$$MyFirstDatewithR$$
$$ExerciseBook$$

Contents

Chapter 1

Introduction

In this document you will find some exercises about these sections:

- *Your First R session*
- *Data Objects*
- *Data Import from external sources*
- *Data Manipulation with R*
- *Data Discovery with R*
- *Data Visualization with R*
- *Statistical Models with R*
- *Data Mining with R*

Chapter 2

Your first R session

2.1 Aritmetic with R

2.1.1 Exercise 1

Calculate your body mass index dividing your body mass (kg) by the square of your body height (m) (kg/m^2)

2.2 Assignment

2.2.1 Exercise 1

a. Assign your age (in number) to `age` variable.
b. Print out the value of the variable `age`.
c. Remove the variable `age` from the workspace, by using `rm()` function.

2.2.2 Exercise 2

Suppose you want to buy 10 roses and 8 sunflowers in a flower shop. The roses cost 3 euros each and the sunflowers 2 euros each.

a. Assign the total cost of roses to `roses_cost` variable and the total cost of sunflowers to `sunflowers_cost` variable.
b. Calculate the total cost of flowers by adding `roses_cost` and `sunflowers_cost` variables and assign it to `flowers_cost` variable.
c. Print out the value of the variable `flowers_cost`.
d. List the objects in the current R session, by using `ls()` function.

Chapter 3

Data Objects

3.1 Data Frames, Vectors and Factors

3.1.1 Exercise 1

a. Generate a data frame, named `df`, corresponding to:

```
##    country population continent
## 1    Italy   59801004    Europe
## 2   France   64668129    Europe
## 3    China 1382323332      Asia
## 4    Japan  126323715      Asia
## 5    Libya    6330159    Africa
## 6 Cameroon   23924407    Africa
```

Remember to maintain character vectors as they are, specifiyng `stringsAsFactors = FALSE`.

b. Supposing `dplyr` package is already installed, convert the previously defined data frame in tbl_df class.

```
require(dplyr)
```

3.1.2 Exercise 2

a. Generate a numeric vector, named `num_vec`, containing the values from 1 to 7.

b. Genarate a character vector, named `char_vec` with the days of the week.

c. Starting from the vector:

   ```
   fac <- c("F", "F", "M", "M", "F", "F", "M")
   ```

   Generate the corresponding factor, named `fac`, with two levels: "F" and "M"

d. Generate a data frame, named `df2`, containing the previously defined: `num_vec`, `char_vec` and `fac`. Remember to maintain character vectors as they are, specifiyng `stringsAsFactors = FALSE`.

e. Supposing `dplyr` package is already installed and loaded, convert the previously defined data frame in tbl_df class.

3.2 Matrices

3.2.1 Exercise 1

Generate a matrix, named `mat`, with 5 rows and 3 columns containing numbers from 1 to 15, using `matrix()` function.

3.3 Lists

3.3.1 Exercise 1

Generate a list, named `my_list` that contains the following R elements:

```
char <- "Veronica"
mat <- matrix(1:9, ncol = 3)
log_vec <- c(TRUE, FALSE, TRUE, TRUE)
```

Chapter 4

Data Import from external sources

First of all, set your working directory in the *data* folder, using `setwd()` function, like in this example

```
setwd("C:/Users/Veronica/Documents/rbase/data")
```

We will work inside this folder.

4.1 Text Files

4.1.1 Exercise 1

a. Import text file named *"tuscany.txt"* and save it in an R object named `tuscany_df`.
   Open the text file before importing it to control if the first row contains column names and to control the field and the decimal separator characters. Remember to not import the character columns as factors.
b. Visualize the first rows of `tuscany_df`

4.1.2 Exercise 2

a. Import text file named *"solar.txt"* and save it in an R object `solar_df`.
   Open the text file before importing it to control if the first row contains column names and to control the field and the decimal separator characters. Remember to not import the character columns as factors.
b. Visualize the first rows of `solar_df`.

4.2 Excel Files

4.2.1 Exercise 1

a. Import `iris` sheet of .xlsx file *"flowers.xlsx"* by using `read_excel` function of `readxl` package and save it in a R object named `flowers`.
   Remember to load `read_excel` package, supposing it is already installed.

```
require(readxl)
```

b. Visualize the first rows of `flowers`

4.3 Databases

4.3.1 Exercise 1

a. Connect to *"plant.sqlite"* SQLite database, using `dbConnect()` function of `RSQLite` package. Save the connection in an R object, named `con`.
   Remember to load `RSQLite` package, supposing it is already installed.

```
require(RSQLite)
```

b. See the list of available tables in *"plant.sqlite"* db, using `dbListTables()` function.

c. See list of fields in *"PlantGrowth"* table of *"plant.sqlite"* db, using `dbListFields()` function.

d. Send query to *"PlantGrowth"* table of *"plant.sqlite"* which select the records with `weight` greater than 5.5.

e. Disconnect from the database, using `dbDisconnect()` function.

Chapter 5

Data Manipulation with R

Load `dplyr` package, supposing it is already installed.

```
require(dplyr)
```

5.1 Data

All the following exercises are based on the `nycflights13` data, taken from the `nycflights13` package. So first of all, install and load this package

```
install.packages("nycflights13")
require(nycflights13)
```

The `nycflights13` package contains information about all flights that departed from NYC (e.g. EWR, JFK and LGA) in 2013: 336,776 flights in total.

```
ls(pos = "package:nycflights13")
```

```
## [1] "airlines" "airports" "flights"  "planes"   "weather"
```

To help understand what causes delays, it includes a number of useful datasets:

- `flights`: information about all flights that departed from NYC
- `weather`: hourly meterological data for each airport;
- `planes`: construction information about each plane;
- `airports`: airport names and locations;
- `airlines`: translation between two letter carrier codes and names.

