IODS course project

Subam Kathet

Contents

Introduction to Open Data Science - Course Project **2**

IODS Course Project **2**

About the project **2**

Week 1: Start me up !! The book and material **2**

Week 2: Regression and model validation **3**
2.0 Installing the packages . . . 3
2.1 Reading data from the web . . . 3
2.2 Scaling variables . . . 4
2.3 Combining variables . . . 5
2.4 Selecting columns . . . 6
2.5 Modifying column names . . . 10
2.6 Excluding observations . . . 10
2.7 Visualizations with ggplot2 . . . 11
2.8 Exploring a data frame . . . 14
2.9 Simple regression . . . 15
2.10 Multiple regression . . . 17
2.11 Graphical model validation . . . 19
2.12 Making predictions . . . 22

Week 3: Logistic Regression **24**
3.0 Loading packages . . . 24
3.1 Data wrangling . . . 24
3.1 Creating new R Markdown file . . . 24
3.2 Importing the data set and exploration . . . 24
3.3 Choosing the variables . . . 26
3.4 Exploring and plotting the choosen variable . . . 26
3.4 Logistic regression . . . 30
3.5 Power of the model . . . 31
3.6 Cross validation (Bonus) . . . 31
3.7 Cross validation of different model (Super-Bonus) . . . 31

Week 4: Clustering and Classification **32**
4.0 Packages for clustering and classification . . . 32
4.1 loading data and Exploring the data . . . 32
4.2 Graphical Overview of the data set . . . 33
4.3 Standardizing the dataset . . . 37
4.4 Fitting the linear discriminant analysis . . . 40
4.5 Predicting the LDA model . . . 42
4.6 Model Optimization and K means clustering . . . 42

4.7 K-means: determine the k 45

Week 5: Dimensionality reduction techniques **47**
5.1 Packages required 47
5.2 Loading the data set from wrangling exercise 48
5.3 Graphical overview 48
5.4 Principal component analysis (PCA) 50
5.5 Intrepretation of the analysis 52
5.6 Lets see the "tea" data set 53
5.7 Multiple Correspondence Analysis (MCA) with "tea" data set 56

Introduction to Open Data Science - Course Project

IODS Course Project

SUBAM KATHET

This course was recommended by a friend who is in the masters program in data science at the University of Helsinki. Data management, mining, data analytic and machine learning among many other within the same sphere are the next generation skill set everyone is recommended to acquire and here I am. I am a bit nervous, very exited and mostly curious to take a deep dive into the world of data science.

About the project

Here is the link to my github webpage.

https://iamsubam.github.io/IODS-project/

And here is the link to my course diary.

https://github.com/iamsubam/IODS-project

Week 1: Start me up !! The book and material

I have only had some time to browse through the R for Health Data Science. Coming from a background of experimental epidemiology, I was drawn immediately by linear and logistic regression because this is something I often rely on in my work. I looked into survival analysis briefly because of my interest and found it quite interesting. Although I need to practice a lot before I can get my hands around the analysis. I think the book gives a great over view of essential statistical analysis required on a fundamental level. Some knowledge of statistics can be of great advantage as R platform is already designed with a steep learning curve.

```
# This is a so-called "R chunk" where you can write R code.

date()
```

```
## [1] "Tue Dec  6 21:30:26 2022"
```

```
# Trying to check if the chunk works or not. It is usually a struggle especially when R version is outd
```

Week 2: Regression and model validation

This set consists of a few numbered exercises. Go to each exercise in turn and do as follows:

1. Read the brief description of the exercise.
2. Run the (possible) pre-exercise-code chunk.
3. Follow the instructions to fix the R code!

2.0 Installing the packages

One or more extra packages (in addition to `tidyverse`) will be needed below.

```
# Select (with mouse or arrow keys) the install.packages("...") and
# run it (by Ctrl+Enter / Cmd+Enter):

library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
```

```
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.8     v stringr 1.4.1
## v tidyr   1.2.0     v forcats 0.5.2
## v readr   2.1.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(ggplot2)
library(gapminder)
library(finalfit)
library(broom)
```

2.1 Reading data from the web

The first step of data analysis with R is reading data into R. This is done with a function. Which function and function arguments to use to do this, depends on the original format of the data.

Conveniently in R, the same functions for reading data can usually be used whether the data is saved locally on your computer or somewhere else behind a web URL.

After the correct function has been identified and data read into R, the data will usually be in R `data.frame` format. Te dimensions of a data frame are (n,d), where n is the number of rows (the observations) and d the number of columns (the variables).

The purpose of this course is to expose you to some basic and more advanced tasks of programming and data analysis with R.

Instructions

- Read the `lrn14` data frame to memory with `read.table()`. There is information related to the data here
- Use `dim()` on the data frame to look at the dimensions of the data. How many rows and colums does the data have?
- Look at the structure of the data with `str()`.

Hint: - For both functions you can pass a data frame as the first (unnamed) argument.

R code

```
# This is a code chunk in RStudio editor.
# Work with the exercise in this chunk, step-by-step. Fix the R code!

# read the data into memory
lrn14 <- read.table("http://www.helsinki.fi/~kvehkala/JYTmooc/JYTOPKYS3-meta.txt", sep="\t", header=TRU
```

```
## Warning in scan(file = file, what = what, sep = sep, quote = quote, dec = dec, :
## EOF within quoted string
```

```
# Look at the dimensions of the data

# Look at the structure of the data
#use .txt file to import data set for better description.
# Preliminary results available at http://www.slideshare.net/kimmovehkalahti/the-relationship-between-l
#Total respondents n=183, total question n=60, so 184 rows including heading and 60 columns
#The code as respective column heading represents a question related to the survey and number. Each SN

dim(lrn14)
```

```
## [1] 28  1
```

```
str(lrn14)
```

```
## 'data.frame':    28 obs. of  1 variable:
##  $ Variable.descriptions.and.other.meta.data.of.JYTOPKYS3..syksy.2016.: chr  "Kimmo Vehkalahti: ASSI
```

2.2 Scaling variables

The next step is wrangling the data into a format that is easy to analyze. We will wrangle our data for the next few exercises.

A neat thing about R is that may operations are *vectorized*. It means that a single operation can affect all elements of a vector. This is often convenient.

The column `Attitude` in `lrn14` is a sum of 10 questions related to students attitude towards statistics, each measured on the Likert scale (1-5). Here we'll scale the combination variable back to the 1-5 scale.

Instructions

- Execute the example codes to see how vectorized division works

- Use vector division to create a new column `attitude` in the `lrn14` data frame, where each observation of `Attitude` is scaled back to the original scale of the questions, by dividing it with the number of questions.

Hint: - Assign 'Attitude divided by 10' to the new column 'attitude.

R code

```
lrn14$attitude <- lrn14$Attitude / 10
```

2.3 Combining variables

Our data includes many questions that can be thought to measure the same *dimension*. You can read more about the data and the variables here. Here we'll combine multiple questions into combination variables. Useful functions for summation with data frames in R are

function	description
`colSums(df)`	returns a sum of each column in `df`
`rowSums(df)`	returns a sum of each row in `df`
`colMeans(df)`	returns the mean of each column in `df`
`rowMeans(df)`	return the mean of each row in `df`

We'll combine the use of `rowMeans()`with the `select()` function from the **dplyr** library to average the answers of selected questions. See how it is done from the example codes.

Instructions

- Access the **dplyr** library
- Execute the example codes to create the combination variables 'deep' and 'surf' as columns in `lrn14`
- Select the columns related to strategic learning from `lrn14`
- Create the combination variable 'stra' as a column in `lrn14`

Hints: - Columns related to strategic learning are in the object `strategic_questions`. Use it for selecting the correct columns. - Use the function `rowMeans()` identically to the examples

R code

```
# Work with the exercise in this chunk, step-by-step. Fix the R code!
# lrn14 is available

# Access the dplyr library
library(dplyr)

# questions related to deep, surface and strategic learning
deep_questions <- c("D03", "D11", "D19", "D27", "D07", "D14", "D22", "D30","D06",  "D15", "D23", "D31")
surface_questions <- c("SU02","SU10","SU18","SU26", "SU05","SU13","SU21","SU29","SU08","SU16","SU24","SU
strategic_questions <- c("ST01","ST09","ST17","ST25","ST04","ST12","ST20","ST28")


# select the columns related to deep learning
deep_columns <- select(lrn14, one_of(deep_questions))
# and create column 'deep' by averaging
lrn14$deep <- rowMeans(deep_columns)
```

```
# select the columns related to surface learning
surface_columns <- select(lrn14, one_of(surface_questions))
# and create column 'surf' by averaging
lrn14$surf <- rowMeans(surface_columns)

# select the columns related to strategic learning
strategic_columns <- select(lrn14, one_of(strategic_questions))
# and create column 'stra' by averaging
lrn14$stra <- rowMeans(strategic_columns)
```

2.4 Selecting columns

Often it is convenient to work with only a certain column or a subset of columns of a bigger data frame. There are many ways to select columns of data frame in R and you saw one of them in the previous exercise: `select()` from **dplyr***.

dplyr is a popular library for *data wrangling*. There are also convenient data wrangling cheatsheets by RStudio to help you get started (dplyr, tidyr etc.)

Instructions

- Access the **dplyr** library
- Create object `keep_columns`
- Use `select()` (possibly together with `one_of()`) to create a new data frame `learning2014` with the columns named in `keep_columns`.
- Look at the structure of the new dataset

Hint: - See the previous exercise or the data wrangling cheatsheet for help on how to select columns

R code

```
# Work with the exercise in this chunk, step-by-step. Fix the R code!

# lrn14 is available

# access the dplyr library
library(dplyr)

# choose a handful of columns to keep
keep_columns <- c("gender","Age","attitude", "deep", "stra", "surf", "Points")

# select the 'keep_columns' to create a new dataset
learning2014 <- select(lrn14,all_of(keep_columns))

# see the structure of the new dataset

print(learning2014)
```

```
##     gender Age attitude     deep  stra     surf Points
## 1        F  53      3.7 3.583333 3.375 2.583333     25
## 2        M  55      3.1 2.916667 2.750 3.166667     12
## 3        F  49      2.5 3.500000 3.625 2.250000     24
## 4        M  53      3.5 3.500000 3.125 2.250000     10
## 5        M  49      3.7 3.666667 3.625 2.833333     22
## 6        F  38      3.8 4.750000 3.625 2.416667     21
```

```
## 7        M  50      3.5 3.833333 2.250 1.916667     21
## 8        F  37      2.9 3.250000 4.000 2.833333     31
## 9        M  37      3.8 4.333333 4.250 2.166667     24
## 10       F  42      2.1 4.000000 3.500 3.000000     26
## 11       M  37      3.9 3.583333 3.625 2.666667     31
## 12       F  34      3.8 3.833333 4.750 2.416667     31
## 13       F  34      2.4 4.250000 3.625 2.250000     23
## 14       F  34      3.0 3.333333 3.500 2.750000     25
## 15       M  35      2.6 4.166667 1.750 2.333333     21
## 16       F  46      2.5 3.083333 3.125 2.666667      0
## 17       F  33      4.1 3.666667 3.875 2.333333     31
## 18       F  32      2.6 4.083333 1.375 2.916667     20
## 19       F  44      2.6 3.500000 3.250 2.500000     22
## 20       M  29      1.7 4.083333 3.000 3.750000      9
## 21       F  30      2.7 4.000000 3.750 2.750000     24
## 22       M  27      3.9 3.916667 2.625 2.333333     28
## 23       M  29      3.4 4.000000 2.375 2.416667     30
## 24       F  31      2.7 4.000000 3.625 3.000000     24
## 25       F  37      2.3 3.666667 2.750 2.416667      9
## 26       F  26      3.7 3.666667 1.750 2.833333     26
## 27       F  26      4.4 4.416667 3.250 3.166667     32
## 28       M  30      4.1 3.916667 4.000 3.000000     32
## 29       F  37      2.4 3.833333 2.125 2.166667      0
## 30       F  33      3.7 3.750000 3.625 2.000000     33
## 31       F  33      2.5 3.250000 2.875 3.500000     29
## 32       M  28      3.0 3.583333 3.000 3.750000     30
## 33       M  26      3.4 4.916667 1.625 2.500000     19
## 34       F  27      3.2 3.583333 3.250 2.083333     23
## 35       F  25      2.0 2.916667 3.500 2.416667     19
## 36       F  31      2.4 3.666667 3.000 2.583333     12
## 37       M  20      4.2 4.500000 3.250 1.583333     10
## 38       F  39      1.6 4.083333 1.875 2.833333     11
## 39       M  38      3.1 3.833333 4.375 1.833333     20
## 40       M  24      3.8 3.250000 3.625 2.416667     26
## 41       M  26      3.8 2.333333 2.500 3.250000     31
## 42       M  25      3.3 3.333333 1.250 3.416667     20
## 43       F  30      1.7 4.083333 4.000 3.416667     23
## 44       F  25      2.5 2.916667 3.000 3.166667     12
## 45       M  30      3.2 3.333333 2.500 3.500000     24
## 46       F  48      3.5 3.833333 4.875 2.666667     17
## 47       F  24      3.2 3.666667 5.000 2.416667     29
## 48       F  40      4.2 4.666667 4.375 3.583333     23
## 49       M  25      3.1 3.750000 3.250 2.083333     28
## 50       F  23      3.9 3.416667 4.000 3.750000     31
## 51       F  25      1.9 4.166667 3.125 2.916667     23
## 52       F  23      2.1 2.916667 2.500 2.916667     25
## 53       M  27      2.5 4.166667 3.125 2.416667     18
## 54       M  25      3.2 3.583333 3.250 3.000000     19
## 55       M  23      3.2 2.833333 2.125 3.416667     22
## 56       F  23      2.6 4.000000 2.750 2.916667     25
## 57       F  23      2.3 2.916667 2.375 3.250000     21
## 58       F  45      3.8 3.000000 3.125 3.250000      9
## 59       F  22      2.8 4.083333 4.000 2.333333     28
## 60       F  23      3.3 2.916667 4.000 3.250000     25
```

```
## 61       M  21      4.8 3.500000 2.250 2.500000     29
## 62       M  21      4.0 4.333333 3.250 1.750000     33
## 63       F  21      4.0 4.250000 3.625 2.250000     33
## 64       F  21      4.7 3.416667 3.625 2.083333     25
## 65       F  26      2.3 3.083333 2.500 2.833333     18
## 66       F  25      3.1 4.583333 1.875 2.833333     22
## 67       F  26      2.7 3.416667 2.000 2.416667     17
## 68       M  21      4.1 3.416667 1.875 2.250000     25
## 69       F  22      3.4 3.500000 2.625 2.333333      0
## 70       F  23      3.4 3.416667 4.000 2.833333     28
## 71       F  22      2.5 3.583333 2.875 2.250000     22
## 72       F  22      2.1 1.583333 3.875 1.833333     26
## 73       F  22      1.4 3.333333 2.500 2.916667     11
## 74       F  23      1.9 4.333333 2.750 2.916667     29
## 75       M  22      3.7 4.416667 4.500 2.083333     22
## 76       M  25      4.1 4.583333 3.500 2.666667      0
## 77       M  23      3.2 4.833333 3.375 2.333333     21
## 78       M  24      2.8 3.083333 2.625 2.416667     28
## 79       F  22      4.1 3.000000 4.125 2.750000     33
## 80       F  23      2.5 4.083333 2.625 3.250000     16
## 81       M  22      2.8 4.083333 2.250 1.750000     31
## 82       M  20      3.8 3.750000 2.750 2.583333     22
## 83       M  22      3.1 3.083333 3.000 3.333333     31
## 84       M  21      3.5 4.750000 1.625 2.833333     23
## 85       F  22      3.6 4.250000 1.875 2.500000     26
## 86       F  23      2.6 4.166667 3.375 2.416667     12
## 87       M  21      4.4 4.416667 3.750 2.416667     26
## 88       M  22      4.5 3.833333 2.125 2.583333     31
## 89       M  29      3.2 3.333333 2.375 3.000000     19
## 90       F  26      2.0 3.416667 1.750 2.333333      0
## 91       F  29      3.9 3.166667 2.750 2.000000     30
## 92       F  21      2.5 3.166667 3.125 3.416667     12
## 93       M  28      3.3 3.833333 3.500 2.833333     17
## 94       M  23      3.5 4.166667 1.625 3.416667      0
## 95       F  21      3.3 4.250000 2.625 2.250000     18
## 96       F  30      3.0 3.833333 3.375 2.750000     19
## 97       F  21      2.9 3.666667 2.250 3.916667     21
## 98       M  23      3.3 3.833333 3.000 2.333333     24
## 99       F  21      3.3 3.833333 4.000 2.750000     28
## 100      F  21      3.5 3.833333 3.500 2.750000     17
## 101      F  20      3.6 3.666667 2.625 2.916667     18
## 102      M  21      4.2 4.166667 2.875 2.666667      0
## 103      M  22      3.7 4.333333 2.500 2.083333     17
## 104      F  35      2.8 4.416667 3.250 3.583333      0
## 105      M  21      4.2 3.750000 3.750 3.666667     23
## 106      M  22      2.2 2.666667 2.000 3.416667      0
## 107      M  21      3.2 4.166667 3.625 2.833333     26
## 108      F  20      5.0 4.000000 4.125 3.416667     28
## 109      M  22      4.7 4.000000 4.375 1.583333     31
## 110      F  20      3.6 4.583333 2.625 2.916667     27
## 111      F  20      3.6 3.666667 4.000 3.000000     25
## 112      M  24      2.9 3.666667 2.750 2.916667     23
## 113      F  20      3.5 3.833333 2.750 2.666667     21
## 114      F  19      4.0 2.583333 1.375 3.000000     27
```

```
## 115      F  21      3.5 3.500000 2.250 2.750000     28
## 116      F  21      3.2 3.083333 3.625 3.083333     23
## 117      F  22      2.6 4.250000 3.750 2.500000     21
## 118      F  25      2.0 3.166667 4.000 2.333333     25
## 119      F  21      2.7 3.083333 3.125 3.000000     11
## 120      F  22      3.2 4.166667 3.250 3.000000     19
## 121      F  25      3.3 2.250000 2.125 4.000000     24
## 122      F  20      3.9 3.333333 2.875 3.250000     28
## 123      M  24      3.3 3.083333 1.500 3.500000     21
## 124      F  20      3.0 2.750000 2.500 3.500000     24
## 125      M  21      3.7 3.250000 3.250 3.833333     24
## 126      F  26      1.4 3.750000 3.250 3.083333      0
## 127      M  28      3.0 4.750000 2.500 2.500000      0
## 128      F  20      2.5 4.000000 3.625 2.916667     20
## 129      F  20      2.9 3.583333 3.875 2.166667     19
## 130      M  31      3.9 4.083333 3.875 1.666667     30
## 131      F  20      3.6 4.250000 2.375 2.083333     22
## 132      F  22      2.9 3.416667 3.000 2.833333     16
## 133      F  22      2.1 3.083333 3.375 3.416667     16
## 134      M  21      3.1 3.500000 2.750 3.333333     19
## 135      F  20      2.4 3.916667 3.125 3.500000      0
## 136      M  22      4.0 3.666667 4.500 2.583333     30
## 137      F  21      3.1 4.250000 2.625 2.833333     23
## 138      F  21      2.3 4.250000 2.750 3.333333     19
## 139      F  21      2.8 3.833333 3.250 3.000000     18
## 140      F  21      3.7 4.416667 4.125 2.583333     28
## 141      F  20      2.6 3.500000 3.375 2.416667     21
## 142      F  21      2.4 3.583333 2.750 3.583333     19
## 143      F  25      3.0 3.666667 4.125 2.083333     27
## 144      F  27      2.9 4.250000 3.000 2.750000      0
## 145      F  20      3.2 3.750000 4.125 3.916667      0
## 146      M  21      2.8 2.083333 3.250 4.333333     24
## 147      F  24      2.9 4.250000 2.875 2.666667     21
## 148      F  20      2.4 3.583333 2.875 3.000000     20
## 149      M  21      3.1 4.000000 2.375 2.666667     28
## 150      F  20      1.9 3.333333 3.875 2.166667     12
## 151      F  20      2.0 3.500000 2.125 2.666667     21
## 152      F  18      3.8 3.166667 4.000 2.250000     28
## 153      F  21      3.4 3.583333 3.250 2.666667     31
## 154      F  19      3.7 3.416667 2.625 3.333333     18
## 155      F  21      2.9 4.250000 2.750 3.500000     25
## 156      F  20      2.3 3.250000 4.000 2.750000     19
## 157      M  21      4.1 4.416667 3.000 2.000000     21
## 158      F  20      2.7 3.250000 3.375 2.833333     16
## 159      F  21      3.5 3.916667 3.875 3.500000      7
## 160      F  20      3.4 3.583333 3.250 2.500000     21
## 161      F  18      3.2 4.500000 3.375 3.166667     17
## 162      M  22      3.3 3.583333 4.125 3.083333     22
## 163      F  22      3.3 3.666667 3.500 2.916667     18
## 164      M  24      3.5 2.583333 2.000 3.166667     25
## 165      F  19      3.2 4.166667 3.625 2.500000     24
## 166      F  20      3.1 3.250000 3.375 3.833333     23
## 167      F  21      2.4 3.583333 2.250 2.833333      0
## 168      F  20      2.8 4.333333 2.125 2.250000     23
```

```
## 169      F  17      1.7 3.916667 4.625 3.416667     26
## 170      M  19      1.9 2.666667 2.500 3.750000     12
## 171      F  23      3.2 3.583333 2.000 1.750000      0
## 172      F  20      3.5 3.083333 2.875 3.000000     32
## 173      F  20      2.4 3.750000 2.750 2.583333     22
## 174      F  25      3.8 4.083333 3.375 2.750000      0
## 175      F  20      2.1 4.166667 4.000 3.333333     20
## 176      F  20      2.9 4.166667 2.375 2.833333     21
## 177      F  19      1.9 3.250000 3.875 3.000000     23
## 178      F  19      2.0 4.083333 3.375 2.833333     20
## 179      F  22      4.2 2.916667 1.750 3.166667     28
## 180      M  35      4.1 3.833333 3.000 2.750000     31
## 181      F  18      3.7 3.166667 2.625 3.416667     18
## 182      F  19      3.6 3.416667 2.625 3.000000     30
## 183      M  21      1.8 4.083333 3.375 2.666667     19
```

2.5 Modifying column names

Sometimes you want to rename your column. You could do this by creating copies of the columns with new names, but you can also directly get and set the column names of a data frame, using the function `colnames()`.

The **dplyr** library has a `rename()` function, which can also be used. Remember the cheatsheets.

Instructions

- Print out the column names of `learning2014`
- Change the name of the second column to 'age'
- Change the name of 'Points' to 'points'
- Print out the column names again to see the changes

Hint: - You can use `colnames()` similarily to the example. Which index matches the column 'Points'?

R code

```
print(names(learning2014))
```

```
## [1] "gender"   "Age"      "attitude" "deep"     "stra"     "surf"     "Points"
```

```
colnames(learning2014)[2] <- "age"
learning2014 <- rename(learning2014, points = Points)
```

```
print(dim(learning2014)) #check the dimension now (must have 166 rown and 7)
```

```
## [1] 183   7
```

2.6 Excluding observations

Often your data includes outliers or other observations which you wish to remove before further analysis. Or perhaps you simply wish to work with some subset of your data.

In the **learning2014** data the variable 'points' denotes the students exam points in a statistics course exam. If the student did not attend an exam, the value of 'points' will be zero. We will remove these observations from the data.

R code

Instructions

- Access the **dplyr** library
- As an example, create object `male_students` by selecting the male students from `learning2014`
- Override `learning2014` and select rows where the 'points' variable is greater than zero.
- If you do not remember how logical comparison works in R, see the 'Logical comparison' exercise from the course 'R Short and Sweet'.

Hint: - The "greater than" logical operator is `>`

```
dim(lrn14)
```

```
## [1] 183  64
```

```
dim(learning2014)
```

```
## [1] 166   7
```

```
#Export csv file
setwd("~/Documents/GitHub/IODS-project")
write_csv(learning2014, 'learning2014.csv')
```

2.7 Visualizations with ggplot2

ggplot2 is a popular library for creating stunning graphics with R. It has some advantages over the basic plotting system in R, mainly consistent use of function arguments and flexible plot alteration. ggplot2 is an implementation of Leland Wilkinson's *Grammar of Graphics* — a general scheme for data visualization.

