R for Beginners Course Exercises

October 21, 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 Objects*
- *Data Import and Export*
- *Data Manipulation with dplyr*
- *Data Visualization with ggplot2*

Chapter 2

Data Import

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.

2.1 Text Files

2.1.1 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`

2.1.2 Exercise 2

Import 7 rows of the text file named *"solar.txt"* skipping the first two rows. Save it in the 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.

2.1.3 Exercise 3

Considering the following data frame, named `df`:

```
df <- data.frame(col1=1:4, col2=4:1, col3=c("one", "two", "three", "four"),
                 stringsAsFactors = FALSE)
```

Save it in a .txt file named *"exercise-3.txt"* in *data* folder.

2.2 Excel Files

2.2.1 Exercise 1

a. Import .xlsx file *"flowers.xlsx"* using `XLConnect` function `loadWorkbook()` and save it in a R workbook object named `flowers`.
   Remember to load `XLConnect` package, supposing it is already installed.

```
require(XLConnect)
```

b. Read *iris* sheet with `readWorksheet()` function and save it in `flower_df` object. Then, visualize its first rows.

2.2.2 Exercise 2

a. Create a new file xlsx, named *"exercise-2.xlsx"*, and save it in the R worksheet object, named `ex_2`. Use: `loadWorkbook()` and `saveWorkbook()` functions of `XLConnect`.

b. Create a sheet, named `df`, in the R workbook object using `createSheet()` function. Remember to save the changes also in .xlsx file (use `saveWorkbook()` function).

c. Considering the following data frame, named `numbers_df`:

```
numbers_df <- data.frame(a= 1:4, b=c("one", "two", "three", "four"),
                         stringsAsFactors = FALSE)
numbers_df
```

```
##   a     b
## 1 1   one
## 2 2   two
## 3 3 three
## 4 4  four
```

Add it to `df` sheet of `ex_2` R workbook object, starting from row 3 and from column 2. Use the function `writeWorksheet()`. Remember to save the changes also in .xlsx file (use `saveWorkbook()` function).

2.3 Databases

2.3.1 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.

2.4 R Data Files

2.4.1 Exercise 1

Given the following data frame, named `df_rdata`:

```
df_rdata <- data.frame(a=1:20, b=20:1)
```

Save it in *.Rda* format in the file *"df_rdata.Rda"*, using `save()` function.

2.4.2 Exercise 2

Load *"drug.Rda"* file into the environment, using `load()` function.

Chapter 3

Data Object

3.1 Vectors

3.1.1 Exercise 1

a. Create a vector, named `vec1`, containing the following values:
   1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 25, 30, 35, 40, 45, 50, 60, 70, 80, 90

b. Select the 5-th element of `vec1`.

c. Select the first 10 elements of `vec1.`

d. Select all the elements of `vec1` apart from the 2nd and the 6th element.

3.1.2 Exercise 2

a. Generate a vector, named `vec2`, containing the numbers from 1 to 10 and of length 8, using the function `seq()`.

b. Select the values of `vec2` which are greater than 4.

c. Select the values of `vec2` which are equal or less than 2 or which are equal or greater than 6.

3.1.3 Exercise 3

a. Generate the following vector using the function `rep()`:
   `vec3 <- c("one", "two", "one", "two", "one", "two")`

b. Generate a new vector, named `vec5`, combining the previous vector, `vec3`, with the following one:

```
vec4 <- c("three", "four")
```

3.2 Matrices

3.2.1 Exercise 1

Generate a matrix, named `mat1`, with 5 rows and 3 columns, using `matrix()` function:

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
## [4,]   10   11   12
## [5,]   13   14   15
```

3.2.2 Exercise 2

Starting from the following vector:

```
mat2 <- 1:8
```

Generate a matrix with 2 rows and 4 columns using `dim()` function.

3.2.3 Exercise 3

a. Generate a matrix, named `mat3`, combining the following columns:

```
a <- 1:3
b <- 7:9
c <- 8:6
```

b. Add the following row to `mat3`:

```
d <- 4:6
```

3.2.4 Exercise 4

Considering the following matrix, named `mat4`:

```
mat4 <- matrix(1:24, nrow = 6, ncol = 4, byrow = TRUE)
mat4
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]    5    6    7    8
## [3,]    9   10   11   12
## [4,]   13   14   15   16
## [5,]   17   18   19   20
## [6,]   21   22   23   24
```

a. Select the third and the fifth row of `mat4`.
b. Select all columns of `mat4` apart from the first.
c. Select second and third rows and second and third columns of `mat4`.

3.3 Lists

3.3.1 Exercise 1

a. Generate a list, named `list1` that contains the following R elements:

