Data Programming Course Exercises

May 4, 2016

Andrea Spanò
andrea.spano@quantide.com[1]

[1] mailto:andrea.spano@quantide.com

Contents

Chapter 1

Introduction

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

- *Data Manipulation*
- *Data Visualization with ggplot2*
- *Writing R functions*

Chapter 2

Data Manipulation with `dplyr`

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

```
require(dplyr)
```

2.1 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")
```

2.1.1 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 ...
```

2.2 Select

2.2.1 Exercise 1

Extract the following information:

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

2.2.2 Exercise 2

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

2.2.3 Exercise 3

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

2.3 Filter

2.3.1 Exercise 1

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

2.3.2 Exercise 2

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

2.3.3 Exercise 3

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

2.4 Arrange

2.4.1 Exercise 1

Sort the flights in chronological order.

2.4.2 Exercise 2

Sort the flights by decreasing arrival delay.

2.4.3 Exercise 3

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

2.5 Mutate

2.5.1 Exercise 1

Add the following new variable to the `flights` dataset:

- the speed in miles per hour, named `speed` (`distance` / `air_time` * 60).

Consider that times are in minutes and distances are in miles.

2.5.2 Exercise 2

Add the following new variables to the `flights` dataset:

- the gained time in minutes (named `gain`), defined as the difference between delay at departure and delay at arrival;
- the gain time per hours, defined as `gain` / (`air_time` / 60)

2.6 Summarise

2.6.1 Exercise 1

Calculate minimum, mean and maximum delay at arrival. Remember to add `na.rm=TRUE` option to all calculations.

2.7 Group_by

2.7.1 Exercise 1

Calculate number of flights, minimum, mean and maximum delay at departure for flights by month.
Remember to add `na.rm=TRUE` option to all calculations.

2.7.2 Exercise 2

Calculate number of flights (using `n()` operator), mean delay at departure and arrival for flights by origin.
Remember to add `na.rm=TRUE` option to mean calculations.

2.8 Chain multiple operations (%>%)

2.8.1 Exercise 1

Calculate number of flights, minimum, mean and maximum delay at departure for flights by month.
Remember to add `na.rm=TRUE` option to all calculations.

2.8.2 Exercise 2

Calculate the monthly mean gained time in minutes, where the gained time is defined as the difference between delay at departure and delay at arrival. Remember to add `na.rm=TRUE` option to mean calculations.

2.8.3 Exercise 3

For each destination, select all days where the mean delay at arrival is greater than 30 minutes.
Remember to add `na.rm=TRUE` option to mean calculations.

Chapter 3

Data Visualization with ggplot2

Load ggplot2 package, supposing it is already installed.

```
require(ggplot2)
```

3.1 Data

3.1.1 iris

Almost all the following exercises are based on the iris data, 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 ...
```

3.1.2 mpg

Some of the exercises are based on `mpg` dataset, taken from the `datasets` package.
It is a base package so it is already installed and loaded.

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

This dataset contains the fuel economy data from 1999 and 2008 for 38 popular models of car.

```
dim(mpg)
```

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

```
head(mpg)
```

```
##   manufacturer model displ year cyl      trans drv cty hwy fl   class
## 1         audi    a4   1.8 1999   4   auto(l5)   f  18  29  p compact
## 2         audi    a4   1.8 1999   4 manual(m5)   f  21  29  p compact
## 3         audi    a4   2.0 2008   4 manual(m6)   f  20  31  p compact
## 4         audi    a4   2.0 2008   4   auto(av)   f  21  30  p compact
## 5         audi    a4   2.8 1999   6   auto(l5)   f  16  26  p compact
## 6         audi    a4   2.8 1999   6 manual(m5)   f  18  26  p compact
```