In ggplot2, plots may be created via the convenience function `qplot()` where arguments and defaults are meant to be similar to base R's `plot()` function. More complex plotting capacity is available via `ggplot()`, which exposes the user to more explicit elements of the grammar. (from wikipedia)

RStudio has a cheatsheet for data visualization with ggplot2.

Instructions

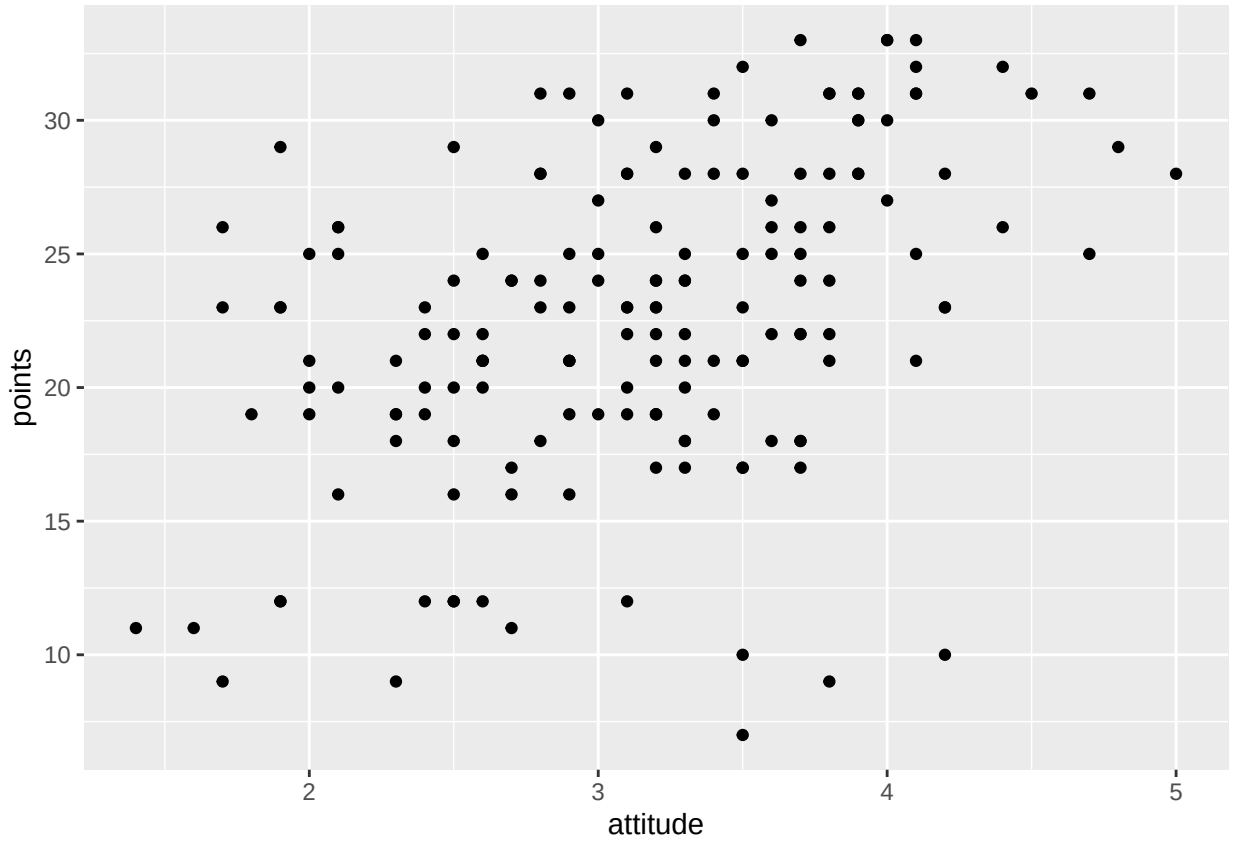
- Access the **ggplot2** library
- Initialize the plot with data and aesthetic mappings
- Adjust the plot initialization: Add an aesthetic element to the plot by defining `col = gender` inside `aes()`.
- Define the visualization type (points)
- Draw the plot to see how it looks at this point
- *Add* a regression line to the plot
- *Add* the title "Student's attitude versus exam points" with `ggtitle("<insert title here>")` to the plot with regression line
- Draw the plot again to see the changes

Hints: - Use `+` to add the title to the plot - The plot with regression line is saved in the object `p3` - You can draw the plot by typing the object name where the plot is saved

R code

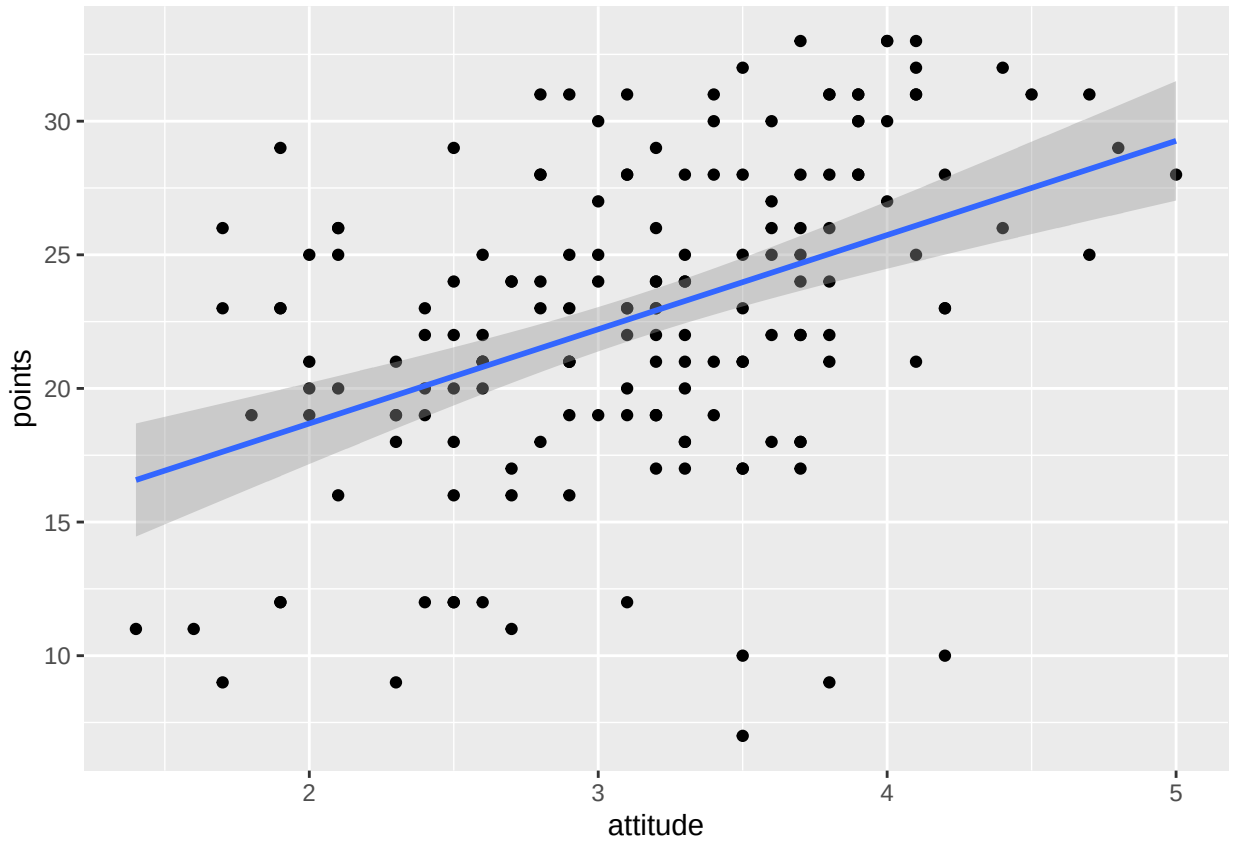
```
# Work with the exercise in this chunk, step-by-step. Fix the R code!
# learning2014 is available

# Access the gglot2 library
library(ggplot2)
```

```r
# initialize plot with data and aesthetic mapping
p1 <- ggplot(learning2014, aes(x = attitude, y = points))

# define the visualization type (points)
p2 <- p1 + geom_point()

# draw the plot
p2
```

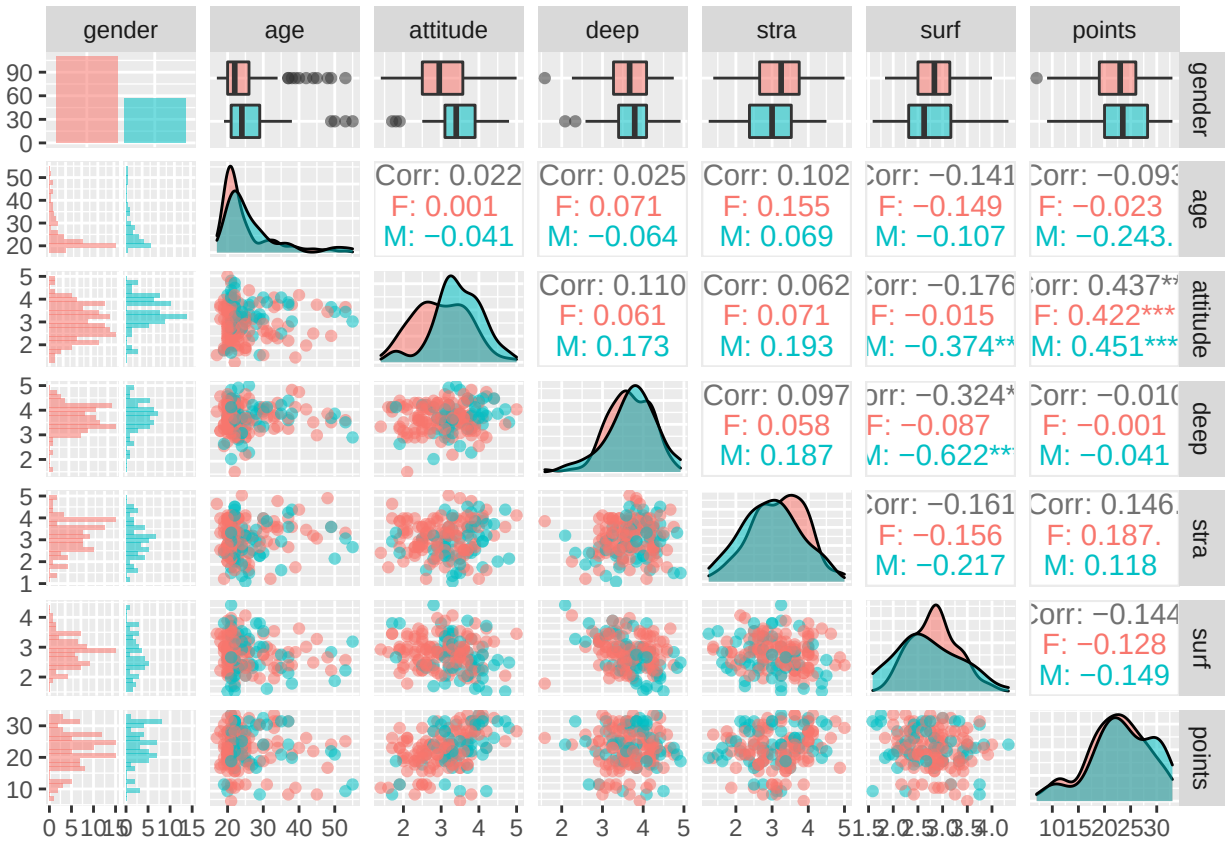



```r
# add a regression line
p3 <- p2 + geom_smooth(method = "lm")

# draw the plot
p3
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
#Lets try and overview summary
p <- ggpairs(learning2014, mapping = aes(col = gender, alpha = 0.3), lower = list(combo = wrap("facethis
 # draw the plot!
p
```



Fitting a regression line in a scatter plot between points and attitude doesn't provide a strong linear relationship. Would be interesting to work on this particular model as exercise to predict the relationship between these two variables.

2.8 Exploring a data frame

Often the most interesting feature of your data are the relationships between the variables. If there are only a handful of variables saved as columns in a data frame, it is possible to visualize all of these relationships neatly in a single plot.

Base R offers a fast plotting function `pairs()`, which draws all possible scatter plots from the columns of a data frame, resulting in a scatter plot matrix. Libraries **GGally** and **ggplot2** together offer a slow but more detailed look at the variables, their distributions and relationships.

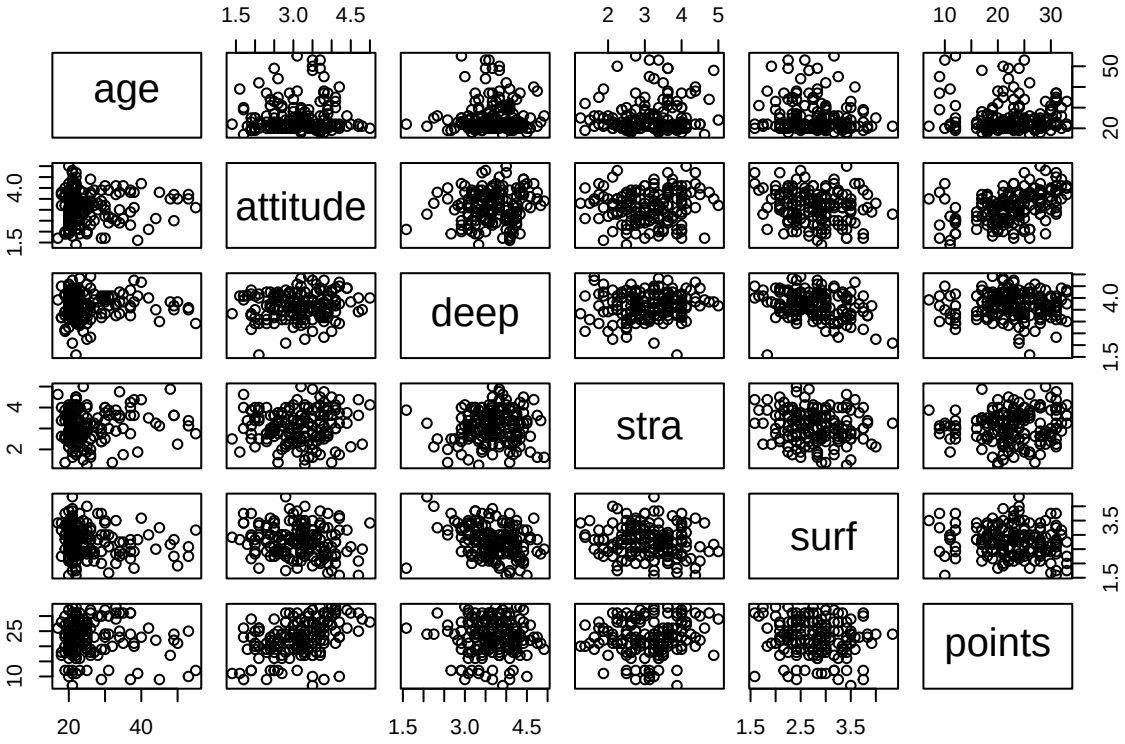
Instructions

- Draw a scatter matrix of the variables in learning2014 (other than gender)
- Adjust the code: Add the argument `col` to the `pairs()` function, defining the colour with the 'gender' variable in learning2014.
- Draw the plot again to see the changes.
- Access the **ggpot2** and **GGally** libraries and create the plot `p` with `ggpairs()`.
- Draw the plot. Note that the function is a bit slow.
- Adjust the argument `mapping` of `ggpairs()` by defining `col = gender` inside `aes()`.
- Draw the plot again.
- Adjust the code a little more: add another aesthetic element `alpha = 0.3` inside `aes()`.
- See the difference between the plots?

Hints: - You can use `$` to access a column of a data frame. - Remember to separate function arguments with a comma - You can draw the plot `p` by simply typing it's name: just like printing R objects.

R code

```
# Work with the exercise in this chunk, step-by-step. Fix the R code!
# learning2014 is available

# draw a scatter plot matrix of the variables in learning2014.
# [-1] excludes the first column (gender)
pairs(learning2014[-1])
```




```
# access the GGally and ggplot2 libraries
library(GGally)
library(ggplot2)

# create a more advanced plot matrix with ggpairs()
p <- ggpairs(learning2014, mapping = aes(), lower = list(combo = wrap("facethist", bins = 20)))
```

2.9 Simple regression

Regression analysis with R is easy once you have your data in a neat data frame. You can simply use the `lm()` function to fit a linear model. The first argument of `lm()` is a `formula`, which defines the target variable and the explanatory variable(s).

The formula should be `y ~ x`, where `y` is the target (or outcome) variable and `x` the explanatory variable (predictor). The second argument of `lm()` is `data`, which should be a data frame where `y` and `x` are columns.

The output of `lm()` is a linear model object, which can be saved for later use. The generic function `summary()` can be used to print out a summary of the model.

Instructions

- Create a scatter plot of 'points' versus 'attitude'.
- Fit a regression model where 'points' is the target and 'attitude' is the explanatory variable

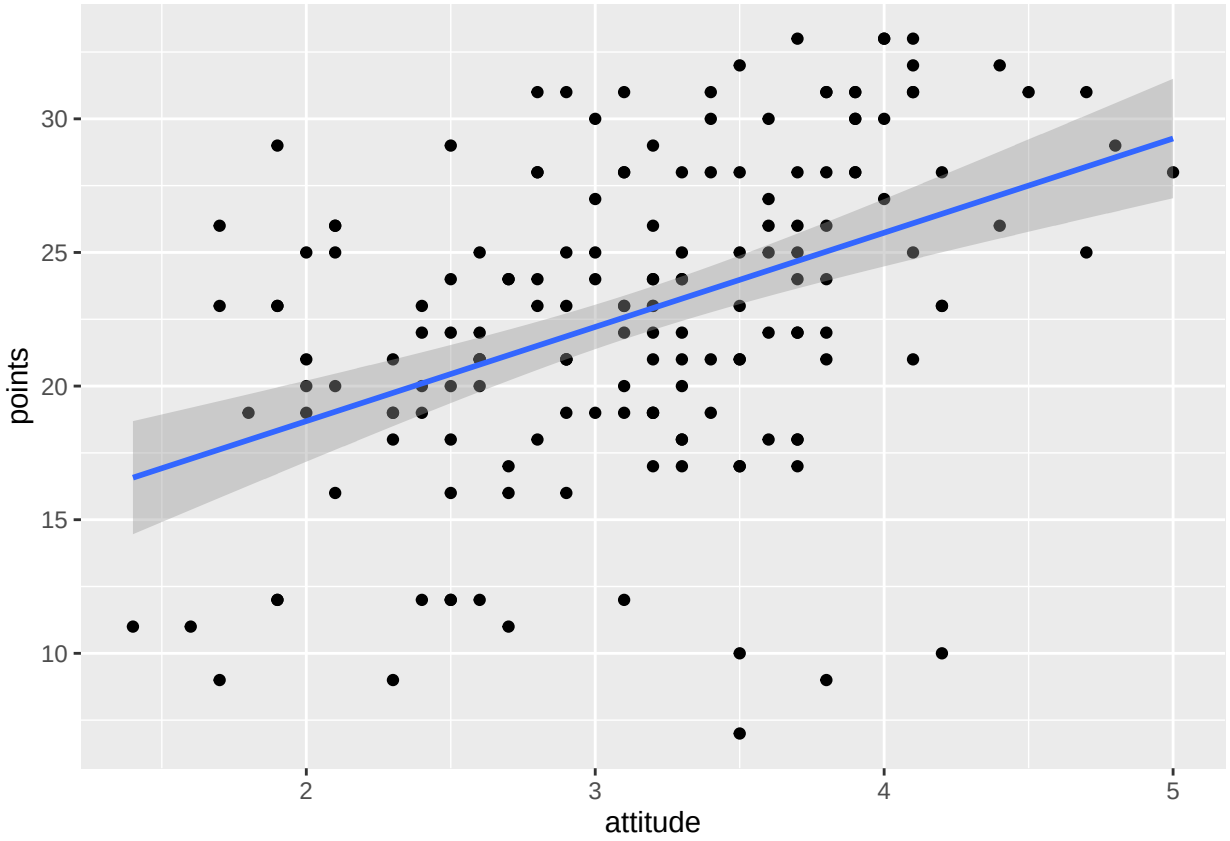
- Print out the summary of the linear model object

Hints: - Replace `1` with the name of the explanatory variable in the formula inside `lm()` - Use `summary()` on the model object to print out a summary

R code

```
# Work with the exercise in this chunk, step-by-step. Fix the R code!
# learning2014 is available

# a scatter plot of points versus attitude
library(ggplot2)
qplot(attitude, points, data = learning2014) + geom_smooth(method = "lm")
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
# fit a linear model
my_model <- lm(points ~ 1, data = learning2014)

# print out a summary of the model
summary(my_model)
```

```
##
## Call:
## lm(formula = points ~ 1, data = learning2014)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.7169  -3.7169   0.2831   5.0331  10.2831
```

```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  22.7169     0.4575   49.65   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.895 on 165 degrees of freedom
```

2.10 Multiple regression

When there are more than one explanatory variables in the linear model, it is called multiple regression. In R, it is easy to include more than one explanatory variables in your linear model. This is done by simply defining more explanatory variables with the `formula` argument of `lm()`, as below

```
y ~ x1 + x2 + ..
```

Here `y` is again the target variable and `x1, x2, ..` are the explanatory variables.

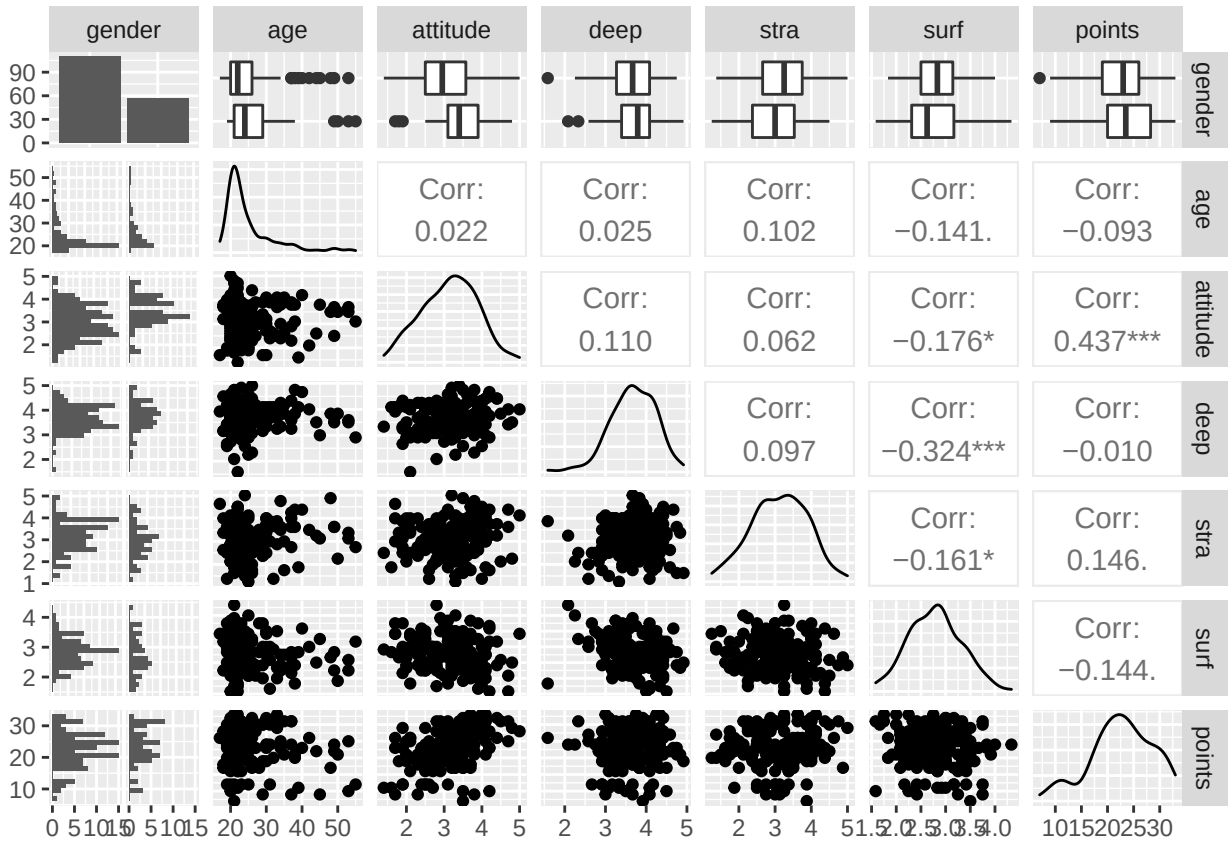
Instructions

- Draw a plot matrix of the learning2014 data with `ggpairs()`
- Fit a regression model where `points` is the target variable and both `attitude` and `stra` are the explanatory variables.
- Print out a summary of the model.
- Adjust the code: Add one more explanatory variable to the model. Based on the plot matrix, choose the variable with the third highest (absolute) correlation with the target variable and use that as the third variable.
- Print out a summary of the new model.