```
vec <- 1:10
mat <- matrix(1:9, ncol = 3)
name <- "Oscar"
```

b. Add to `list1` the following element:

```
letters <- c("a", "b", "c", "d")
```

3.3.2 Exercise 2

Given the following list, named `list2`:

```
list2 <- list(vec = c(1,3,5,7,8), mat = matrix(1:12, ncol = 4),
              sub_list = list(names = c("Veronica", "Enrico", "Andrea", "Anna"),
                              numbers = 1:4))
list2
```

```
## $vec
## [1] 1 3 5 7 8
##
## $mat
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
##
## $sub_list
## $sub_list$names
## [1] "Veronica" "Enrico"   "Andrea"   "Anna"
##
## $sub_list$numbers
## [1] 1 2 3 4
```

a. Entract the first element of `list2`.

b. Extract the objects contained in the first element of `list2`.

c. Extract the element named `sub_list` of `list2`.

d. Extract the second rows of the matrix included in the second element of `list2`.

3.4 Factors

3.4.1 Exercise 1

Starting from the vector:

```
fac1 <- c("F", "F", "M", "M" , "F")
```

Generate the corresponding factor with two levels: “F” and “M”

3.4.2 Exercise 2

Starting from the vector:

```
fac2 <- c(1, 1, 1, 2, 2, 2)
```

a. Generate the corresponding factor considering that 1 = “Female”, 2 = “Male” e 3 = “Trans”.
b. Select the all elements of `fac2` apart from “Male”.

3.5 Data Frames

3.5.1 Exercise 1

a. Generate a data frame, named `df1`, corresponding to:

```
##    id     name class mean
## 1   1     Luca    5A  6.0
## 2   2   Chiara    5A  7.0
## 3   3     Lisa    5A  5.0
## 4   4   Matteo    5A  6.5
## 5   5    Alice    5A  7.5
## 6   6    Marco    5B  4.5
## 7   7 Veronica    5B  9.0
## 8   8   Nicola    5B  8.0
## 9   9    Elena    5B  8.5
## 10 10  Daniele    5B  7.0
```

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

b. Select the first 3 rows of `df1`.

c. Select the last 6 rows and the first 3 columns of `df1`.

d. Select the column `class` of `df1`.

e. Convert the column `class` of `df1` in a factor with levels: "5A" and "5B"

f. How many columns and rows `df1` has?

g. Generate another dataframe, named `df2` composed by the columns `name` and `mean` of `df1`, specifying the argument `stringsAsFactors = FALSE`.

h. Show the first rows and the structure of `df2`.

Chapter 4

Data Manipulation with `dplyr`

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

```
require(dplyr)
```

4.1 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")
```

4.1.1 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 ...
```

4.2 Select

4.2.1 Exercise 1

Extract the following information:

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

4.2.2 Exercise 2

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

4.2.3 Exercise 3

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

4.3 Filter

4.3.1 Exercise 1

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

4.3.2 Exercise 2

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

4.3.3 Exercise 3

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

4.4 Arrange

4.4.1 Exercise 1

Sort the flights in chronological order.

4.4.2 Exercise 2

Sort the flights by decreasing arrival delay.

4.4.3 Exercise 3

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

4.5 Mutate

4.5.1 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.

4.5.2 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)

4.6 Summarise

4.6.1 Exercise 1

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

4.7 Group_by

4.7.1 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.

4.7.2 Exercise 2

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

4.8 Chain multiple operations (%>%)

4.8.1 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.

4.8.2 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.

4.8.3 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 5

Data Visualization with ggplot2

Load ggplot2 package, supposing it is already installed.

```
require(ggplot2)
```

5.1 Data

5.1.1 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 ...
```

5.1.2 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

```r
dim(mpg)
```

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

```r
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
```

```r
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" ...
```

5.2 Scatterplot

Let us consider `iris` dataset.

5.2.1 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").

