540	0.90239	-0.4184781	-0.015186	0.015555
M5000	0.078990	0.37121	-0.921125	-0.08648	0.0031867	0.000642	0.000858
M10K	0.180437	0.90259	0.385347	-0.06522	0.0031326	-0.002815	-0.000183
M42K	0.980290	-0.19749	0.004152	0.00345	-0.0000589	0.000652	-0.000359
	PC8						
M100	0.8996602						
M200	-0.4330211						
M400	-0.0551806						
M800	-0.0018789						
M1500	0.0073064						
M5000	0.0012362						
M10K	-0.0009295						
M42K	0.0000788						

Only one component is needed to explain more than 99% of the variation. From the loadings we see that the loading weight for M42K (Marathon) is totally dominating with a weight of 0.98. All other weights are small. PC1 is therefore more or less identical to the Marathon variable.

PC1 is located in the direction of largest variability, and due to the large scale of marathon times, this variable totally dominates the PCA. In such cases it may be better to use standardized variables (which is equivalent to running the Eigenvalue decomposition on the correlation matrix instead of the covariance matrix of the variables).

- Re-run the PCA with option `scale=TRUE` in `prcomp()`. How many variables are needed to explain 99% of the variation in this case? How much is explained by the

two first components? How would you interpret the loadings of PC1, PC2 and PC3?

```
pr <- prcomp(runMen[,1:8], scale = TRUE)
summary(pr)
```

Importance of components:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7
Standard deviation	2.563	0.953	0.4076	0.3615	0.2859	0.26585	0.21933
Proportion of Variance	0.821	0.114	0.0208	0.0163	0.0102	0.00883	0.00601
Cumulative Proportion	0.821	0.935	0.9557	0.9720	0.9822	0.99106	0.99707

	PC8
Standard deviation	0.15305
Proportion of Variance	0.00293
Cumulative Proportion	1.00000

```
pr
```

Standard deviations (1, ..., p=8):

```
[1] 2.563 0.953 0.408 0.361 0.286 0.266 0.219 0.153
```

Rotation (n x k) = (8 x 8):

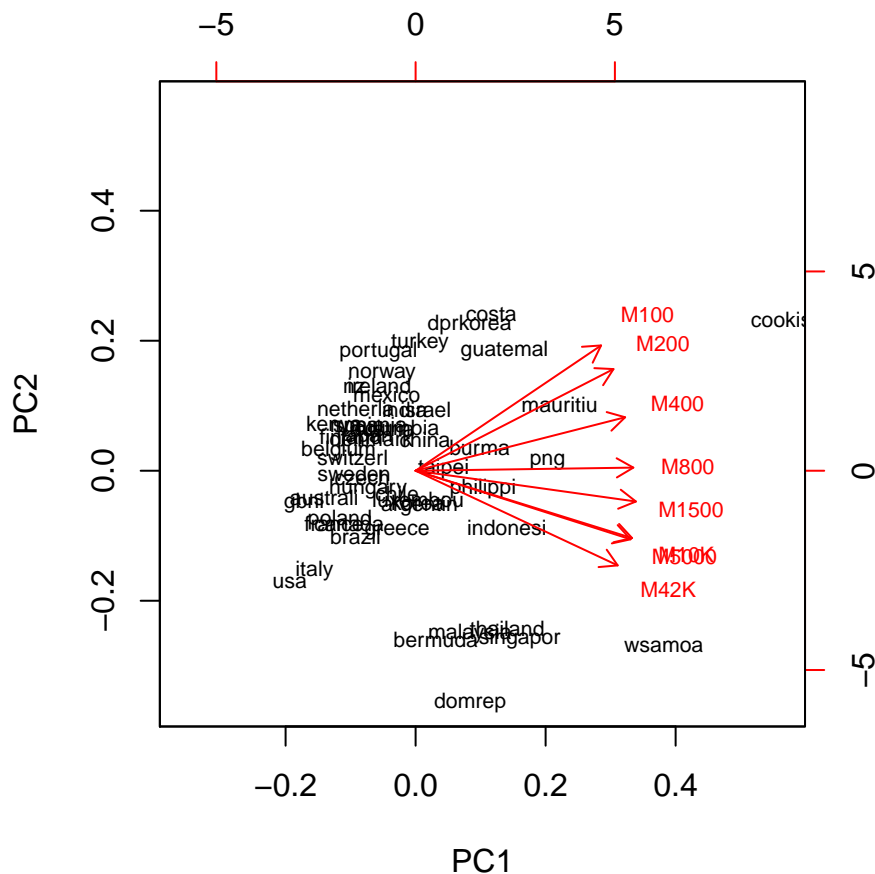
	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
M100	0.315	0.5716	0.322	-0.1783	0.269	-0.5781	0.1425	-0.10781
M200	0.336	0.4638	0.372	0.2485	-0.158	0.6498	-0.1218	0.09974
M400	0.356	0.2439	-0.615	-0.5946	-0.236	0.1619	-0.0119	0.00229
M800	0.369	0.0142	-0.496	0.5214	0.541	-0.0318	-0.2233	0.03623
M1500	0.374	-0.1404	-0.105	0.4103	-0.494	-0.1747	0.6048	-0.14352
M5000	0.365	-0.3129	0.194	-0.0463	-0.237	-0.1448	-0.5974	-0.54333
M10K	0.367	-0.3071	0.180	-0.0985	-0.120	-0.2202	-0.1728	0.79734
M42K	0.343	-0.4319	0.228	-0.3176	0.489	0.3413	0.4028	-0.15986

Now we need 6 components to achieve 99% explained variance. Two components explain about 93.5% of the total variance. The loadings for PC1 are almost identical for all variables, hence PC1 is close to identical to the average run record across all distances for each country. PC2 has weights ranging from highly negative for marathon to highly positive for M100. This PC therefore contrasts short versus long runs. PC3 is a component that contrasts medium long distances (400, 800 and 1500 m) and either short or long distances. This component appears to extract information about how these distances differ from both

sprint and endurance distances.

