Applied Methods in Statistics

Thore Egeland and Kathrine Frey Frøslie

Year: 2019

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 secion, 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 arraywe 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]` with give three, four, five, the elements at postion 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
101   86.4    185  25          83.4
381   43.2    160  20          59.5
402   72.3    165  28          74.2
107   83.2    180  32          88.6
93    70.5    165  26          81.5
```

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
```

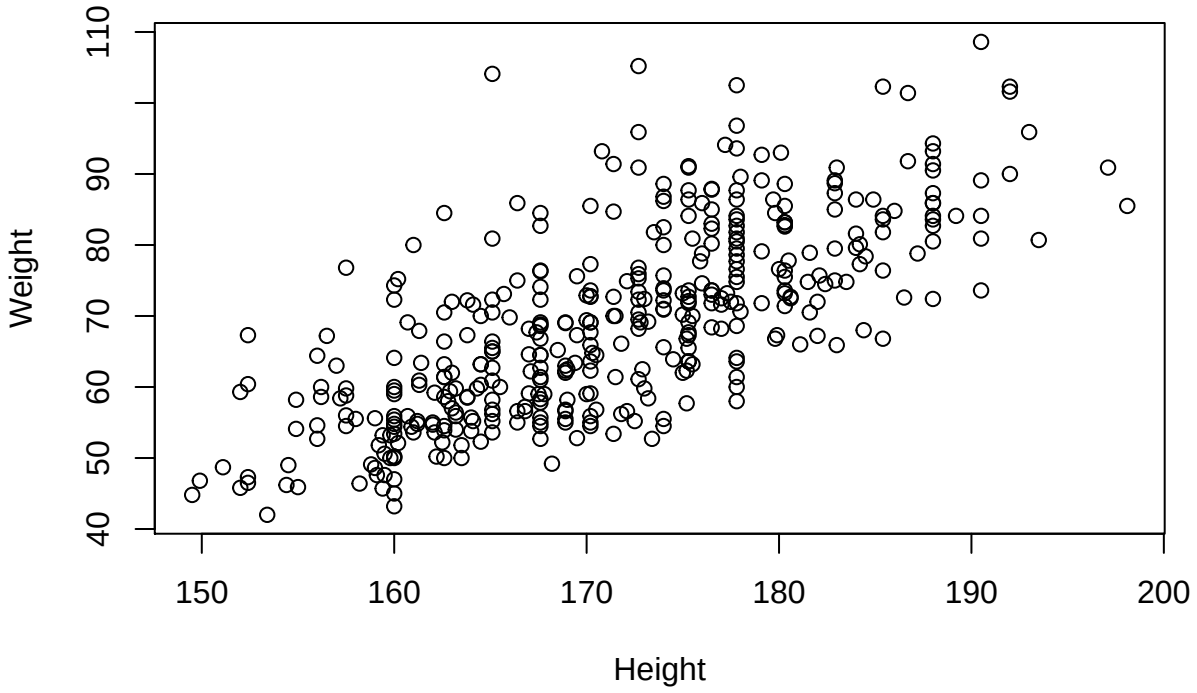
```
    Weight Height Age Circumference
136   76.4    185  62          94.8
189   73.6    175  60          90.5
207   66.8    168  62          81.5
231   80.0    174  65          98.6
```