```
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" ...
```

3.2 Scatterplot

3.2.1 Exercise 1

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

b. Set the size of the point as 3 and their colour (`colour` and `fill` arguments as “green”).

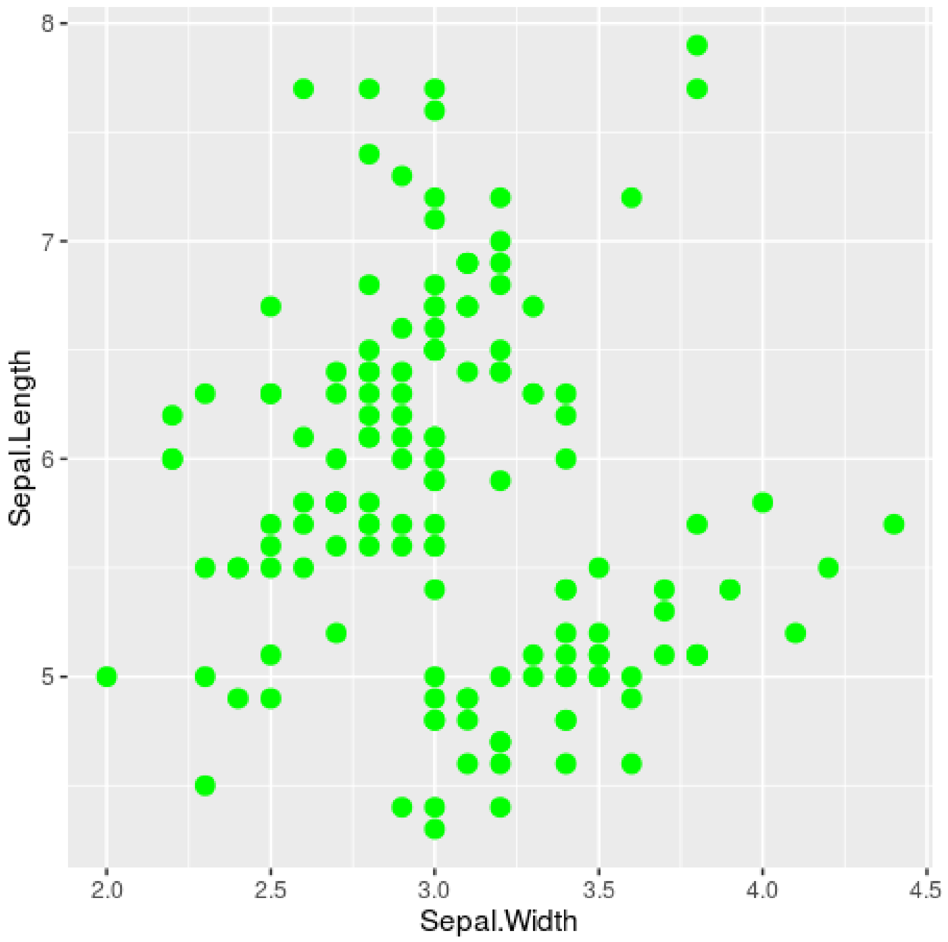

Figure 3.1:

3.2.2 Exercise 2