Let us explore the features of `flights` datasets, which will be used in the following exercises.

```
data("flights")
```

5.1.1 flights

This dataset contains on-time data for all flights that departed from NYC (i.e. JFK, LGA or EWR) in 2013. The data frame has 16 variables and 336776 observations. The variables are organised as follow:

- Date of departure: `year`, `month`, `day`;

- Departure and arrival times (local tz): `dep_time`, `arr_time`;
- Departure and arrival delays, in minutes: `dep_delay`, `arr_delay` (negative times represent early departures/arrivals);
- Time of departure broken in to hour and minutes: `hour`, `minute`;
- Two letter carrier abbreviation: `carrier`;
- Plane tail number: `tailnum`;
- Flight number: `flight`;
- Origin and destination: `origin`, `dest`;
- Amount of time spent in the air: `air_time`;
- Distance flown: `distance`.

```
dim(flights)
```

```
## [1] 336776     16
```

```
head(flights)
```

```
##   year month day dep_time dep_delay arr_time arr_delay carrier tailnum flight
## 1 2013     1   1      517         2      830        11      UA  N14228   1545
## 2 2013     1   1      533         4      850        20      UA  N24211   1714
## 3 2013     1   1      542         2      923        33      AA  N619AA   1141
## 4 2013     1   1      544        -1     1004       -18      B6  N804JB    725
## 5 2013     1   1      554        -6      812       -25      DL  N668DN    461
## 6 2013     1   1      554        -4      740        12      UA  N39463   1696
##   origin dest air_time distance hour minute
## 1    EWR  IAH      227     1400    5     17
## 2    LGA  IAH      227     1416    5     33
## 3    JFK  MIA      160     1089    5     42
## 4    JFK  BQN      183     1576    5     44
## 5    LGA  ATL      116      762    5     54
## 6    EWR  ORD      150      719    5     54
```

```
str(flights)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    336776 obs. of  16 variables:
##  $ year     : int  2013 2013 2013 2013 2013 2013 2013 2013 2013 2013 ...
##  $ month    : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ day      : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ dep_time : int  517 533 542 544 554 554 555 557 557 558 ...
##  $ dep_delay: num  2 4 2 -1 -6 -4 -5 -3 -3 -2 ...
##  $ arr_time : int  830 850 923 1004 812 740 913 709 838 753 ...
##  $ arr_delay: num  11 20 33 -18 -25 12 19 -14 -8 8 ...
##  $ carrier  : chr  "UA" "UA" "AA" "B6" ...
##  $ tailnum  : chr  "N14228" "N24211" "N619AA" "N804JB" ...
##  $ flight   : int  1545 1714 1141 725 461 1696 507 5708 79 301 ...
##  $ origin   : chr  "EWR" "LGA" "JFK" "JFK" ...
##  $ dest     : chr  "IAH" "IAH" "MIA" "BQN" ...
##  $ air_time : num  227 227 160 183 116 150 158 53 140 138 ...
##  $ distance : num  1400 1416 1089 1576 762 ...
##  $ hour     : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ minute   : num  17 33 42 44 54 54 55 57 57 58 ...
```

5.2 `select()`

5.2.1 Exercise 1

Extract the following information:

- month;
- day;
- air_time;
- distance.

5.2.2 Exercise 2

Extract all information about `flights` except hour and minute.

5.2.3 Exercise 3

Extract `tailnum` variable and rename it into `tail_num`

5.3 `filter()`

5.3.1 Exercise 1

Select all flights which delayed more than 1000 minutes at departure.

5.3.2 Exercise 2

Select all flights which delayed more than 1000 minutes at departure or at arrival.

5.3.3 Exercise 3

Select all flights which took off from "EWR" and landed in "IAH".

5.4 `arrange()`

5.4.1 Exercise 1

Sort the flights in chronological order.

5.4.2 Exercise 2

Sort the flights by decreasing arrival delay.

5.4.3 Exercise 3

Sort the flights by origin (in alphabetical order) and decreasing arrival delay.

Chapter 6

Data Discovery with R

Load `dplyr` package, supposing it is already installed.

```
require(dplyr)
```

6.1 Data

Also these exercises are based on the `nycflights13` data, taken from the `nycflights13` package.
Load `nycflights13` package, supposing it is already installed.

```
require(nycflights13)
```

The `nycflights13` package contains information about all flights that departed from NYC (e.g. EWR, JFK and LGA) in 2013: 336,776 flights in total. For more information see *Data Manipulation with R* section.

The following exercises refers to `flights` dataset:

```
data("flights")
```

6.2 Descriptive statistics with `summarise()` and `group_by()`

6.2.1 Exercise 1

Calculate the mean delay at arrival (`arr_delay` variable). Remember to add `na.rm=TRUE` option to all calculations.

6.2.2 Exercise 2

Calculate the summary (minimum, first quartile, median, mean, third quartile, maximum and standard deviation) of delay at departure (`dep_delay` variable) for flights.
Remember to add `na.rm=TRUE` option to mean calculations.

