# 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 (PCA, PCR) | 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.   : 42.00   Min.   :149.5   Min.   :18.00   Min.   : 57.90
 1st Qu.: 58.45   1st Qu.:163.9   1st Qu.:23.00   1st Qu.: 67.95
 Median : 68.60   Median :171.4   Median :27.00   Median : 75.60
 Mean   : 69.19   Mean   :171.3   Mean   :29.91   Mean   : 76.91
 3rd Qu.: 78.80   3rd Qu.:177.8   3rd Qu.:35.00   3rd Qu.: 84.30
 Max.   :108.60   Max.   :198.1   Max.   :67.00   Max.   :113.20
```

## 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
483   63.6  175.3  25          67.0
187   85.0  176.5  54          98.5
183   84.1  185.4  22          89.2
115   84.1  175.3  44          90.0
49    74.9  172.1  25          79.7
```

## 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
```

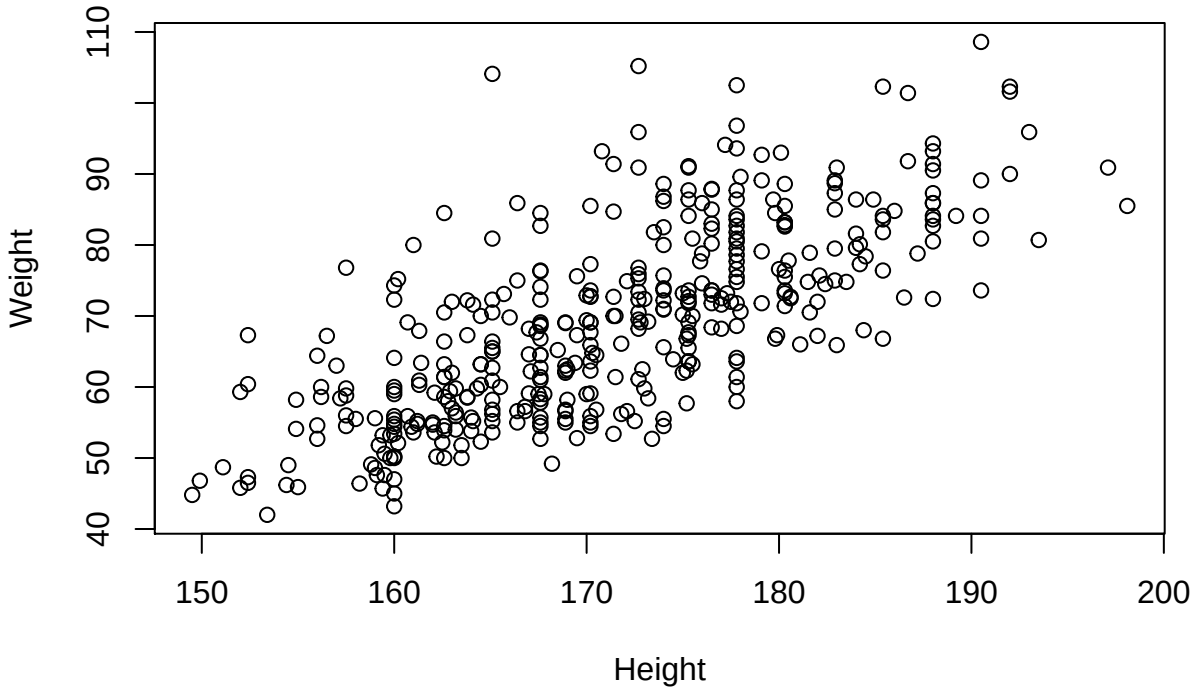
```
    Weight Height Age Circumference
136   76.4  185.4  62          94.8
189   73.6  175.3  60          90.5
207   66.8  167.6  62          81.5
231   80.0  174.0  65          98.6
```

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

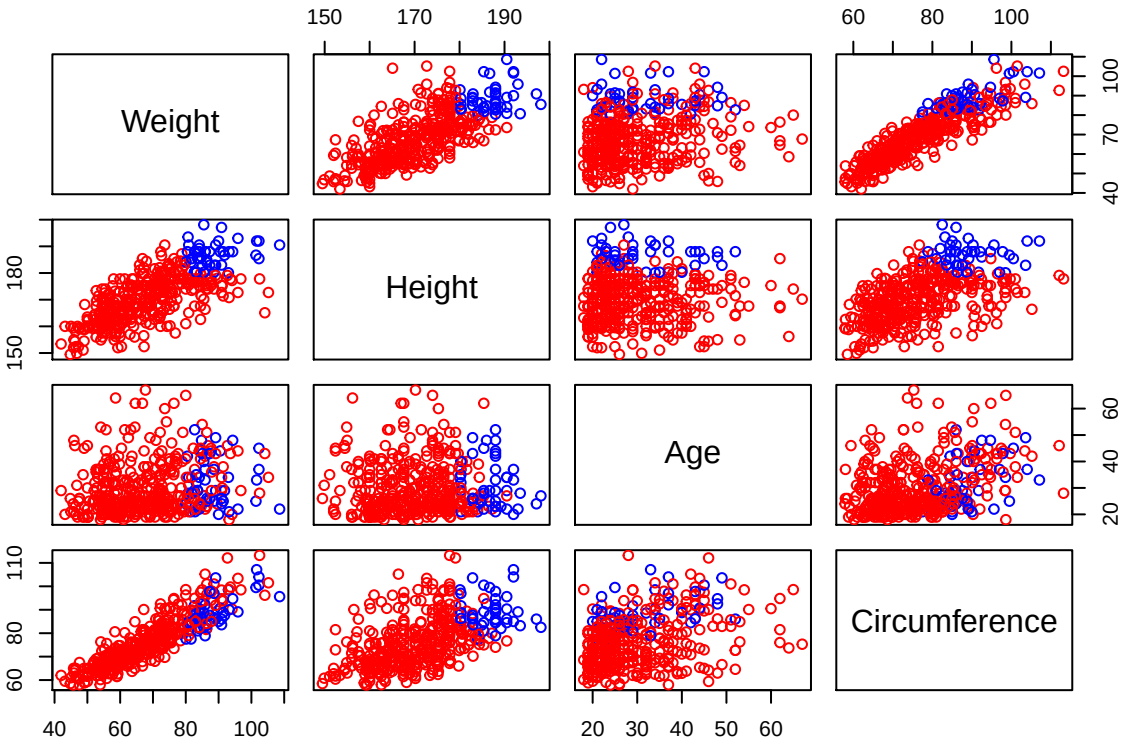
```
mycolors <- ifelse((isHeavy & isTall), "blue", "red")
```

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        1.0000000 0.72080269 0.18701340     0.8994638
Height        0.7208027 1.00000000 0.04822479     0.5447980
Age           0.1870134 0.04822479 1.00000000     0.3547390
Circumference 0.8994638 0.54479800 0.35473898     1.0000000
```