a. Generate a scatterplot to analyze the relationship between `Petal.Width` and `Petal.Length` variables according to iris species, mapped as `colour` aes.

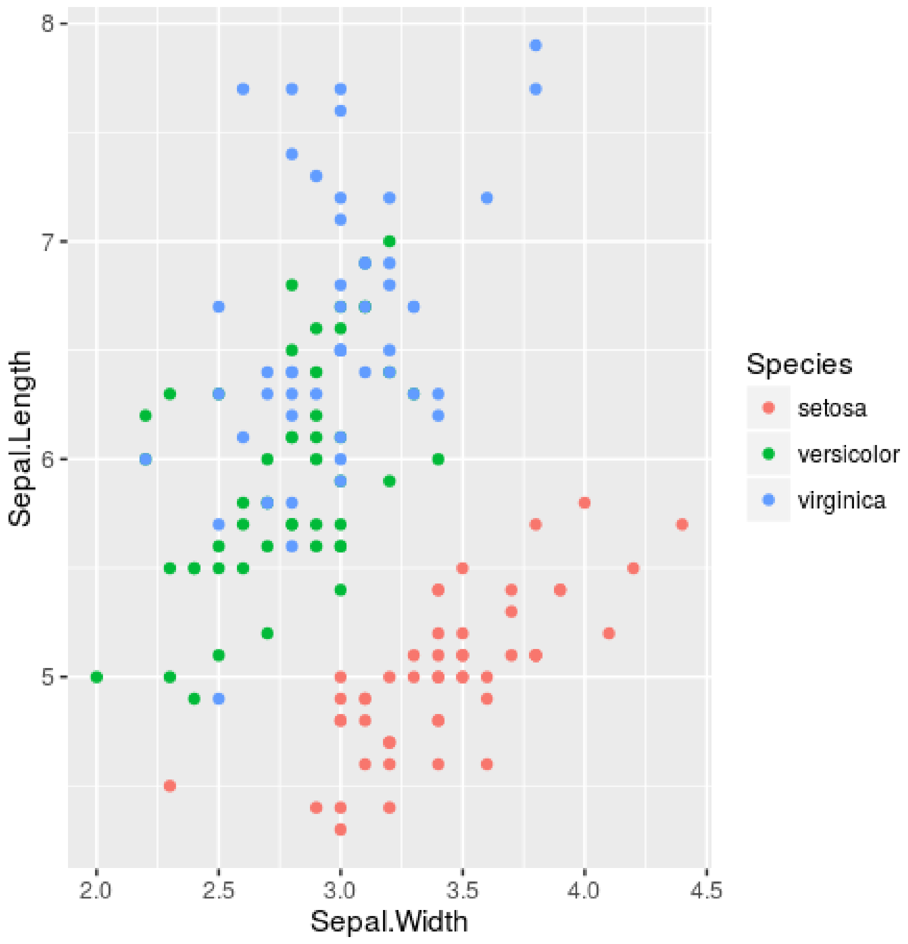

Figure 3.2:

3.3 Box Plot

3.3.1 Exercise 1

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

b. Set the fill of boxes as “#00FFFF”, the colour as “#0000FF” and the outlier colours as “red”.

c. Add the plot title: “Boxplot of Sepal.Width vs Species”