Make a biplot with the first two components. You may use the argument “xlabs” in biplot to indicate that “Nations” should be used to label the scores from each country. Give comments to how the nations cluster relative to the loadings.

```
biplot(pr, xlabs = runMen$Nation, cex = 0.7)
```



On the first axis (PC1) we observe that Cook's Islands and West Samoa have the largest weights, and they therefore have on average poor (long time) national records for all distance. At the other end we find USA and others with on average good national records. Along PC2 (vertically) we find those with relatively poor times on long distances, but relatively good times on short, at the bottom (Dom. Repub, Bermuda, Singapore, Malaysia, Thailand and West Samoa) whereas at the top we find countries with poor records on short distances compared to their long distance records (Costa Rica, North-Korea, Cook's Island and others).

- Let's try to predict the 3000M national records for women using the men's data. First use a least squares approach using the men's variables directly as predictor for

women's M3000. To accomplish this use `runWomen$M3000` as response in `lm()`. Are there any significant variables? What is the R^2 and the adjusted R^2 values?

```
lmmod <- lm(runWomen$M3000 ~ M100 + M200 + M400 + M800 + M1500 +
            M5000 + M10K + M42K, data = runMen)
(lm_sumry <- summary(lmmod))
```

Call:

```
lm(formula = runWomen$M3000 ~ M100 + M200 + M400 + M800 + M1500 +
    M5000 + M10K + M42K, data = runMen)
```

Residuals:

Min	1Q	Median	3Q	Max
-0.7333	-0.2060	-0.0537	0.2496	0.7184

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-6.09205	1.98465	-3.07	0.0037 **
M100	-23.17895	23.87260	-0.97	0.3370
M200	18.93566	13.76942	1.38	0.1762
M400	-11.69187	5.19182	-2.25	0.0295 *
M800	6.06528	2.25105	2.69	0.0100 *
M1500	1.70738	1.16573	1.46	0.1503
M5000	0.20068	0.28632	0.70	0.4871
M10K	0.08099	0.14783	0.55	0.5866
M42K	-0.00202	0.01715	-0.12	0.9070

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

s: 0.357 on 43 degrees of freedom

Multiple R-squared: 0.84,

Adjusted R-squared: 0.81

F-statistic: 28.2 on 8 and 43 DF, p-value: 1e-14

```
lm_sumry[c("r.squared", "adj.r.squared")]
```

\$r.squared

```
[1] 0.84
```

```
$adj.r.squared
```

```
[1] 0.81
```

Two variables are significant at 5% level in this fitted model, M800 and M400. The R^2 -values indicate more than 80% explained variance. There is a slight difference between the adjusted and the non-adjusted R^2 indicating that there may be too many variables included in the model.

- Either perform a manual elimination of insignificant variables, or run `backward()` from the `mixlm`-package to find a reduced model with only significant effects (5% testlevel). Which variables do you end up having in your model?

```
red.mod <- backward(lmmod, alpha = 0.05)
```

```
Backward elimination, alpha-to-remove: 0.05
```

```
Full model: runWomen$M3000 ~ M100 + M200 + M400 + M800 + M1500 + M5000 +
  M10K + M42K
<environment: 0x7faef1904cd8>
```

	Step	RSS	AIC	R2pred	Cp	F value	Pr(>F)
M42K	1	5.49	-101	0.644	7.01	0.01	0.907
M10K	2	5.53	-102	0.687	5.32	0.32	0.576
M100	3	5.64	-104	0.730	4.15	0.86	0.359
M200	4	5.75	-104	0.746	3.03	0.92	0.343
M5000	5	6.20	-103	0.755	4.55	3.67	0.061 .

```
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(red.mod)
```

```
Call:
```

```
lm(formula = runWomen$M3000 ~ M400 + M800 + M1500, data = runMen)
```

```
Residuals:
```

```
      Min       1Q   Median       3Q      Max
```

```
-0.6139 -0.2798 -0.0541  0.2591  0.7221
```

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-6.930	1.687	-4.11	0.00015	***
M400	-11.241	4.286	-2.62	0.01166	*
M800	6.661	2.211	3.01	0.00412	**
M1500	3.556	0.806	4.41	0.000058	***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

s: 0.359 on 48 degrees of freedom

Multiple R-squared: 0.819,

Adjusted R-squared: 0.808

F-statistic: 72.6 on 3 and 48 DF, p-value: <2e-16

In addition to M400 and M800 I find M1500 to be highly significant after removing other variables. This “covered” effect from the full model was because of the inflated variances due to multicollinear variables in the full model.