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", ...)
}
```

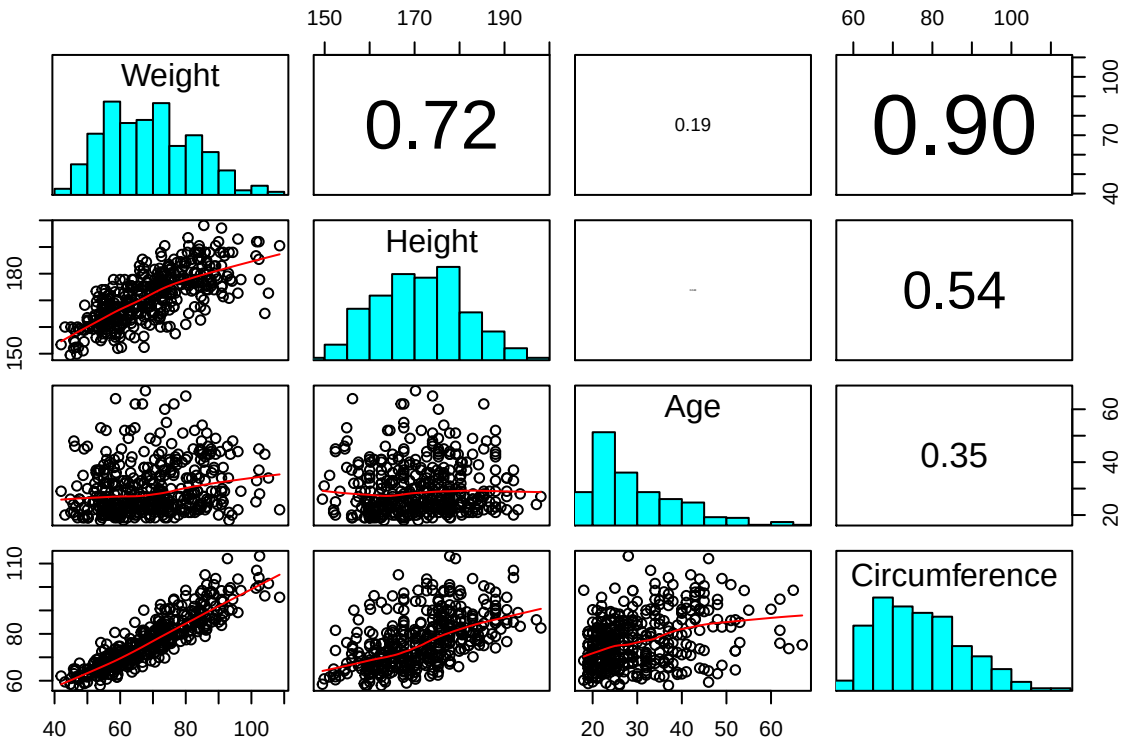
Now, create a function that will display correlation on `pairs` plot.

```
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)
```

```
  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
```

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

```
summary(birth)
```

```
  LOW           AGE             LWT          SMK           BWT
 NO :130   Min.   :14.00   Min.   : 80.0   NO :115   Min.   : 709
 YES: 59   1st Qu.:19.00   1st Qu.:110.0   YES: 74   1st Qu.:2414
           Median :23.00   Median :121.0             Median :2977
```

```
      Mean   :23.24   Mean   :129.8            Mean   :2945
      3rd Qu.:26.00   3rd Qu.:140.0            3rd Qu.:3475
      Max.   :45.00   Max.   :250.0            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
3054.957 2773.243
```

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
752.4090 660.0752
```

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
-2192.18  -503.63    -3.91   508.25  2075.53