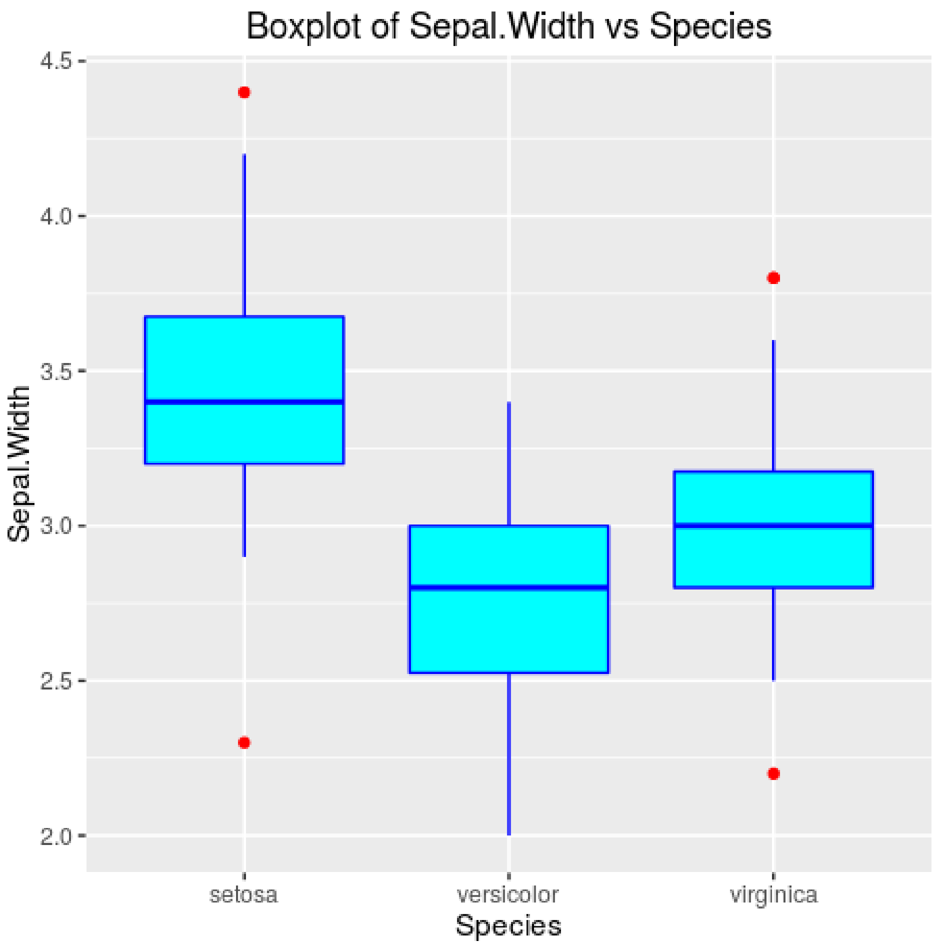

Figure 3.3:

3.4 Histogram

3.4.1 Exercise 1

a. Represent the distribution of sepal length with an histogram.

b. Set bins fill as “hotpink” and colour as “deeppink”.

c. Set the number of bins as 15.

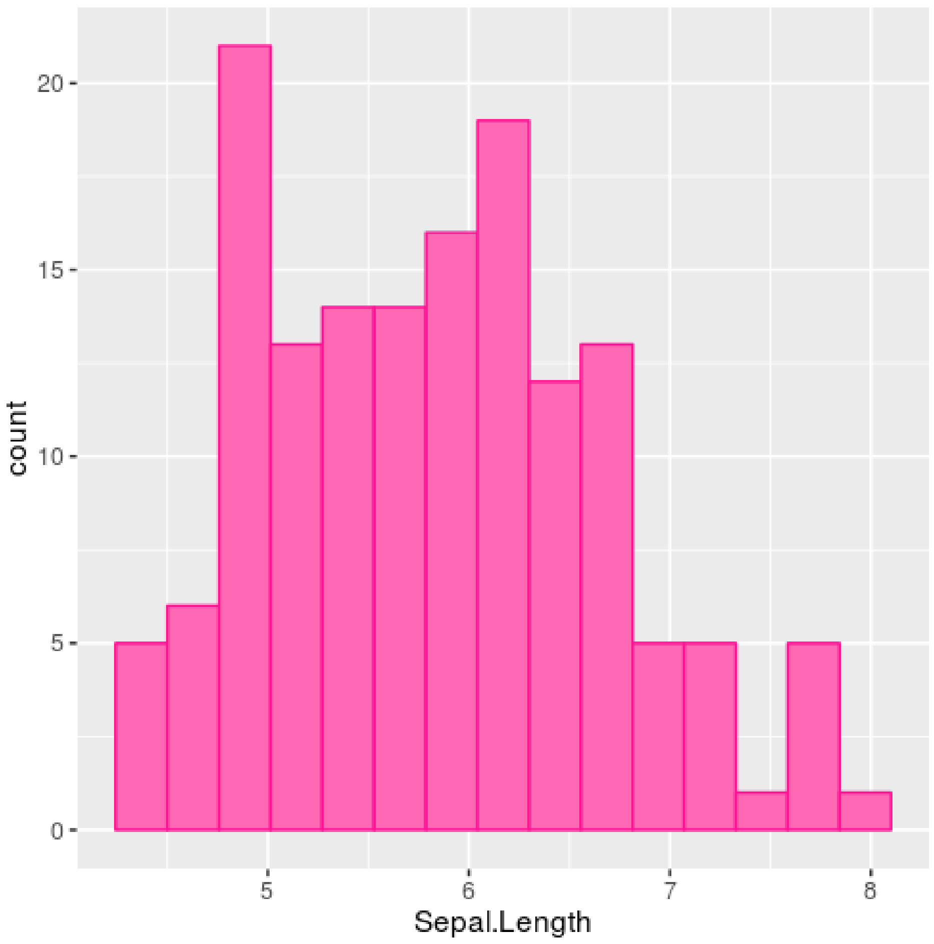

Figure 3.4:

3.5 Lineplot

3.5.1 Exercise 1