- Fit another model using all principal component scores from the men’s data as predictors for women’s M3000. The scores are stored in the principal component object as element names x. Which components are significant at a 5% test level? Compare the R^2 values with those from the full model using all original variables as predictors.

```
#You need to transform the scores into a data.frame first
PCdata <- as.data.frame(pr$x)
pcrmod <- lm(runWomen$M3000 ~ PC1 + PC2 + PC3 + PC4 + PC5 +
             PC6 + PC7 + PC8, data = PCdata)

summary(pcrmod)
```

Call:

```
lm(formula = runWomen$M3000 ~ PC1 + PC2 + PC3 + PC4 + PC5 + PC6 +
    PC7 + PC8, data = PCdata)
```


Residuals:

Min	1Q	Median	3Q	Max
-0.7333	-0.2060	-0.0537	0.2496	0.7184

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	9.49673	0.04956	191.61	< 2e-16 ***
PC1	0.27603	0.01952	14.14	< 2e-16 ***
PC2	-0.17201	0.05249	-3.28	0.00208 **
PC3	0.03614	0.12279	0.29	0.76990
PC4	0.53593	0.13844	3.87	0.00036 ***
PC5	0.00982	0.17503	0.06	0.95551
PC6	0.04261	0.18825	0.23	0.82201
PC7	-0.09547	0.22818	-0.42	0.67772
PC8	0.04242	0.32698	0.13	0.89737

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

s: 0.357 on 43 degrees of freedom

Multiple R-squared: 0.84,

Adjusted R-squared: 0.81

F-statistic: 28.2 on 8 and 43 DF, p-value: 1e-14

Components 1,2 and 4 are highly significant, the others not. The R^2 values are identical to the full model on the original variables since we use all available components. The information content in PC1-PC8 is therefore the same as in the eight original variables.

- Perform a model reduction also for the PCR-model by excluding Principal components until you have only PC's significant at 5% test level. Compare the estimated regression coefficients between the full and the reduced PCR-models. Why do you think the effects don't change for the variables retained in the model?

```
pcr.red.mod <- backward(pcrmod, alpha = 0.05)
```

Backward elimination, alpha-to-remove: 0.05

Full model: runWomen\$M3000 ~ PC1 + PC2 + PC3 + PC4 + PC5 + PC6 + PC7 + PC8
<environment: 0x7faeef9eed48>

	Step	RSS	AIC	R2pred	Cp	F value	Pr(>F)
PC5	1	5.49	-101	0.632	7.003	0.00	0.96
PC8	2	5.50	-103	0.683	5.020	0.02	0.90
PC6	3	5.50	-105	0.733	3.071	0.05	0.82
PC3	4	5.51	-107	0.749	1.158	0.09	0.76
PC7	5	5.54	-108	0.763	-0.667	0.19	0.66

```
summary(pcr.red.mod)
```

Call:

```
lm(formula = runWomen$M3000 ~ PC1 + PC2 + PC4, data = PCdata)
```

Residuals:

Min	1Q	Median	3Q	Max
-0.7232	-0.2207	-0.0461	0.2415	0.6907

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	9.4967	0.0471	201.67	< 2e-16 ***
PC1	0.2760	0.0186	14.88	< 2e-16 ***
PC2	-0.1720	0.0499	-3.45	0.00118 **
PC4	0.5359	0.1315	4.07	0.00017 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

s: 0.34 on 48 degrees of freedom

Multiple R-squared: 0.839,

Adjusted R-squared: 0.829

F-statistic: 83.3 on 3 and 48 DF, p-value: <2e-16

The reduced model has only PC1, PC2 and PC4 as predictors, and the R^2 is almost as high as for the full model. The estimated effects are identical due to the fact that the PC's are orthogonal to each other and explain independent variability.

- For special interested: Use the estimated effects (alphas from the lecture) from the reduced PCR-model to compute PCR-estimated regression coefficients (betas in

the lecture) for the original variables back back-rotation. Compare the estimated regression coefficients from the PCR-approach with Least squares estimates using original (BUT SCALED) variables.

```
#PCR-estimated betas using components 1,2 and 4. The %*% operator is the inner-product
#multiplier in R
PCRest <- pr$rotation[,c(1,2,4)] %*% (coef(pcr.red.mod)[-1])

#Scaling the original variables and re-running the least squares model:
runMen.scaled <- as.data.frame(scale(runMen[, -9]))
lmmod2 <- lm(runWomen$M3000 ~ M100 + M200 + M400 + M800 + M1500 +
             M5000 + M10K + M42K, data = runMen.scaled)

#Estimates without intercept.
Estimates <- cbind(coef(lmmod2)[-1], PCRest)
colnames(Estimates) <- c("Least Squares", "PCR")
Estimates
```

	Least Squares	PCR
M100	-0.1355	-0.10698
M200	0.2015	0.14607
M400	-0.2789	-0.26247
M800	0.3879	0.37900
M1500	0.2673	0.34725
M5000	0.1622	0.12968
M10K	0.1477	0.10142
M42K	-0.0189	-0.00125

For most of the variables the PCR-estimates are closer to zero (shrinkage effect) which induces a bias, but as lectured, the PCR-model may have smaller variance for the estimates due to avoidance of the multicollinearity problem. It seems like PCR has down-weighted the short and long distances compared to the Least Squares approach, which seems reasonable.

Chapter 6

Multivariate Analysis (PCR, PLS)

In this exercise we will study a data set used in the paper by Liland et al (2009) (<http://www.sciencedirect.com/science/article/pii/S0169743909001476>) where PLS-regression was used to predict the percentage of cow-milk in mixtures of cow, goat and ewe milk.

```
load("_data/malddidata.rdata")
```

An excerpt from the paper explains why this is interesting:

“Quality assurance is an important issue in modern food production. The products are expected to have the right taste, smell, texture and appearance. In addition they should be safe, wholesome, authentic and have a composition that complies with regulations. As a practical example this paper will analyse data simulating milk adulteration. In real life such adulteration could occur where one type of milk is replaced by, or mixed with, another deliberately, by accident or because of failing routines.