Coefficients:
            Estimate Std. Error t value            Pr(>|t|)
(Intercept) 2369.672    228.431  10.374 <0.0000000000000002 ***
LWT            4.429      1.713   2.586              0.0105 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

s: 718.2 on 187 degrees of freedom
Multiple R-squared: 0.03452,
Adjusted R-squared: 0.02935
F-statistic: 6.686 on 1 and 187 DF,  p-value: 0.01048
```

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 $\beta_1$ 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?

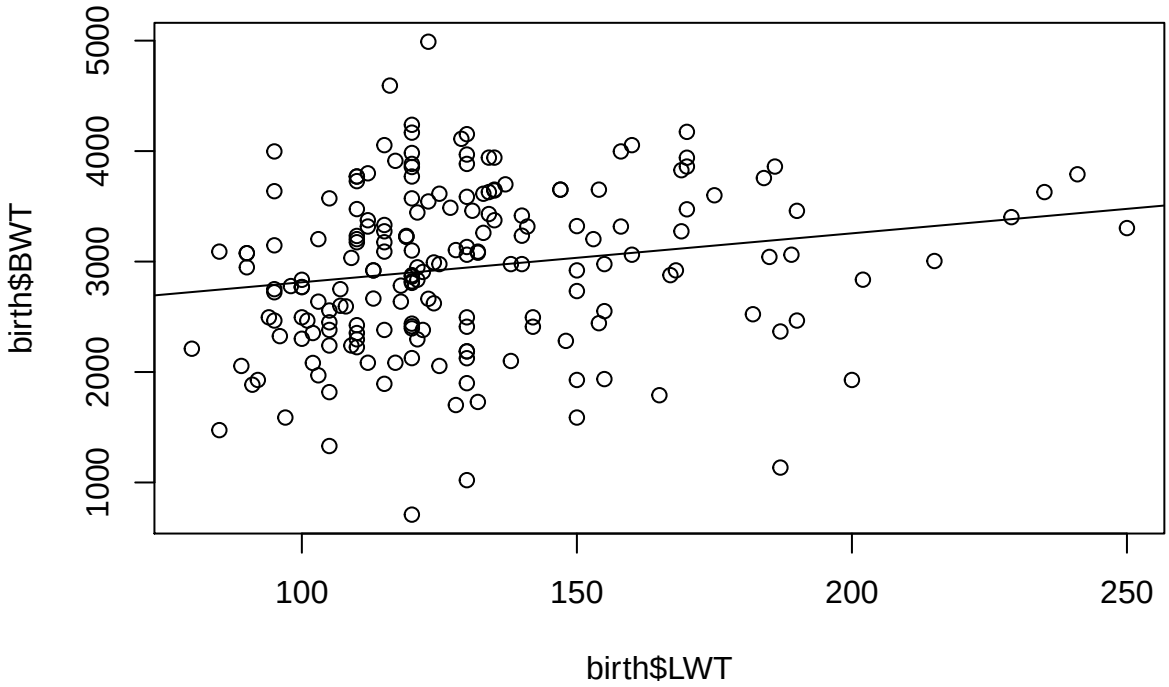
The r-squared ($R^2$) for the model is 0.03, this shows that only 3% of variation present in birth weight (BWT) is explained by weight of mother (LWD). Here, the model fits the data poorely.

**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) 1919.039836 2820.3043
LWT            1.049927    7.8086
```

- Also find 99% confidence intervals

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

```
                   0.5 %      99.5 %
(Intercept) 1775.20974036 2964.134385
LWT           -0.02866992    8.887197
```

- Comment on the intervals

1. It is 95% certain that the interval (1.05, 7.809) covers the true $\beta_1$. Similary, it is 99% certain that the interval (-0.029, 8.887) covers the true $\beta_1$.
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 $\beta_1$ 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")
```

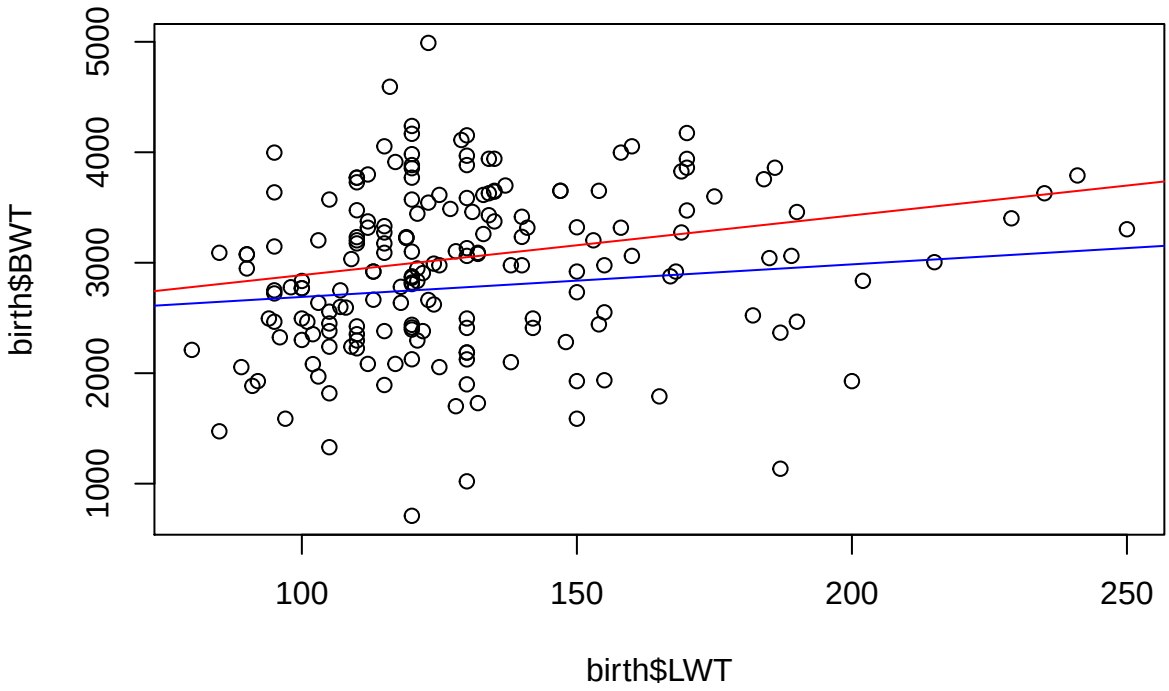
Fit the same model as `birth1`, but separate models for non-smokers and smokers, and call the models `birth2` and `birth3`. (Hint: select observations by the subset argument in the `lm`-function using the `smokeYes` variable.)