Hint: - The variable with the third highest absolute correlation with `points` is `surf`.

R code

```
# Work with the exercise in this chunk, step-by-step. Fix the R code!
# learning2014 is available

library(GGally)
library(ggplot2)
# create an plot matrix with ggpairs()
ggpairs(learning2014, lower = list(combo = wrap("facethist", bins = 20)))
```




```
# create a regression model with multiple explanatory variables
my_model2 <- lm(points ~ attitude + stra, data = learning2014)

# print out a summary of the model
summary(my_model2)
```

```
##
## Call:
## lm(formula = points ~ attitude + stra, data = learning2014)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -17.6436  -3.3113   0.5575   3.7928  10.9295
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   8.9729     2.3959   3.745  0.00025 ***
## attitude      3.4658     0.5652   6.132 6.31e-09 ***
## stra          0.9137     0.5345   1.709  0.08927 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.289 on 163 degrees of freedom
## Multiple R-squared:  0.2048, Adjusted R-squared:  0.1951
## F-statistic: 20.99 on 2 and 163 DF,  p-value: 7.734e-09
```

2.11 Graphical model validation

R makes it easy to graphically explore the validity of your model assumptions. If you give a linear model object as the first argument to the `plot()` function, the function automatically assumes you want diagnostic plots and will produce them. You can check the help page of plotting an lm object by typing `?plot.lm` or `help(plot.lm)` to the R console.

In the plot function you can then use the argument `which` to choose which plots you want. `which` must be an integer vector corresponding to the following list of plots:

which	graphic
1	Residuals vs Fitted values
2	Normal QQ-plot
3	Standardized residuals vs Fitted values
4	Cook's distances
5	Residuals vs Leverage
6	Cook's distance vs Leverage

We will focus on plots 1, 2 and 5: Residuals vs Fitted values, Normal QQ-plot and Residuals vs Leverage.

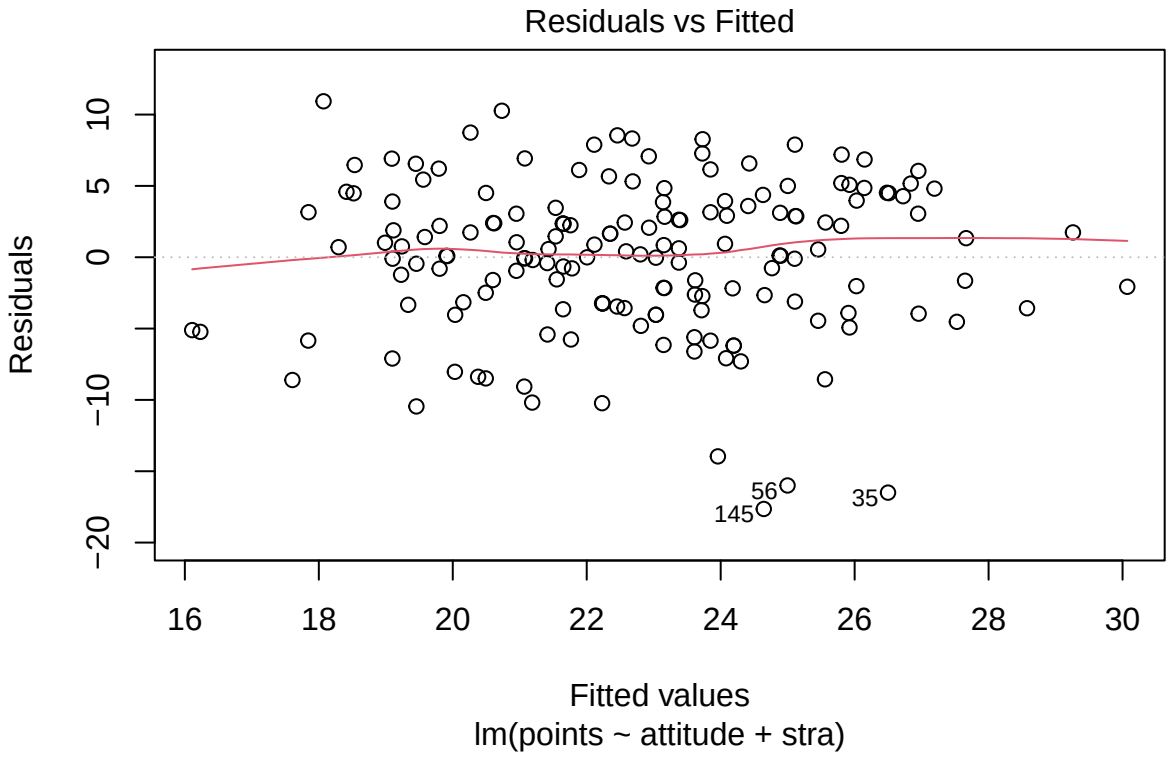
Instructions

- Create the linear model object `my_model2`
- Produce the following diagnostic plots using the `plot()` function: Residuals vs Fitted values, Normal QQ-plot and Residuals vs Leverage using the argument `which`.
- Before the call to the `plot()` function, add the following: `par(mfrow = c(2,2))`. This will place the following 4 graphics to the same plot. Execute the code again to see the effect.

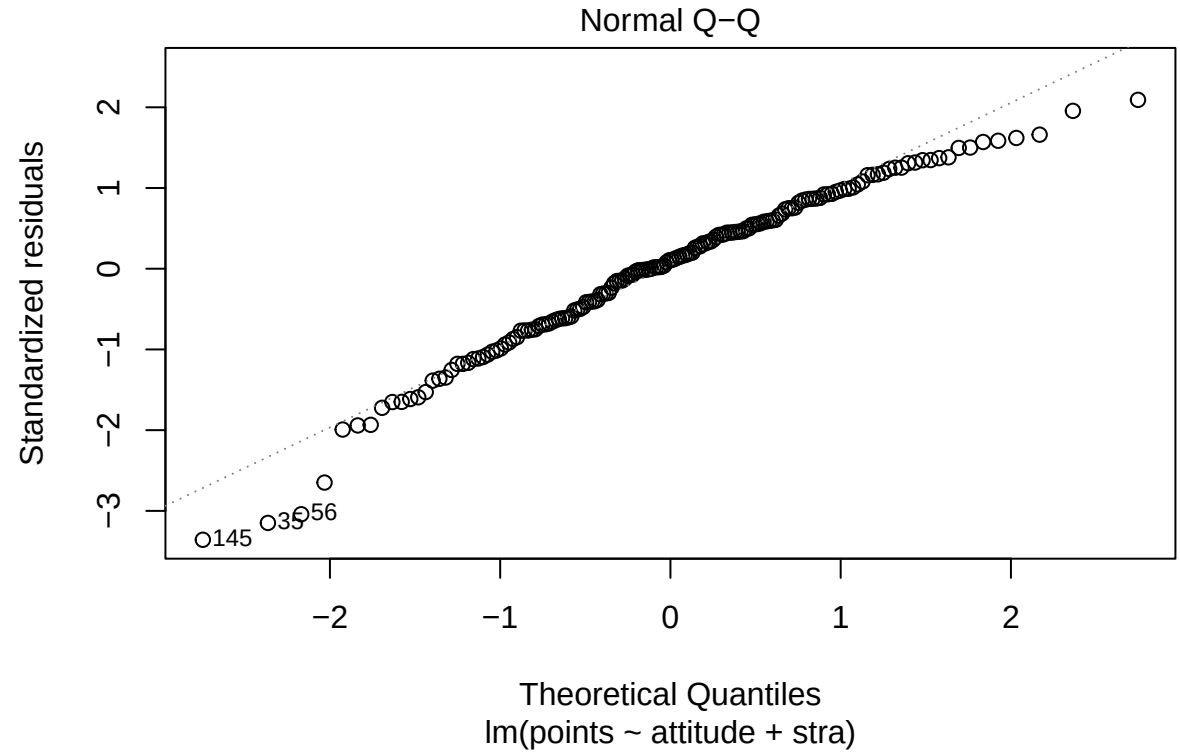
Hint: - You can combine integers to an integer vector with `c()`. For example: `c(1,2,3)`.

R code

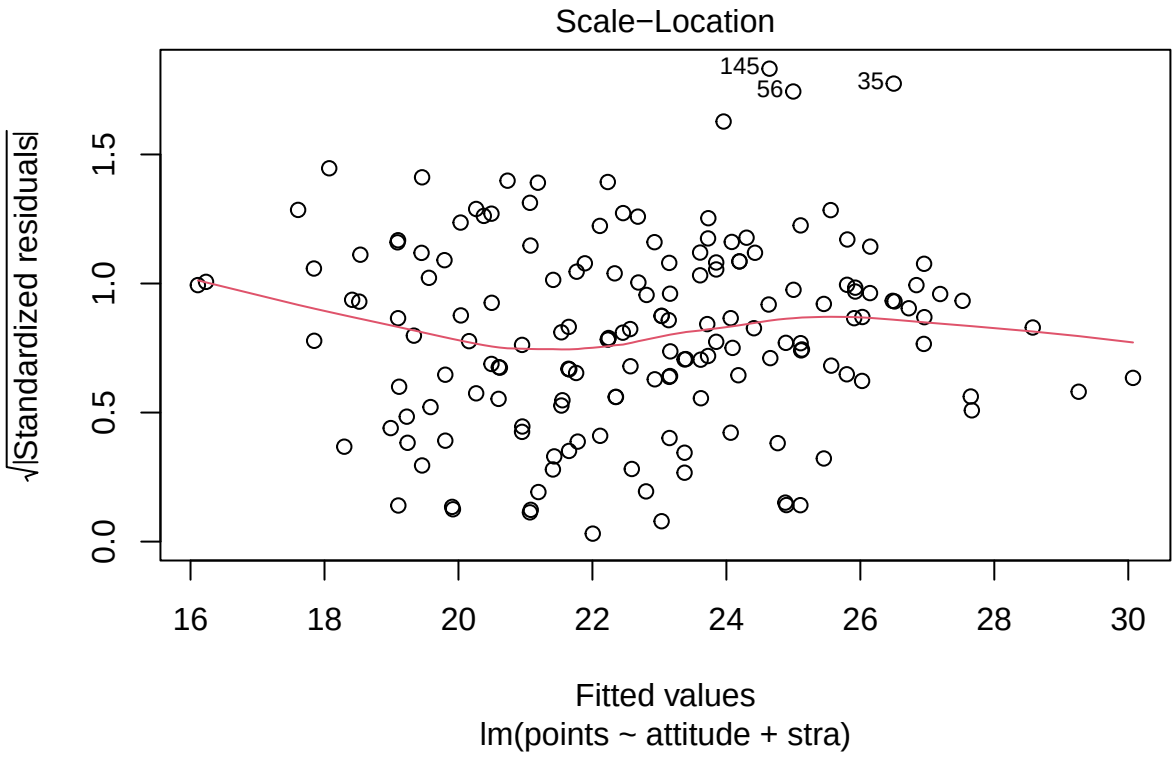
```
# Work with the exercise in this chunk, step-by-step. Fix the R code!
# learning2014 is available

# create a regression model with multiple explanatory variables
my_model2 <- lm(points ~ attitude + stra, data = learning2014)

# draw diagnostic plots using the plot() function. Choose the plots 1, 2 and 5
plot(my_model2, which = 1)
```



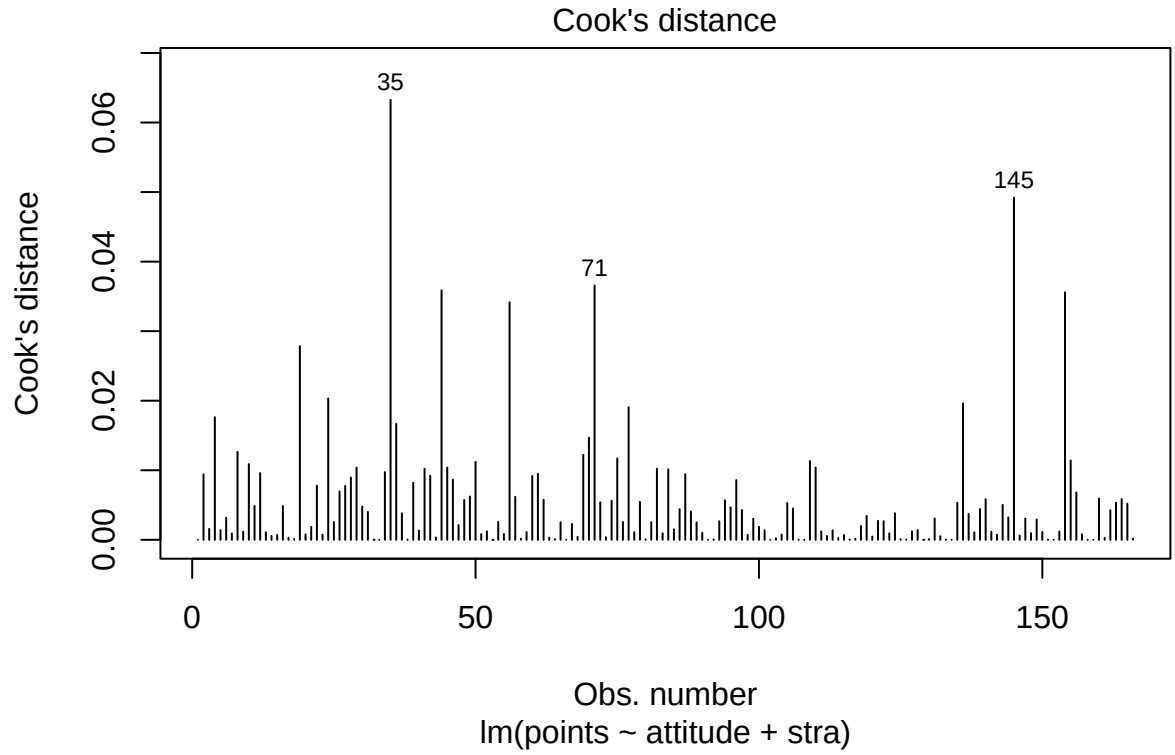

```
plot(my_model2, which = 2)
```



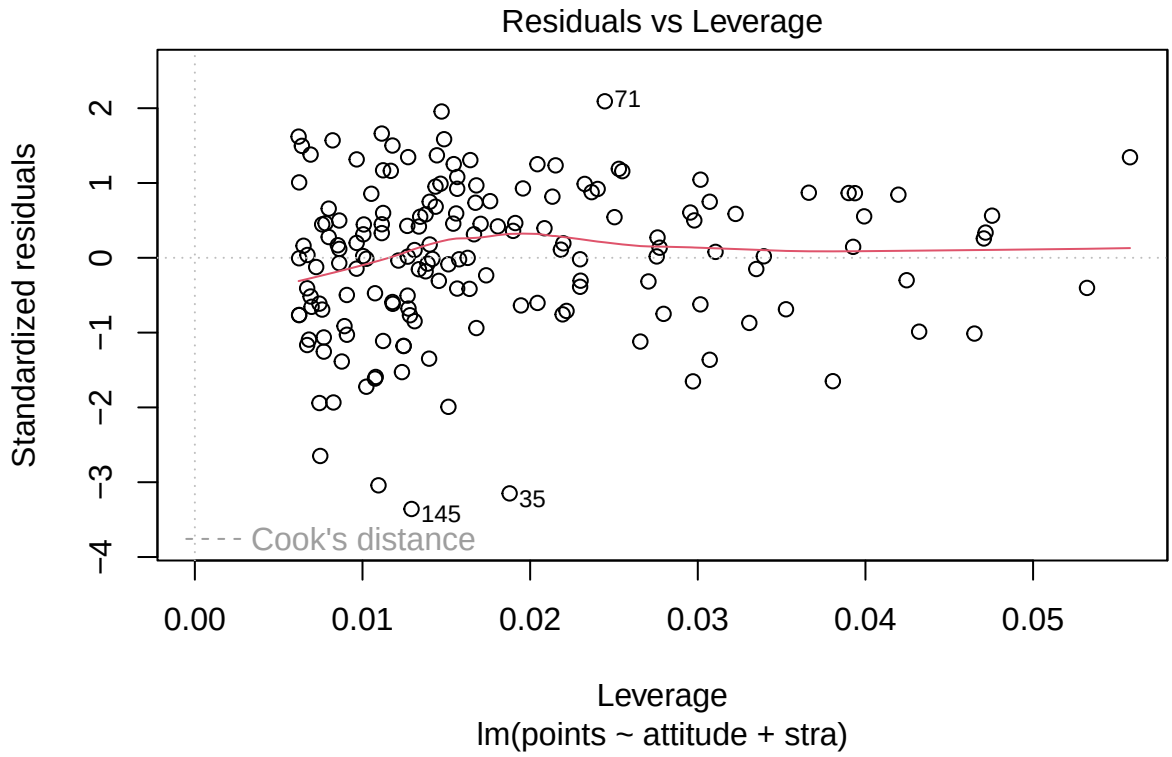

```
plot(my_model2, which = 3)
```



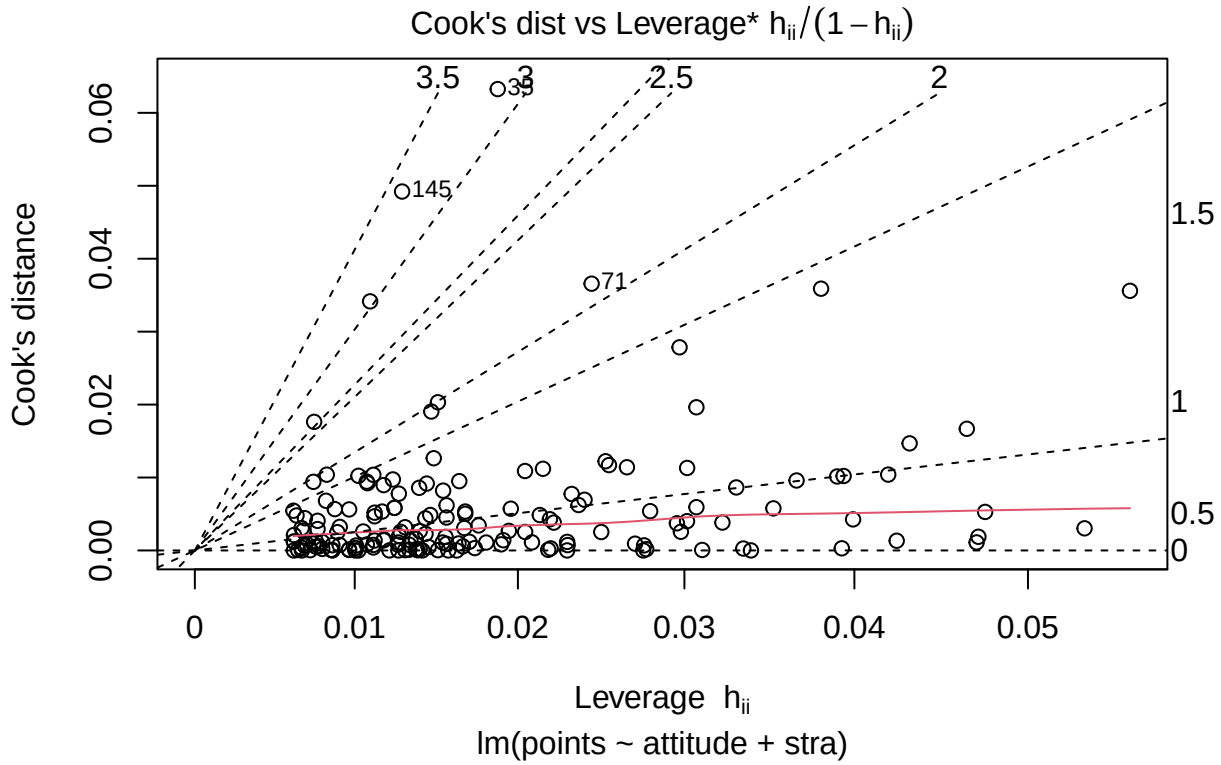

```
plot(my_model2, which = 4)
```




```
plot(my_model2, which = 5)
```




```
plot(my_model2, which = 6)
```



2.12 Making predictions

Okay, so let's assume that we have a linear model which seems to fit our standards. What can we do with it?

The model quantifies the relationship between the explanatory variable(s) and the dependent variable. The model can also be used for predicting the dependent variable based on new observations of the explanatory variable(s).

In R, predicting can be done using the `predict()` function. (see `?predict`). The first argument of predict is a model object and the argument `newdata` (a data.frame) can be used to make predictions based on new observations. One or more columns of `newdata` should have the same name as the explanatory variables in the model object.

Instructions

- Create object `m` and print out a summary of the model
- Create object `new_attitudes`
- Adjust the code: Create a new data frame with a column named 'attitude' holding the new attitudes defined in `new_attitudes`
- Print out the new data frame
- `predict()` the new student's exam points based on their attitudes, using the `newdata` argument

Hints: - Type `attitude = new_attitudes` inside the `data.frame()` function. - Give the `new_data` data.frame as the `newdata` argument for `predict()`

R code

```
# Work with the exercise in this chunk, step-by-step. Fix the R code!
# learning2014 is available

# Create model object m
m <- lm(points ~ attitude, data = learning2014)

# print out a summary of the model
summary(m)
```

```
##
## Call:
## lm(formula = points ~ attitude, data = learning2014)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.9763  -3.2119   0.4339   4.1534  10.6645
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  11.6372     1.8303   6.358 1.95e-09 ***
## attitude      3.5255     0.5674   6.214 4.12e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.32 on 164 degrees of freedom
## Multiple R-squared:  0.1906, Adjusted R-squared:  0.1856
## F-statistic: 38.61 on 1 and 164 DF,  p-value: 4.119e-09
```

```
# New observations
new_attitudes <- c("Mia" = 3.8, "Mike"= 4.4, "Riikka" = 2.2, "Pekka" = 2.9)
new_data <- data.frame(attitude = new_attitudes)
```

```
# Print out the new data
summary(new_data)
```

```
##     attitude
##  Min.   :2.200
##  1st Qu.:2.725
##  Median :3.350
##  Mean   :3.325
##  3rd Qu.:3.950
##  Max.   :4.400
```

```
# Predict the new students exam points based on attitude
predict(m, newdata = new_data)
```

```
##      Mia     Mike   Riikka    Pekka
## 25.03390 27.14918 19.39317 21.86099
```

Week 3: Logistic Regression

3.0 Loading packages

```
library("boot")
library("readr")
```

3.1 Data wrangling

Data wrangling completed using the UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets.html). csv files downloaded from the repository and final data set "alc.csv" has been exported into the project folder. Rcodes in the learning diary has been updated.

3.1 Creating new R Markdown file

New Rmd file has been created with tittle "chapter3" and now saved in the project folder.