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

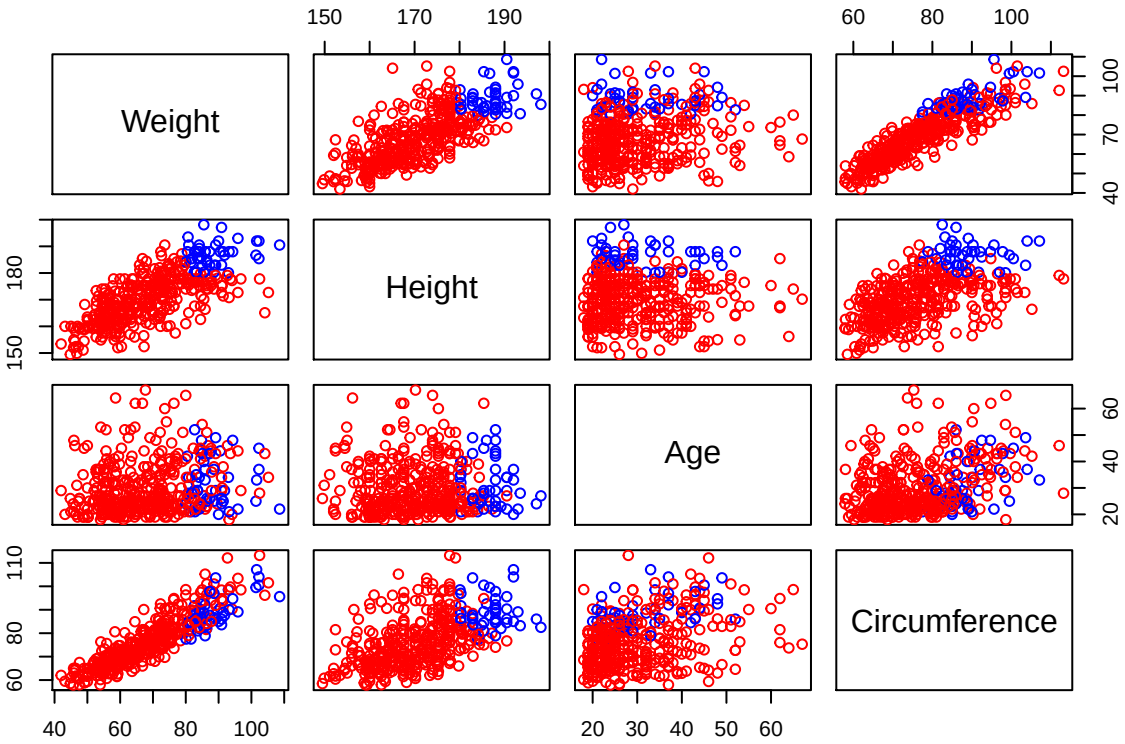
```
mycolors <- ifelse((isHeavy & isTall), "blue", "red")
```

Here, `isHeady & isTall` returns a logical vector. The `ifelse` function returns `blue` if `TRUE`and `red` if `FALSE` for each elements 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", ...)
}
```

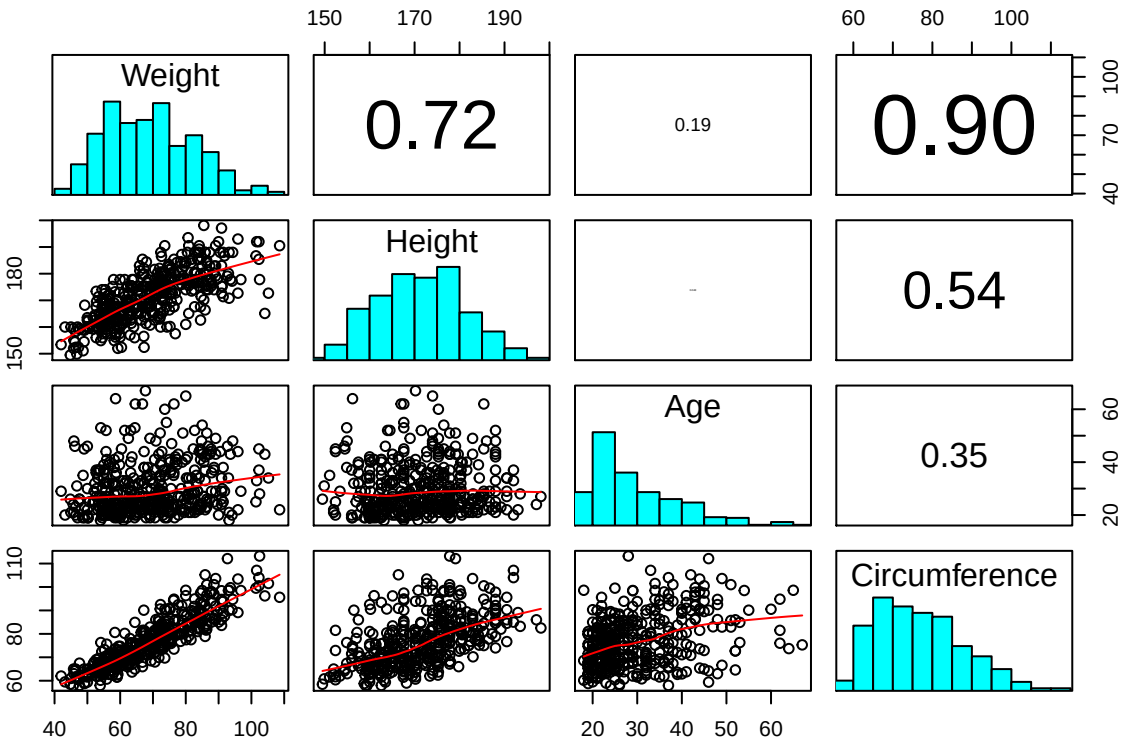
Now, create a function that will display correlation on `pairs` plot.