```
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
-2040.25  -416.25    33.92   472.18  1488.75

Coefficients:
            Estimate Std. Error t value        Pr(>|t|)
(Intercept) 2395.373    301.474   7.946 0.0000000000194 ***
LWT            2.949      2.276   1.296           0.199
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

s: 657 on 72 degrees of freedom
Multiple R-squared: 0.02279,
Adjusted R-squared: 0.009213
F-statistic: 1.679 on 1 and 72 DF,  p-value: 0.1992
```

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

- What is the interpretation of the regression coefficients?

1. $\beta_1$ 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 $\beta_1$ grams for people of the same AGE.
2. $\beta_2$ 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
-2232.84  -500.50    32.13   520.32  1899.26

Coefficients:
            Estimate Std. Error t value         Pr(>|t|)
(Intercept) 2215.760    299.238   7.405 0.00000000000443 ***
LWT            4.179      1.743   2.397           0.0175 *
AGE            8.021     10.060   0.797           0.4263
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

s: 718.9 on 186 degrees of freedom
Multiple R-squared: 0.03781,
Adjusted R-squared: 0.02746
F-statistic: 3.654 on 2 and 186 DF,  p-value: 0.02776
```

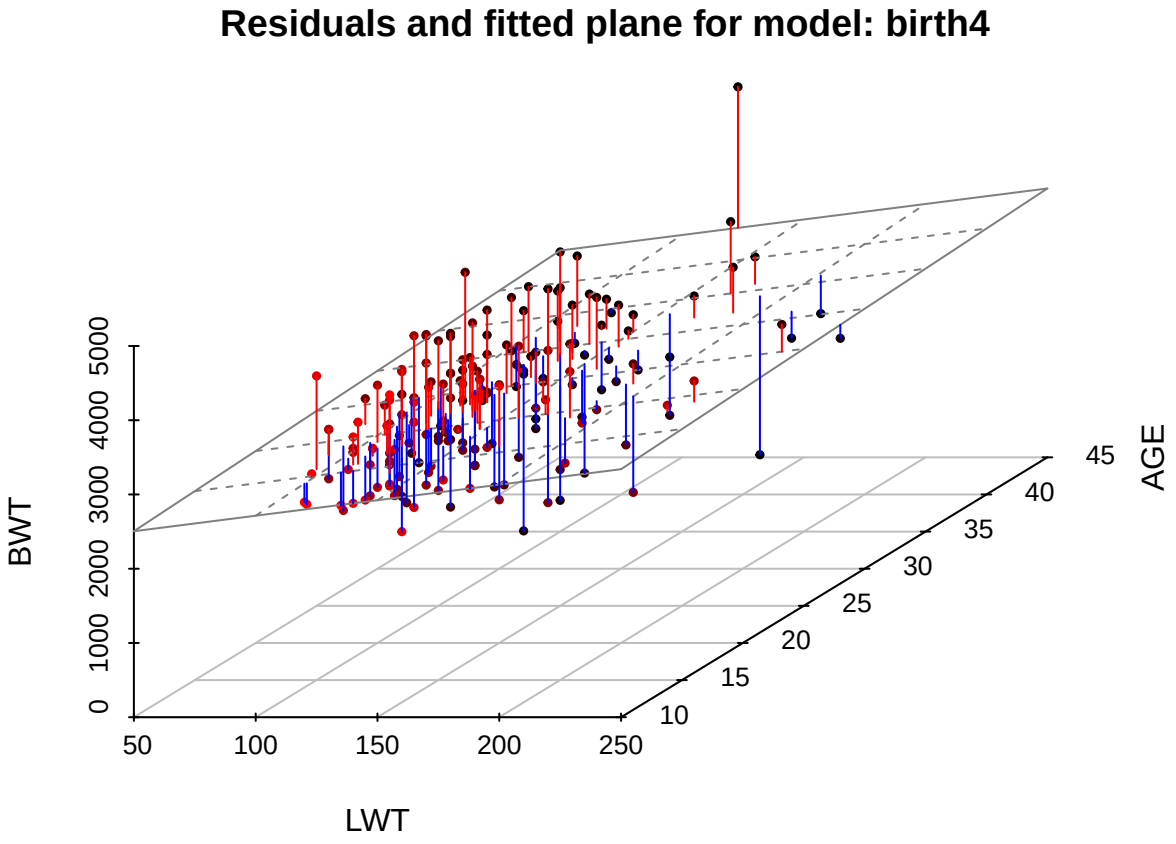
The summary output shows that LWT is significant at 5% level of significance but not at 1%. AGE has very high p-value and thus is not significant, i.e. there is not any linear relationship of AGE with BWT. The explained variation is still very low with an $R^2 = 0.038$.