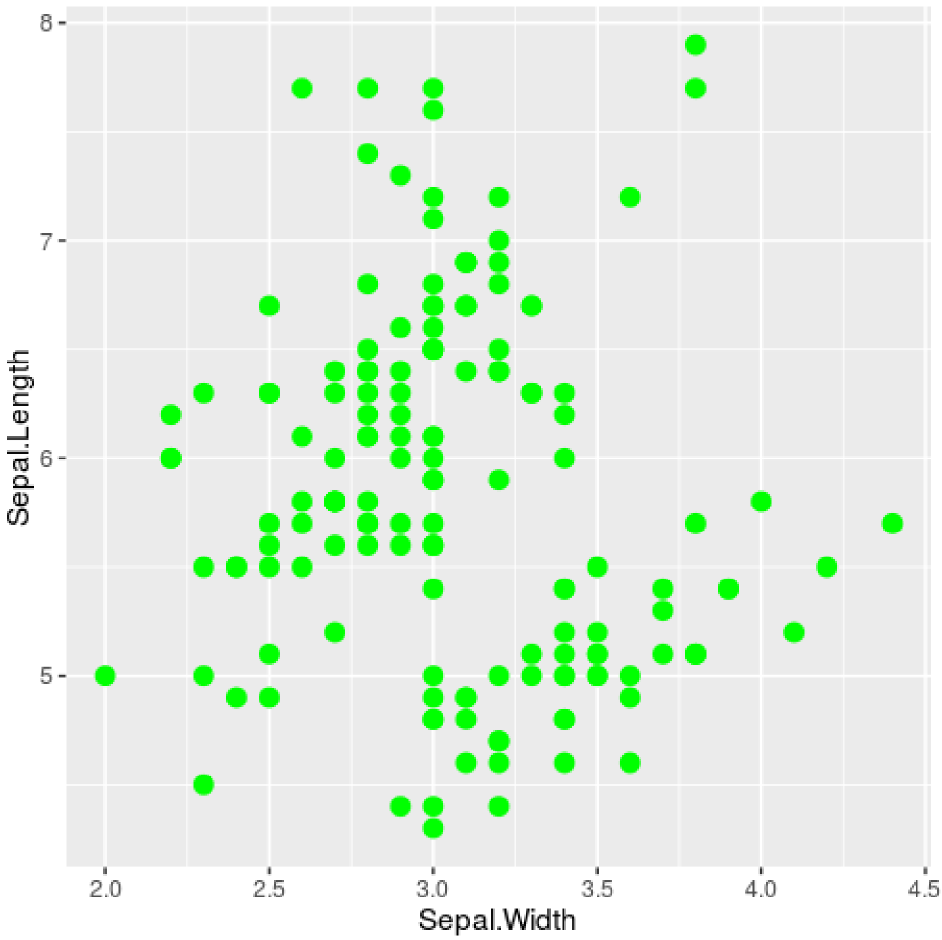

Figure 5.1:

5.2.2 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.

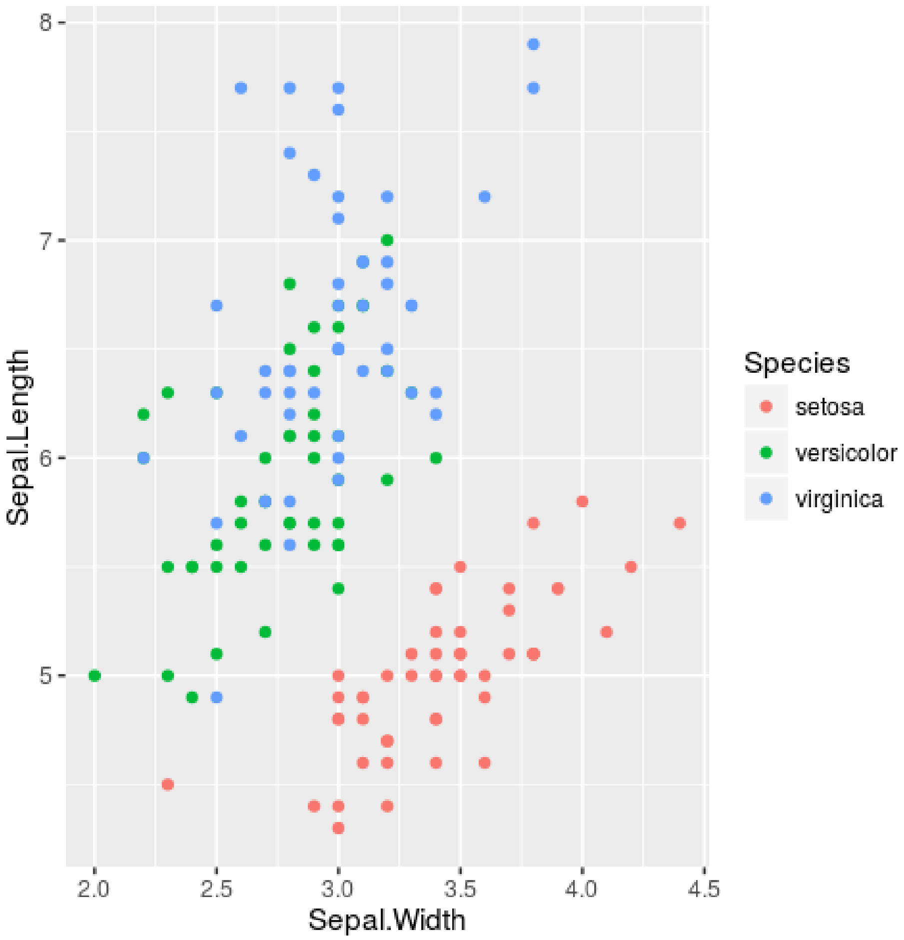

Figure 5.2:

5.3 Box Plot

Let us consider `iris` dataset.