There are several reasons why detection of the concentrations of cow, goat and ewe milk is of importance. Pure products of goat milk may be used as a supplement of milk for humans who are born with allergic reactions towards cow milk. Some mixed milk products are produced with regard to specifications which specify the mixture of milk from cow, goat and/or ewe. Professionals and consumers want to control the origin of milk in order to be sure that they get products following specifications and labelling. Farmers producing more than one type of milk might be tempted to add cow milk to goat or ewe milk as

this would result in higher quantities of the better paid milk variants"

For the exercise we will need pls package.

```
library(pls)
```

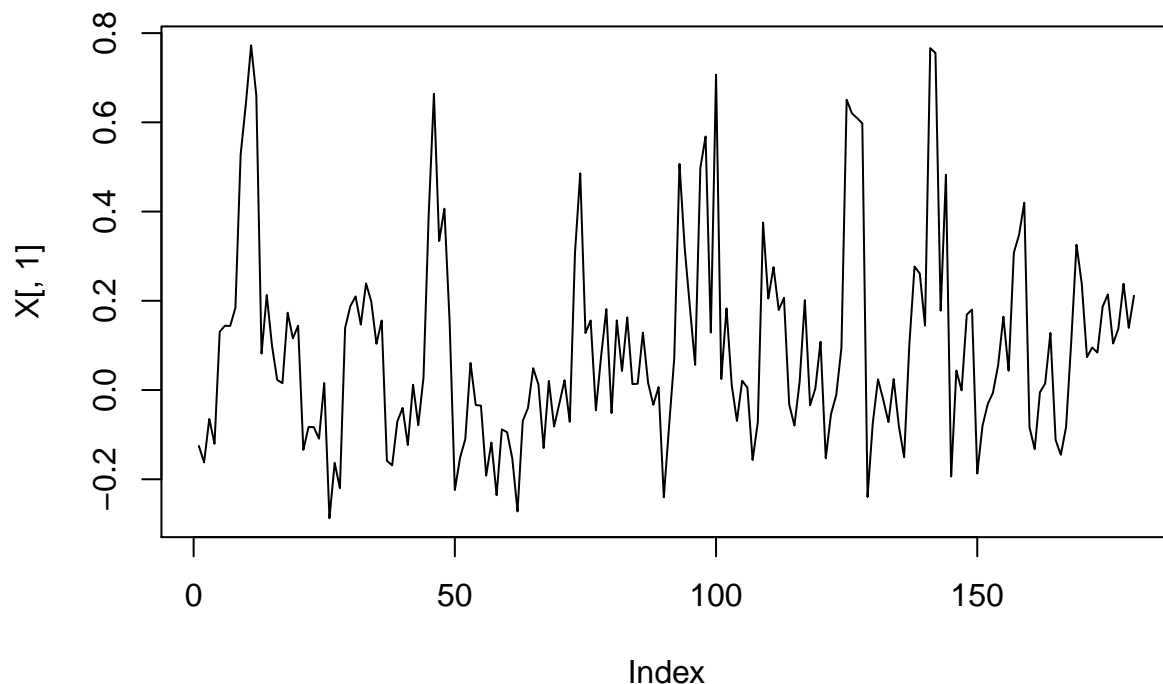
Prediction of cow milk percentage

In the data file "maldidata.rdata" you find four objects:

- Y : The percentage of cow-milk for 4 replicates of 45 different milk mixtures
- X : Mass Spectrometry data (MALDI-TOF) for the milk samples. The 6179 variables is a quantification of molecule "size" and "charge" in a sample. For simplicity we may say that the size of molecules increases from variable 1 to variable 6179. The measurements are then amounts of molecules of different sizes. The method is used to separate proteins, peptides and other ionizable compounds.
- Ytest : Cow-milk percentages for 45 extra test samples
- Xtest : The MALDI-TOF values for the test samples.

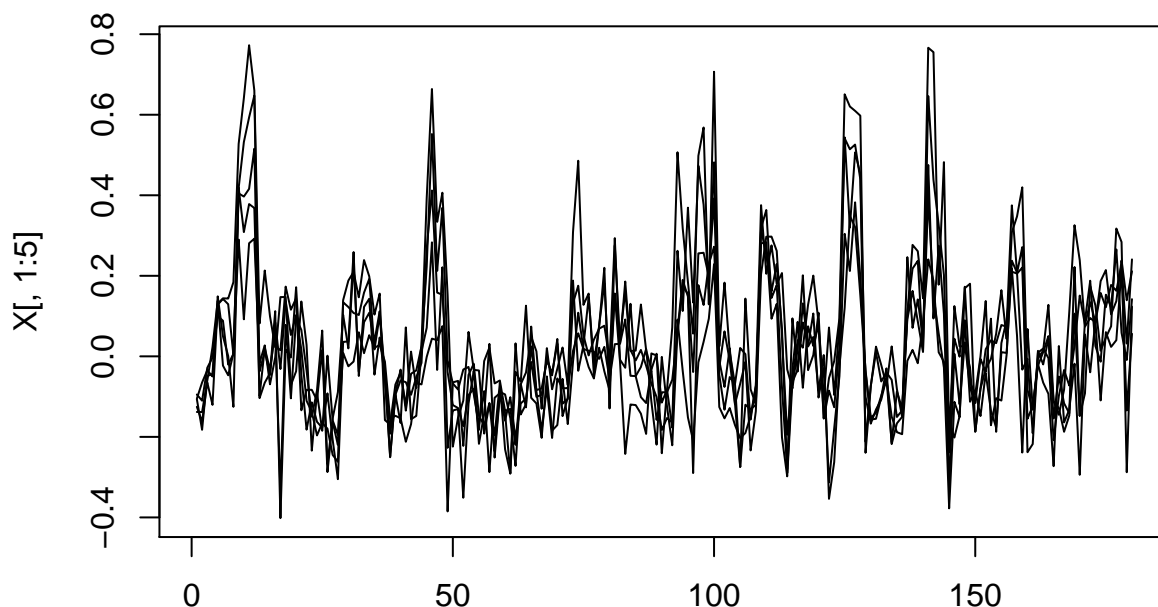
You may plot the spectra in X one-by-one as,:

```
plot(X[, 1], type = "l")
```