**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 $\beta$ 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
-11.5160  -2.5964   0.0261   2.9141  11.7454

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -38.2310    38.3605  -0.997  0.33216
Height        0.6386     0.2123   3.007  0.00757 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

s: 5.839 on 18 degrees of freedom
Multiple R-squared: 0.3344,
Adjusted R-squared: 0.2974
F-statistic: 9.043 on 1 and 18 DF,  p-value: 0.007566
```

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
-15.0839  -3.9176   0.8978   4.2513  12.4386

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  80.7448    14.5625   5.545 0.000029 ***
Age          -0.1592     0.6253  -0.255    0.802
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

s: 7.144 on 18 degrees of freedom
Multiple R-squared: 0.003587,
```

```
Adjusted R-squared: -0.05177
F-statistic: 0.06481 on 1 and 18 DF,  p-value: 0.8019
```

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)     3.6737    17.1036   0.215 0.832343
Circumference   0.9137     0.2125   4.300 0.000431 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

s: 5.027 on 18 degrees of freedom
Multiple R-squared: 0.5067,
Adjusted R-squared: 0.4793
F-statistic: 18.49 on 1 and 18 DF,  p-value: 0.0004311
```

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        1.000000000 0.334400375 0.003587475482 0.506712708016
Height        0.334400375 1.000000000 0.001672757208 0.061161161867
Age           0.003587475 0.001672757 1.000000000000 0.000004216559
Circumference 0.506712708 0.061161162 0.000004216559 1.000000000000
```

- 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] 0.3344004
```

- 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, $\sigma^2$.

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     1 308.31 308.313  9.0433 0.007566 **
Residuals 18 613.67  34.093
---
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   3.31   3.308  0.0648 0.8019
Residuals 18 918.68  51.038
```

```
anova(model3)
```

```
Analysis of Variance Table

Response: Weight
              Df Sum Sq Mean Sq F value    Pr(>F)
Circumference  1 467.18  467.18   18.49 0.0004311 ***
Residuals     18 454.80   25.27
---
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
-5.3189 -3.5363 -0.7817  2.8028  6.3974

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   -70.8202    28.4518  -2.489 0.023464 *
Height          0.4731     0.1566   3.021 0.007698 **
Circumference   0.7777     0.1820   4.273 0.000515 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

s: 4.172 on 17 degrees of freedom
Multiple R-squared: 0.679,
Adjusted R-squared: 0.6413
F-statistic: 17.98 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)*)

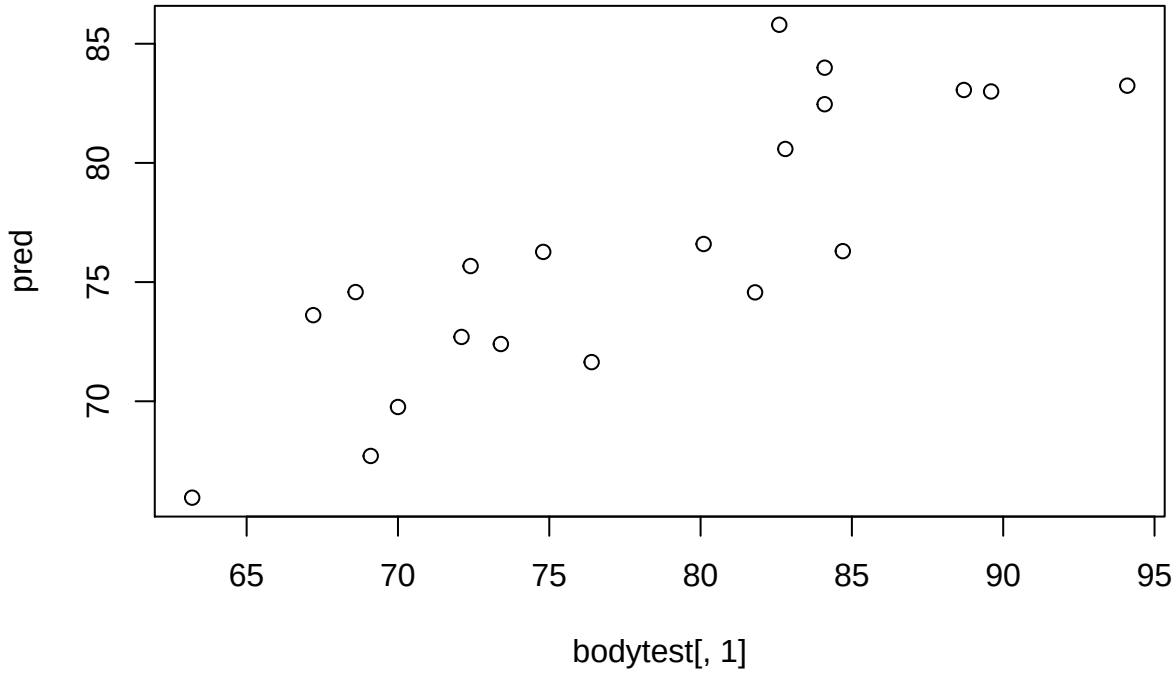
```
bodytest <- bodydata[21:40, ]
```

- 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] 0.7137211
```

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
-5.4088 -3.3900 -0.9123  3.0403  6.9827

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   -66.5785    30.0098  -2.219 0.041330 *
Height          0.4768     0.1600   2.981 0.008831 **
Circumference   0.7769     0.1858   4.181 0.000706 ***
Age            -0.2094     0.3731  -0.561 0.582352
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

s: 4.259 on 16 degrees of freedom
Multiple R-squared: 0.6852,
Adjusted R-squared: 0.6262
F-statistic: 11.61 on 3 and 16 DF,  p-value: 0.0002729
```

- 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
-4.899 -2.936 -1.176  2.956  6.472

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)   -45.21104   34.42108  -1.313  0.21173
Height          0.37156    0.18504   2.008  0.06589 .
Circumference   0.77853    0.23841   3.265  0.00614 **
Age            -0.32771    0.41016  -0.799  0.43866
Junk1          -0.15687    0.13940  -1.125  0.28080
Junk2           0.16415    0.13669   1.201  0.25122
Junk3          -0.07714    0.13614  -0.567  0.58065
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

s: 4.344 on 13 degrees of freedom
Multiple R-squared: 0.734,
Adjusted R-squared: 0.6112
F-statistic: 5.978 on 6 and 13 DF,  p-value: 0.003455
```