3.2 Importing the data set and exploration

```
library(tidyverse)
stu_alc2014 <- read_csv("alc.csv" , show_col_types= FALSE)
spec(stu_alc2014)
```

```
## cols(
##   school = col_character(),
##   sex = col_character(),
##   age = col_double(),
##   address = col_character(),
##   famsize = col_character(),
##   Pstatus = col_character(),
##   Medu = col_double(),
##   Fedu = col_double(),
##   Mjob = col_character(),
##   Fjob = col_character(),
##   reason = col_character(),
```

```
##   guardian = col_character(),
##   traveltime = col_double(),
##   studytime = col_double(),
##   schoolsup = col_character(),
##   famsup = col_character(),
##   activities = col_character(),
##   nursery = col_character(),
##   higher = col_character(),
##   internet = col_character(),
##   romantic = col_character(),
##   famrel = col_double(),
##   freetime = col_double(),
##   goout = col_double(),
##   Dalc = col_double(),
##   Walc = col_double(),
##   health = col_double(),
##   failures = col_double(),
##   paid = col_character(),
##   absences = col_double(),
##   G1 = col_double(),
##   G2 = col_double(),
##   G3 = col_double(),
##   alc_use = col_double(),
##   high_use = col_logical()
## )
```

```
#looking at the data
dim(stu_alc2014)
```

```
## [1] 370  35
```

```
colnames(stu_alc2014)
```

```
##  [1] "school"     "sex"        "age"        "address"    "famsize"
##  [6] "Pstatus"    "Medu"       "Fedu"       "Mjob"       "Fjob"
## [11] "reason"     "guardian"   "traveltime" "studytime"  "schoolsup"
## [16] "famsup"     "activities" "nursery"    "higher"     "internet"
## [21] "romantic"   "famrel"     "freetime"   "goout"      "Dalc"
## [26] "Walc"       "health"     "failures"   "paid"       "absences"
## [31] "G1"         "G2"         "G3"         "alc_use"    "high_use"
```

```
glimpse(stu_alc2014)
```

```
## Rows: 370
## Columns: 35
## $ school     <chr> "GP", "GP", "GP", "GP", "GP", "GP", "GP", "GP", "GP", "GP",~
## $ sex        <chr> "F", "F", "F", "F", "F", "M", "M", "F", "M", "M", "F", "F",~
## $ age        <dbl> 18, 17, 15, 15, 16, 16, 16, 17, 15, 15, 15, 15, 15, 15, 15,~
## $ address    <chr> "U", "U", "U", "U", "U", "U", "U", "U", "U", "U", "U", "U",~
## $ famsize    <chr> "GT3", "GT3", "LE3", "GT3", "GT3", "LE3", "LE3", "GT3", "LE~
## $ Pstatus    <chr> "A", "T", "T", "T", "T", "T", "T", "A", "A", "T", "T", "T",~
## $ Medu       <dbl> 4, 1, 1, 4, 3, 4, 2, 4, 3, 3, 4, 2, 4, 4, 2, 4, 4, 3, 3, 4,~
## $ Fedu       <dbl> 4, 1, 1, 2, 3, 3, 2, 4, 2, 4, 4, 1, 4, 3, 2, 4, 4, 3, 2, 3,~
## $ Mjob       <chr> "at_home", "at_home", "at_home", "health", "other", "servic~
## $ Fjob       <chr> "teacher", "other", "other", "services", "other", "other", ~
## $ reason     <chr> "course", "course", "other", "home", "home", "reputation", ~
```

```
## $ guardian   <chr> "mother", "father", "mother", "mother", "father", "mother",~
## $ traveltime <dbl> 2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 3, 1, 2, 1, 1, 1, 3, 1, 1,~
## $ studytime  <dbl> 2, 2, 2, 3, 2, 2, 2, 2, 2, 2, 2, 3, 1, 2, 3, 1, 3, 2, 1, 1,~
## $ schoolsup  <chr> "yes", "no", "yes", "no", "no", "no", "no", "yes", "no", "n~
## $ famsup     <chr> "no", "yes", "no", "yes", "yes", "yes", "no", "yes", "yes",~
## $ activities <chr> "no", "no", "no", "yes", "no", "yes", "no", "no", "no", "ye~
## $ nursery    <chr> "yes", "no", "yes", "yes", "yes", "yes", "yes", "yes", "yes~
## $ higher     <chr> "yes", "yes", "yes", "yes", "yes", "yes", "yes", "yes", "ye~
## $ internet   <chr> "no", "yes", "yes", "yes", "no", "yes", "yes", "no", "yes",~
## $ romantic   <chr> "no", "no", "no", "yes", "no", "no", "no", "no", "no", "no"~
## $ famrel     <dbl> 4, 5, 4, 3, 4, 5, 4, 4, 4, 5, 3, 5, 4, 5, 4, 4, 3, 5, 5, 3,~
## $ freetime   <dbl> 3, 3, 3, 2, 3, 4, 4, 1, 2, 5, 3, 2, 3, 4, 5, 4, 2, 3, 5, 1,~
## $ goout      <dbl> 4, 3, 2, 2, 2, 2, 4, 4, 2, 1, 3, 2, 3, 3, 2, 4, 3, 2, 5, 3,~
## $ Dalc       <dbl> 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1,~
## $ Walc       <dbl> 1, 1, 3, 1, 2, 2, 1, 1, 1, 1, 2, 1, 3, 2, 1, 2, 2, 1, 4, 3,~
## $ health     <dbl> 3, 3, 3, 5, 5, 5, 3, 1, 1, 5, 2, 4, 5, 3, 3, 2, 2, 4, 5, 5,~
## $ failures   <dbl> 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0,~
## $ paid       <chr> "no", "no", "yes", "yes", "yes", "yes", "no", "no", "yes", ~
## $ absences   <dbl> 5, 3, 8, 1, 2, 8, 0, 4, 0, 0, 1, 2, 1, 1, 0, 5, 8, 3, 9, 5,~
## $ G1         <dbl> 2, 7, 10, 14, 8, 14, 12, 8, 16, 13, 12, 10, 13, 11, 14, 16,~
## $ G2         <dbl> 8, 8, 10, 14, 12, 14, 12, 9, 17, 14, 11, 12, 14, 11, 15, 16~
## $ G3         <dbl> 8, 8, 11, 14, 12, 14, 12, 10, 18, 14, 12, 12, 13, 12, 16, 1~
## $ alc_use    <dbl> 1.0, 1.0, 2.5, 1.0, 1.5, 1.5, 1.0, 1.0, 1.0, 1.0, 1.5, 1.0,~
## $ high_use   <lgl> FALSE, FALSE, TRUE, FALSE, FALSE, FALSE, FALSE, FALSE, FALS~
```

3.3 Choosing the variables

```
stu_alc2014_2 <- select(stu_alc2014, absences, G3, age, freetime, high_use)
str(stu_alc2014_2)
```

```
## tibble [370 x 5] (S3: tbl_df/tbl/data.frame)
##  $ absences: num [1:370] 5 3 8 1 2 8 0 4 0 0 ...
##  $ G3      : num [1:370] 8 8 11 14 12 14 12 10 18 14 ...
##  $ age     : num [1:370] 18 17 15 15 16 16 16 17 15 15 ...
##  $ freetime: num [1:370] 3 3 3 2 3 4 4 1 2 5 ...
##  $ high_use: logi [1:370] FALSE FALSE TRUE FALSE FALSE FALSE ...
```

```
colnames(stu_alc2014_2)
```

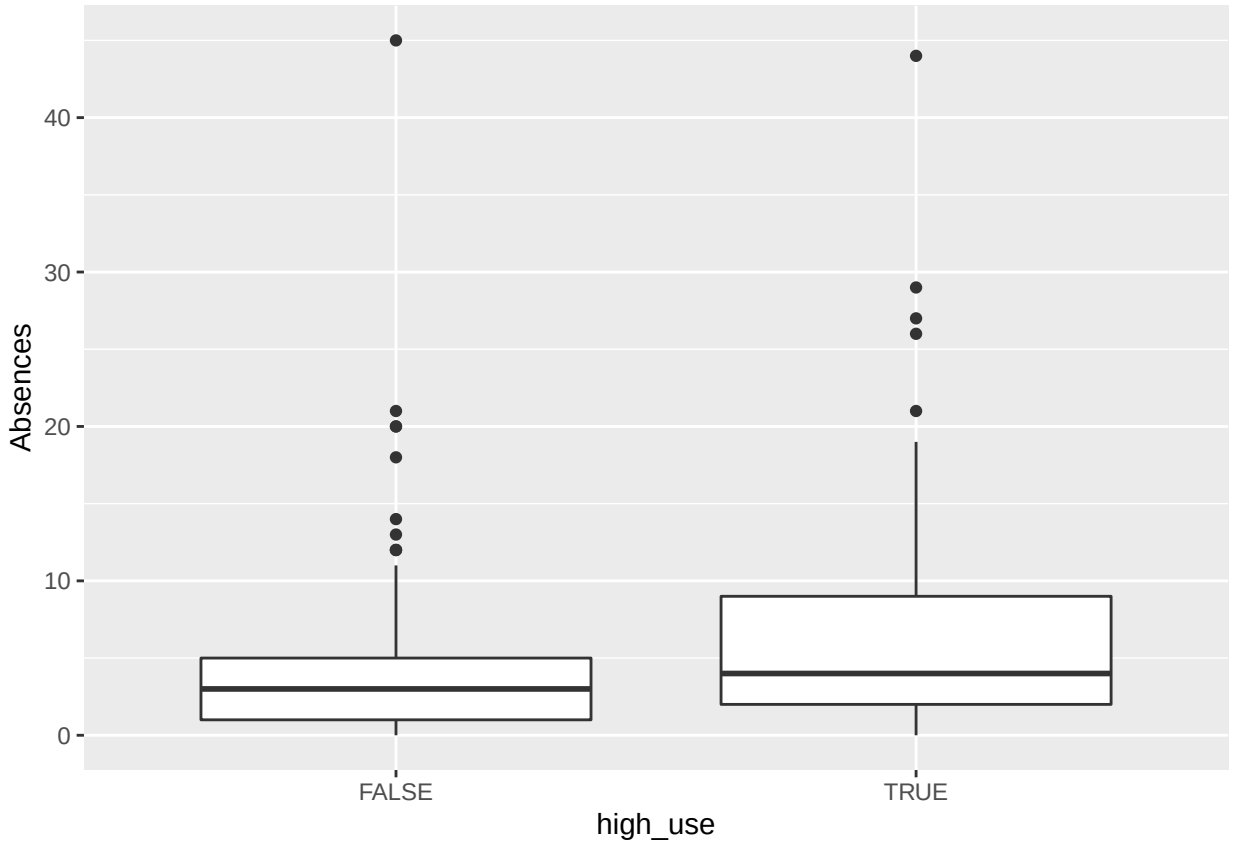
```
## [1] "absences" "G3"       "age"      "freetime" "high_use"
```

In above data set we have 370 observations and 35 variables. These data set belong to a survey from questionnaire in two secondary school from Portugal in which different variables, demographics and socio-economic features measures students association with alcohol consumption. Here I choose 4 interesting variable that I think has greater influence in the level of alcohol consumption among these school kids. My rationale behind choosing these variables is that certain age groups have higher access to alcohol, ages above 16 lets say can access alcohol easily than ages below 16 so I wish to see the relation here. Also free time activities and amount of free time influences alcohol consumption. Likewise final grade and absences can directly correlate with higher use. So I wish to test my model with these variables.

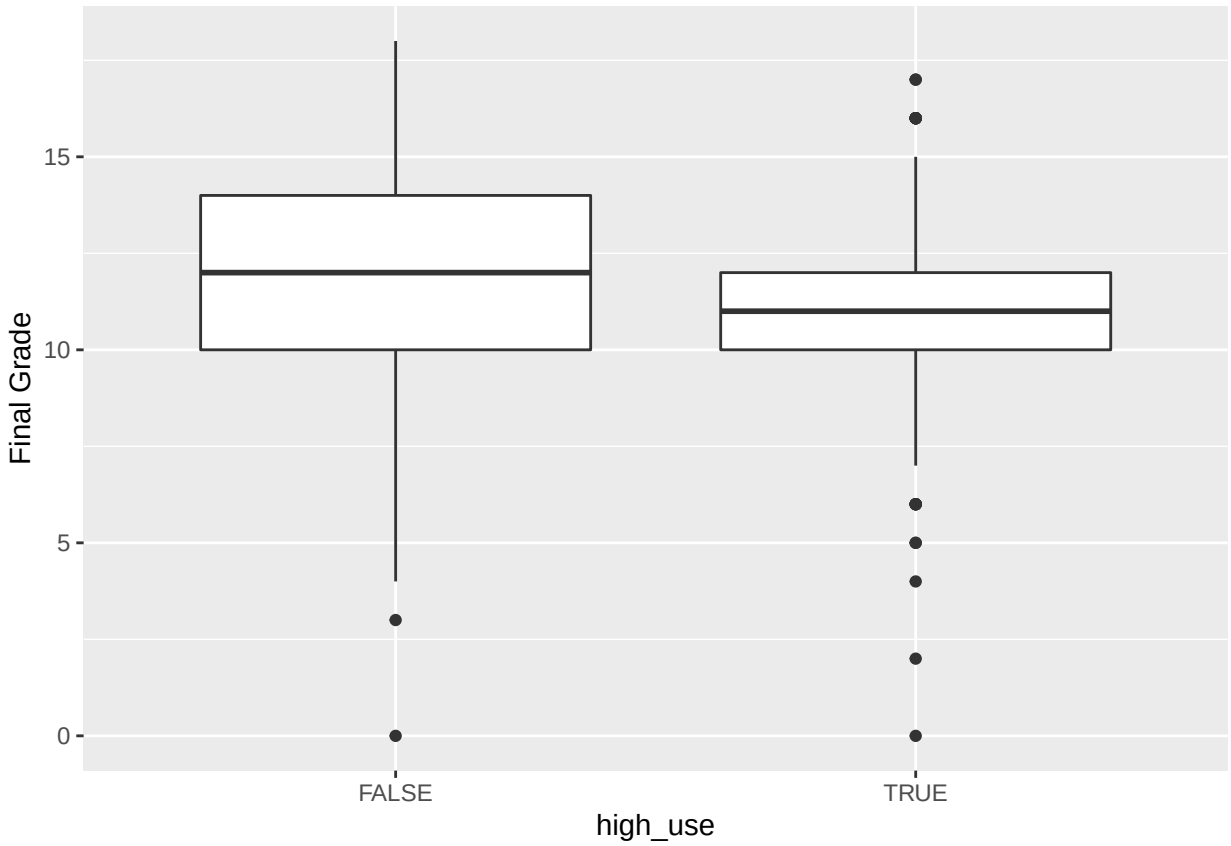
3.4 Exploring and plotting the choosen variable

```
#Let's explore the choose variables using box plots

##Let's see for absences
```

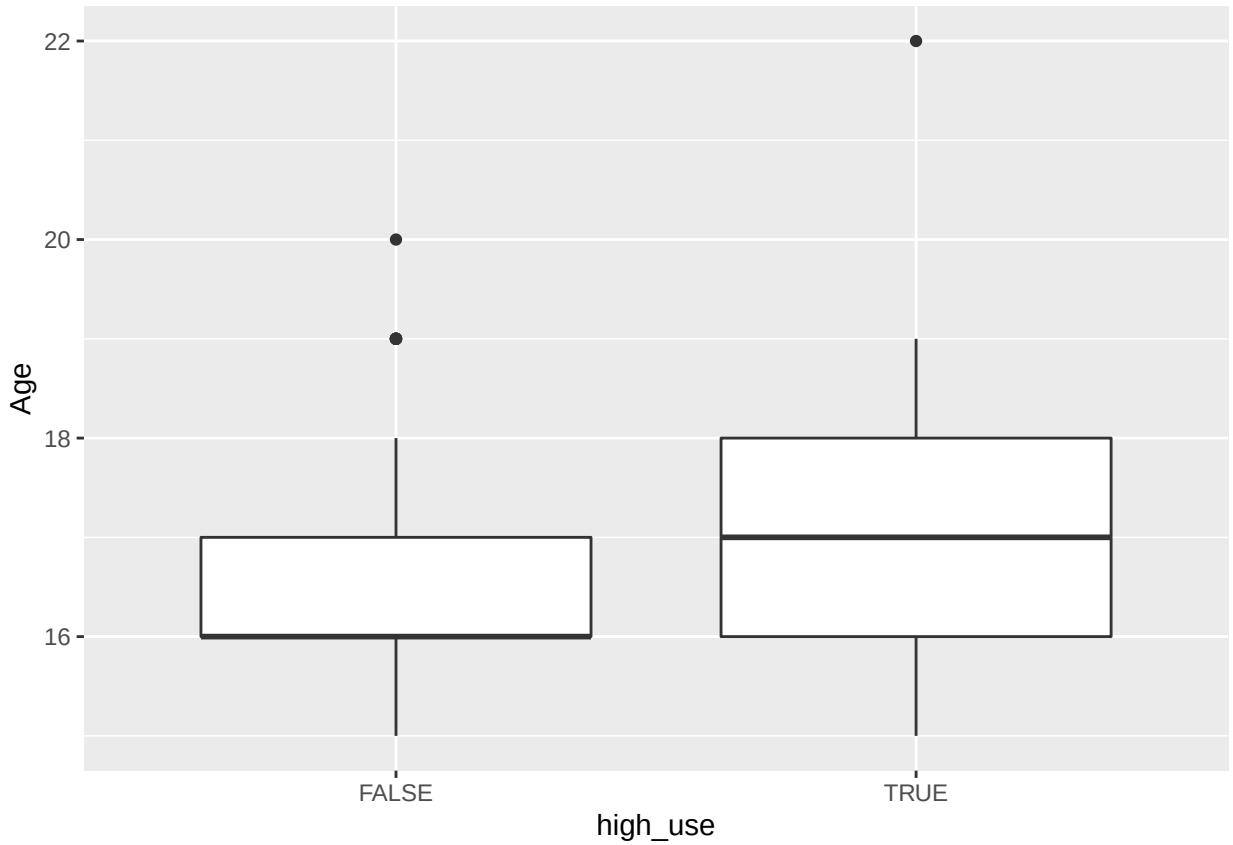
```
g1 <- ggplot(stu_alc2014, aes(x = high_use, y = absences))
g1 + geom_boxplot() + ylab("Absences")
```



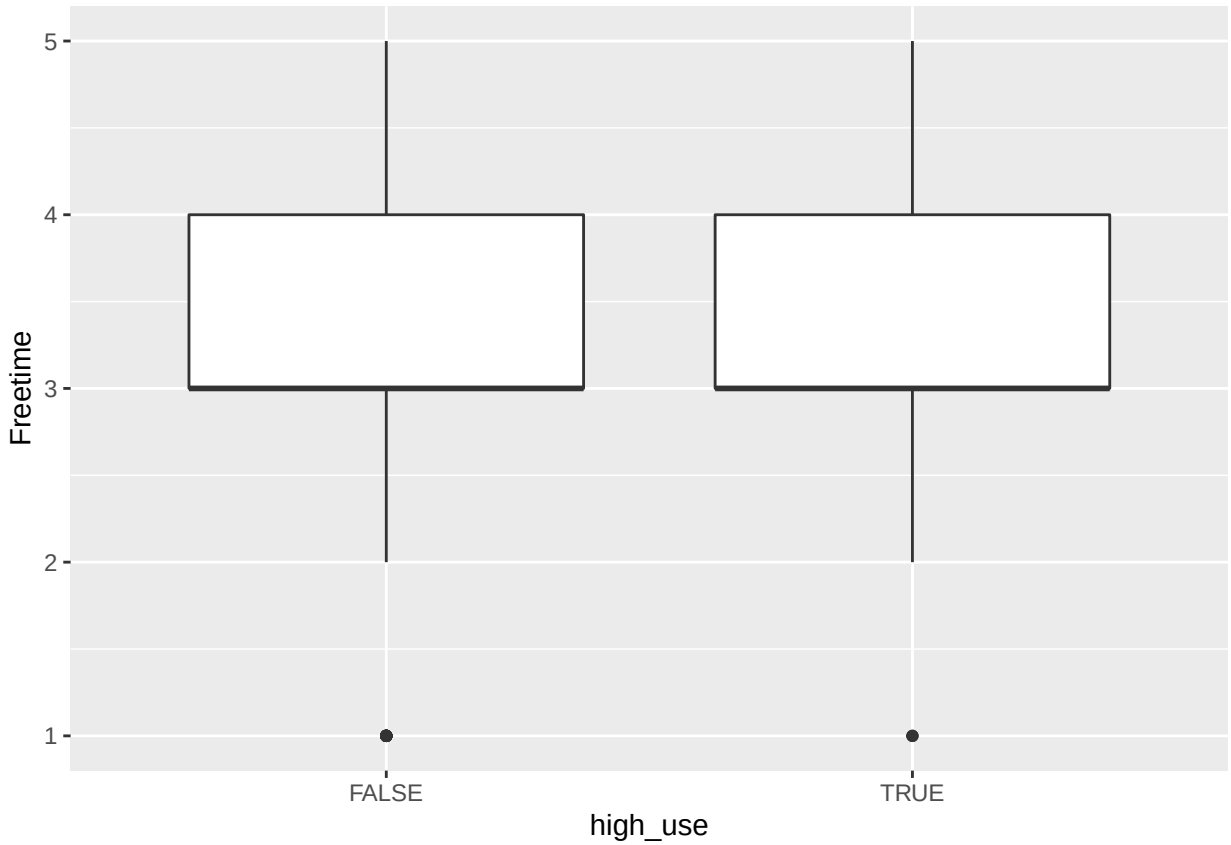

```
##Let's see for G3(Final Grade)
g1 <- ggplot(stu_alc2014, aes(x = high_use, y = G3))
g1 + geom_boxplot() + ylab("Final Grade")
```




```
##Let's see for age
g1 <- ggplot(stu_alc2014, aes(x = high_use, y = age))
g1 + geom_boxplot() + ylab("Age")
```




```
##And for freetime
g1 <- ggplot(stu_alc2014, aes(x = high_use, y = freetime))
g1 + geom_boxplot() + ylab("Freetime")
```



General overview from the plot infers some association between high alcohol use and absence and also age (holds true). It would be interesting to fit this kind of model to see the relationship. Final grade and alcohol consumption shows some association but there appears to be some difference in their mean for true and false.Free time doesn't seem to have much effect on alcohol consumption.

3.4 Logistic regression

```
##Lets call this model-1 (m1) which explores 4 variable
m1 <- glm(high_use ~ absences + G3 + age + freetime, data = stu_alc2014_2, family = "binomial")
summary(m1)
```

```
##
## Call:
## glm(formula = high_use ~ absences + G3 + age + freetime, family = "binomial",
##     data = stu_alc2014_2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0110  -0.8181  -0.6584   1.1345   2.0343
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.80128    1.87360  -2.029 0.042472 *
## absences     0.08428    0.02301   3.663 0.000249 ***
## G3          -0.06623    0.03666  -1.807 0.070838 .
## age          0.11958    0.10233   1.169 0.242592
## freetime     0.39844    0.12559   3.172 0.001511 **
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 452.04  on 369  degrees of freedom
## Residual deviance: 417.94  on 365  degrees of freedom
## AIC: 427.94
##
## Number of Fisher Scoring iterations: 4
```

This seems to be an interesting outcome. Fitted model seem to match my above hypothesis based on box plot and distribution for some variable, absences for instant is the most significant and free time is second most significant among other variable. Absence has the highest significance (p = 0.000249) and free time is significant with p = 0.001511. Final grade and age however doesn't have the same lever of significance as other two. Final grade has p = 0.05 which can be considered significant result but comparatively, two other variable stands out the most.

3.5 Power of the model

```
coef(m1)
```

```
## (Intercept)    absences          G3         age    freetime
## -3.80128143  0.08428256 -0.06622629  0.11957546  0.39844165
```

```
OR <- coef(m1) %>% exp
CI <- confint(m1) %>% exp
```

```
## Waiting for profiling to be done...
```

```
#Print OR and CI
cbind(OR, CI)
```

```
##                     OR        2.5 %    97.5 %
## (Intercept) 0.02234212 0.0005401794 0.8536065
## absences    1.08793626 1.0421786444 1.1408573
## G3          0.93591905 0.8705458921 1.0055377
## age         1.12701829 0.9228580718 1.3797569
## freetime    1.48950173 1.1689712226 1.9147718
```

Coefficient and Confidence Interval: Looking at the coefficient, final grade has negative value suggesting an opposite association between higher alcohol consumption which makes sense because higher grade can result in lower alcohol consumption. absences, age and free time shows positive association with free time being the most cause of higher alcohol consumption followed by age and absences. This is also supported by the odds ratio and confidence interval for each tested variable.
The above hypothesis therefore holds true for most variable. It is quite obviously important to explore other variable to see effect of an additional variable on an outcome through multivariate analysis.