We can also plot multiple (all) spectra together, takes more time.

```
matplot(X[, 1:5], type = "l", lty = 1, col = "black")
```



The peaks show molecule sizes that are abundant in the sample.

- First run a PCA on X. How many components are needed to explain 80% and 90% of the variability in X?

```
pr <- prcomp(X)
summary(pr)$importance[, 1:14]
```

	PC1	PC2	PC3	PC4	PC5	PC6	PC7
Standard deviation	12.502	4.4833	3.0218	2.5373	2.0112	1.7430	1.30637
Proportion of Variance	0.648	0.0833	0.0379	0.0267	0.0168	0.0126	0.00708
Cumulative Proportion	0.648	0.7313	0.7691	0.7958	0.8126	0.8252	0.83228

	PC8	PC9	PC10	PC11	PC12	PC13
Standard deviation	1.0875	1.05133	0.96666	0.91363	0.84363	0.83145
Proportion of Variance	0.0049	0.00458	0.00387	0.00346	0.00295	0.00287
Cumulative Proportion	0.8372	0.84176	0.84564	0.84910	0.85205	0.85491

	PC14
Standard deviation	0.81638
Proportion of Variance	0.00276
Cumulative Proportion	0.85768

We need 5 components to explain 80% and 36 to explain 90% (check yourself). Hopefully

we do not need to use all X-information to predict Y.

- Compute the correlations between Y and the principal components. How can you use this get an idea of how many components PLS-regression will require to make a good model for cow-milk prediction?

```
cor(Y, pr$x)
```

```

      PC1    PC2    PC3    PC4    PC5    PC6    PC7    PC8    PC9
[1,] 0.528 -0.754 0.0338 -0.225 0.0744 -0.0503 0.015 -0.166 -0.00757
      PC10   PC11   PC12   PC13   PC14   PC15   PC16   PC17   PC18
[1,] -0.0754 -0.137 -0.0432 0.0183 0.0167 -0.0155 -0.0263 0.0222 -0.0187
      PC19   PC20   PC21   PC22   PC23   PC24   PC25   PC26
[1,] -0.00495 -0.0413 0.00649 -0.0187 -0.0442 -0.0069 -0.0411 0.00412
      PC27   PC28   PC29   PC30   PC31   PC32   PC33   PC34   PC35
[1,] -0.0123 0.017 -0.0147 0.0387 0.000618 0.0185 -0.0213 0.0478 -0.00964
      PC36   PC37   PC38   PC39   PC40   PC41   PC42   PC43   PC44
[1,] -0.0106 -0.00503 0.021 -0.011 -0.00874 -0.0191 -0.014 0.0082 -0.0124
      PC45   PC46   PC47   PC48   PC49   PC50   PC51   PC52
[1,] 0.0176 0.00597 0.0141 -0.0193 0.0217 -0.00107 -0.00454 0.00781
      PC53   PC54   PC55   PC56   PC57   PC58   PC59   PC60
[1,] 0.00674 -0.0041 0.0027 -0.00404 -0.0269 0.00521 -0.00471 -0.000586
      PC61   PC62   PC63   PC64   PC65   PC66   PC67   PC68
[1,] 0.0124 -0.0142 0.0125 -0.0181 -0.00766 0.00276 0.0146 -0.0221
      PC69   PC70   PC71   PC72   PC73   PC74   PC75   PC76
[1,] -0.00245 -0.00385 -0.00334 -0.00403 -0.038 -0.0394 0.00702 0.0103
      PC77   PC78   PC79   PC80   PC81   PC82   PC83   PC84
[1,] -0.017 -0.0104 -0.0116 -0.00895 0.00664 -0.00716 0.0086 -0.0205
      PC85   PC86   PC87   PC88   PC89   PC90   PC91   PC92   PC93
[1,] -0.0217 0.018 0.0284 0.00673 0.0166 0.00445 -0.00696 -0.0109 -0.00113
      PC94   PC95   PC96   PC97   PC98   PC99   PC100   PC101
[1,] 0.0244 0.0304 -0.0186 -0.00388 0.0214 -0.00707 -0.00238 0.000283
      PC102   PC103   PC104   PC105   PC106   PC107   PC108   PC109   PC110
[1,] 0.0251 -0.00757 -0.0175 -0.0259 0.0244 -0.026 0.00272 -0.0157 0.0142
      PC111   PC112   PC113   PC114   PC115   PC116   PC117   PC118   PC119
[1,] -0.0115 -0.009 -0.0115 0.0299 0.00924 0.00246 -0.01 -0.00907 -0.0242
      PC120   PC121   PC122   PC123   PC124   PC125   PC126   PC127   PC128
```



```

[1,] 0.00516 -0.0066 0.0147 0.00467 -0.0254 -0.0224 -0.0124 0.00816 0.0101
      PC129  PC130   PC131  PC132   PC133  PC134   PC135  PC136
[1,] -0.0142 -0.0119 -0.00018 -0.0108 -0.00172 0.00275 -0.00635 0.0103
      PC137  PC138   PC139   PC140   PC141  PC142   PC143  PC144
[1,] -0.0087 0.00552 0.00742 -0.00805 -0.00327 0.0159 -0.00109 0.00291
      PC145   PC146   PC147   PC148   PC149   PC150  PC151  PC152
[1,] -0.00378 -0.00148 -0.00736 -0.00656 0.00354 -0.00579 0.0034 0.00252
      PC153   PC154   PC155  PC156  PC157   PC158   PC159  PC160
[1,] 0.0015 -0.00918 -0.0111 0.0232 0.0108 -0.00734 -0.00148 0.00503
      PC161   PC162   PC163  PC164   PC165  PC166   PC167  PC168
[1,] -0.00242 -0.00583 -0.00701 0.00558 -0.00344 0.00436 0.00368 0.00529
      PC169  PC170   PC171   PC172   PC173  PC174   PC175   PC176
[1,] -0.015 -0.0092 -0.0134 -0.00933 -0.00255 0.0113 0.00024 -0.000418
      PC177  PC178  PC179  PC180
[1,] -0.0057 -0.0103 0.0218 0.0288