```
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 Sqaures App

Play around with the least squares app on 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 fronter 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?

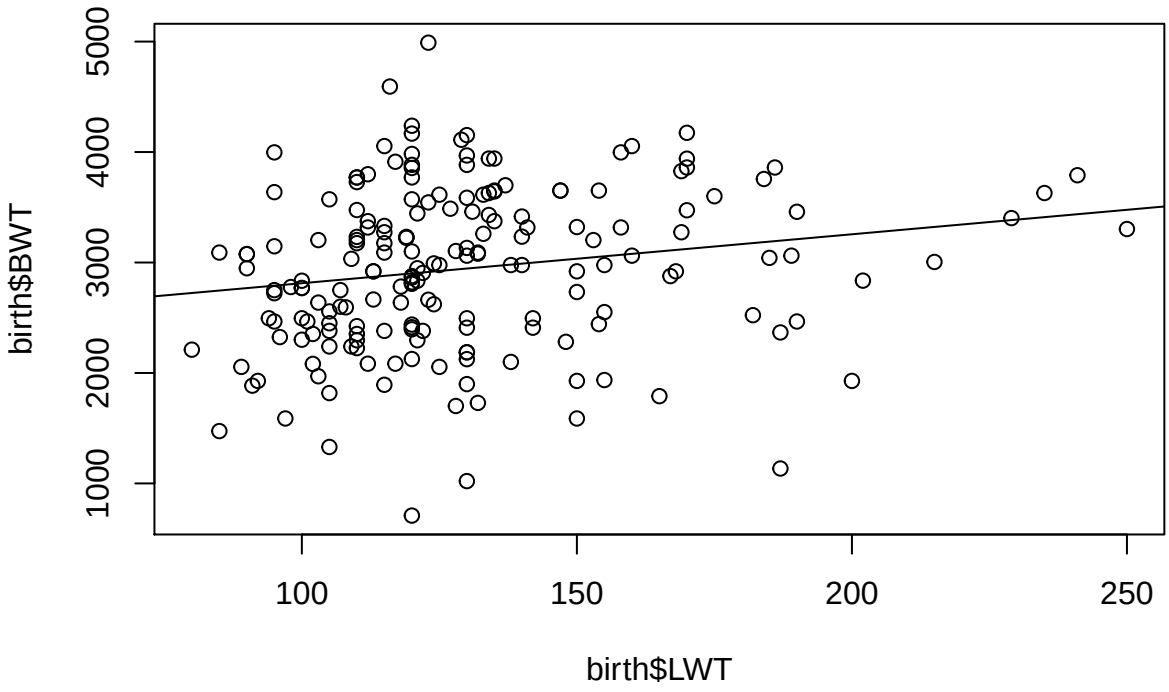
The r-squared (R^2) for the model is 0.03, this shows that only 3% of variation present in birth weight (BWT) is explained by weight of mother (LWD). Here, the model fits the data poorely.