3.6 Cross validation (Bonus)

3.7 Cross validation of different model (Super-Bonus)

Week 4: Clustering and Classification

The topics of this chapter - clustering and classification - are handy and visual tools of exploring statistical data. Clustering means that some points (or observations) of the data are in some sense closer to each other than some other points. In other words, the data points do not comprise a homogeneous sample, but instead, it is somehow clustered.

In general, the clustering methods try to find these clusters (or groups) from the data. One of the most typical clustering methods is called k-means clustering. Also hierarchical clustering methods quite popular, giving tree-like dendrograms as their main output.

As such, clusters are easy to find, but what might be the "right" number of clusters? It is not always clear. And how to give these clusters names and interpretations?

Based on a successful clustering, we may try to classify new observations to these clusters and hence validate the results of clustering. Another way is to use various forms of discriminant analysis, which operates with the (now) known clusters, asking: "what makes the difference(s) between these groups (clusters)?"

In the connection of these methods, we also discuss the topic of distance (or dissimilarity or similarity) measures. There are lots of other measures than just the ordinary Euclidean distance, although it is one of the most important ones. Several discrete and even binary measures exist and are widely used for different purposes in various disciplines.

4.0 Packages for clustering and classification

```
library(tidyverse)
library(GGally)
library(dplyr)
library(ggplot2)
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

4.1 loading data and Exploring the data

```
#Loading the Boston data from the MASS package
data("Boston")

# Exploring the structure and the dimensions of the data set

str(Boston)
```

```
## 'data.frame':    506 obs. of  14 variables:
##  $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
```

```
##  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
##  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ black  : num  397 397 393 395 397 ...
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

```
dim(Boston)
```

```
## [1] 506  14
```

The data set "Boston" from MASS library presents geographical, demographic, structural, economic and cultural description of different suburbs in Boston and their implication on housing values within different suburbs. Data set contains 14 variables (factors) and 506 observation and most variables are numerical.

The variables in the data set which influences the housing prices are:

crim = per capita crime rate by town. zn = proportion of residential land zoned for lots over 25,000 sq.ft. indus = proportion of non-retail business acres per town. chas = Charles River dummy variable (= 1 if tract bounds river; 0 otherwise). nox = nitrogen oxides concentration (parts per 10 million). rm = average number of rooms per dwelling. age = proportion of owner-occupied units built prior to 1940. dis = weighted mean of distances to five Boston employment centres. rad = index of accessibility to radial highways. tax = full-value property-tax rate per $10,000. ptratio = pupil-teacher ratio by town. black = 1000(Bk - 0.63)^2 where BkBk is the proportion of blacks by town. lstat = lower status of the population (percent). medv = median value of owner-occupied homes in $1000s.

Data is sourced from Harrison, D. and Rubinfeld, D.L. (1978) Hedonic prices and the demand for clean air. J. Environ. Economics and Management 5, 81–102. Belsley D.A., Kuh, E. and Welsch, R.E. (1980) Regression Diagnostics. Identifying Influential Data and Sources of Collinearity. New York: Wiley.

For more information about the "Boston" data set follow the link https://stat.ethz.ch/R-manual/R-devel/library/MASS/html/Boston.html

4.2 Graphical Overview of the data set

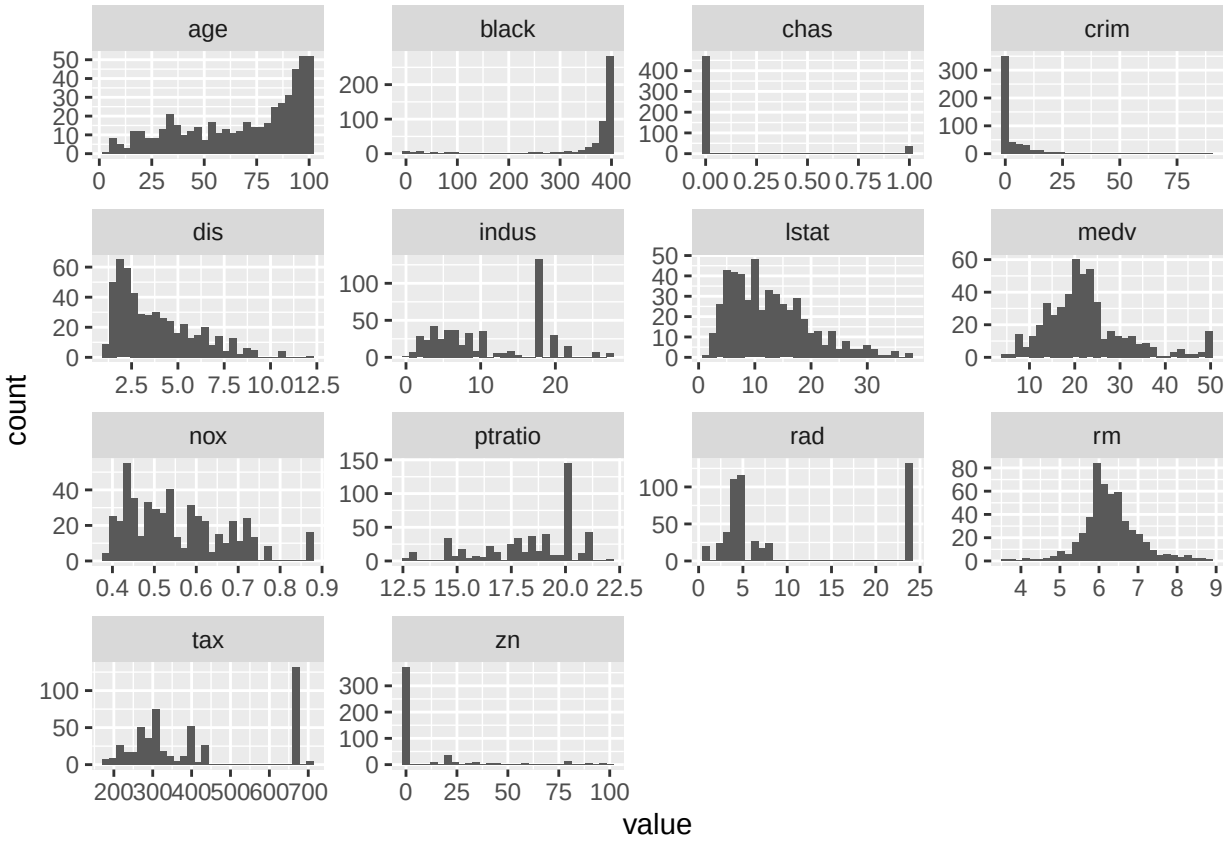
```
#overview
summary(Boston)
```

```
##       crim                zn              indus            chas        
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000  
##  1st Qu.: 0.08205   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000  
##  Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000  
##  Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.06917  
##  3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000  
##  Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000  
##       nox               rm              age              dis        
##  Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130  
##  1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100  
##  Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207  
##  Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795  
##  3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188  
##  Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127  
##       rad              tax           ptratio          black       
##  Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32  
```

```
##  1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38
##  Median : 5.000   Median :330.0   Median :19.05   Median :391.44
##  Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67
##  3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23
##  Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90
##      lstat            medv
##  Min.   : 1.73   Min.   : 5.00
##  1st Qu.: 6.95   1st Qu.:17.02
##  Median :11.36   Median :21.20
##  Mean   :12.65   Mean   :22.53
##  3rd Qu.:16.95   3rd Qu.:25.00
##  Max.   :37.97   Max.   :50.00
```

```
#Plotting
long_Boston <- pivot_longer(Boston, cols=everything(), names_to = 'variable', values_to = 'value')
p1 <- ggplot(data=long_Boston)
p1 + geom_histogram(mapping= aes(x=value)) + facet_wrap(~variable, scales="free")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

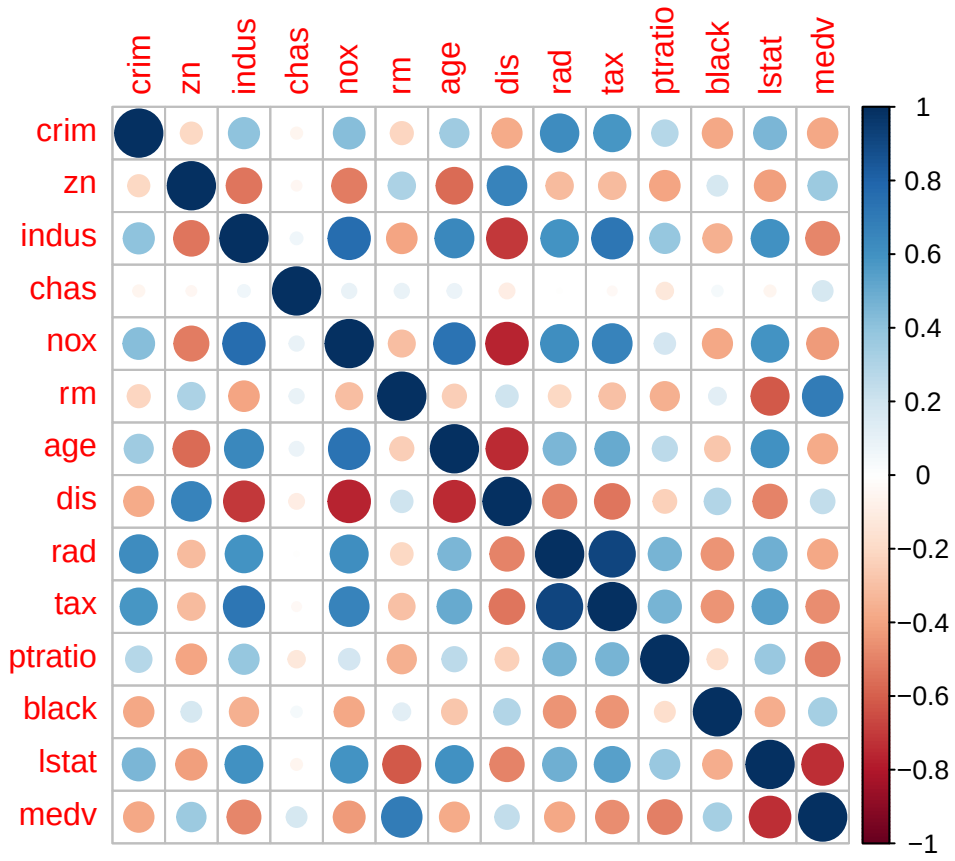



```
#Relationship between the variables
cor_matrix <- cor(Boston)
cor_matrix
```

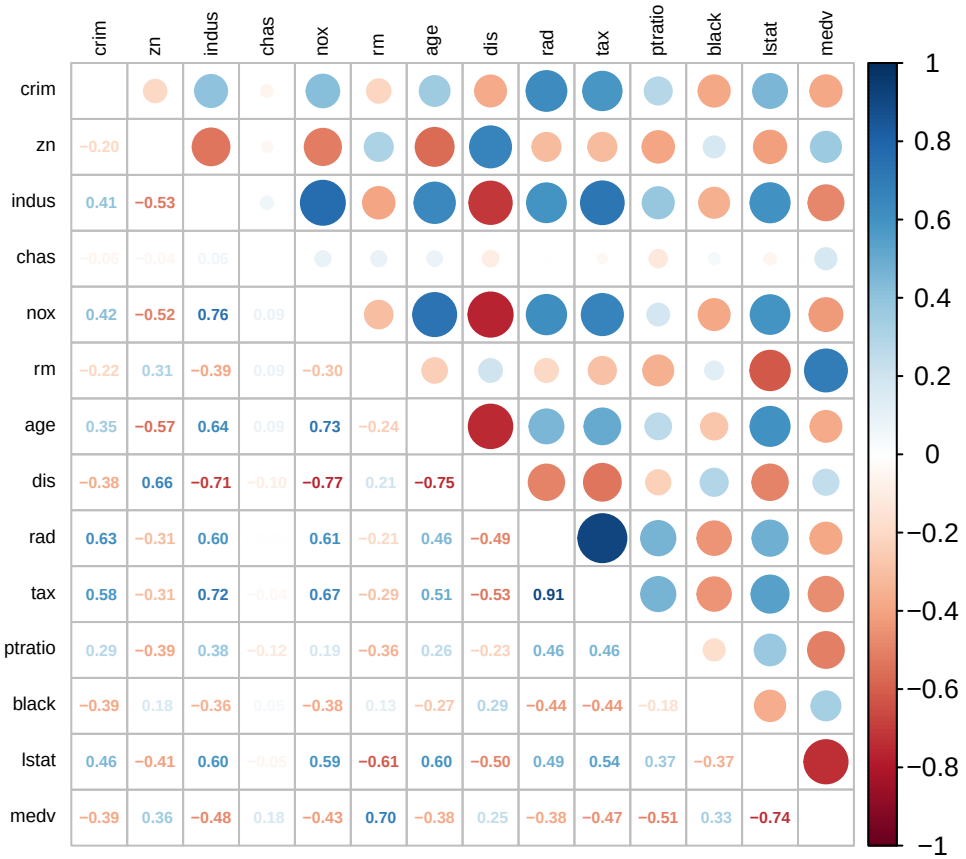
```
##                crim          zn       indus         chas         nox
## crim     1.00000000 -0.20046922  0.40658341 -0.055891582  0.42097171
## zn      -0.20046922  1.00000000 -0.53382819 -0.042696719 -0.51660371
## indus    0.40658341 -0.53382819  1.00000000  0.062938027  0.76365145
## chas    -0.05589158 -0.04269672  0.06293803  1.000000000  0.09120281
```

```
## nox      0.42097171 -0.51660371  0.76365145  0.091202807  1.00000000
## rm      -0.21924670  0.31199059 -0.39167585  0.091251225 -0.30218819
## age      0.35273425 -0.56953734  0.64477851  0.086517774  0.73147010
## dis     -0.37967009  0.66440822 -0.70802699 -0.099175780 -0.76923011
## rad      0.62550515 -0.31194783  0.59512927 -0.007368241  0.61144056
## tax      0.58276431 -0.31456332  0.72076018 -0.035586518  0.66802320
## ptratio  0.28994558 -0.39167855  0.38324756 -0.121515174  0.18893268
## black   -0.38506394  0.17552032 -0.35697654  0.048788485 -0.38005064
## lstat    0.45562148 -0.41299457  0.60379972 -0.053929298  0.59087892
## medv    -0.38830461  0.36044534 -0.48372516  0.175260177 -0.42732077
##                 rm         age         dis          rad         tax    ptratio
## crim    -0.21924670  0.35273425 -0.37967009  0.625505145  0.58276431  0.2899456
## zn       0.31199059 -0.56953734  0.66440822 -0.311947826 -0.31456332 -0.3916785
## indus   -0.39167585  0.64477851 -0.70802699  0.595129275  0.72076018  0.3832476
## chas     0.09125123  0.08651777 -0.09917578 -0.007368241 -0.03558652 -0.1215152
## nox     -0.30218819  0.73147010 -0.76923011  0.611440563  0.66802320  0.1889327
## rm       1.00000000 -0.24026493  0.20524621 -0.209846668 -0.29204783 -0.3555015
## age     -0.24026493  1.00000000 -0.74788054  0.456022452  0.50645559  0.2615150
## dis      0.20524621 -0.74788054  1.00000000 -0.494587930 -0.53443158 -0.2324705
## rad     -0.20984667  0.45602245 -0.49458793  1.000000000  0.91022819  0.4647412
## tax     -0.29204783  0.50645559 -0.53443158  0.910228189  1.00000000  0.4608530
## ptratio -0.35550149  0.26151501 -0.23247054  0.464741179  0.46085304  1.0000000
## black    0.12806864 -0.27353398  0.29151167 -0.444412816 -0.44180801 -0.1773833
## lstat   -0.61380827  0.60233853 -0.49699583  0.488676335  0.54399341  0.3740443
## medv     0.69535995 -0.37695457  0.24992873 -0.381626231 -0.46853593 -0.5077867
##               black      lstat       medv
## crim    -0.38506394  0.4556215 -0.3883046
## zn       0.17552032 -0.4129946  0.3604453
## indus   -0.35697654  0.6037997 -0.4837252
## chas     0.04878848 -0.0539293  0.1752602
## nox     -0.38005064  0.5908789 -0.4273208
## rm       0.12806864 -0.6138083  0.6953599
## age     -0.27353398  0.6023385 -0.3769546
## dis      0.29151167 -0.4969958  0.2499287
## rad     -0.44441282  0.4886763 -0.3816262
## tax     -0.44180801  0.5439934 -0.4685359
## ptratio -0.17738330  0.3740443 -0.5077867
## black    1.00000000 -0.3660869  0.3334608
## lstat   -0.36608690  1.0000000 -0.7376627
## medv     0.33346082 -0.7376627  1.0000000
```

```
corrplot(cor_matrix, method="circle")
```




```
#Let's try a mixed plot with correlation values (closer the values to 1, stronger the correlation)
corrplot.mixed(cor_matrix, lower = 'number', upper = 'circle',tl.pos ="lt", tl.col='black', tl.cex=0.6,
```



In the correlation matrix, Color blue represents positive and red represents the negative correlation. The values closer to +1 and -1 implies a stronger positive and negative correlation respectively. The threshold is represented by different shades of blue and black and decimal points which is indexed on the right side of the plot.

Based on the plot we can see:

Positive correlation between some variables for example:

- Proportional of non-retail business acres per town (indus) and nitrogen oxides concentration in ppm (nox)
- Index of accessibility to radial highways (rad) and full-value property-tax rate per $10,000 (tax)

Negative correlation between some variables for example:

- Proportional of non-retail business acres per town (indus) and weighted mean of distances to five Boston employment centers (dis)
- Nitrogen oxides concentration in ppm (nox) and weighted mean of distances to five Boston employment centers (dis)

4.3 Standardizing the dataset

Lets beging the scaling exercise to standardize the data set. Since the Boston data contains only numerical values, so we can use the function `scale()` to standardize the whole dataset as mentioned in the exercise 4 instruction.

We also have the following equation from exercise which basically gives us the idea on scaling i.e., subtract the column means from corresponding means and dividing with the standard deviation.

$$scaled(x) = \frac{x - mean(x)}{sd(x)}$$

```
#Saving the scaled data to the object
boston_scaled <- scale(Boston)

#Lets see
summary(boston_scaled)
```

```
##       crim                 zn               indus              chas        
##  Min.   :-0.419367   Min.   :-0.48724   Min.   :-1.5563   Min.   :-0.2723  
##  1st Qu.:-0.410563   1st Qu.:-0.48724   1st Qu.:-0.8668   1st Qu.:-0.2723  
##  Median :-0.390280   Median :-0.48724   Median :-0.2109   Median :-0.2723  
##  Mean   : 0.000000   Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000  
##  3rd Qu.: 0.007389   3rd Qu.: 0.04872   3rd Qu.: 1.0150   3rd Qu.:-0.2723  
##  Max.   : 9.924110   Max.   : 3.80047   Max.   : 2.4202   Max.   : 3.6648  
##       nox                rm                age               dis         
##  Min.   :-1.4644   Min.   :-3.8764   Min.   :-2.3331   Min.   :-1.2658  
##  1st Qu.:-0.9121   1st Qu.:-0.5681   1st Qu.:-0.8366   1st Qu.:-0.8049  
##  Median :-0.1441   Median :-0.1084   Median : 0.3171   Median :-0.2790  
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000  
##  3rd Qu.: 0.5981   3rd Qu.: 0.4823   3rd Qu.: 0.9059   3rd Qu.: 0.6617  
##  Max.   : 2.7296   Max.   : 3.5515   Max.   : 1.1164   Max.   : 3.9566  
##       rad               tax             ptratio            black        
##  Min.   :-0.9819   Min.   :-1.3127   Min.   :-2.7047   Min.   :-3.9033  
##  1st Qu.:-0.6373   1st Qu.:-0.7668   1st Qu.:-0.4876   1st Qu.: 0.2049  
##  Median :-0.5225   Median :-0.4642   Median : 0.2746   Median : 0.3808  
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000  
##  3rd Qu.: 1.6596   3rd Qu.: 1.5294   3rd Qu.: 0.8058   3rd Qu.: 0.4332  
##  Max.   : 1.6596   Max.   : 1.7964   Max.   : 1.6372   Max.   : 0.4406  
##      lstat              medv        
##  Min.   :-1.5296   Min.   :-1.9063  
##  1st Qu.:-0.7986   1st Qu.:-0.5989  
##  Median :-0.1811   Median :-0.1449  
##  Mean   : 0.0000   Mean   : 0.0000  
##  3rd Qu.: 0.6024   3rd Qu.: 0.2683  
##  Max.   : 3.5453   Max.   : 2.9865
```

```
# class of the boston_scaled object
class(boston_scaled)
```

```
## [1] "matrix" "array"
```

```
# change the object to data frame
boston_scaled <- as.data.frame(boston_scaled)
```

Scaling the data set with various scales makes the analyses easier. Scale() transforms the data into a matrix and array so for further analysis we can change it back to the data frame

Let's scale further by creating a quantile vector of crim and print it

```
bins <- quantile(boston_scaled$crim)
bins
```

```
##           0%          25%          50%          75%         100% 
## -0.419366929 -0.410563278 -0.390280295  0.007389247  9.924109610
```

```
# create a categorical variable 'crime'
crime <- cut(boston_scaled$crim, breaks = bins, include.lowest = TRUE)
```

```
# look at the table of the new factor crime
table(crime)
```

```
## crime
## [-0.419,-0.411]  (-0.411,-0.39] (-0.39,0.00739]  (0.00739,9.92]
##             127             126             126             127
```

```
# remove original crim from the dataset
boston_scaled <- dplyr::select(boston_scaled, -crim)

# add the new categorical value to scaled data
boston_scaled <- data.frame(boston_scaled, crime)

# let's see the new data set now !!
summary(boston_scaled)
```

```
##        zn               indus              chas              nox
##  Min.   :-0.48724   Min.   :-1.5563   Min.   :-0.2723   Min.   :-1.4644
##  1st Qu.:-0.48724   1st Qu.:-0.8668   1st Qu.:-0.2723   1st Qu.:-0.9121
##  Median :-0.48724   Median :-0.2109   Median :-0.2723   Median :-0.1441
##  Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.04872   3rd Qu.: 1.0150   3rd Qu.:-0.2723   3rd Qu.: 0.5981
##  Max.   : 3.80047   Max.   : 2.4202   Max.   : 3.6648   Max.   : 2.7296
##        rm                age               dis               rad
##  Min.   :-3.8764   Min.   :-2.3331   Min.   :-1.2658   Min.   :-0.9819
##  1st Qu.:-0.5681   1st Qu.:-0.8366   1st Qu.:-0.8049   1st Qu.:-0.6373
##  Median :-0.1084   Median : 0.3171   Median :-0.2790   Median :-0.5225
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.4823   3rd Qu.: 0.9059   3rd Qu.: 0.6617   3rd Qu.: 1.6596
##  Max.   : 3.5515   Max.   : 1.1164   Max.   : 3.9566   Max.   : 1.6596
##       tax             ptratio            black             lstat
##  Min.   :-1.3127   Min.   :-2.7047   Min.   :-3.9033   Min.   :-1.5296
##  1st Qu.:-0.7668   1st Qu.:-0.4876   1st Qu.: 0.2049   1st Qu.:-0.7986
##  Median :-0.4642   Median : 0.2746   Median : 0.3808   Median :-0.1811
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 1.5294   3rd Qu.: 0.8058   3rd Qu.: 0.4332   3rd Qu.: 0.6024
##  Max.   : 1.7964   Max.   : 1.6372   Max.   : 0.4406   Max.   : 3.5453
##       medv                     crime
##  Min.   :-1.9063   [-0.419,-0.411]:127
##  1st Qu.:-0.5989   (-0.411,-0.39] :126
##  Median :-0.1449   (-0.39,0.00739]:126
##  Mean   : 0.0000   (0.00739,9.92] :127
##  3rd Qu.: 0.2683
##  Max.   : 2.9865
```

Now as mention in the exercise set “Divide and conquer”, lets divide train and test sets: training -> 80% and testing -> 20%