```

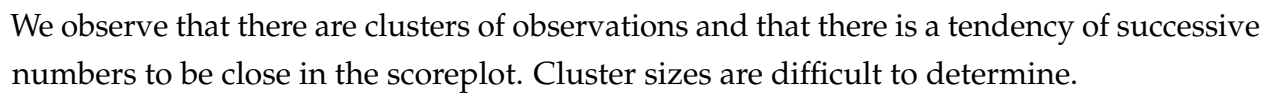
Components 1, 2, 4 and perhaps 8 are moderately to highly correlated with Y. Supposedly 3 to 4 components will be necessary for PLSR.

- Fit a PLS-regression model using Y as response and X as predictor (You may simply write $Y \sim X$ as your model call in `plsr`. Also use `ncomp=10` as extra argument to only fit 1 to 10 components). Use the `scoreplot()` function to make a scoreplot. Check the help-file for this function to see how you can chose the component numbers to plot, and how you can put labels to your observations. Plot component 1 against 2 and put observation numbers 1:180 as labels. If the noise level of the measurements is low, the replicates should group in clusters of size four (obs 1,2,3 and 4 are from the same mixture, and so on). Do you see any tendency to this?

```

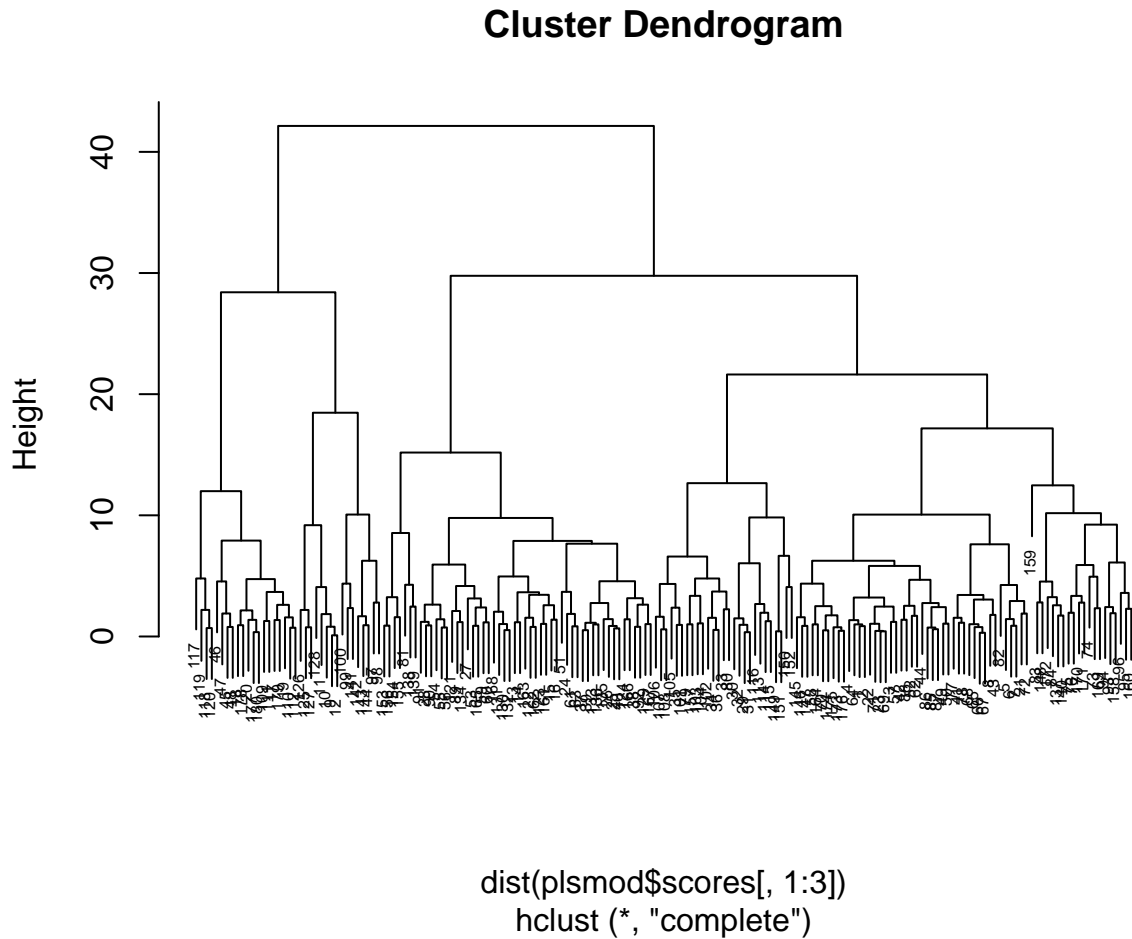
plsmod <- plsr(Y ~ X, ncomp = 10)
scoreplot(plsmod, comps = c(1,2), labels = 'names', pch = 0.7)