**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
-4.471 -2.067  1.039  1.923  4.337

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)   -94.53920   34.61011  -2.732  0.02114 *
Height          0.61173    0.19013   3.217  0.00922 **
Circumference   0.78183    0.20054   3.899  0.00297 **
Age            -0.07007    0.34876  -0.201  0.84480
Junk1           0.08223    0.16526   0.498  0.62955
Junk2          -0.02088    0.14686  -0.142  0.88975
Junk3           0.04883    0.13228   0.369  0.71971
Junk4          -0.29641    0.11810  -2.510  0.03092 *
Junk5           0.16262    0.12798   1.271  0.23263
Junk6          -0.04694    0.08255  -0.569  0.58216
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

s: 3.552 on 10 degrees of freedom
Multiple R-squared: 0.8632,
Adjusted R-squared:  0.74
F-statistic:  7.01 on 9 and 10 DF,  p-value: 0.002688
```

- 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 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  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
```

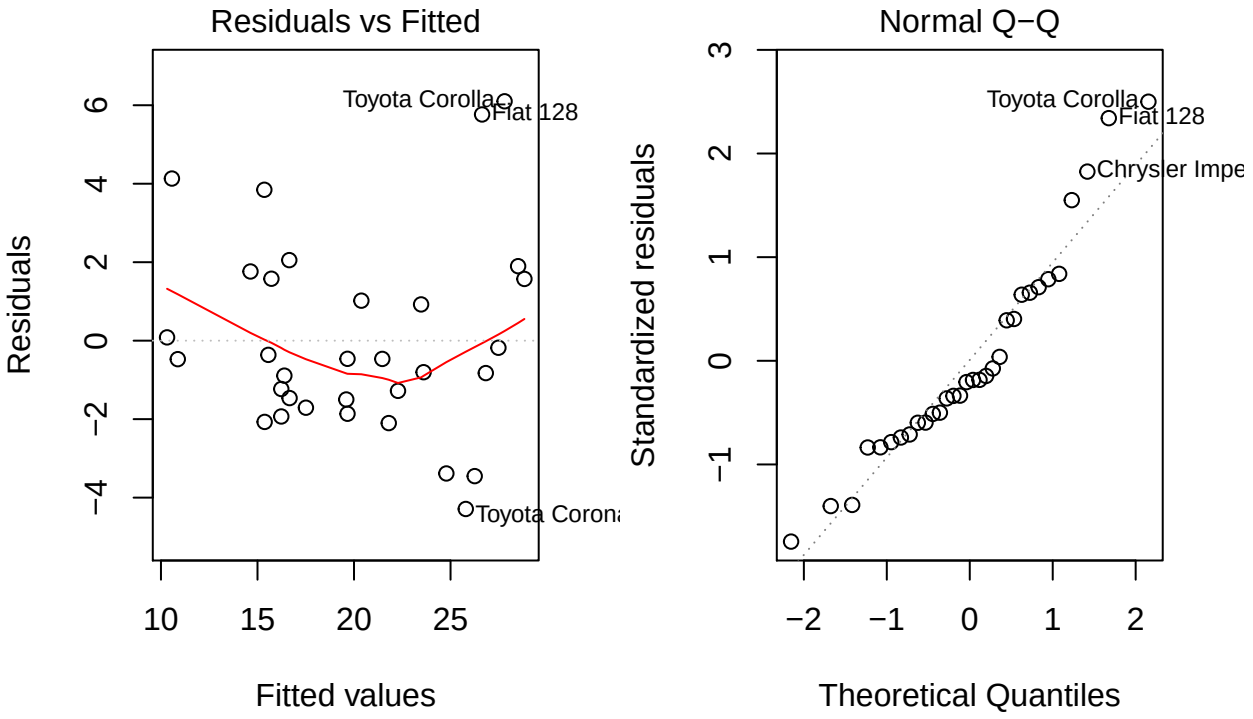
### Ex-1: Model Fitting

- 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
-4.2893 -1.5512 -0.4684  1.5743  6.1004

Coefficients:
            Estimate Std. Error t value             Pr(>|t|)
(Intercept)  39.6863     1.7150  23.141 < 0.0000000000000002 ***
cyl          -1.5078     0.4147  -3.636             0.001064 **
wt           -3.1910     0.7569  -4.216             0.000222 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

s: 2.568 on 29 degrees of freedom
```

```
Multiple R-squared: 0.8302,
Adjusted R-squared: 0.8185
F-statistic: 70.91 on 2 and 29 DF,  p-value: 0.000000000006809
```

```r
anova(cars1)
```

```
Analysis of Variance Table

Response: mpg
          Df Sum Sq Mean Sq F value             Pr(>F)
cyl        1 817.71  817.71 124.044 0.000000000005424 ***
wt         1 117.16  117.16  17.773          0.000222 ***
Residuals 29 191.17    6.59
---
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:

```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 $y$ = mpg, $x_1$ = cyl, $x_2$ = wt, $x_3$ = 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
-3.4621 -1.4913 -0.7879  1.3959  5.3499

Coefficients:
            Estimate Std. Error t value         Pr(>|t|)
(Intercept)  34.2830     2.7965  12.259 0.00000000000152 ***
cyl          -1.1814     0.3803  -3.106          0.00442 **
wt           -2.3689     0.8244  -2.874          0.00782 **
am           11.9385     3.8453   3.105          0.00444 **
wt:am        -4.1974     1.3115  -3.200          0.00350 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

s: 2.265 on 27 degrees of freedom
Multiple R-squared: 0.877,
Adjusted R-squared: 0.8588
F-statistic: 48.13 on 4 and 27 DF,  p-value: 0.000000000006643
```

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

$$\hat{y} = 34.28 - 1.18x_1 - 2.37x_2 + 11.94 \cdot 1 - 4.20x_2 \cdot 1$$

$$= 46.22 - 1.18x_1 - 6.57x_2$$

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 betwen 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) were 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     29 191.17
2     27 138.51  2    52.666 5.1333 0.0129 *
---
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 $\alpha$ 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] 3.354131
```

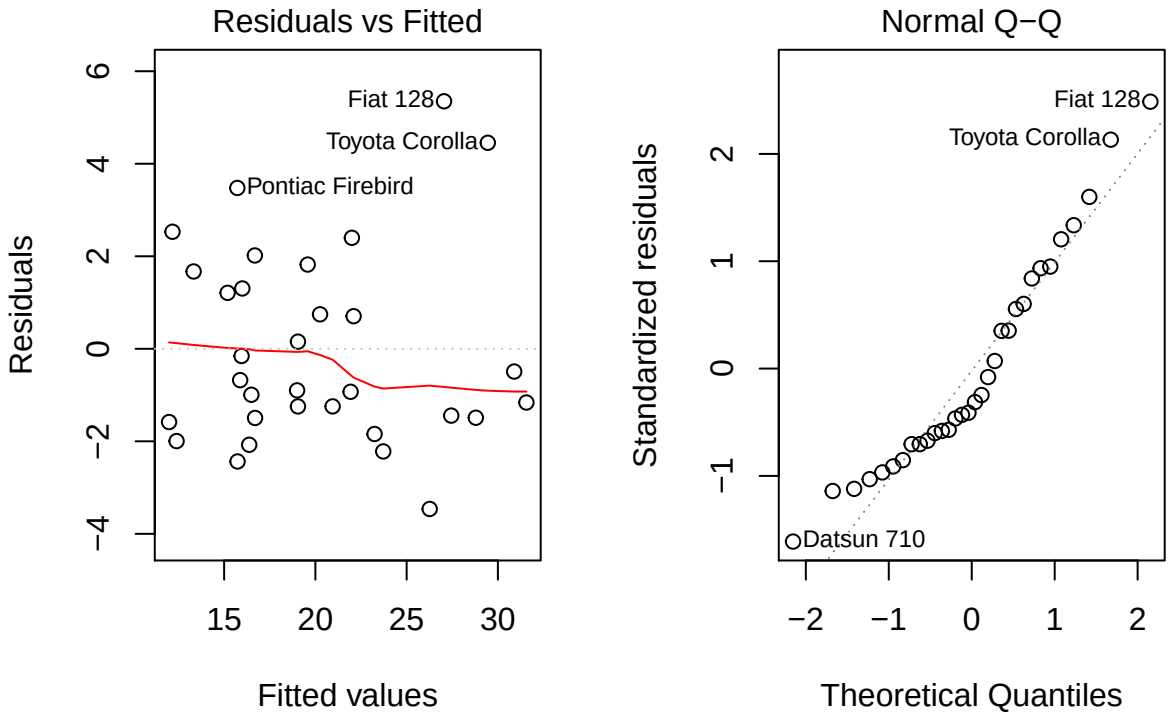
See `?FDist` for help-file for the Fisher distribution.

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 accoring 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.

```r
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
-3.9682 -1.5795 -0.4353  1.1662  5.5272

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 26.30736   14.62994   1.798  0.08424 .
cyl         -0.81856    0.81156  -1.009  0.32282
disp         0.01320    0.01204   1.097  0.28307
hp          -0.01793    0.01551  -1.156  0.25846
drat         1.32041    1.47948   0.892  0.38065
wt          -4.19083    1.25791  -3.332  0.00269 **
qsec         0.40146    0.51658   0.777  0.44436
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
s: 2.557 on 25 degrees of freedom
Multiple R-squared: 0.8548,
Adjusted R-squared:  0.82
F-statistic: 24.53 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 )                   *      278.3219 0.7528328 0.7445939 14.562907
1  ( 2 )   *                      308.3342 0.7261800 0.7170527 19.152594
1  ( 3 )       *                  317.1587 0.7183433 0.7089548 20.502090
1  ( 4 )          *               447.6743 0.6024373 0.5891853 40.461438
1  ( 5 )               *          603.5667 0.4639952 0.4461283 64.301580
2  ( 1 )   *              *       191.1720 0.8302274 0.8185189  3.235333
2  ( 2 )          *       *       195.0478 0.8267855 0.8148396  3.828045
2  ( 3 )                  *     * 195.4636 0.8264161 0.8144448  3.891644
2  ( 4 )       *          *       246.6825 0.7809306 0.7658223 11.724387
2  ( 5 )               *  *       269.2413 0.7608970 0.7444071 15.174232
3  ( 1 )   *       *      *       176.6205 0.8431500 0.8263446  3.010026
3  ( 2 )   *              *     * 180.6046 0.8396119 0.8224275  3.619295
3  ( 3 )               *  *     * 183.5216 0.8370214 0.8195594  4.065383
3  ( 4 )          *    *  *       183.6819 0.8368791 0.8194018  4.089899
3  ( 5 )          *       *     * 186.0593 0.8347678 0.8170643  4.453469
4  ( 1 )   *    *  *       *       170.4444 0.8486348 0.8262103  4.065526
4  ( 2 )          *    *  *     * 174.1035 0.8453853 0.8224794  4.625103
4  ( 3 )   *      *    *  *       174.3752 0.8451439 0.8222023  4.666661
4  ( 4 )   *      *       *     * 175.2195 0.8443942 0.8213415  4.795768
4  ( 5 )   *    *         *     * 175.6684 0.8439955 0.8208838  4.864420
5  ( 1 )   *    *  *    *  *       167.4261 0.8513152 0.8227219  5.603955
```

```
5  ( 2 )   *    *  *       *    * 168.6853 0.8501969 0.8213886  5.796524
5  ( 3 )        *  *    *  *    * 170.1291 0.8489147 0.8198599  6.017318
5  ( 4 )   *       *    *  *    * 171.3467 0.8478334 0.8185706  6.203520
5  ( 5 )   *    *       *  *    * 172.2208 0.8470572 0.8176451  6.337198
6  ( 1 )   *    *  *    *  *    * 163.4768 0.8548224 0.8199798  7.000000
```

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 scource 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 ttoal 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)
```

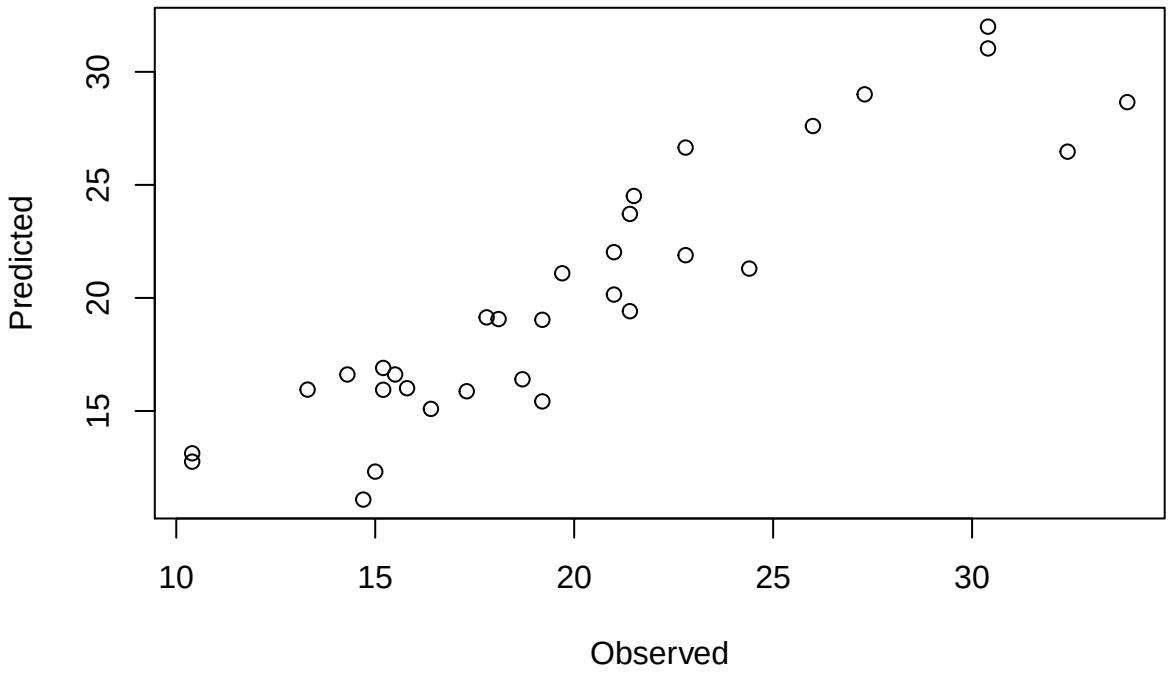
```
print(res)
```

```
$pred
 [1] 22.02465 20.14940 26.64983 19.41364 16.40368 19.06674 16.61396
 [8] 21.29786 21.89094 19.03386 19.14117 15.09325 15.87195 15.93892
[15] 13.12962 12.75906 11.08191 26.47037 31.03522 28.65971 24.50916
[22] 16.61597 16.90330 15.94565 15.42444 29.00620 27.60453 31.99763
[29] 16.00725 21.09180 12.31828 23.71533

$msep
[1] 6.088405

$r.squared.pred
[1] 0.8282658
```

The `CV()` returns a list with three elements, the predictions, the Mean Square Error of Prediction and 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] 22.96794 22.07539 26.25638 20.90993 16.97878 20.41187 15.65251
 [8] 23.64555 23.38461 20.02785 20.09677 14.96997 16.05497 16.07208
[15] 11.01924 10.08662  8.73892 26.32479 28.71190 27.34888 25.82503
[22] 17.69247 18.08644 14.79267 15.57424 27.70402 26.61982 27.72238
[29] 16.57704 21.28440 12.89349 24.61230

$msep
[1] 7.189898
```

```
$r.squared.pred
[1] 0.7972019
```

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] 22.27053 20.08943 26.96290 19.46344 16.23783 18.84980 16.55208
 [8] 21.69562 21.88502 19.23336 19.23336 15.25603 16.13734 16.12464
[15] 13.21458 10.77692 11.00258 26.45393 31.20857 28.76698 23.51149
[22] 16.77591 16.84913 15.98576 15.28733 29.43170 26.89460 32.35720
[29] 15.98236 21.25588 12.27157 23.24905

$msep
[1] 5.826322

$r.squared.pred
[1] 0.8368866
```

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.