```
# number of rows in the Boston data set
n <- nrow(boston_scaled)
n
```

```
## [1] 506
```

```
# choose randomly 80% of the rows
ind <- sample(n,  size = n * 0.8)
```

```r
# create train set
train <- boston_scaled[ind,]

# create test set
test <- boston_scaled[-ind,]

# save the correct classes from test data
correct_classes <- test$crime

# remove the crime variable from test data
test <- dplyr::select(test, -crime)
```

4.4 Fitting the linear discriminant analysis

Let's move on fitting LDA on the training set.

Our target variable: crime(categorical)

```r
# linear discriminant analysis
# Other variables are designated (.)
lda.fit <- lda(crime ~ ., data = train)

# print the lda.fit object
lda.fit
```

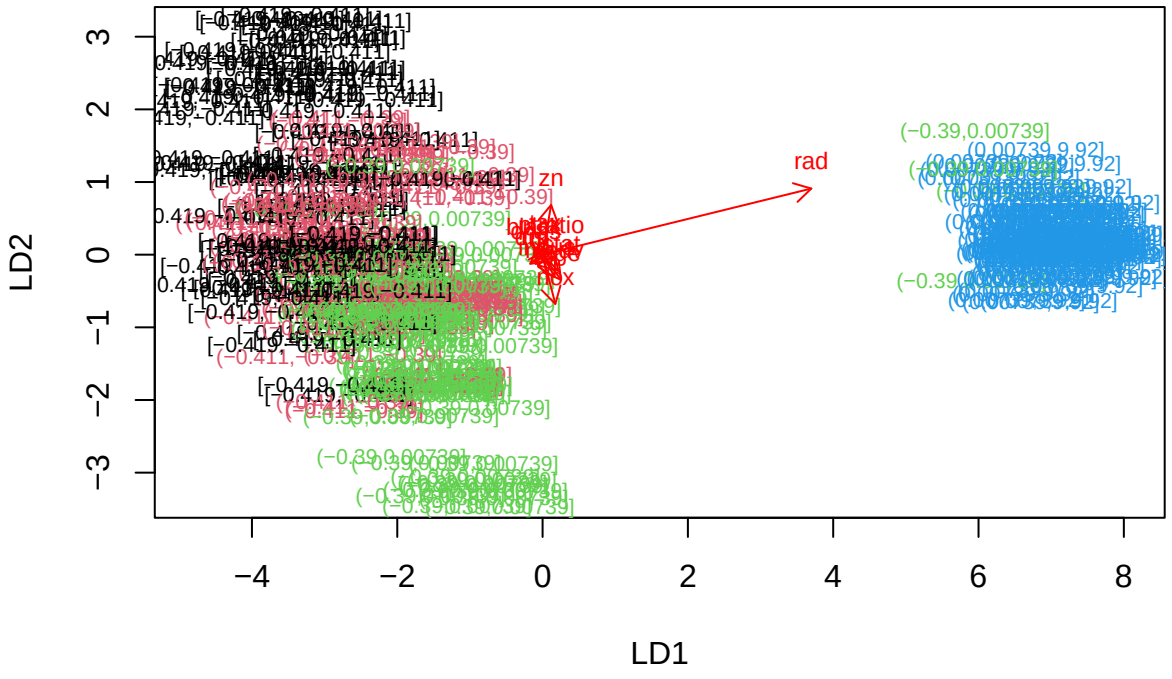
```
## Call:
## lda(crime ~ ., data = train)
##
## Prior probabilities of groups:
## [-0.419,-0.411]  (-0.411,-0.39] (-0.39,0.00739]  (0.00739,9.92]
##       0.2500000       0.2425743       0.2475248       0.2599010
##
## Group means:
##                           zn      indus        chas        nox          rm
## [-0.419,-0.411]  1.0100634 -0.9333074 -0.07742312 -0.9055726  0.44385575
## (-0.411,-0.39]  -0.1503484 -0.3231141 -0.03128211 -0.5615718 -0.14251629
## (-0.39,0.00739] -0.3920529  0.1953138  0.20012296  0.3865709  0.06926334
## (0.00739,9.92]  -0.4872402  1.0149946 -0.04735191  1.0225084 -0.47191738
##                          age        dis        rad        tax     ptratio
## [-0.419,-0.411] -0.9327937  0.9634777 -0.6862262 -0.7245783 -0.48115401
## (-0.411,-0.39]  -0.2862480  0.2790324 -0.5470944 -0.4908530 -0.03979123
## (-0.39,0.00739]  0.4266282 -0.3991745 -0.4214193 -0.3203280 -0.23304679
## (0.00739,9.92]   0.8263331 -0.8626629  1.6596029  1.5294129  0.80577843
##                       black       lstat        medv
## [-0.419,-0.411]  0.38094308 -0.76967977  0.54062063
## (-0.411,-0.39]   0.31760900 -0.15639674 -0.01699184
## (-0.39,0.00739]  0.09998228  0.05823179  0.11712314
## (0.00739,9.92]  -0.73911771  0.89917052 -0.73537088
##
## Coefficients of linear discriminants:
##                 LD1         LD2         LD3
## zn       0.11749867  0.68474127 -0.91909444
## indus    0.05974138 -0.27386740  0.09033255
## chas    -0.09382843 -0.04962562  0.03838033
## nox      0.17574770 -0.67815435 -1.50907668
```

```
## rm      -0.09629512 -0.12523575 -0.23334671
## age      0.25151266 -0.34530926 -0.02513006
## dis     -0.16965361 -0.12449252 -0.11255346
## rad      3.69895352  0.91007926 -0.02438438
## tax      0.03769299  0.06710313  0.56780370
## ptratio  0.12166474  0.03287276 -0.28850792
## black   -0.11774628  0.01242576  0.13099670
## lstat    0.21890677 -0.22353615  0.11531688
## medv     0.16495494 -0.30630739 -0.34067271
##
## Proportion of trace:
##    LD1    LD2    LD3
## 0.9563 0.0338 0.0099
```

There are three discriminant function LD(proportion of trace) as follow:

LD1 LD2 LD3 0.9489 0.0362 0.0150

```
# the function for lda biplot arrows
lda.arrows <- function(x, myscale = 1, arrow_heads = 0.1, color = "red", tex = 0.75, choices = c(1,2)){
  heads <- coef(x)
  arrows(x0 = 0, y0 = 0,
         x1 = myscale * heads[,choices[1]],
         y1 = myscale * heads[,choices[2]], col=color, length = arrow_heads)
  text(myscale * heads[,choices], labels = row.names(heads),
       cex = tex, col=color, pos=3)
}

# target classes as numeric
classes <- as.numeric(train$crime)

# plot the lda results
plot(lda.fit, dimen = 2, col = classes, pch = classes)
lda.arrows(lda.fit, myscale = 1)
```



4.5 Predicting the LDA model

```
library(MASS)

ind <- sample(nrow(boston_scaled),  size = nrow(boston_scaled) * 0.8)

train <- boston_scaled[ind,]

test <- boston_scaled[-ind,]

correct_classes <- test$crime

test <- dplyr::select(test, -crime)

lda.fit = lda(crime ~ ., data=train)

# predict classes with test data
lda.pred <- predict(lda.fit, newdata = test)

# cross tabulate the results
table(correct = correct_classes, predicted = lda.pred$class)
```

```
##                  predicted
## correct           [-0.419,-0.411] (-0.411,-0.39] (-0.39,0.00739] (0.00739,9.92]
##   [-0.419,-0.411]              16             12               3              0
##   (-0.411,-0.39]                5             13               8              0
##   (-0.39,0.00739]               0             10              14              1
##   (0.00739,9.92]                0              0               0             20
```

4.6 Model Optimization and K means clustering

Let's work with clustering now Let's calculate also the distances between observations to assess the similarity between data points. As instructed we calculated two distance matrix euclidean and manhattan.

```
#start by reloading the Boston data set
data("Boston")

boston_scaled <- scale(Boston)
summary(boston_scaled)
```

```
##       crim                 zn               indus              chas        
##  Min.   :-0.419367   Min.   :-0.48724   Min.   :-1.5563   Min.   :-0.2723  
##  1st Qu.:-0.410563   1st Qu.:-0.48724   1st Qu.:-0.8668   1st Qu.:-0.2723  
##  Median :-0.390280   Median :-0.48724   Median :-0.2109   Median :-0.2723  
##  Mean   : 0.000000   Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000  
##  3rd Qu.: 0.007389   3rd Qu.: 0.04872   3rd Qu.: 1.0150   3rd Qu.:-0.2723  
##  Max.   : 9.924110   Max.   : 3.80047   Max.   : 2.4202   Max.   : 3.6648  
##       nox                rm                age               dis         
##  Min.   :-1.4644   Min.   :-3.8764   Min.   :-2.3331   Min.   :-1.2658  
##  1st Qu.:-0.9121   1st Qu.:-0.5681   1st Qu.:-0.8366   1st Qu.:-0.8049  
##  Median :-0.1441   Median :-0.1084   Median : 0.3171   Median :-0.2790  
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000  
##  3rd Qu.: 0.5981   3rd Qu.: 0.4823   3rd Qu.: 0.9059   3rd Qu.: 0.6617  
##  Max.   : 2.7296   Max.   : 3.5515   Max.   : 1.1164   Max.   : 3.9566  
##       rad               tax             ptratio            black        
```

```
##  Min.   :-0.9819   Min.   :-1.3127   Min.   :-2.7047   Min.   :-3.9033
##  1st Qu.:-0.6373   1st Qu.:-0.7668   1st Qu.:-0.4876   1st Qu.: 0.2049
##  Median :-0.5225   Median :-0.4642   Median : 0.2746   Median : 0.3808
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 1.6596   3rd Qu.: 1.5294   3rd Qu.: 0.8058   3rd Qu.: 0.4332
##  Max.   : 1.6596   Max.   : 1.7964   Max.   : 1.6372   Max.   : 0.4406
##      lstat              medv
##  Min.   :-1.5296   Min.   :-1.9063
##  1st Qu.:-0.7986   1st Qu.:-0.5989
##  Median :-0.1811   Median :-0.1449
##  Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.6024   3rd Qu.: 0.2683
##  Max.   : 3.5453   Max.   : 2.9865
```

```
#class of Boston_scaled object
class(boston_scaled)
```

```
## [1] "matrix" "array"
```

```
# change the object to data frame for futher analysis
boston_scaled <- as.data.frame(boston_scaled)
```

```
# Calculating distances between the observation
# euclidean distance matrix
dist_eu <- dist(boston_scaled, method = "euclidean")

# look at the summary of the distances
summary(dist_eu)
```

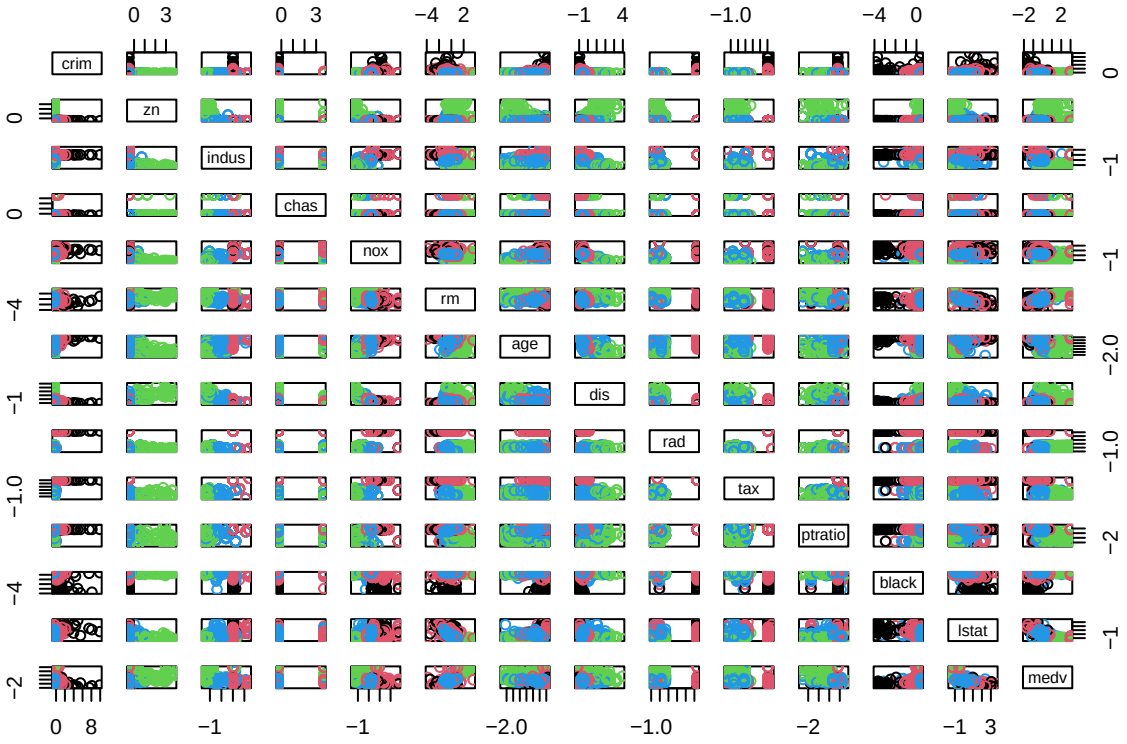
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1343  3.4625  4.8241  4.9111  6.1863 14.3970
```

```
# Manhattan distance matrix
dist_man <- dist(boston_scaled, method = "manhattan")

# look at the summary of the distances
summary(dist_man)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.2662  8.4832 12.6090 13.5488 17.7568 48.8618
```

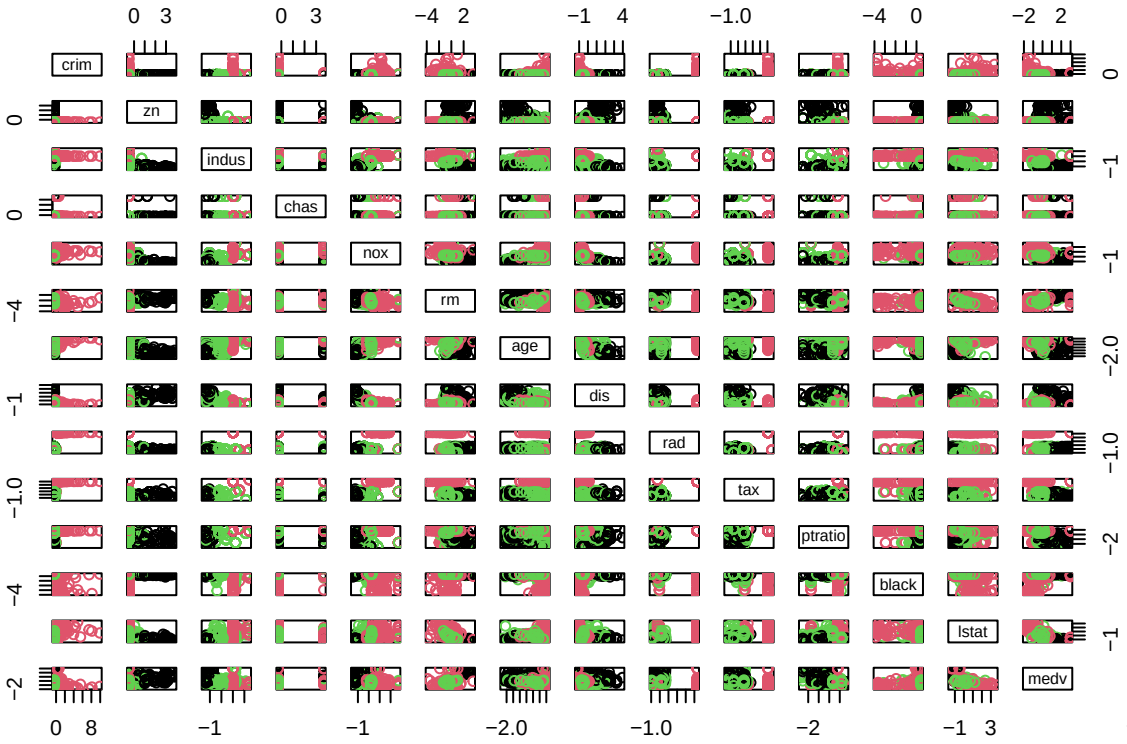
Lets do K-means clustering (first with 4 clusters and then 3)

```
set.seed(123) #function set.seed() is used here to deal with the random assigning of the initial cluste

# k-means clustering
km <- kmeans(boston_scaled, centers = 4)

# plotting the scaled Boston data set with clusters
pairs(boston_scaled, col = km$cluster)
```



With 4 clusters, it appears that there is some overlapping between clusters. lets try 3 clusters and check

```
# k-means clustering
km <- kmeans(boston_scaled, centers = 3)

# plotting the scaled Boston data set with clusters
pairs(boston_scaled, col = km$cluster)
```

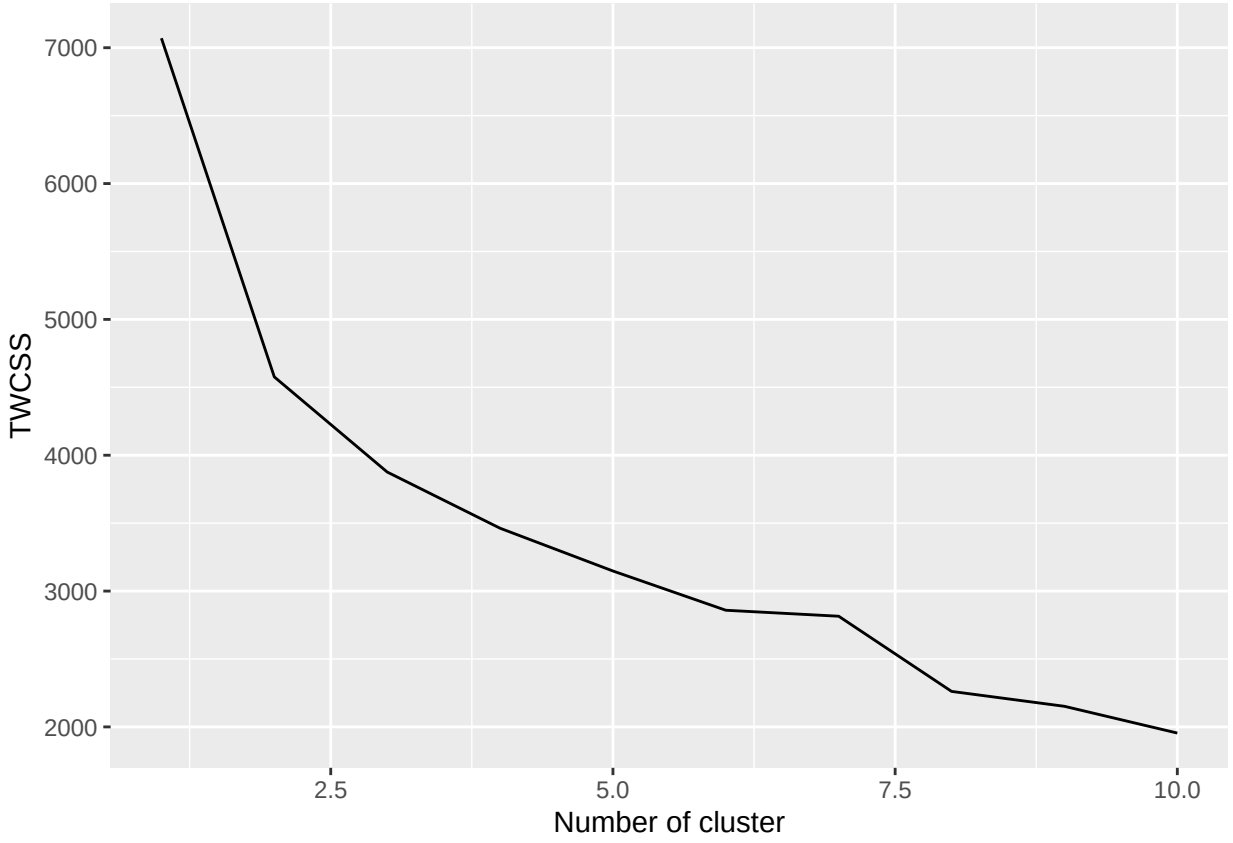


with 3 clusters it is even worse !! We can try to determine the optimal number of cluster for our model and see how it

works

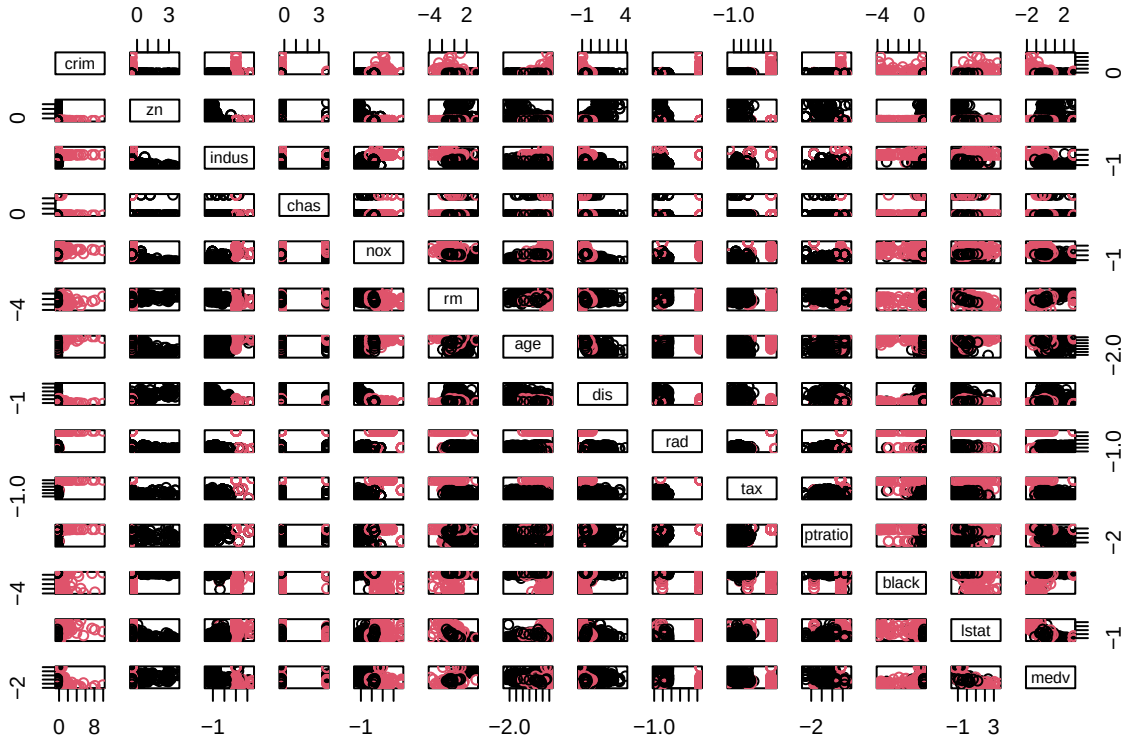
4.7 K-means: determine the k

```
#Investigating the optimal number of clusters
set.seed(123) #setseed function again

# determine the number of clusters
k_max <- 10

# calculate the total within sum of squares
twcss <- sapply(1:k_max, function(k){kmeans(boston_scaled, k)$tot.withinss})

# visualize the results
qplot(x = 1:k_max, y = twcss, geom = 'line', ylab = "TWCSS", xlab = "Number of cluster")
```

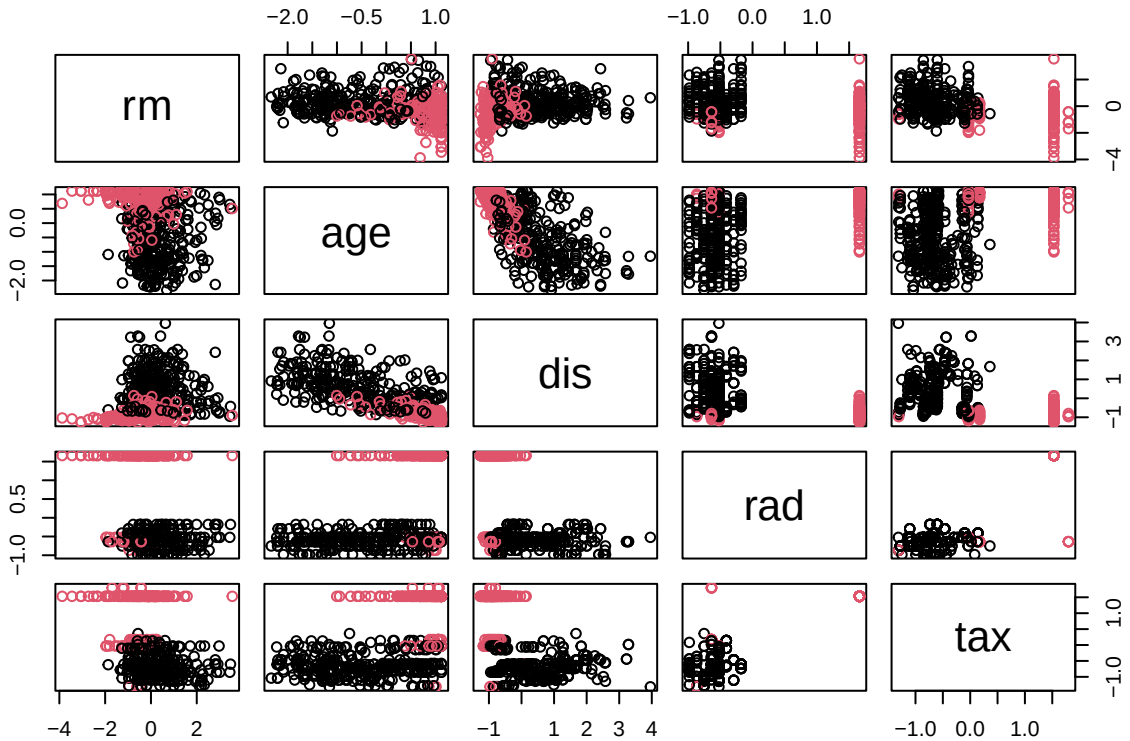


From the plot it looks like the optimal number of clusters for our model is 2-3 as there is a sharp drop in TWCSS values. We can try with 2 because 2 seems more optimal as by 2.5 the drop is significant and we can see the substantial change.

```
# k-means clustering
km <- kmeans(boston_scaled, centers = 2)

# plot the Boston data set with clusters predicted
pairs(boston_scaled, col = km$cluster)
```




```
pairs(boston_scaled[6:10], col = km$cluster)
```



With 2 clusters, the model seem better, there is good separation for example rad and tax. rm and age seems ok as well.