6.2.3 Exercise 3

Calculate minimum and maximum delay at departure (`arr_delay` variable) for flights by month.
Remember to add `na.rm=TRUE` option to all calculations.

6.3 Multiple operations

6.3.1 Exercise 1

For each destination (`dest` variable), compute the mean delay at arrival (`arr_delay` variable) and filter the mean delays greater than 30 minutes.
Remember to add `na.rm=TRUE` option to mean calculations.

6.3.2 Exercise 2

Filter the observations recorded on June 13 and count the number of flights (use `n()` function inside `summarise()`) for each destination. Then sort the result in ascending order.

Chapter 7

Data Visualization with ggplot2

Load ggplot2 package, supposing it is already installed.

```
require(ggplot2)
```

7.1 Data

7.1.1 iris

Almost all the following exercises are based on the iris dataset, taken from the datasets package.
It is a base package so it is already installed and loaded.

```
data("iris")
```

This dataset gives the measurements in centimeters of length and width of sepal and petal, respectively, for 50 flowers from each of 3 species of iris. The species are Iris setosa, versicolor, and virginica.

iris dataset contains the following variables:

- Sepal.Length: length of iris sepal
- Sepal.Width: width of iris sepal
- Petal.Length: length of iris petal
- Petal.Width: width of iris petal
- Species: species of iris

```
dim(iris)
```

```
## [1] 150   5
```

```
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

7.1.2 mpg

Some of the exercises are based on `mpg` dataset, taken from the `ggplot2` package.

```
data("mpg")
```

This dataset contains the fuel economy data from 1999 and 2008 for 38 popular models of car.
`mpg` dataset contains the following variables:

- `manufacturer`
- `model`
- `displ`: engine displacement, in litres
- `year`
- `cyl`: number of cylinders
- `trans`: type of transmission
- `drv`: drivetrain type, f = front-wheel drive, r = rear wheel drive, 4 = 4wd
- `cty`: city miles per gallon
- `hwy`: highway miles per gallon
- `fl`: fuel type

```
dim(mpg)
```

```
## [1] 234  11
```

```
head(mpg)
```

```
## # A tibble: 6 × 11
##   manufacturer model displ  year   cyl      trans   drv   cty   hwy    fl
##          <chr> <chr> <dbl> <int> <int>      <chr> <chr> <int> <int> <chr>
## 1         audi    a4   1.8  1999     4   auto(l5)     f    18    29     p
## 2         audi    a4   1.8  1999     4 manual(m5)     f    21    29     p
## 3         audi    a4   2.0  2008     4 manual(m6)     f    20    31     p
## 4         audi    a4   2.0  2008     4   auto(av)     f    21    30     p
## 5         audi    a4   2.8  1999     6   auto(l5)     f    16    26     p
## 6         audi    a4   2.8  1999     6 manual(m5)     f    18    26     p
## # ... with 1 more variables: class <chr>
```