5.3.1 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 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”

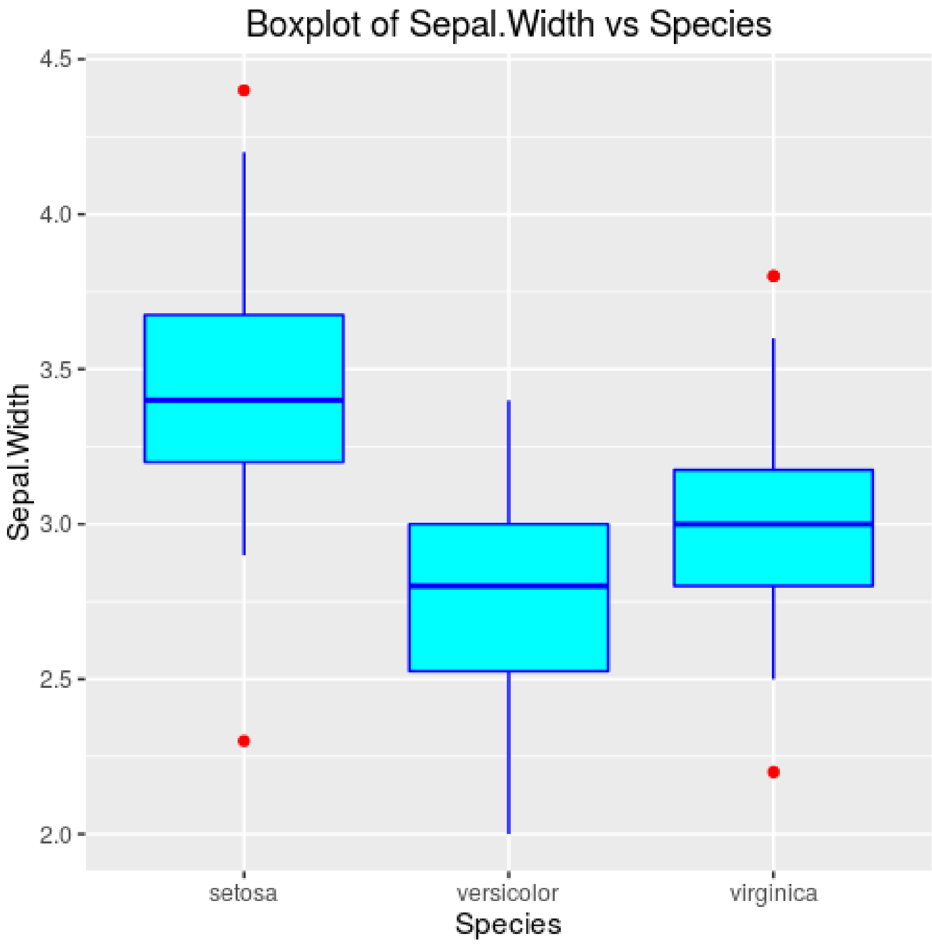

Figure 5.3:

5.4 Histogram

Let us consider `iris` dataset.

5.4.1 Exercise 1

a. Represent the distribution of `Sepal_Length` variable with an histogram.
b. Set bins fill colour as "hotpink" and bins line colour as "deeppink".
c. Set the number of bins as 15.