Week 5: Dimensionality reduction techniques

Actually, a fairly large selection of statistical methods can be listed under the title "dimensionality reduction techniques". Most often (nearly always, that is!) the real-world phenomena are multidimensional: they may consist of not just two or three but 5 or 10 or 20 or 50 (or more) dimensions. Of course, we are living only in a three-dimensional (3D) world, so those multiple dimensions may really challenge our imagination. It would be easier to reduce the number of dimensions in one way or another.

We shall now learn the basics of two data science based ways of reducing the dimensions. The principal method here is principal component analysis (PCA), which reduces any number of measured (continuous) and correlated variables into a few uncorrelated components that collect together as much variance as possible from the original variables. The most important components can be then used for various purposes, e.g., drawing scatterplots and other fancy graphs that would be quite impossible to achieve with the original variables and too many dimensions.

Multiple correspondence analysis (MCA) and other variations of CA bring us similar possibilities in the world of discrete variables, even nominal scale (classified) variables, by finding a suitable transformation into continuous scales and then reducing the dimensions quite analogously with the PCA. The typical graphs show the original classes of the discrete variables on the same "map", making it possible to reveal connections (correspondences) between different things that would be quite impossible to see from the corresponding cross tables (too many numbers!).

Briefly stated, these methods help to visualize and understand multidimensional phenomena by reducing their dimensionality that may first feel impossible to handle at all.

```
date()
```

```
## [1] "Tue Dec  6 21:31:21 2022"
```

Lets start !!

5.1 Packages required

```
#load the packages

library(tidyverse)
library(GGally)
library(dplyr)
library(ggplot2)
library(corrplot)
library(stringr)
library(psych)
```

```
##
## Attaching package: 'psych'
```

```
## The following object is masked from 'package:boot':
##
##     logit
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
library(FactoMineR)
library(tidyr)
```

5.2 Loading the data set from wrangling exercise

```
#loading from project folder
human_ <- read.csv('human_.csv', row.names = 1)

#Alternatively url is given in the instruction page, so we can also use that !!
# human <- read.csv("https://raw.githubusercontent.com/KimmoVehkalahti/Helsinki-Open-Data-Science/maste

#lets check how the data looks
str(human_);dim(human_)
```

```
## 'data.frame':    155 obs. of  8 variables:
##  $ SeEdu_FM: num  1.007 0.997 0.983 0.989 0.969 ...
##  $ LFR_FM  : num  0.891 0.819 0.825 0.884 0.829 ...
##  $ Life_Exp: num  81.6 82.4 83 80.2 81.6 80.9 80.9 79.1 82 81.8 ...
##  $ Exp_Edu : num  17.5 20.2 15.8 18.7 17.9 16.5 18.6 16.5 15.9 19.2 ...
##  $ GNI     : int  64992 42261 56431 44025 45435 43919 39568 52947 42155 32689 ...
##  $ MMR     : int  4 6 6 5 6 7 9 28 11 8 ...
##  $ ABR     : num  7.8 12.1 1.9 5.1 6.2 3.8 8.2 31 14.5 25.3 ...
##  $ X.PR    : num  39.6 30.5 28.5 38 36.9 36.9 19.9 19.4 28.2 31.4 ...
```

```
## [1] 155   8
```

```
colnames(human_)
```

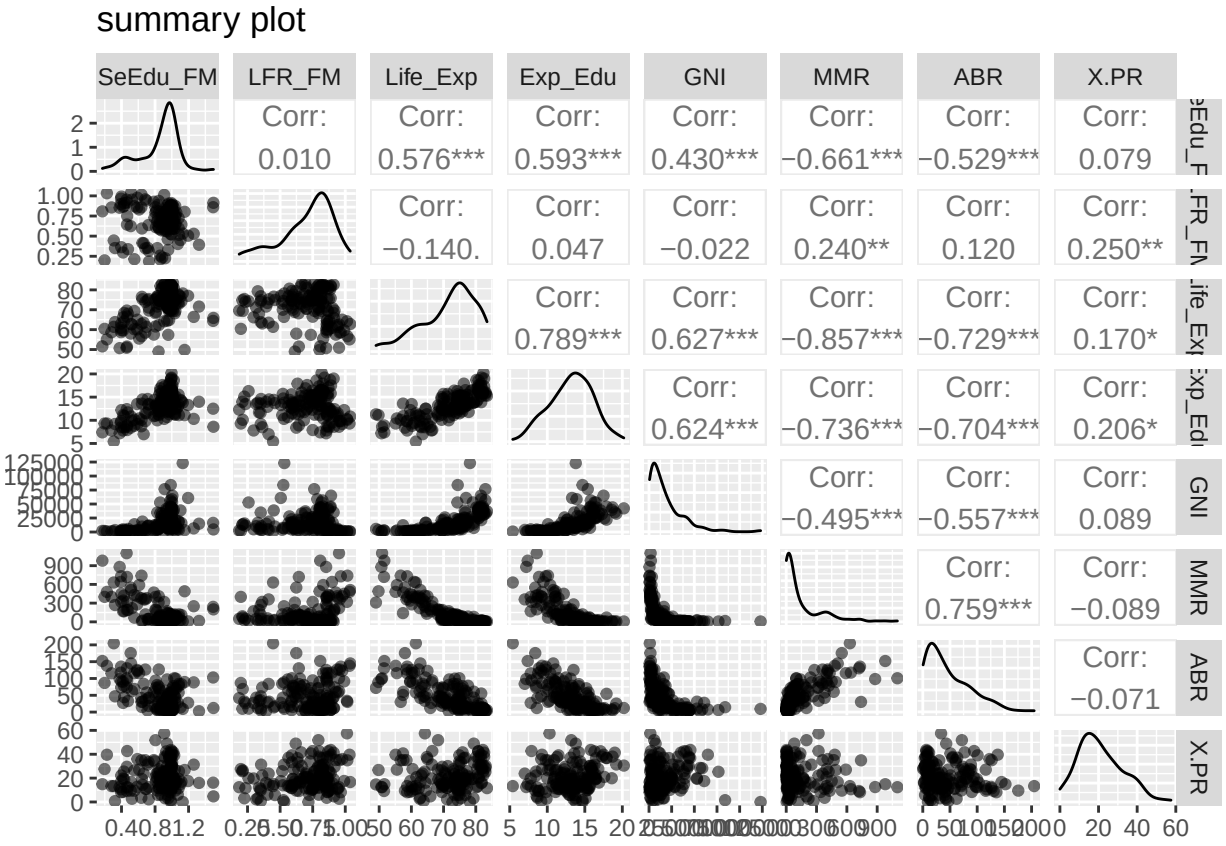
```
## [1] "SeEdu_FM" "LFR_FM"   "Life_Exp" "Exp_Edu"  "GNI"      "MMR"      "ABR"
## [8] "X.PR"
```

5.3 Graphical overview

```
summary(human_)
```

```
##     SeEdu_FM          LFR_FM          Life_Exp        Exp_Edu
##  Min.   :0.1717   Min.   :0.1857   Min.   :49.00   Min.   : 5.40
##  1st Qu.:0.7264   1st Qu.:0.5984   1st Qu.:66.30   1st Qu.:11.25
##  Median :0.9375   Median :0.7535   Median :74.20   Median :13.50
##  Mean   :0.8529   Mean   :0.7074   Mean   :71.65   Mean   :13.18
##  3rd Qu.:0.9968   3rd Qu.:0.8535   3rd Qu.:77.25   3rd Qu.:15.20
##  Max.   :1.4967   Max.   :1.0380   Max.   :83.50   Max.   :20.20
##       GNI              MMR              ABR              X.PR
##  Min.   :   581   Min.   :   1.0   Min.   :  0.60   Min.   : 0.00
##  1st Qu.:  4198   1st Qu.:  11.5   1st Qu.: 12.65   1st Qu.:12.40
##  Median : 12040   Median :  49.0   Median : 33.60   Median :19.30
##  Mean   : 17628   Mean   : 149.1   Mean   : 47.16   Mean   :20.91
##  3rd Qu.: 24512   3rd Qu.: 190.0   3rd Qu.: 71.95   3rd Qu.:27.95
##  Max.   :123124   Max.   :1100.0   Max.   :204.80   Max.   :57.50
```

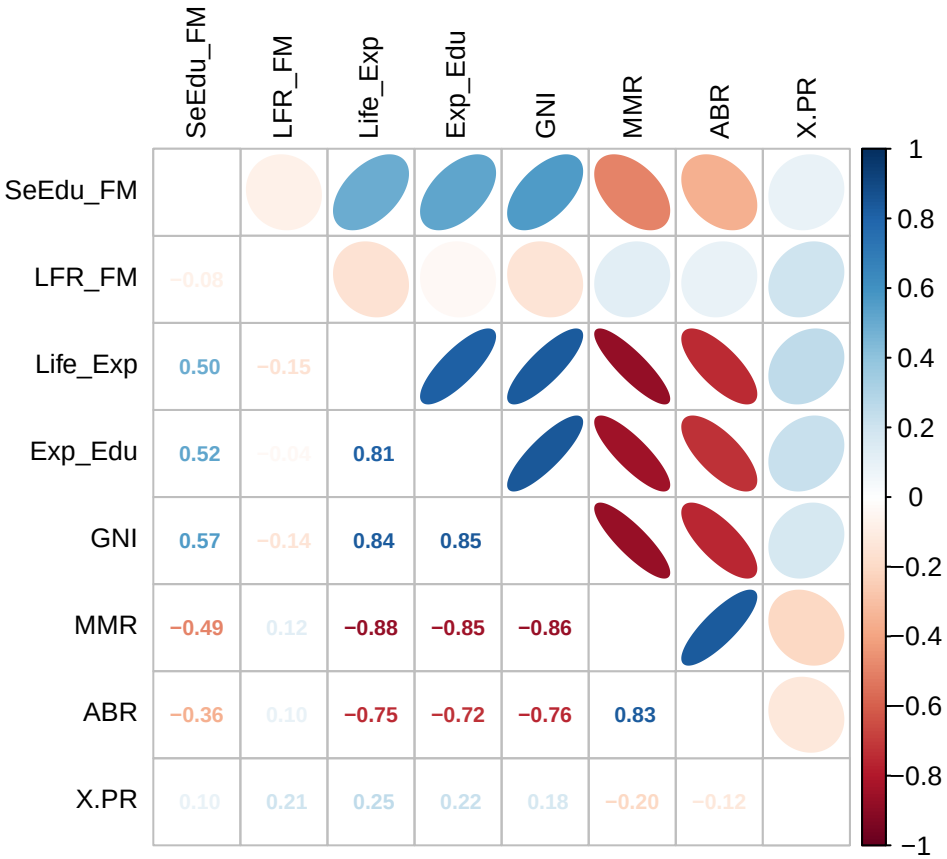
```
Plot1 <- p1 <- ggpairs(human_, mapping = aes(alpha=0.5), title="summary plot",lower = list(combo = wrap
Plot1
```




```
#Lets see the correlation matrix with corrplot, I have used same method as last week's exercise with so
Plot2 <- cor(human_, method='spearman')
Plot2
```

```
##              SeEdu_FM      LFR_FM   Life_Exp     Exp_Edu         GNI         MMR
## SeEdu_FM  1.00000000 -0.07525177  0.4987487  0.52316296  0.5663235 -0.4937468
## LFR_FM   -0.07525177  1.00000000 -0.1535502 -0.03956409 -0.1414489  0.1213061
## Life_Exp  0.49874873 -0.15355015  1.0000000  0.81053640  0.8361208 -0.8753753
## Exp_Edu   0.52316296 -0.03956409  0.8105364  1.00000000  0.8495071 -0.8472241
## GNI       0.56632346 -0.14144887  0.8361208  0.84950709  1.0000000 -0.8647968
## MMR      -0.49374677  0.12130614 -0.8753753 -0.84722412 -0.8647968  1.0000000
## ABR      -0.35514985  0.09705612 -0.7468356 -0.72251512 -0.7562608  0.8315049
## X.PR      0.09722070  0.20545990  0.2523837  0.22396862  0.1778011 -0.2028040
##                  ABR       X.PR
## SeEdu_FM -0.35514985  0.0972207
## LFR_FM    0.09705612  0.2054599
## Life_Exp -0.74683557  0.2523837
## Exp_Edu  -0.72251512  0.2239686
## GNI      -0.75626078  0.1778011
## MMR       0.83150492 -0.2028040
## ABR       1.00000000 -0.1214131
## X.PR     -0.12141308  1.0000000
```

```
corrplot.mixed(Plot2, lower = 'number', upper = 'ellipse',tl.pos ="lt", tl.col='black', tl.cex=0.8, numb
```



In above, correlation plot, I have used ellipse method to visualize the relationship between different variables. The correlation is stronger when the ellipse are narrower and two color spectrum blue and red represents positive and negative correlation respectively.

Statistically significant strong positive correlation from two plots are between variables

- Life expectancy (Life_Exp) and Expected years of schooling (Exp_Edu)
- Life expectancy (Life_Exp) and Gross National Income per capita (GNI)
- Expected years of schooling (Exp_Edu) and Gross National Income per capita (GNI)
- Maternal Mortality Rates (MMR) and Adolescent Birth Rate (ABR)

Statistically significant strong negative correlation from two plots are between variables

- Life expectancy (Life_Exp) and Maternal Mortality Rate (MMR)
- Expected years of schooling (Exp_Edu) and Maternal Mortality Rate (MMR)
- Gross National Income (GNI) and Maternal Mortality Rate (MMR)

Two variables; labor Force Mortality of male and female combined (LFR_FM) and percentage of female representatives in the parliament (%PR) doesn't show any correlation and therefore are not associated with any other variables. Like wise secondary education of male and female combines (SeEdu_FM) isn't associated strongly with any other variables.

5.4 Principal component analysis (PCA)

Principal Component Analysis (PCA) can be performed by two sightly different matrix decomposition methods from linear algebra: the Eigenvalue Decomposition and the Singular Value Decomposition (SVD).

There are two functions in the default package distribution of R that can be used to perform PCA: `princomp()` and `prcomp()`. The `prcomp()` function uses the SVD and is the preferred, more numerically accurate method. Both methods quite literally *decompose* a data matrix into a product of smaller matrices, which let's us

extract the underlying **principal components**. This makes it possible to approximate a lower dimensional representation of the data by choosing only a few principal components.

Lets follow the instruction from course material and

```
# lets create `human_std` by standardizing the variables in `human`
human_std <- scale(human_)

# print out summaries of the standardized variables
summary(human_std)
```

```
##     SeEdu_FM           LFR_FM           Life_Exp          Exp_Edu
##  Min.   :-2.8189   Min.   :-2.6247   Min.   :-2.7188   Min.   :-2.7378
##  1st Qu.:-0.5233   1st Qu.:-0.5484   1st Qu.:-0.6425   1st Qu.:-0.6782
##  Median : 0.3503   Median : 0.2316   Median : 0.3056   Median : 0.1140
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.5958   3rd Qu.: 0.7350   3rd Qu.: 0.6717   3rd Qu.: 0.7126
##  Max.   : 2.6646   Max.   : 1.6632   Max.   : 1.4218   Max.   : 2.4730
##       GNI                MMR                ABR               X.PR
##  Min.   :-0.9193   Min.   :-0.6992   Min.   :-1.1325   Min.   :-1.8203
##  1st Qu.:-0.7243   1st Qu.:-0.6496   1st Qu.:-0.8394   1st Qu.:-0.7409
##  Median :-0.3013   Median :-0.4726   Median :-0.3298   Median :-0.1403
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.3712   3rd Qu.: 0.1932   3rd Qu.: 0.6030   3rd Qu.: 0.6127
##  Max.   : 5.6890   Max.   : 4.4899   Max.   : 3.8344   Max.   : 3.1850
```

```
# perform principal component analysis (with the SVD method)
pca_human <- prcomp(human_std)
pca_human
```

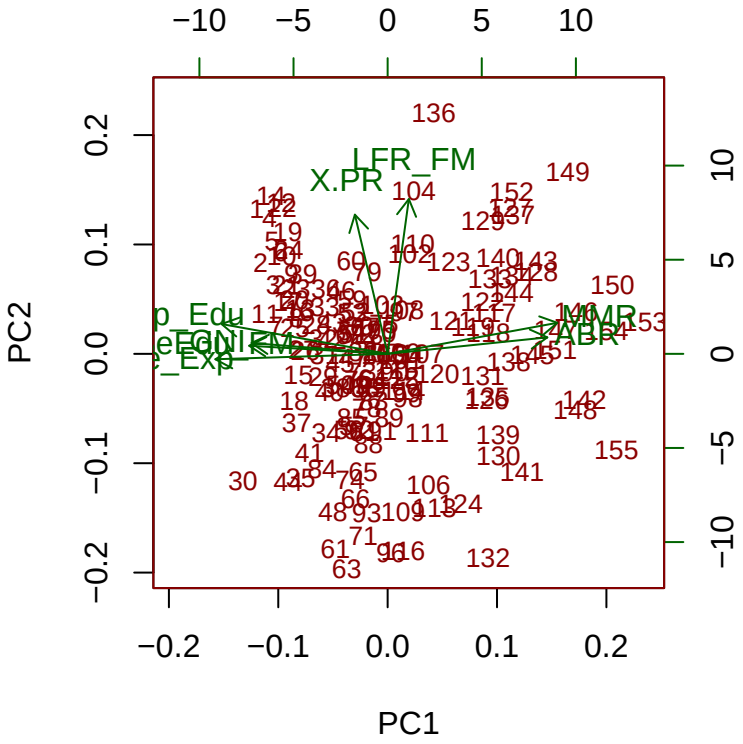
```
## Standard deviations (1, .., p=8):
## [1] 2.0708380 1.1397204 0.8750485 0.7788630 0.6619563 0.5363061 0.4589994
## [8] 0.3222406
##
## Rotation (n x k) = (8 x 8):
##                  PC1         PC2         PC3         PC4        PC5         PC6
## SeEdu_FM -0.35664370  0.03796058 -0.24223089  0.62678110 -0.5983585  0.17713316
## LFR_FM    0.05457785  0.72432726 -0.58428770  0.06199424  0.2625067 -0.03500707
## Life_Exp -0.44372240 -0.02530473  0.10991305 -0.05834819  0.1628935 -0.42242796
## Exp_Edu  -0.42766720  0.13940571 -0.07340270 -0.07020294  0.1659678 -0.38606919
## GNI      -0.35048295  0.05060876 -0.20168779 -0.72727675 -0.4950306  0.11120305
## MMR       0.43697098  0.14508727 -0.12522539 -0.25170614 -0.1800657  0.17370039
## ABR       0.41126010  0.07708468  0.01968243  0.04986763 -0.4672068 -0.76056557
## X.PR     -0.08438558  0.65136866  0.72506309  0.01396293 -0.1523699  0.13749772
##                  PC7         PC8
## SeEdu_FM  0.05773644  0.16459453
## LFR_FM   -0.22729927 -0.07304568
## Life_Exp -0.43406432  0.62737008
## Exp_Edu   0.77962966 -0.05415984
## GNI      -0.13711838 -0.16961173
## MMR       0.35380306  0.72193946
## ABR      -0.06897064 -0.14335186
## X.PR      0.00568387 -0.02306476
```

```
summary(pca_human)
```

```
## Importance of components:
```

```
##                            PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.0708 1.1397 0.87505 0.77886 0.66196 0.53631 0.45900
## Proportion of Variance 0.5361 0.1624 0.09571 0.07583 0.05477 0.03595 0.02634
## Cumulative Proportion  0.5361 0.6984 0.79413 0.86996 0.92473 0.96069 0.98702
##                            PC8
## Standard deviation     0.32224
## Proportion of Variance 0.01298
## Cumulative Proportion  1.00000
```

```
# draw a biplot of the principal component representation and the original variables
biplot(pca_human, choices = 1:2, cex = c(0.8, 1), col = c("darkred", "darkgreen"))
```



From the plot we can see the variability captured by the principal components which seems to have a realistic distribution between the principal components.

Summary of the result

From the plot (rounded with %)

PC1 = 53.61 % variance PC2 = 16.24 % variance PC3 = 9.57 % variance PC4 = 7.58 % variance PC5 = 5.47 % variance PC6 = 3.59 % variance PC7 = 2.63 % variance PC8 = 1.29 % variance

And the standard deviation (SD)

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
Standard deviation	2.0708	1.1397	0.87505	0.77886	0.66196	0.53631	0.45900	0.32224

5.5 Intrepretation of the analysis

Summary of PC1-PC8 rounded with percentage (2 Decimal points only) is elaborated above

PC1 gives the most (53,61%) and PC8 gives the least (1.29%) of the variability in the data set

The variables affect mostly based PC1-PC8 are (explained as an example from table in the summary)

Exp_Edu (positive effect) GNI (positive effect) Life_exp (positive effect) SeEdu_FM (positive effect) MMR (negative effect) ABR (negative effect)

5.6 Lets see the "tea" data set

The tea data comes from the FactoMineR package and it is measured with a questionnaire on tea: 300 individuals were asked how they drink tea (18 questions) and what are their product's perception (12 questions). In addition, some personal details were asked (4 questions).

The Factominer package contains functions dedicated to multivariate explanatory data analysis. It contains for example methods *(Multiple) Correspondence analysis* , *Multiple Factor analysis* as well as PCA.

In the next exercises we are going to use the `tea` dataset. The dataset contains the answers of a questionnaire on tea consumption.