```
str(mpg)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    234 obs. of  11 variables:
##  $ manufacturer: chr  "audi" "audi" "audi" "audi" ...
##  $ model       : chr  "a4" "a4" "a4" "a4" ...
##  $ displ       : num  1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...
```

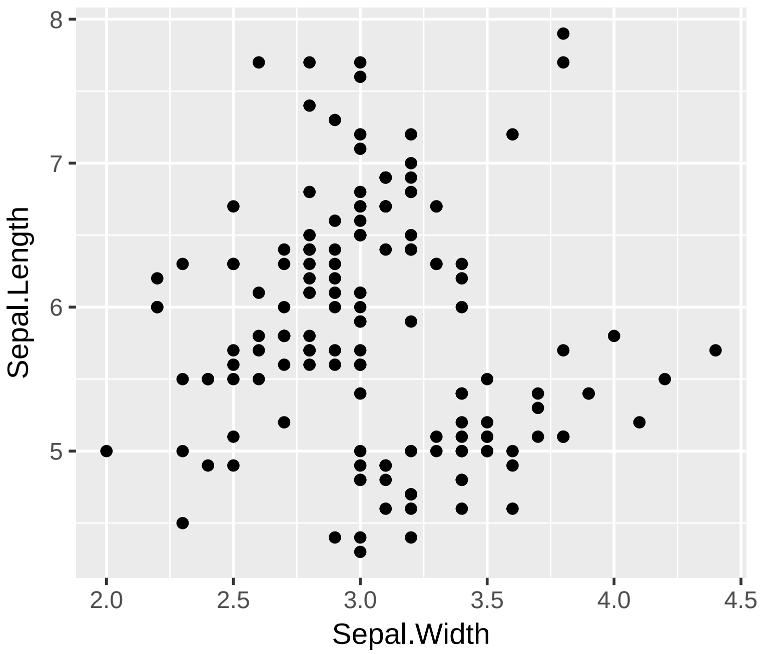
```
## $ year        : int  1999 1999 2008 2008 1999 1999 2008 1999 1999 2008 ...
## $ cyl         : int  4 4 4 4 6 6 6 4 4 4 ...
## $ trans       : chr  "auto(l5)" "manual(m5)" "manual(m6)" "auto(av)" ...
## $ drv         : chr  "f" "f" "f" "f" ...
## $ cty         : int  18 21 20 21 16 18 18 18 16 20 ...
## $ hwy         : int  29 29 31 30 26 26 27 26 25 28 ...
## $ fl          : chr  "p" "p" "p" "p" ...
## $ class       : chr  "compact" "compact" "compact" "compact" ...
```