```



- Perform hierarchical clustering of the samples using the 3 first PLS-component scores as input variables. Try both “complete” and “average” agglomeration method and make a dendrogram. Are the replicate clusters more apparent in the dendrogram than in the scoreplot? Is there any samples that are very different from all others?

```
clust1 <- hclust(dist(plsmod$scores[,1:3]), method = "complete")
plot(clust1, cex = 0.5)
```



From the dendrogram we observe several clusters of size four, and some of three with successive observation numbers. This implies that the replicates are more similar to each other than samples from different milk samples. Sample 159 seems to be an outlier, very different from all others.

- Use K-means clustering with K=45 on the three first PLS-component scores. How do the replicates cluster?

```

clust2 <- kmeans(plsmod$scores[,1:3], centers = 45)
clust2$cluster

```

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
39	39	39	39	13	13	13	44	20	20	20	20	31	31	31	31	38	15
19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
38	15	13	1	1	40	25	25	14	17	2	2	2	2	22	32	22	32
37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54
25	17	25	25	44	23	9	34	3	3	3	3	11	11	19	23	17	24

55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72
24	24	30	30	14	14	40	23	40	39	9	9	9	9	1	30	1	13
73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90
12	45	45	45	44	44	33	22	5	13	12	35	34	34	34	23	14	17
91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108
17	17	6	6	28	28	42	42	42	42	22	32	43	43	7	7	7	7
109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126
38	38	38	38	4	4	4	4	37	37	37	37	41	8	8	41	29	29
127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144
29	26	25	16	16	16	25	41	25	25	35	5	5	35	42	42	42	42
145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162
21	21	21	21	36	36	36	36	18	18	18	18	33	6	27	28	8	8
163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180
8	30	14	41	41	16	12	45	45	35	10	30	10	10	15	15	3	15

Many replicates fall into the same clusters, but some clusters observations from more than one “true” cluster. E.g. cluster number 43 contains observations (97, 98, 99, 100) and (141, 142, 143, 144), which is a mixture of four true replica clusters. We conclude that the replicates are somewhat similar, but there is some noise in the MALDI-TOF data which makes similar milk mixtures hard to distinguish.

- We will use cross-validation to estimate the expected prediction error of PLSR and to choose the appropriate number of components. Instead of using Leave-One-Out Cross-validation we will exclude all four replicates in a “Leave-Four-Out” type of Cross-validation. Why is this smart?

Since the replicates are from the same mixture, they are not independent. If we use Leave-One-Out CV the three replicates still contained in the training set will make the model “too good” to predict the given mixture, and we will under-estimate the prediction error of new samples.

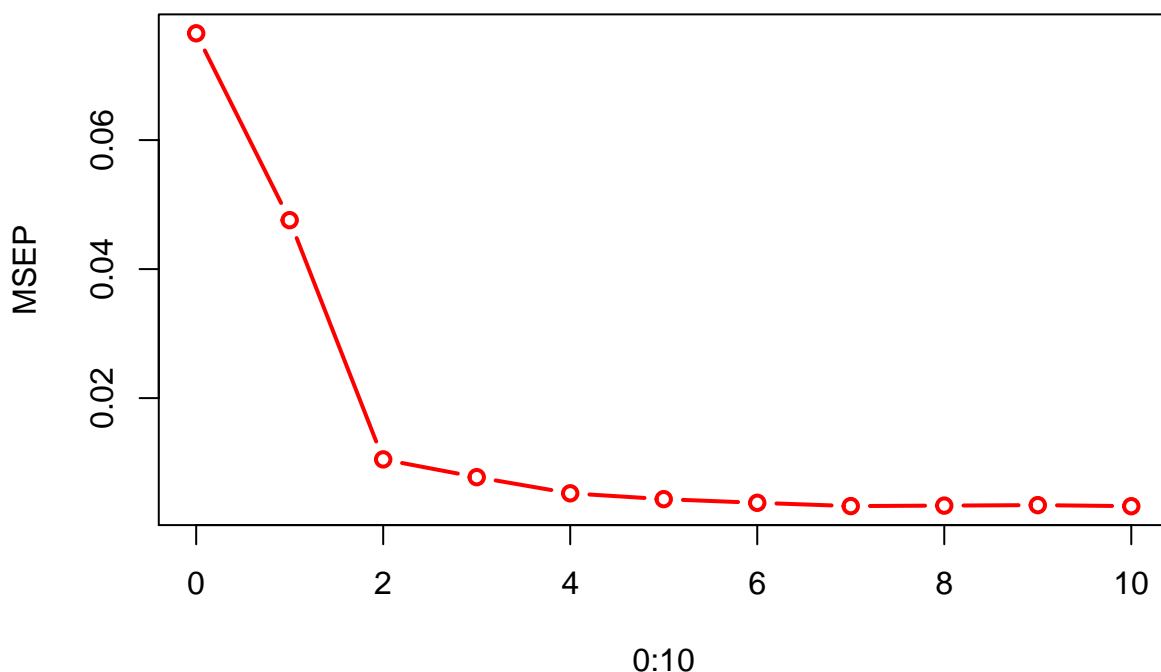
- Refit the `plsmodel` from exercise c. but add the arguments `validation="CV"`, `segments=45` and `segment.type="consecutive"` in the `plsmodel` call. This sets up the “Leave-Four-Out” CV to be performed.

```
plsmod <- pls(Y ~ X, ncomp = 10, validation = "CV",
              segments = 45, segment.type = "consecutive")
```