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. Similary, 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 cofficient 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")
```

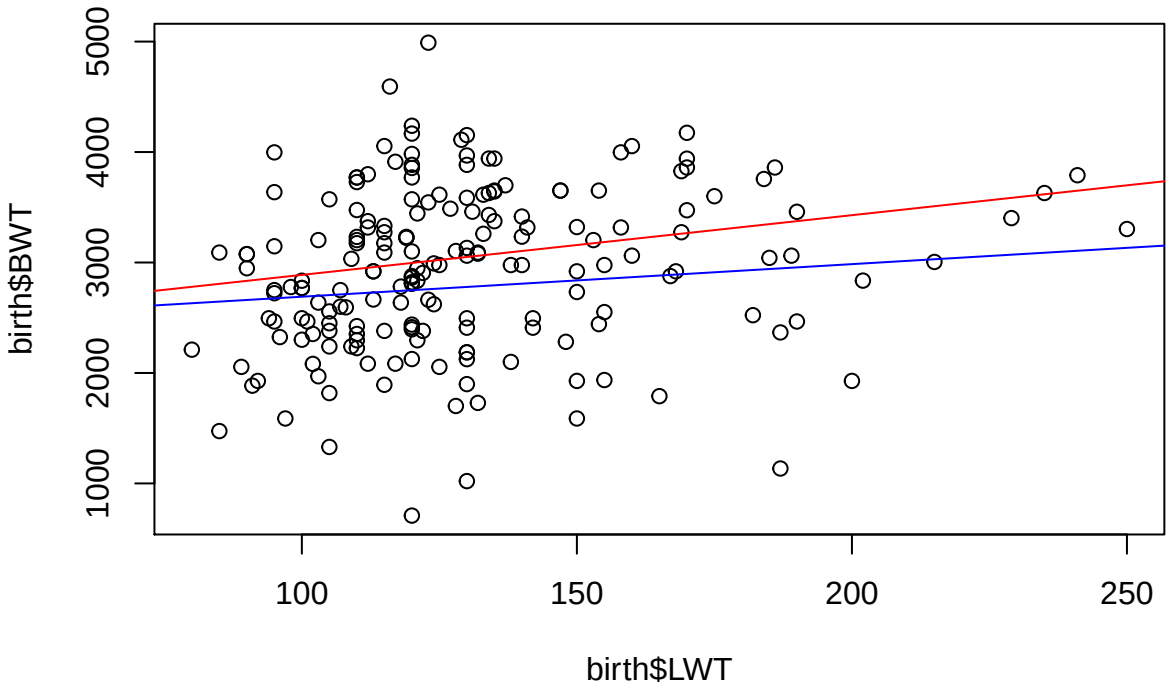
Fit the same model as `birth1`, but separate models for non-smokers and smokers, and call the models `birth2` and `birth3`. (Hint: select observations by the subset argument in the `lm`-function using the `smokeYes` variable.)

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

Interpreate these models

- Make a scatter plot of LWT vs BWT and add two fitted lines form 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 se 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 0.0000000000044 ***
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 $R^2 = 0.038$.

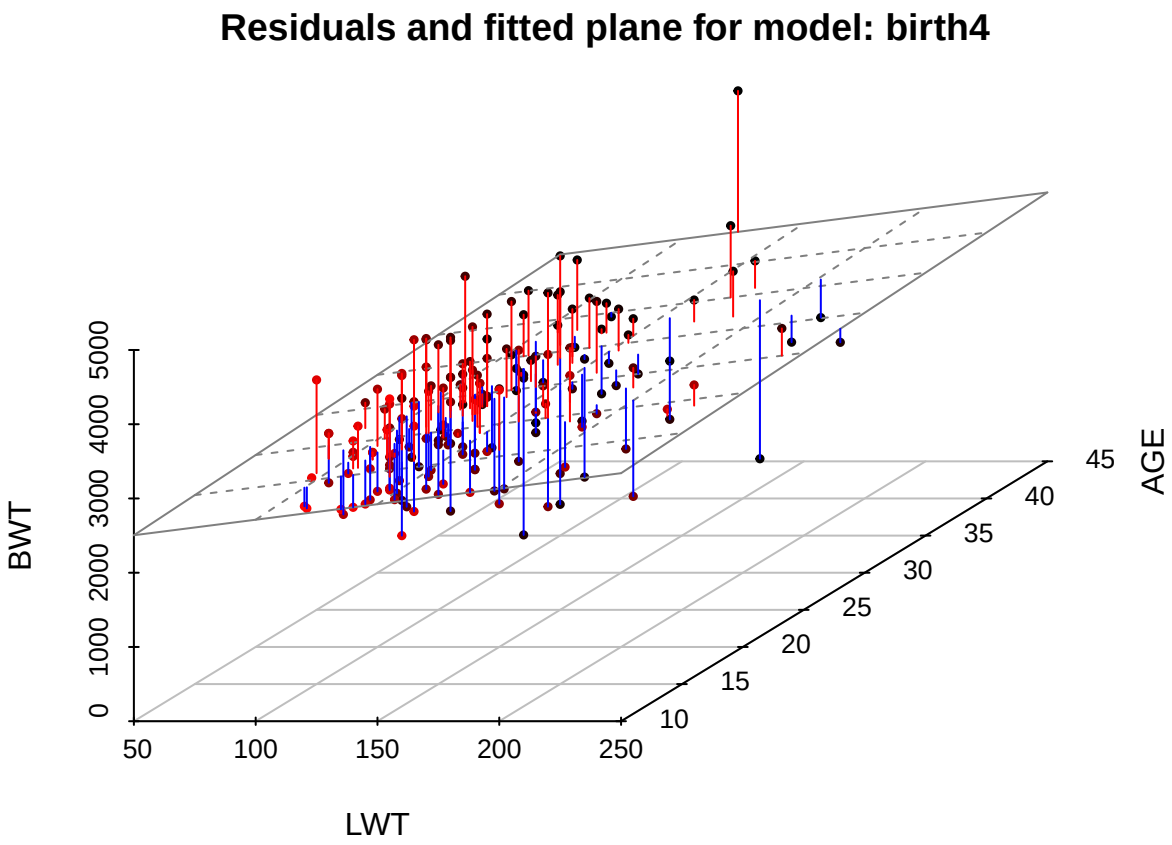
Optional:

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])
})
```