7.2 Scatterplot

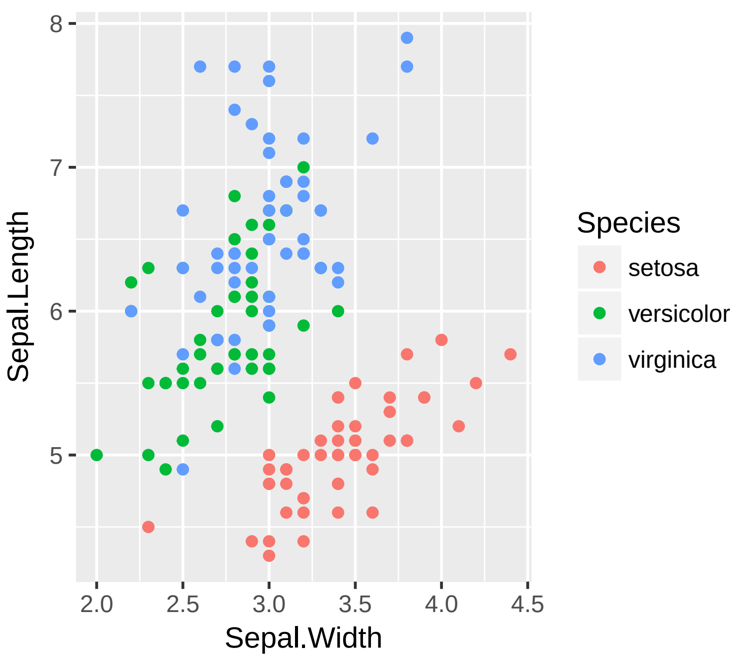
7.2.1 Exercise 1

Let us consider `iris` dataset.

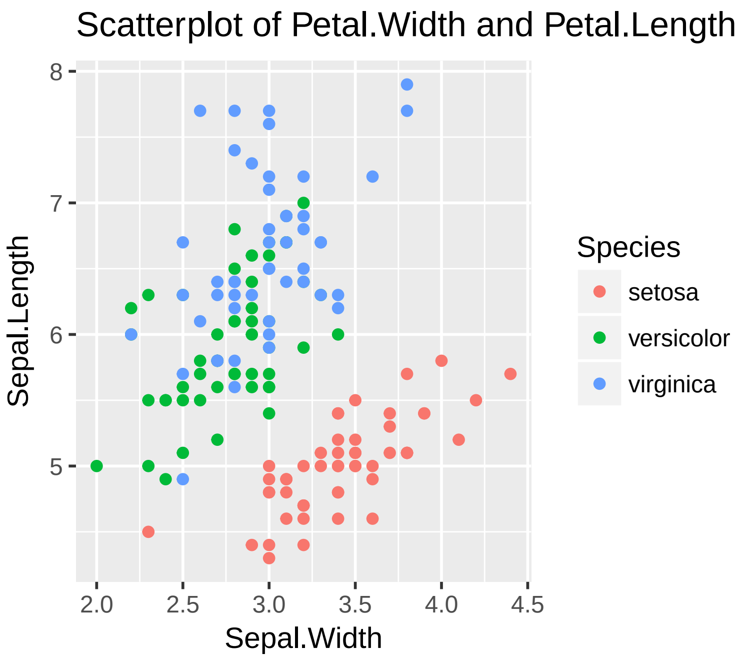
a. Generate a scatterplot to analyze the relationship between `Sepal.Width` and `Sepal.Length` variables.



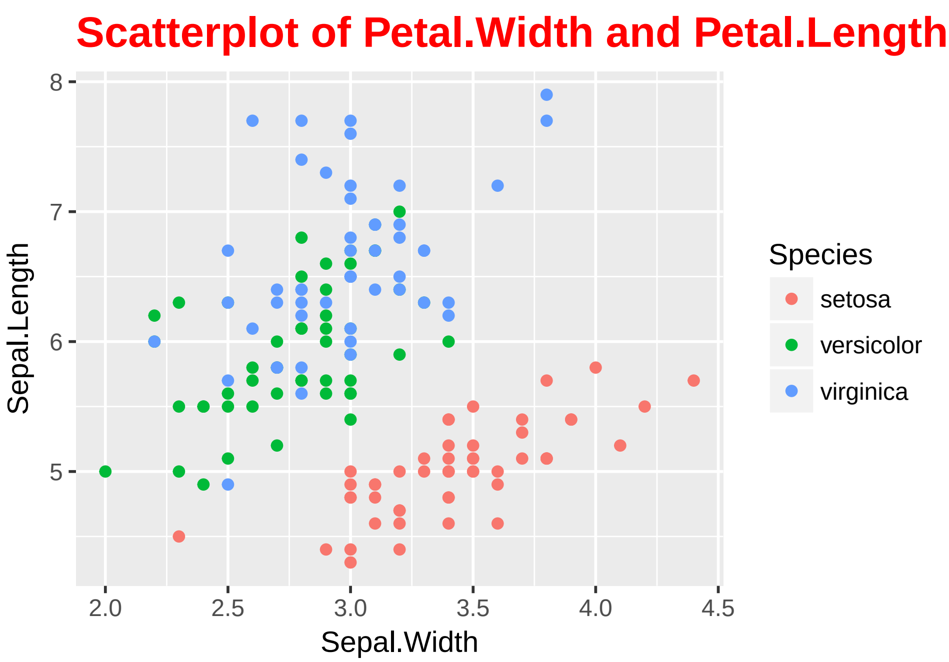
b. Map `Species` to `colour` in `aes()`.



c. Add the title to the plot: `"Scatterplot of Petal.Width and Petal.Length"` (use `ggtitle()` function).



d. Customize plot title by adding `theme(plot.title = element_text())` to the plot and setting `colour` argument to `"red"`, `size` to 16 and `face` to `"bold"`.

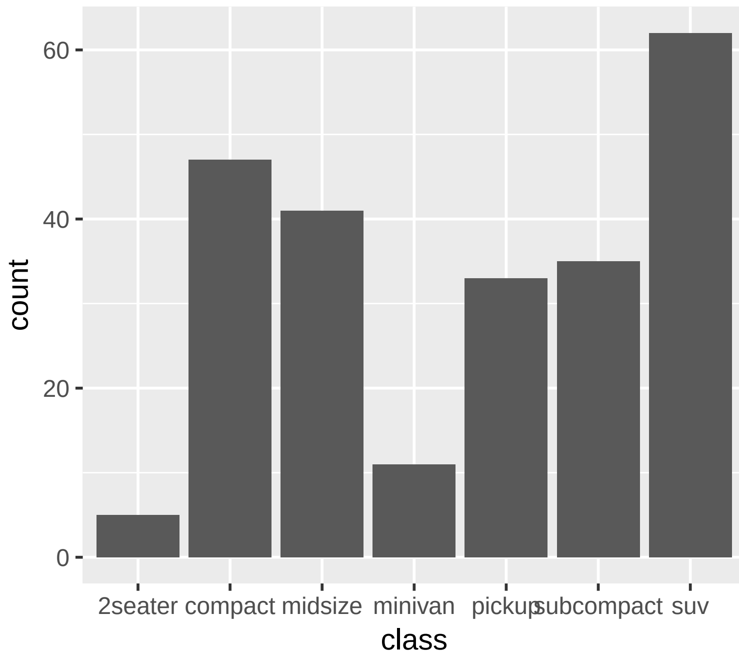


7.3 Barplot

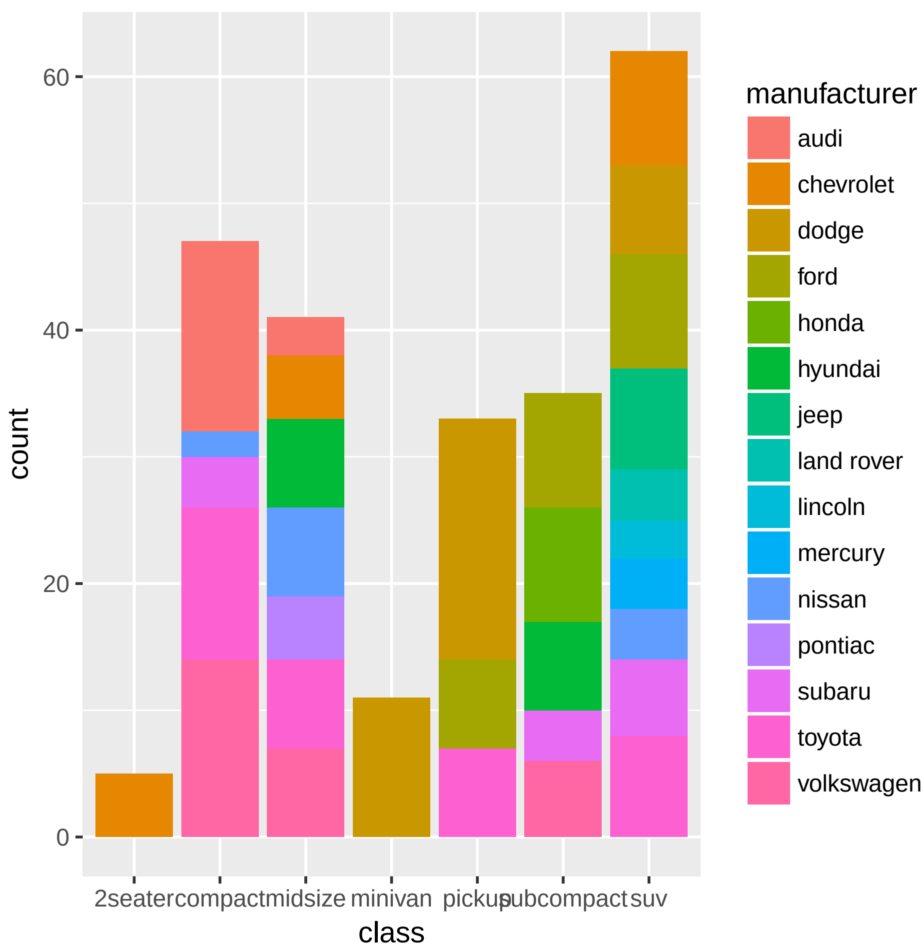
7.3.1 Exercise 1

Let us consider `mpg` dataset.

a. Represent graphically with a barplot the number of cars for each class.



b. Represent graphically with a barplot, the distribution of manufacturer for each class (map `manufacturer` variable to `fill`).

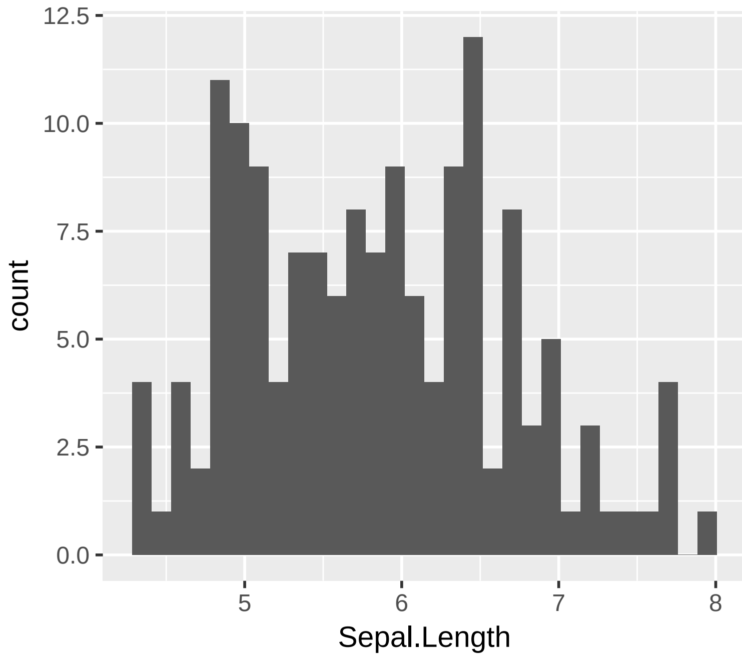


7.4 Histogram

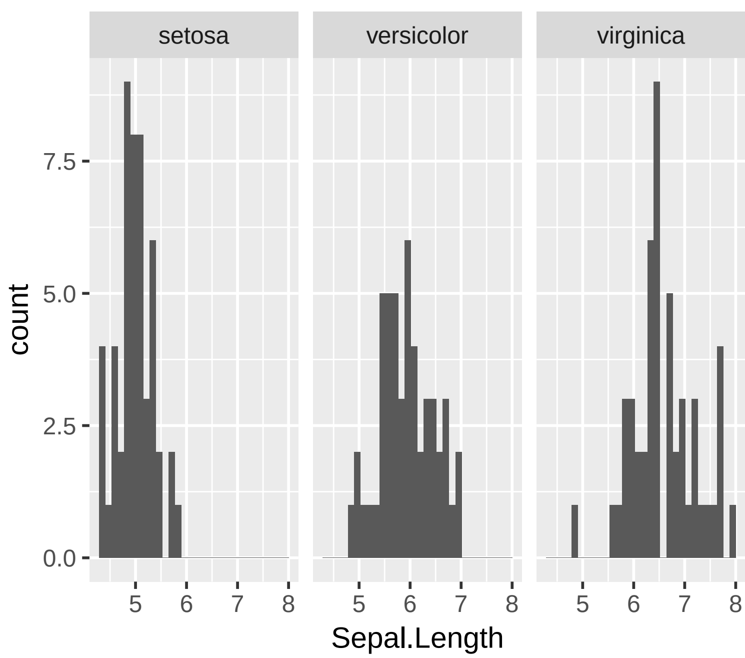
7.4.1 Exercise 1

Let us consider `iris` dataset.

a. Represent the distribution of `Sepal_Length` variable with an histogram.



b. Represent each level of `Species` variable in a different panel. Use `facet_grid()` function.

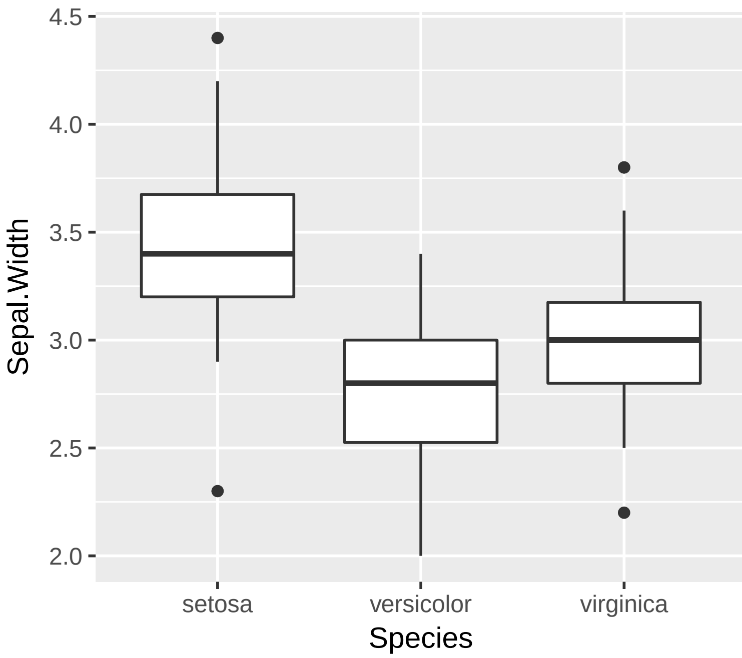


7.5 Boxplot

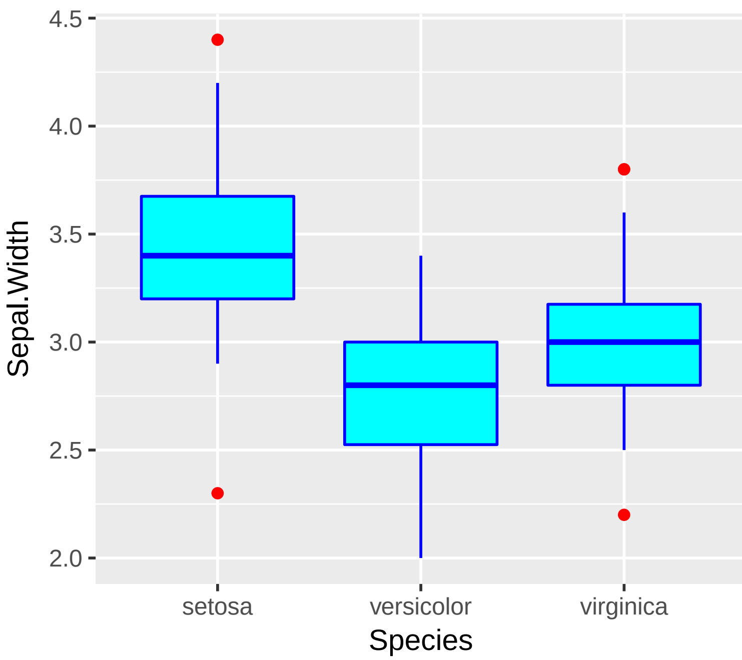
7.5.1 Exercise 1

Let us consider `iris` dataset.

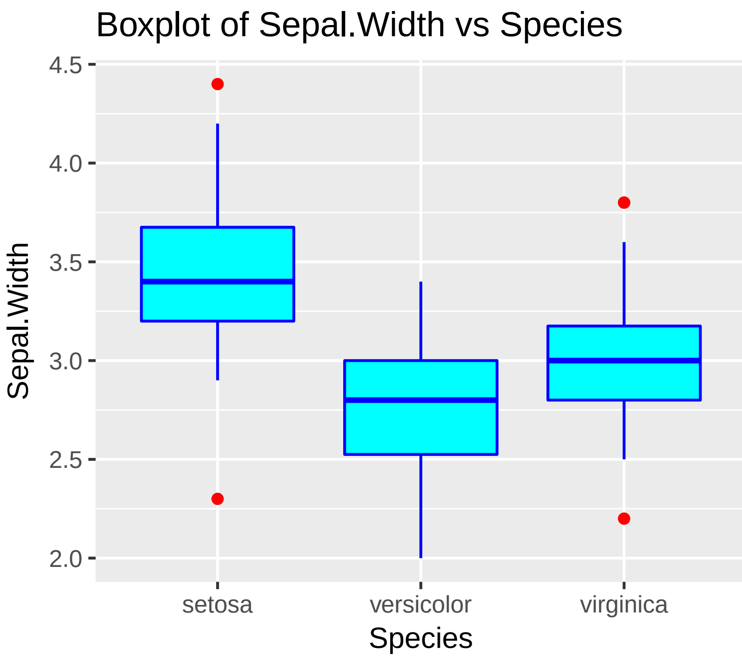
a. Build a boxplot to compare the differences of sepal width accordingly to the type of iris species.



b. Set the fill colour of boxes as `"#00FFFF"`, the lines colour of boxes as `"#0000FF"` and the outliers colour as `"red"`.



c. Add the plot title: `"Boxplot of Sepal.Width vs Species"`.


Boxplot of Sepal.Width vs Species
4.5
4.0
3.5
3.0
2.5
2.0
Sepal.Width
setosa
versicolor
virginica
Species

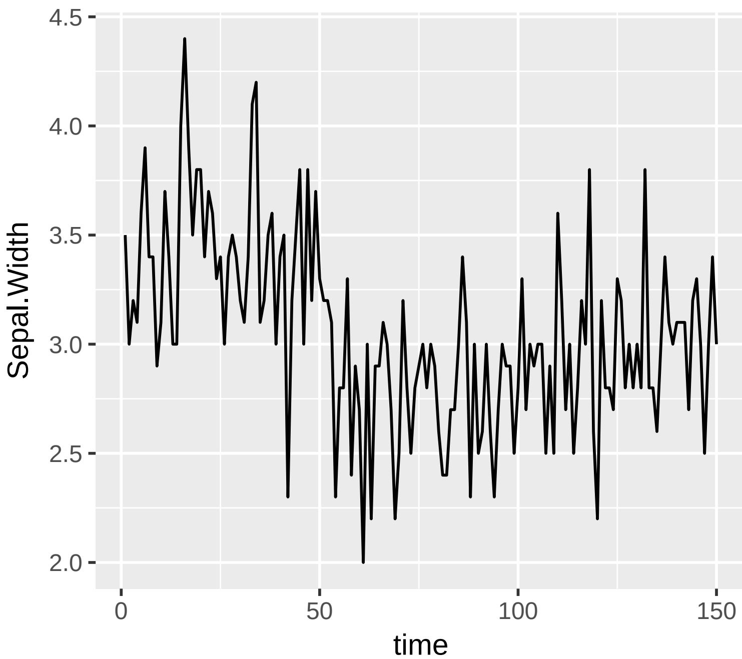
7.6 Lineplot

7.6.1 Exercise 1

Let us suppose that the observations on `iris` are taken along time.
So let us consider the following dataset, named `iris2`, in which `time` variable is added:

```
require(dplyr)
iris2 <- iris %>% mutate(time=1:150)
```

a. Build a lineplot to visualize the measures of `Sepal.Length` variable along time.



Chapter 8

Statistical models

Before starting the exercises, load the following libraries, supposing they are already installed.

```
require(dplyr)
require(ggplot2)
require(qdata)
```

8.1 Linear Models

8.1.1 Exercise 1

The number of impurities (lumps) present in the containers of paint depends on the rate of agitation applied to the container. A researcher wants to determine the relation between the rate of agitation and the number of lumps, so he conducts an experiment. He applies different rates of agitation (`Stirrate`) to 12 containers of paint and he counts the number of impurities (lumps) present in the containers of paint (`Impurity`).

```
data(paint)
head(paint)
```

```
## # A tibble: 6 × 2
##   Stirrate Impurity
##      <int>    <dbl>
## 1       20      8.4
## 2       38     16.5
## 3       36     16.4
## 4       40     18.9
## 5       42     18.5
## 6       26     10.4
```

a. Let us compute the main descriptive statistics of `Impurity`.
b. Let us graphically represent the relation between `Impurity` and `Stirrate` variables (add regression line to the scatterplot).
c. Let us compute a simple linear regression between `Impurity` and `Stirrate`.
d. Does `Stirrate` influence `Impurity`? How? Let us analyze the model fitted by using `summary()` function.
e. Let us check (final) models residuals.

8.1.2 Exercise 2

A pressure switch has a membrane whose thickness (in mm) influences the pressure required to trigger the switch itself. The aim is to determine the thickness of the membrane for which the switch "trig" with a pressure equal to 165 ± 15 KPa. 25 switches with different thickness (`DThickness`) of the membrane was analysed, measuring the pressure at which each switch opens (KPa) (`SetPoint`).

```
data(switcht)
head(switcht)
```

```
## # A tibble: 6 × 2
##   DThickness SetPoint
##        <dbl>    <dbl>
## 1        0.9  223.523
## 2        0.6  157.131
## 3        0.5  149.307
## 4        0.8  200.146
## 5        0.8  199.974
## 6        0.7  166.919
```

a. Let us compute the descriptive statistics of `SetPoint` variable.
b. Let us graphically represent the relation between `DThickness` and `SetPoint`(add regression line to the graph).
c. Let us compute a linear regression between `DThickness` and `SetPoint` and check the residuals of the fitted model.
d. Does `DThickness` influences `SetPoint`? Let us analyze the model fitted by using `summary()` function.
e. Let us check (final) models residuals.

Chapter 9

Data Mining

Before starting the exercises, load the following libraries, supposing they are already installed.

```
require(qdata)
require(dplyr)
require(ggplot2)
require(nnet)
```

9.1 Neural Networks

9.1.1 Exercise 1

Consider `iris` dataset.

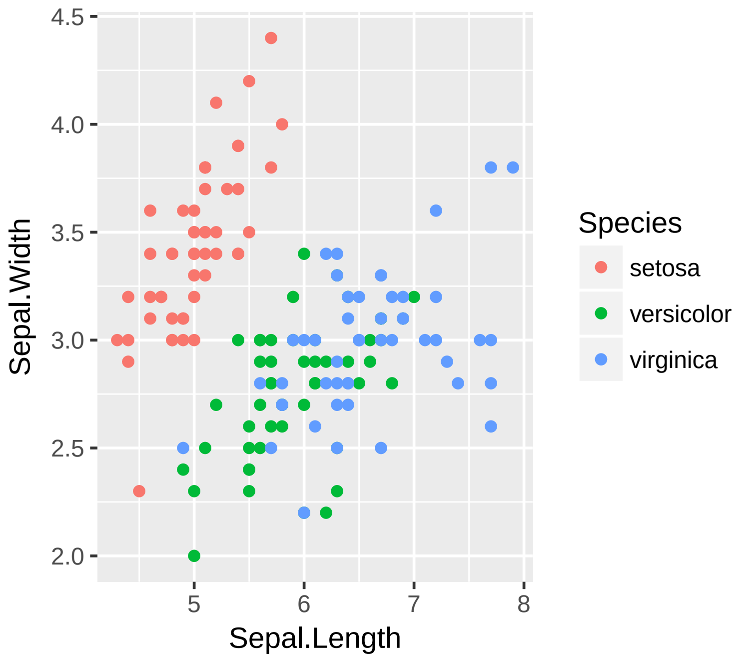
```
data(iris)
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

A botanist wants to to find a prediction model to assess the probability of belonging to a specific species, for each flower, based on its sepal and petal features.

a. Analyze the relationship between `Species` and the other variables of `iris` dataset. The following lines of code produces a scatterplot of `Sepal.Length` and `Sepal.Width` by `Species`.

   ```
   ggplot(data=iris, mapping=aes (x=Sepal.Length, y=Sepal.Width, colour=Species)) +
     geom_point()
   ```



Generate a scatterplot to analyze the relationship between `Petal.Length` and `Petal.Width` by `Species`. Comment the results.

b. Divide the dataset in train and test dataset in this way:

```
set.seed(1)
samp <- c(sample(1:50,25), sample(51:100,25), sample(101:150,25))
train <- iris[samp,]
test <- iris[-samp,]
```

and estimate a Neural Network model on train sample to assess the probability of belonging to a specific species, for each flower, based on its measures of `Sepal.Length`, `Sepal.Width`, `Petal.Length`, and `Petal.Width`. Use `nnet()` function and set the `size` (number of units in the hidden layer) to 2.

c. Use `predict()` function to gain the predictions on test sample. Add `type = "class"` argument to `predict()` function. Add the prediction estimated to `test` dataset.

d. Built a frequency table to compare the original distribution of `Species` and that predicted in `test` data. Comment the results.