Let's dwell in teas for a bit!

```
tea <- read.csv("https://raw.githubusercontent.com/KimmoVehkalahti/Helsinki-Open-Data-Science/master/da
view(tea)
str(tea);dim(tea)
```

```
## 'data.frame':    300 obs. of  36 variables:
##  $ breakfast       : Factor w/ 2 levels "breakfast","Not.breakfast": 1 1 2 2 1 2 1 2 1 1 ...
##  $ tea.time        : Factor w/ 2 levels "Not.tea time",..: 1 1 2 1 1 1 2 2 2 1 ...
##  $ evening         : Factor w/ 2 levels "evening","Not.evening": 2 2 1 2 1 2 2 1 2 1 ...
##  $ lunch           : Factor w/ 2 levels "lunch","Not.lunch": 2 2 2 2 2 2 2 2 2 2 ...
##  $ dinner          : Factor w/ 2 levels "dinner","Not.dinner": 2 2 1 1 2 1 2 2 2 2 ...
##  $ always          : Factor w/ 2 levels "always","Not.always": 2 2 2 2 1 2 2 2 2 2 ...
##  $ home            : Factor w/ 2 levels "home","Not.home": 1 1 1 1 1 1 1 1 1 1 ...
##  $ work            : Factor w/ 2 levels "Not.work","work": 1 1 2 1 1 1 1 1 1 1 ...
##  $ tearoom         : Factor w/ 2 levels "Not.tearoom",..: 1 1 1 1 1 1 1 1 1 2 ...
##  $ friends         : Factor w/ 2 levels "friends","Not.friends": 2 2 1 2 2 2 1 2 2 2 ...
##  $ resto           : Factor w/ 2 levels "Not.resto","resto": 1 1 2 1 1 1 1 1 1 1 ...
##  $ pub             : Factor w/ 2 levels "Not.pub","pub": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Tea             : Factor w/ 3 levels "black","Earl Grey",..: 1 1 2 2 2 2 2 1 2 1 ...
##  $ How             : Factor w/ 4 levels "alone","lemon",..: 1 3 1 1 1 1 1 3 3 1 ...
##  $ sugar           : Factor w/ 2 levels "No.sugar","sugar": 2 1 1 2 1 1 1 1 1 1 ...
##  $ how             : Factor w/ 3 levels "tea bag","tea bag+unpackaged",..: 1 1 1 1 1 1 1 1 2 2 ...
##  $ where           : Factor w/ 3 levels "chain store",..: 1 1 1 1 1 1 1 1 2 2 ...
##  $ price           : Factor w/ 6 levels "p_branded","p_cheap",..: 4 6 6 6 6 3 6 6 5 5 ...
##  $ age             : int  39 45 47 23 48 21 37 36 40 37 ...
##  $ sex             : Factor w/ 2 levels "F","M": 2 1 1 2 2 2 2 1 2 2 ...
##  $ SPC             : Factor w/ 7 levels "employee","middle",..: 2 2 4 6 1 6 5 2 5 5 ...
##  $ Sport           : Factor w/ 2 levels "Not.sportsman",..: 2 2 2 1 2 2 2 2 2 1 ...
##  $ age_Q           : Factor w/ 5 levels "+60","15-24",..: 4 5 5 2 5 2 4 4 4 4 ...
##  $ frequency       : Factor w/ 4 levels "+2/day","1 to 2/week",..: 3 3 1 3 1 3 4 2 1 1 ...
##  $ escape.exoticism: Factor w/ 2 levels "escape-exoticism",..: 2 1 2 1 1 2 2 2 2 2 ...
##  $ spirituality    : Factor w/ 2 levels "Not.spirituality",..: 1 1 1 2 2 1 1 1 1 1 ...
##  $ healthy         : Factor w/ 2 levels "healthy","Not.healthy": 1 1 1 1 2 1 1 1 2 1 ...
##  $ diuretic        : Factor w/ 2 levels "diuretic","Not.diuretic": 2 1 1 2 1 2 2 2 2 1 ...
##  $ friendliness    : Factor w/ 2 levels "friendliness",..: 2 2 1 2 1 2 2 1 2 1 ...
##  $ iron.absorption : Factor w/ 2 levels "iron absorption",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ feminine        : Factor w/ 2 levels "feminine","Not.feminine": 2 2 2 2 2 2 2 1 2 2 ...
##  $ sophisticated   : Factor w/ 2 levels "Not.sophisticated",..: 1 1 1 2 1 1 1 2 2 1 ...
##  $ slimming        : Factor w/ 2 levels "No.slimming",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ exciting        : Factor w/ 2 levels "exciting","No.exciting": 2 1 2 2 2 2 2 2 2 2 ...
```

```
##  $ relaxing        : Factor w/ 2 levels "No.relaxing",..: 1 1 2 2 2 2 2 2 2 2 ...
##  $ effect.on.health: Factor w/ 2 levels "effect on health",..: 2 2 2 2 2 2 2 2 2 2 ...
```

```
## [1] 300  36
```

```
colnames(tea)
```

```
##  [1] "breakfast"        "tea.time"         "evening"          "lunch"
##  [5] "dinner"           "always"           "home"             "work"
##  [9] "tearoom"          "friends"          "resto"            "pub"
## [13] "Tea"              "How"              "sugar"            "how"
## [17] "where"            "price"            "age"              "sex"
## [21] "SPC"              "Sport"            "age_Q"            "frequency"
## [25] "escape.exoticism" "spirituality"     "healthy"          "diuretic"
## [29] "friendliness"     "iron.absorption"  "feminine"         "sophisticated"
## [33] "slimming"         "exciting"         "relaxing"         "effect.on.health"
```

```
summary(tea)
```

```
##          breakfast           tea.time          evening          lunch
##  breakfast    :144   Not.tea time:131   evening    :103   lunch    : 44
##  Not.breakfast:156   tea time    :169   Not.evening:197   Not.lunch:256
##
##
##
##
##
##         dinner           always          home           work
##  dinner    : 21   always    :103   home    :291   Not.work:213
##  Not.dinner:279   Not.always:197   Not.home:  9   work    : 87
##
##
##
##
##
##         tearoom           friends          resto          pub
##  Not.tearoom:242   friends    :196   Not.resto:221   Not.pub:237
##  tearoom    : 58   Not.friends:104   resto    : 79   pub    : 63
##
##
##
##
##
##         Tea         How           sugar                     how
##  black    : 74   alone:195   No.sugar:155   tea bag           :170
##  Earl Grey:193   lemon: 33   sugar   :145   tea bag+unpackaged: 94
##  green    : 33   milk : 63                  unpackaged        : 36
##                  other:  9
##
##
##
##                   where                 price          age        sex
##  chain store         :192   p_branded      : 95   Min.   :15.00   F:178
##  chain store+tea shop: 78   p_cheap        :  7   1st Qu.:23.00   M:122
##  tea shop            : 30   p_private label: 21   Median :32.00
##                             p_unknown      : 12   Mean   :37.05
```

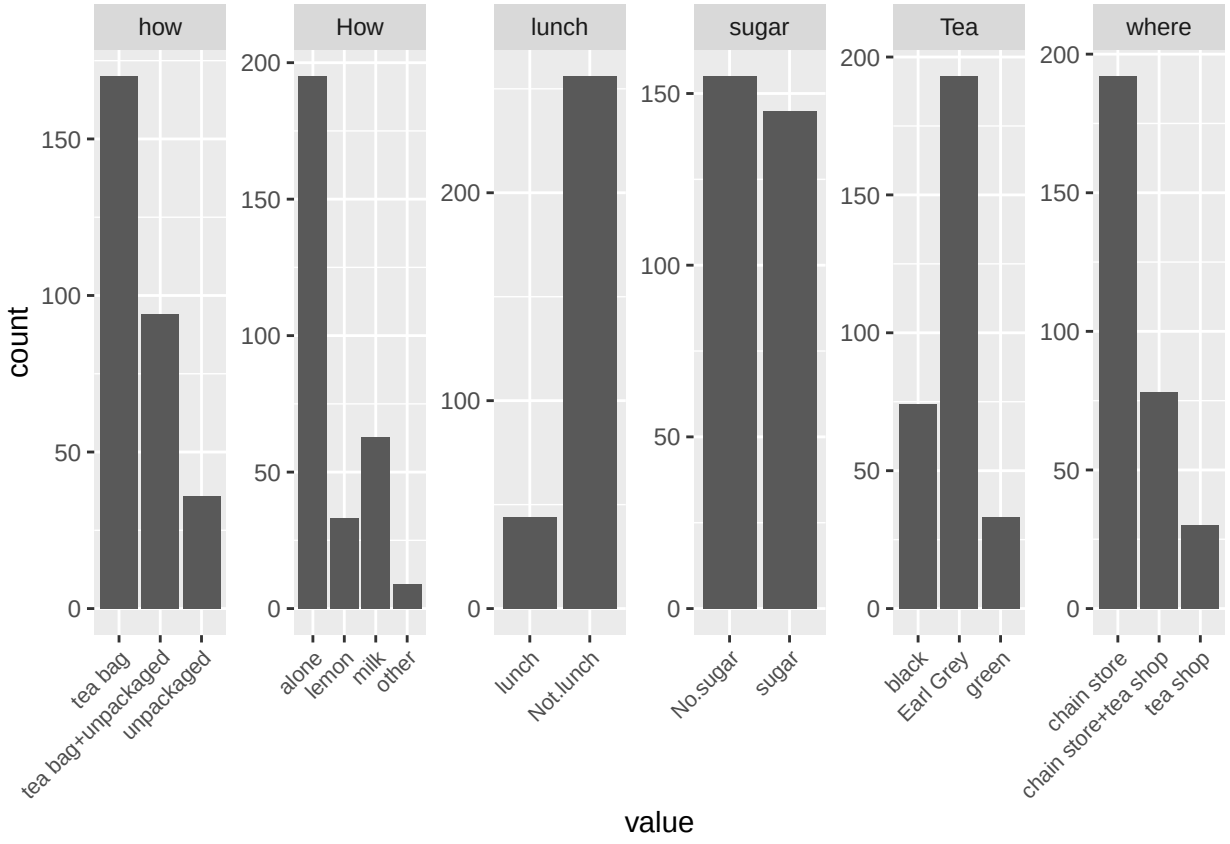
```
##                            p_upscale      : 53   3rd Qu.:48.00
##                            p_variable     :112   Max.   :90.00
##
##            SPC               Sport       age_Q          frequency
##  employee    :59   Not.sportsman:121   +60  :38   +2/day     :127
##  middle      :40   sportsman    :179   15-24:92   1 to 2/week: 44
##  non-worker  :64                       25-34:69   1/day      : 95
##  other worker:20                       35-44:40   3 to 6/week: 34
##  senior      :35                       45-59:61
##  student     :70
##  workman     :12
##              escape.exoticism           spirituality        healthy
##  escape-exoticism    :142     Not.spirituality:206   healthy    :210
##  Not.escape-exoticism:158     spirituality    : 94   Not.healthy: 90
##
##
##
##
##
##          diuretic             friendliness            iron.absorption
##  diuretic    :174   friendliness    :242   iron absorption    : 31
##  Not.diuretic:126   Not.friendliness: 58   Not.iron absorption:269
##
##
##
##
##
##          feminine             sophisticated        slimming          exciting
##  feminine    :129   Not.sophisticated: 85   No.slimming:255   exciting   :116
##  Not.feminine:171   sophisticated    :215   slimming   : 45   No.exciting:184
##
##
##
##
##
##         relaxing              effect.on.health
##  No.relaxing:113   effect on health   : 66
##  relaxing   :187   No.effect on health:234
##
##
##
##
##
```

```
# lets work with some variables
keep_columns <- c("Tea", "How", "how", "sugar", "where", "lunch")


# select the 'keep_columns' to create a new data set
tea_time <- dplyr::select(tea, all_of(keep_columns))

# look at the summaries and structure of the data
summary(tea_time)
```

```
##         Tea         How                      how           sugar
```

```
##  black    : 74   alone:195   tea bag           :170   No.sugar:155
##  Earl Grey:193   lemon: 33   tea bag+unpackaged: 94   sugar   :145
##  green    : 33   milk : 63   unpackaged        : 36
##                  other:  9
##                   where           lunch
##  chain store         :192   lunch    : 44
##  chain store+tea shop: 78   Not.lunch:256
##  tea shop            : 30
##
```

```
# visualize the data set
pivot_longer(tea_time, cols = everything()) %>%
  ggplot(aes(value)) + facet_wrap("name", scales = "free", ncol=6) +
  geom_bar() + theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 8))
```



5.7 Multiple Correspondence Analysis (MCA) with "tea" data set

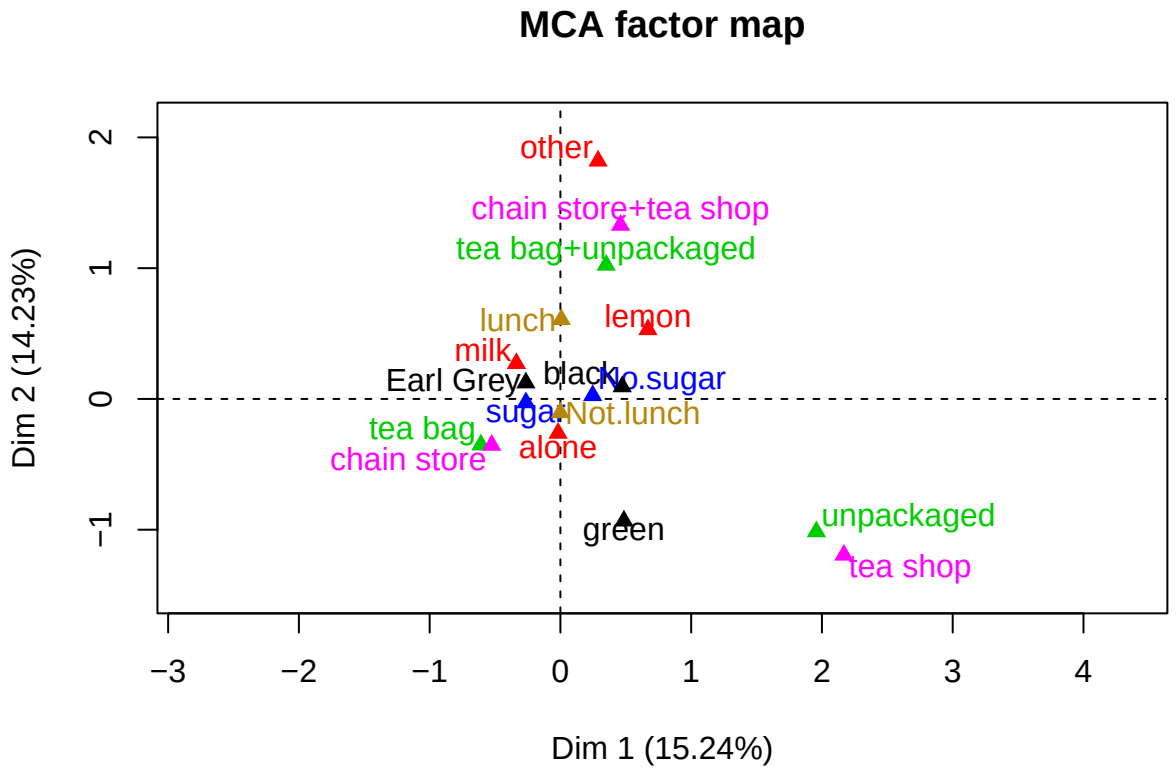
```
# multiple correspondence analysis
#library(FactoMineR), package is loaded above already, this just as note !!

mca <- MCA(tea_time, graph = FALSE)

# summary of the model
summary(mca)
```

```
##
## Call:
## MCA(X = tea_time, graph = FALSE)
```

```
##
##
## Eigenvalues
##                        Dim.1   Dim.2   Dim.3   Dim.4   Dim.5   Dim.6   Dim.7
## Variance               0.279   0.261   0.219   0.189   0.177   0.156   0.144
## % of var.             15.238  14.232  11.964  10.333   9.667   8.519   7.841
## Cumulative % of var.  15.238  29.471  41.435  51.768  61.434  69.953  77.794
##                        Dim.8   Dim.9  Dim.10  Dim.11
## Variance               0.141   0.117   0.087   0.062
## % of var.              7.705   6.392   4.724   3.385
## Cumulative % of var.  85.500  91.891  96.615 100.000
##
## Individuals (the 10 first)
##                       Dim.1    ctr   cos2    Dim.2    ctr   cos2    Dim.3
## 1                  | -0.298  0.106  0.086 | -0.328  0.137  0.105 | -0.327
## 2                  | -0.237  0.067  0.036 | -0.136  0.024  0.012 | -0.695
## 3                  | -0.369  0.162  0.231 | -0.300  0.115  0.153 | -0.202
## 4                  | -0.530  0.335  0.460 | -0.318  0.129  0.166 |  0.211
## 5                  | -0.369  0.162  0.231 | -0.300  0.115  0.153 | -0.202
## 6                  | -0.369  0.162  0.231 | -0.300  0.115  0.153 | -0.202
## 7                  | -0.369  0.162  0.231 | -0.300  0.115  0.153 | -0.202
## 8                  | -0.237  0.067  0.036 | -0.136  0.024  0.012 | -0.695
## 9                  |  0.143  0.024  0.012 |  0.871  0.969  0.435 | -0.067
## 10                 |  0.476  0.271  0.140 |  0.687  0.604  0.291 | -0.650
##                      ctr   cos2
## 1                  0.163  0.104 |
## 2                  0.735  0.314 |
## 3                  0.062  0.069 |
## 4                  0.068  0.073 |
## 5                  0.062  0.069 |
## 6                  0.062  0.069 |
## 7                  0.062  0.069 |
## 8                  0.735  0.314 |
## 9                  0.007  0.003 |
## 10                 0.643  0.261 |
##
## Categories (the 10 first)
##                         Dim.1     ctr    cos2  v.test     Dim.2     ctr    cos2
## black                |   0.473   3.288   0.073   4.677 |   0.094   0.139   0.003
## Earl Grey            |  -0.264   2.680   0.126  -6.137 |   0.123   0.626   0.027
## green                |   0.486   1.547   0.029   2.952 |  -0.933   6.111   0.107
## alone                |  -0.018   0.012   0.001  -0.418 |  -0.262   2.841   0.127
## lemon                |   0.669   2.938   0.055   4.068 |   0.531   1.979   0.035
## milk                 |  -0.337   1.420   0.030  -3.002 |   0.272   0.990   0.020
## other                |   0.288   0.148   0.003   0.876 |   1.820   6.347   0.102
## tea bag              |  -0.608  12.499   0.483 -12.023 |  -0.351   4.459   0.161
## tea bag+unpackaged   |   0.350   2.289   0.056   4.088 |   1.024  20.968   0.478
## unpackaged           |   1.958  27.432   0.523  12.499 |  -1.015   7.898   0.141
##                       v.test     Dim.3     ctr    cos2  v.test
## black                  0.929 |  -1.081  21.888   0.382 -10.692 |
## Earl Grey              2.867 |   0.433   9.160   0.338  10.053 |
## green                 -5.669 |  -0.108   0.098   0.001  -0.659 |
## alone                 -6.164 |  -0.113   0.627   0.024  -2.655 |
## lemon                  3.226 |   1.329  14.771   0.218   8.081 |
```

```
## milk                 2.422 |   0.013   0.003   0.000   0.116 |
## other                5.534 |  -2.524  14.526   0.197  -7.676 |
## tea bag             -6.941 |  -0.065   0.183   0.006  -1.287 |
## tea bag+unpackaged  11.956 |   0.019   0.009   0.000   0.226 |
## unpackaged          -6.482 |   0.257   0.602   0.009   1.640 |
##
## Categorical variables (eta2)
##                       Dim.1 Dim.2 Dim.3
## Tea                 | 0.126 0.108 0.410 |
## How                 | 0.076 0.190 0.394 |
## how                 | 0.708 0.522 0.010 |
## sugar               | 0.065 0.001 0.336 |
## where               | 0.702 0.681 0.055 |
## lunch               | 0.000 0.064 0.111 |
```

```
# visualize MCA
plot(mca, invisible=c("ind"), graph.type = "classic", habillage = "quali")
```

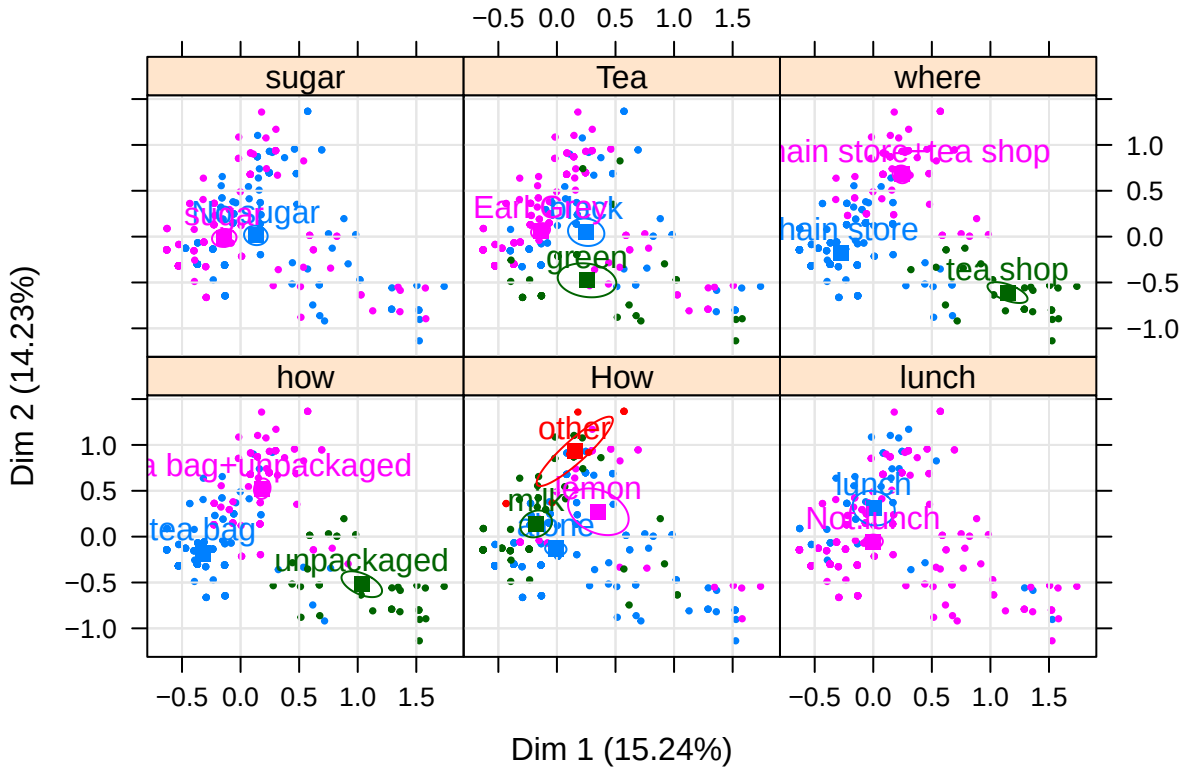



```
mca
```

```
## **Results of the Multiple Correspondence Analysis (MCA)**
## The analysis was performed on 300 individuals, described by 6 variables
## *The results are available in the following objects:
##
##    name              description
## 1  "$eig"            "eigenvalues"
## 2  "$var"            "results for the variables"
## 3  "$var$coord"      "coord. of the categories"
## 4  "$var$cos2"       "cos2 for the categories"
## 5  "$var$contrib"    "contributions of the categories"
## 6  "$var$v.test"     "v-test for the categories"
```

```
## 7  "$ind"            "results for the individuals"
## 8  "$ind$coord"      "coord. for the individuals"
## 9  "$ind$cos2"       "cos2 for the individuals"
## 10 "$ind$contrib"    "contributions of the individuals"
## 11 "$call"           "intermediate results"
## 12 "$call$marge.col" "weights of columns"
## 13 "$call$marge.li"  "weights of rows"
```

```
plotellipses(mca)
```



I have only chosen selected variables here. From the selected categories, in category where , chain store and tea shop seem to be favored. Likewise in category how, milk tea, alone and other (undefined) seemed preferred. Also how, tea bag, un-packaged seem to be preferred.