- The sum of squared prediction errors (PRESS) for different number of components

is given in the `validation$PRESS` element of the fitted pls-model. The null-model PRESS (prediction using the mean-response) is given in the `validation$PRESS0` element. You find the MSEP-values (Mean Squared Error of Prediction) by dividing the PRESS by the number of observations ($N=180$). Make a prediction error plot of MSEP with 0 to 10 components. How many components do you think gives satisfying prediction of cow-milk content in new mixtures? Remember that simple models are often better than complex.

```
MSEP <- c(plsmode$validation$PRESS0, plsmode$validation$PRESS)/180
plot(0:10, MSEP, type = "b", col = 2, lwd = 2)
```



The prediction error is heavily reduced as we introduce components 1 and 2, but there is a small gain by adding components 3, 4 and 5. Simple models are usually more robust, so I would not go any further than 5 components here.

- Predict the cow-milk content of the test samples using `Xtest` as newdata in the `predict`-function and use the number of components you found as best from the previous exercise. Save the predictions into an object called `pred`. (See `?predict.mvr` for the help-file to predict for pls-objects.). The `predict`-function returns an array of dimension `[45,1,1]` of predictions. You can extract all predictions by `pred[,1,1]`.

```
pred <- predict(plsmode, newdata = Xtest, ncomp = 5)
```

Copy the following code into R and execute:

```
MSEPfunc <- function(y, yhat){
  mean((y - yhat) ^ 2)
}
```

- Use `MSEPfunc()` to compute the MSEP-value for the test-predictions using `Ytest` and the predicted values as inputs. Did the cross-validation under-estimate the prediction error?

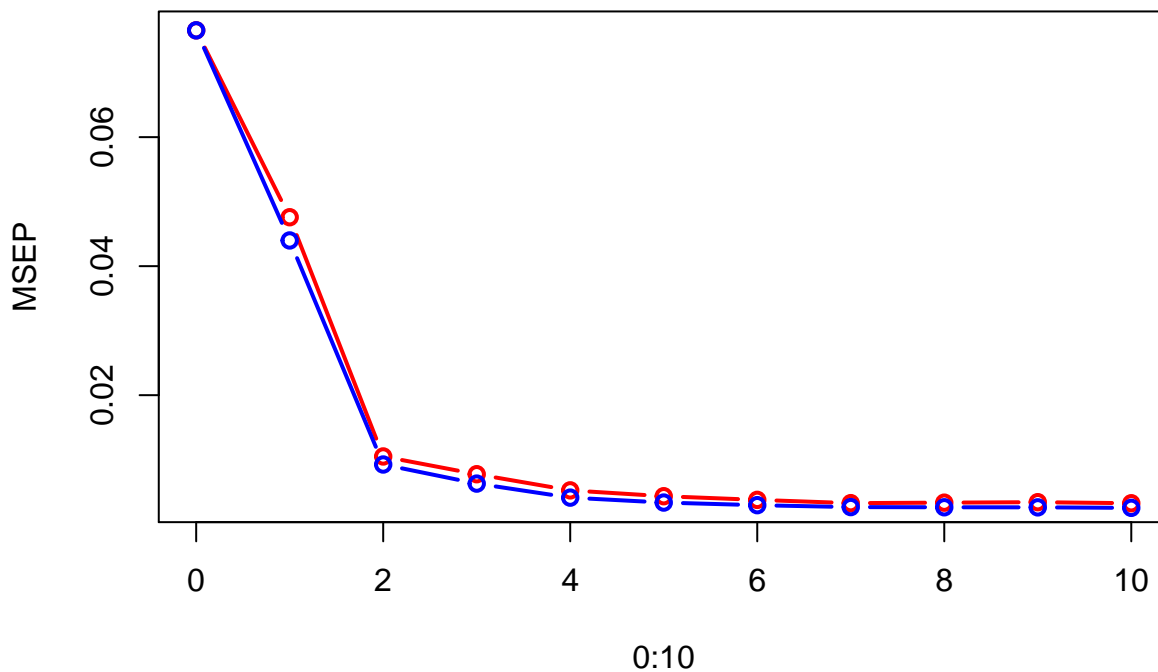
```
MSEPfunc(Ytest, pred[,1,1])
```

```
[1] 0.00692
```

The MSEP-value is slightly larger than the value found for five components using Cross-validation, but the order of magnitude is the same.

- EXTRA for those interested. Redo the exercises g. and h. with Leave-One-Out CV. (See also lecture notes Lesson 7.) Compare the MSEP-values you find with the previous CV-routine.

```
plsm2 <- pls(Y ~ X, ncomp = 10, validation = "LOO")
MSEP2 <- c(plsm2$validation$PRESS0, plsm2$validation$PRESS)/180
plot(0:10, MSEP, type = "b", col = 2, lwd = 2)
points(0:10, MSEP2, type = "b", col = 4, lwd = 2)
```



We see that LOO-CV underestimates the prediction error, as commented in exercise f.