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. Similary, 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.

```r
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)_)

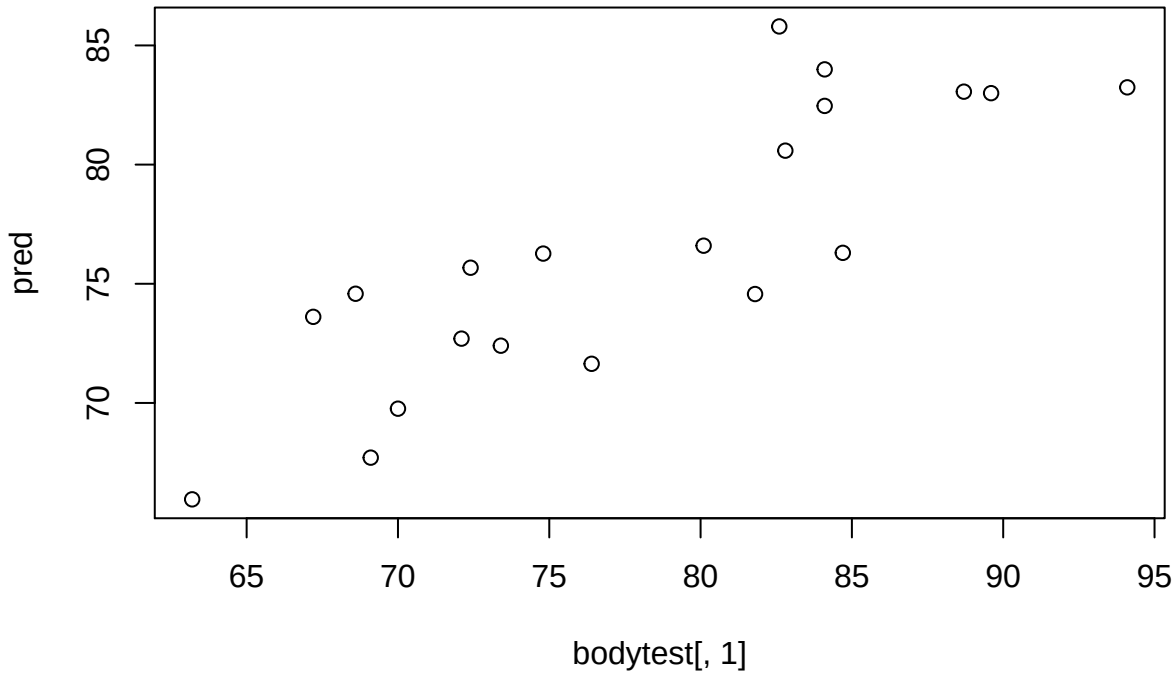
```
bodytest <- bodydata[21:40, ]
```

- 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 adjuste 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.