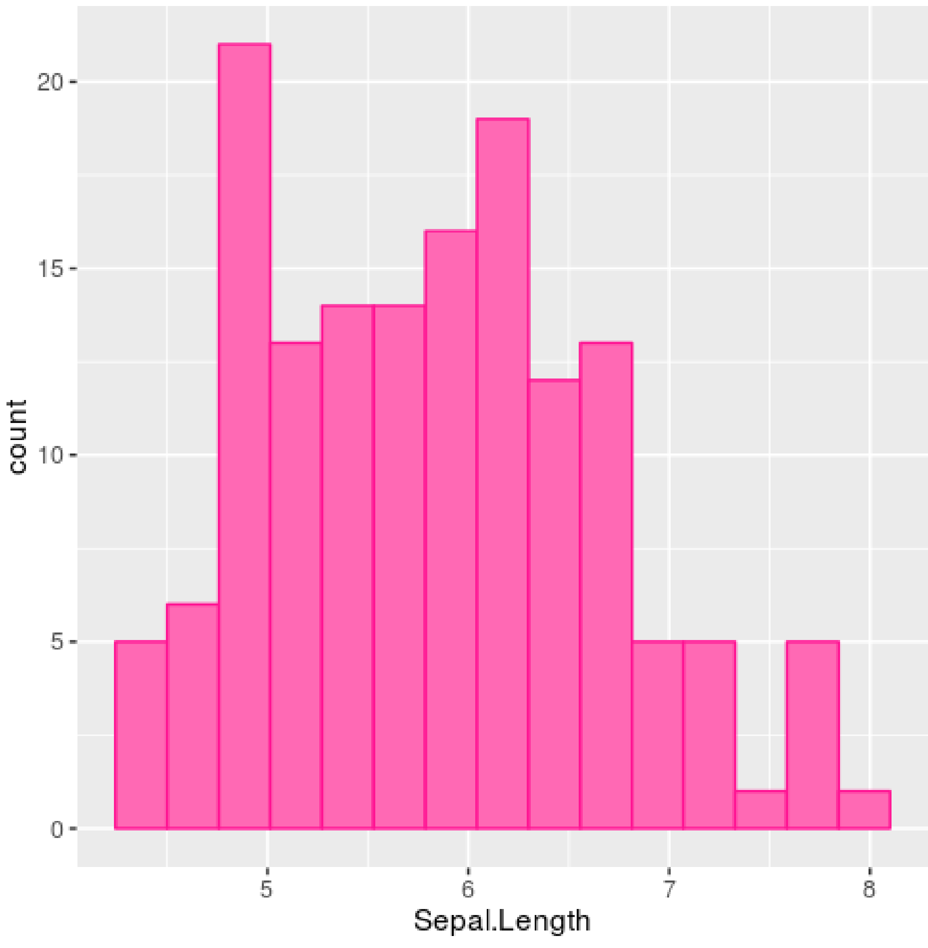

Figure 5.4:

5.5 Line graph

5.5.1 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 line graph to visualize the measures of `Sepal.Length` variable along time.

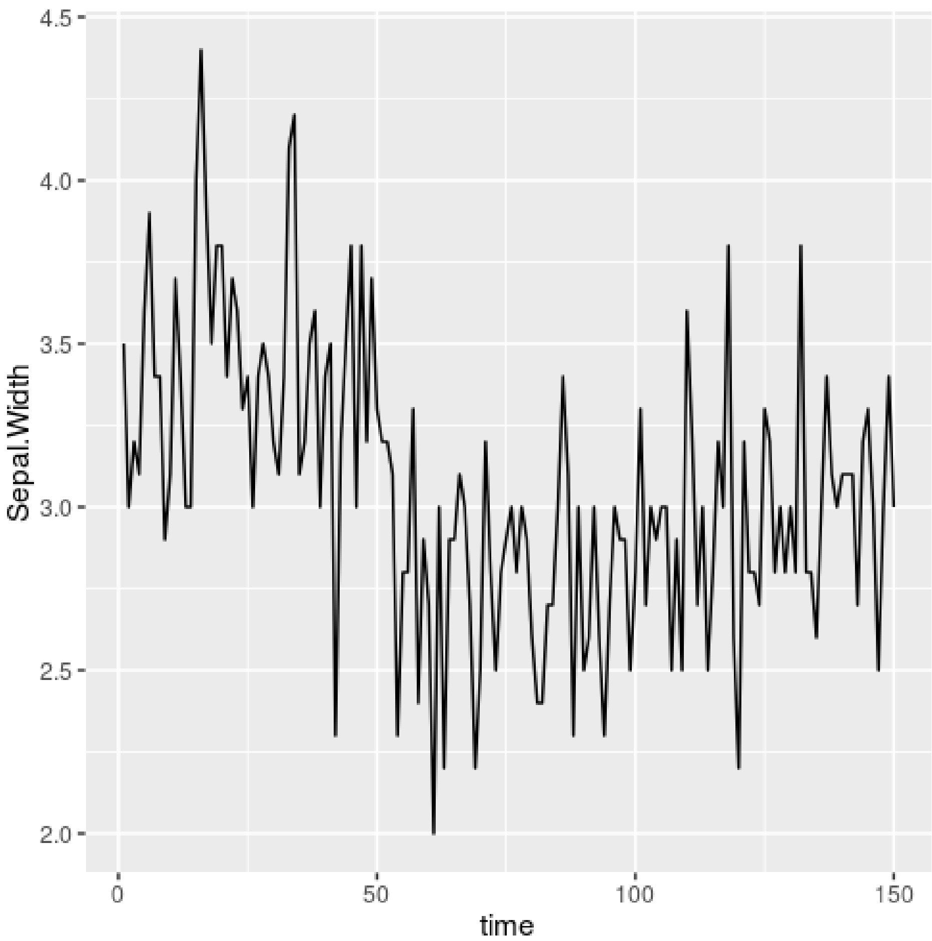

Figure 5.5:

5.5.2 Exercise 2

Let us suppose that the observations on `iris` are taken along time.
So let us consider the following dataset, named `iris3`, in which `time` variable is added:

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

a. Build a line graph to visualize the measures of `Sepal.Length` variable along time, according to the `Species` variable, mapped as `colour` aes.

b. Set linetype as "twodash".

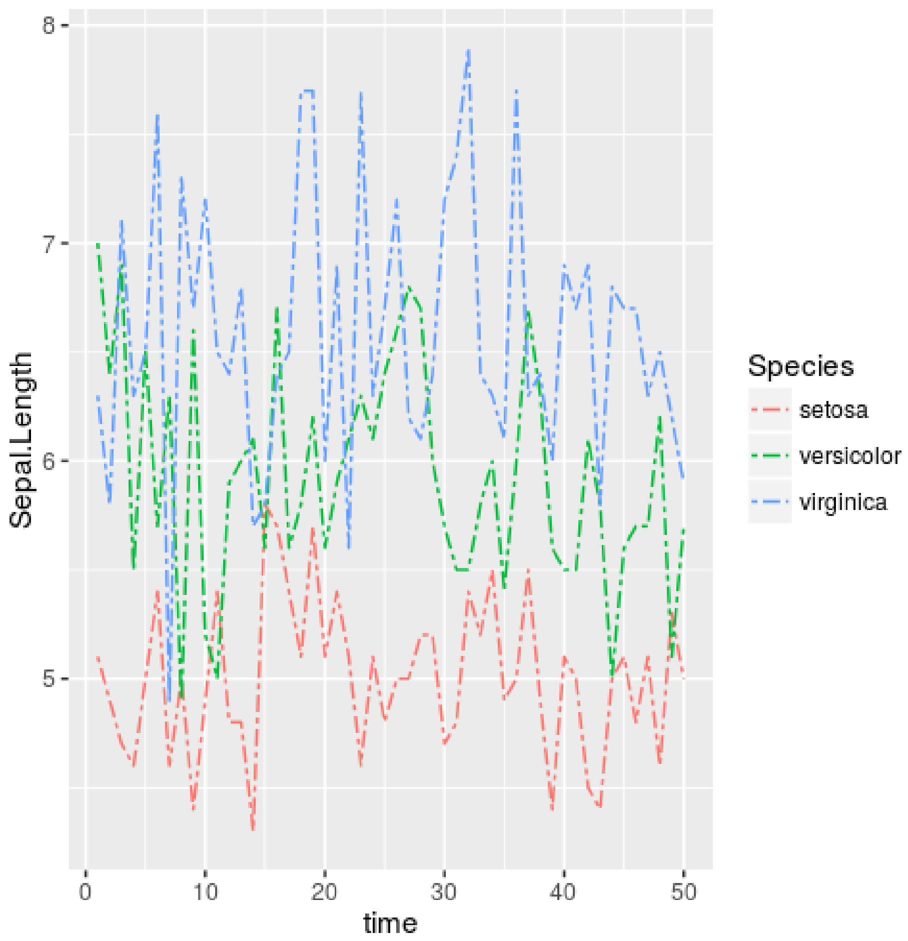

Figure 5.6:

5.6 Bar graph

Let us consider `mpg` dataset.