Let us suppose that the observations on flowers are taken along time, so let us consider the following dataset:

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

a. Build a line plot to visualize the `Sepal.Length` along time.

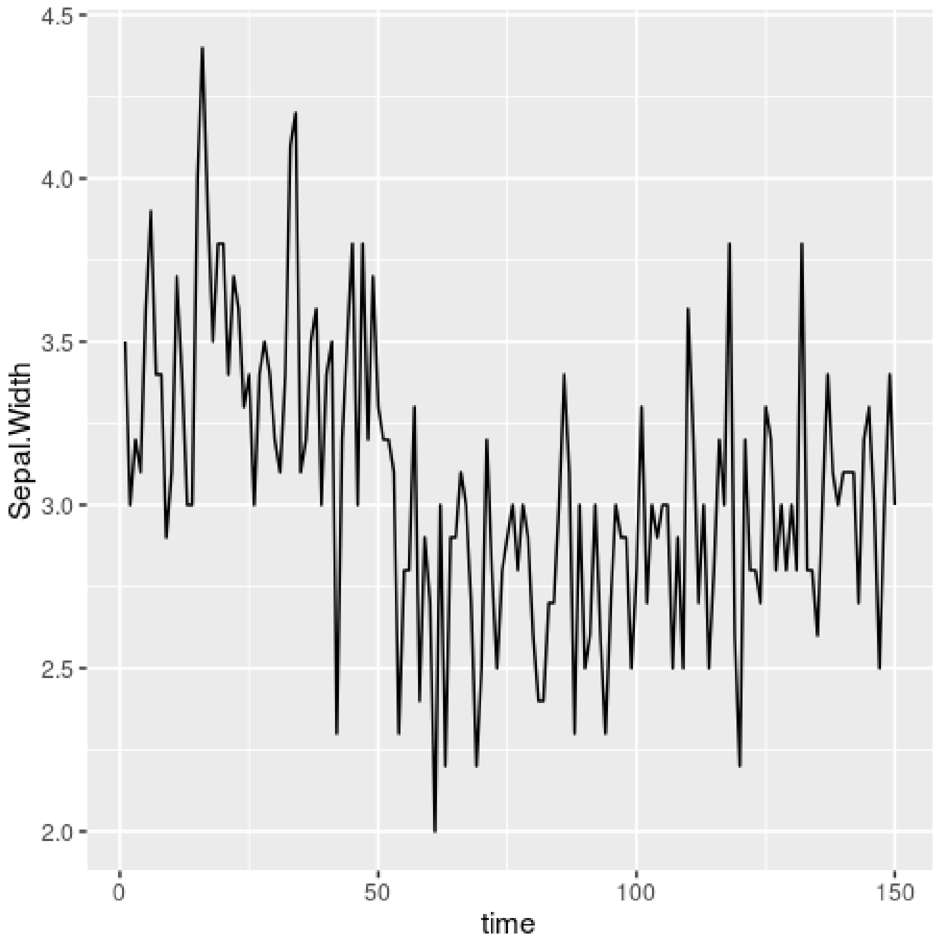

Figure 3.5:

3.5.2 Exercise 2

Let us suppose that the observations on flowers are taken along time, so let us consider the following dataset:

```
iris3 <- iris %>% mutate(time=rep(1:50, times=3))
```

a. Build a line plot to visualize the `Sepal.Length` along time, according to the `Species`.

b. Set linetype as “twodash”.

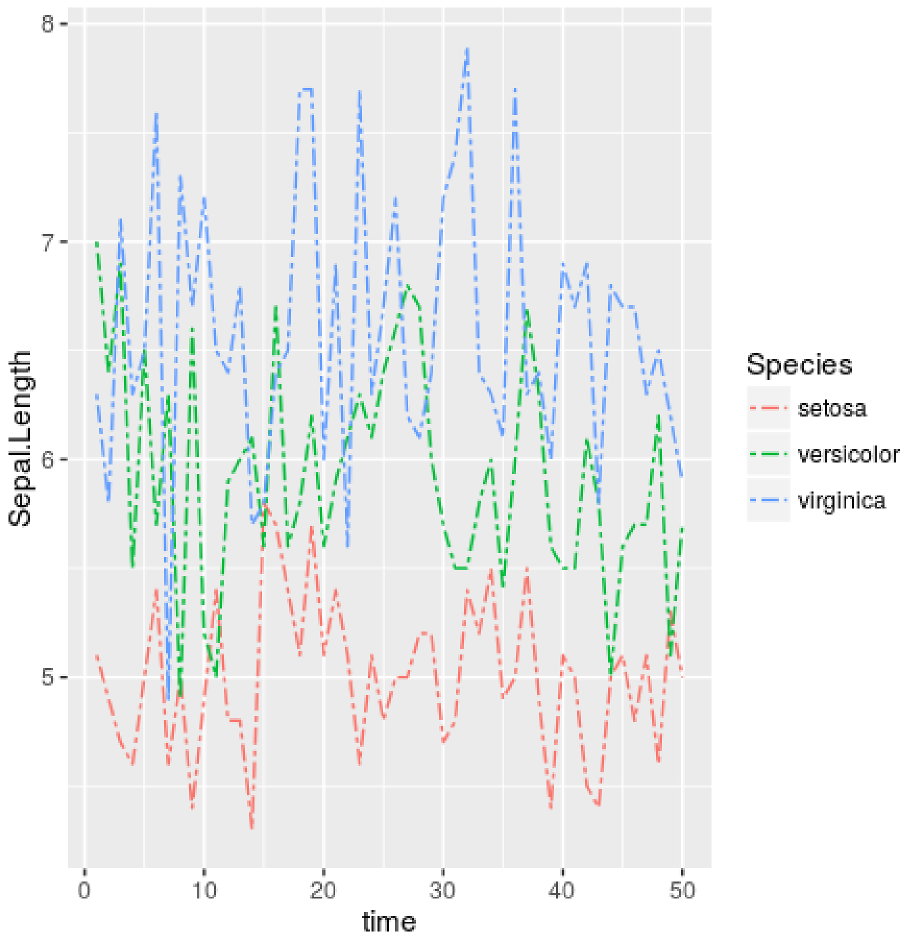

Figure 3.6:

3.6 Bar graph

Let us consider `mpg` dataset.

3.6.1 Exercise 1

a. Represent graphically with a bar graph, how many cars there are for each class.
b. Represent horizontal bar and set bar width as 0.6

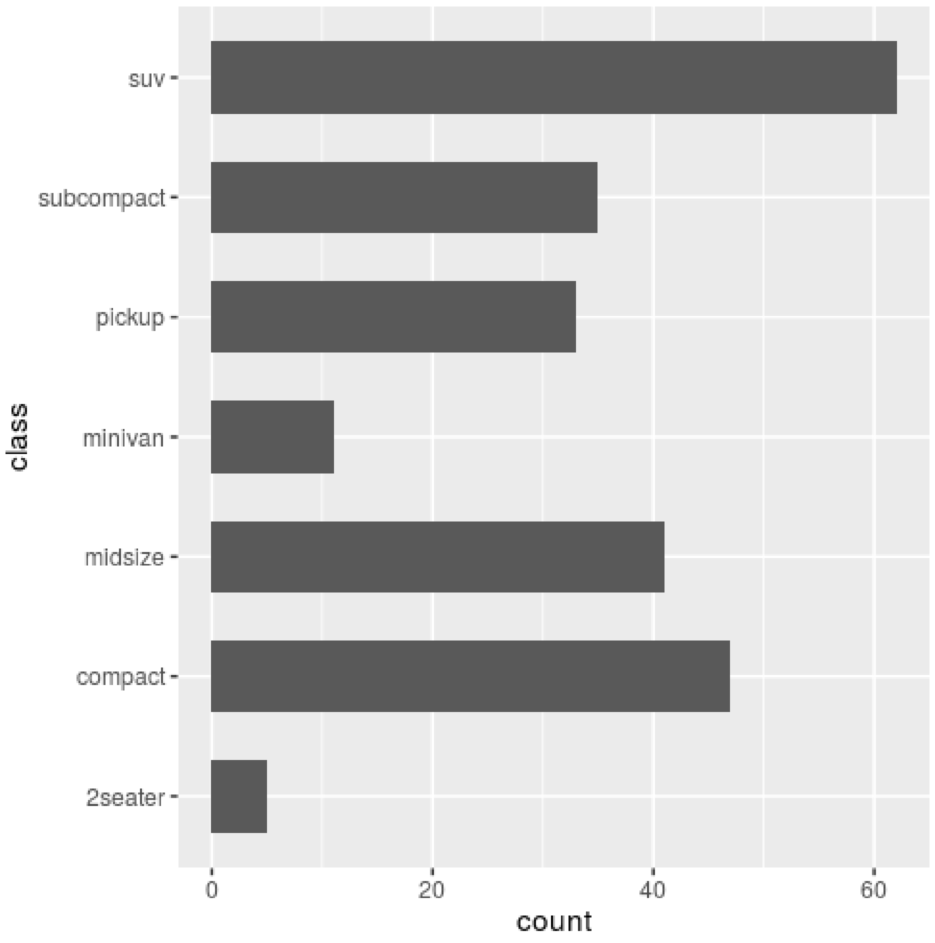

Figure 3.7:

3.6.2 Exercise 2

a. Represent graphically with a bar graph, how many cars there are for each class according to manifacturer.

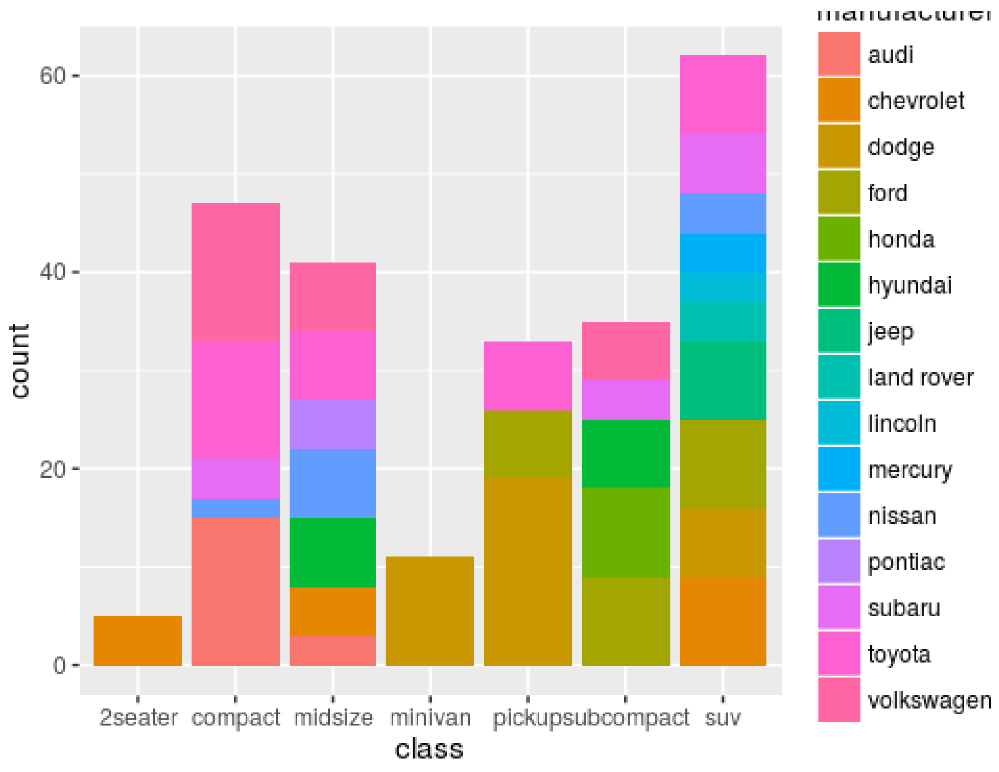

Figure 3.8:

b. Represent graphically with a bar graph, the distribution of manifacturerfor each class.

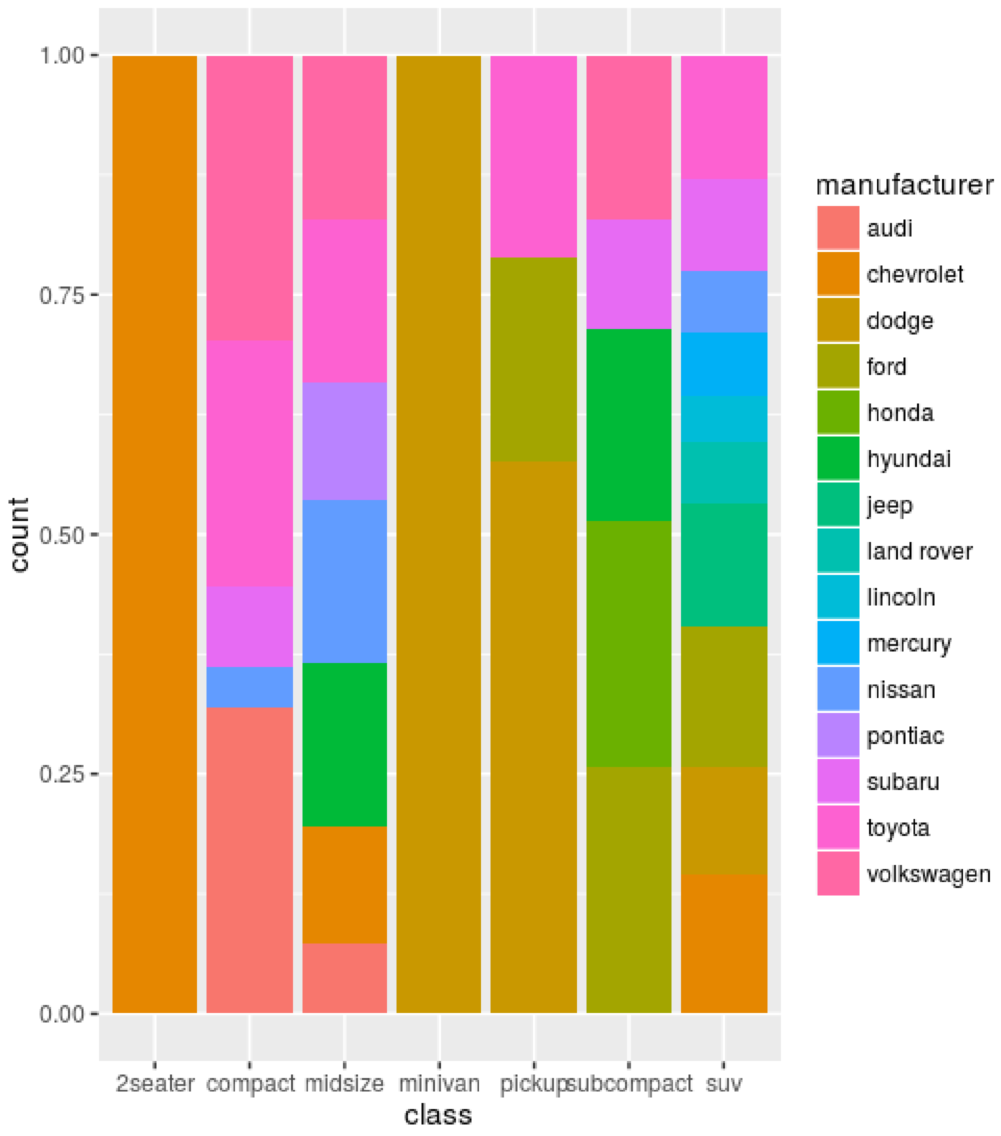

Figure 3.9:

Chapter 4

Writing R functions

4.1 Writing R functions

4.1.1 Exercise 1

Write a function, named `compute_summary`, which computes: sum, subtraction, multiplication and division of two numbers. The function arguments should be the two numbers, named as: `x` and `y`. The function should return all amounts computed.

4.1.2 Exercise 2

Write a function, named `compute_gain`, which computes the income by multiplying the amount produced for sale price and then computes the gain by subtracting the costs to income.
The function arguments should be: `amount`, `price`, and `costs`; `price` should have a default value equal to 5. The function should return the gain.