5.6.1 Exercise 1

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

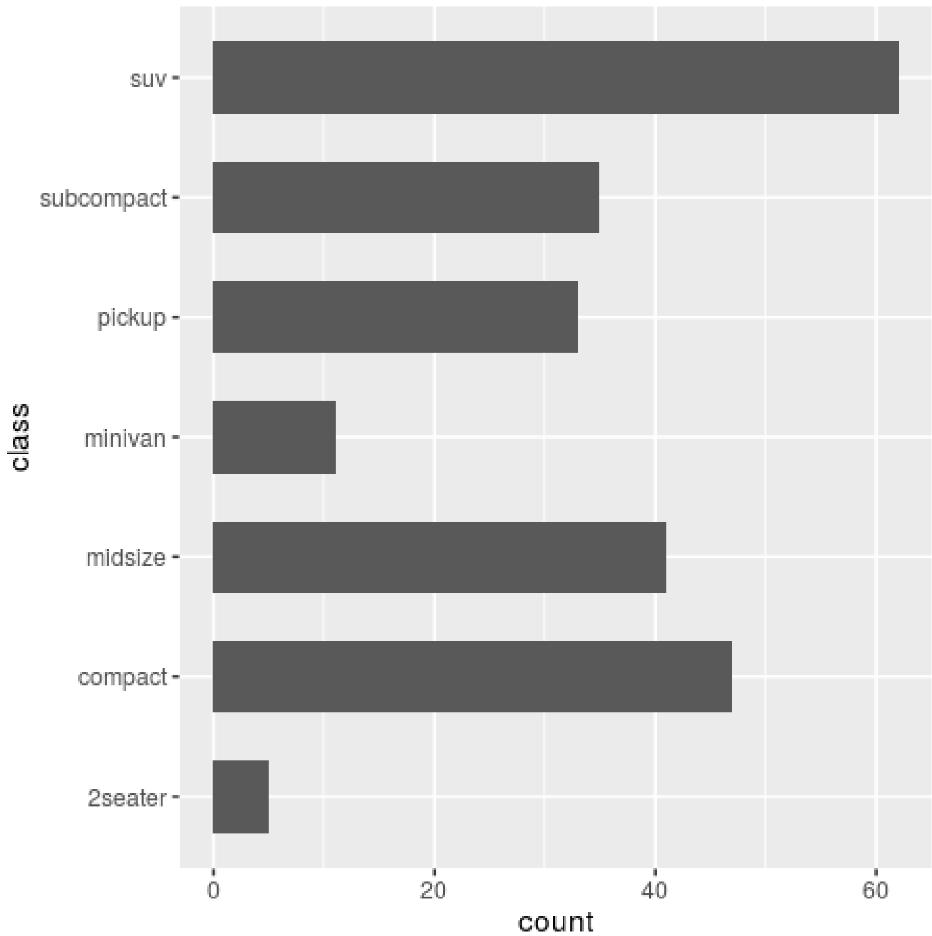

Figure 5.7:

5.6.2 Exercise 2

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

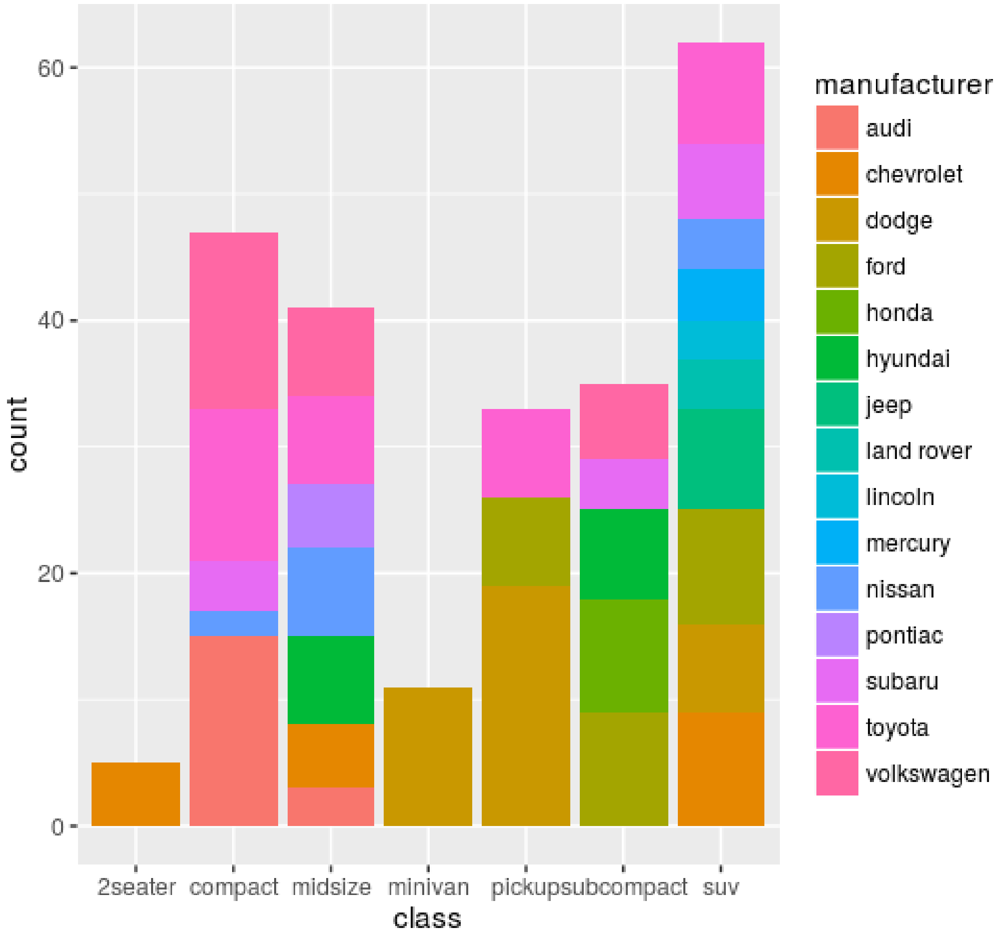

Figure 5.8:

b. Represent graphically with a bar graph, the distribution of manifacturerfor each class (set `position` argument of `geom_bar`).

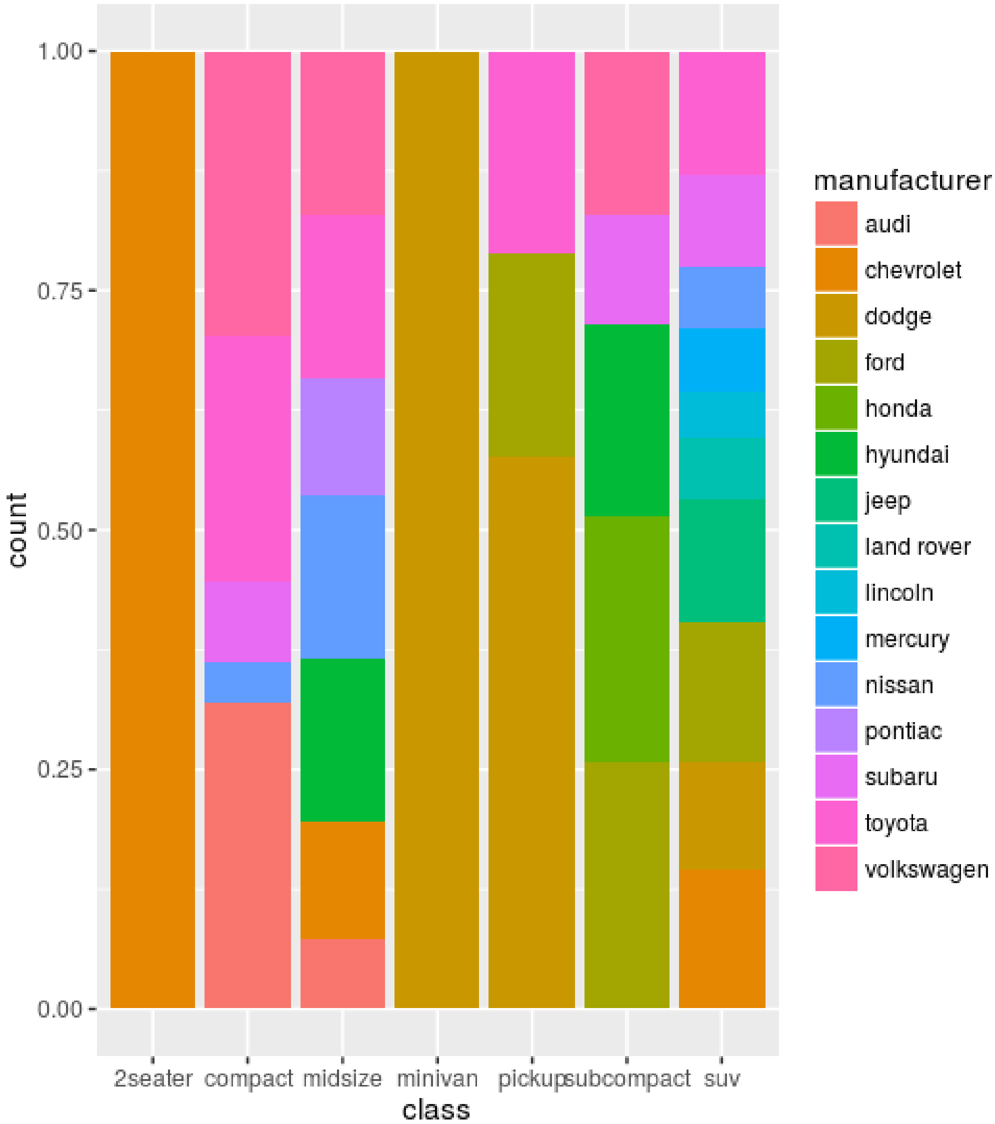

Figure 5.9: