Introduction to R

author: Marcelo Gehara

This tutorial is an intrduction to R by a self-thaught R user, so it is based on my perspective of what is the basics in R coding. R is a very useful and friendly language, with a huge community and a lot of materials online. You can learn a lot just by trying things yourself and googling the errors you get. Or just by googling the type of thing you want to do, like: "How to run anova in R". For the vast majority of cases, you will find codes online with explanations of how to do it. In fact that was how I learned. There are several other R tutorials online, youtube videos etc. I recommend you check them out later. You can find an official R tutorial here[https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf].

How to read this tutorial

All the R code in this tutorial is highlighted in gray. The resuting R output is right below, after the ## mark. You should be able to copy and paste the code in your R console and get the same output. However, I advise you to type the code yourself. A lot of errors come from typos, especially in the beginning. It is important to develop awareness of what is written and what should be written. After some trainning you should be able to spot a misplaced comma or the absence of a required comma. It is a matter of trainning your eyes but it is also a mindset. *If something doesn't work the first thing you should do is to look for typos!*

General R help

```
help.start() # in this specific case the output should be in your viewer. It is the R help.
```

```
## starting httpd help server ... done
```

```
## If the browser launched by 'xdg-open' is already running, it is
##     *not* restarted, and you must switch to its window.
## Otherwise, be patient ...
```

Assigning numbers to the objects "x" and "y". The "<-" symbol is used to assign data to object. You may also point it to the right like "->". In this case your object will be on the right of the code line.

```
x <- 5
y <- 2
x
```

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

```
y
```

```
## [1] 2
```

See objects in environment

```
objects()
```

```
## [1] "x" "y"
```

Arithmetic

```
x + 1 # sum
```

```
## [1] 6
```

```
x + y
```

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

```
x * y # multiplication
```

```
## [1] 10
```

```
x ^ y # power
```

```
## [1] 25
```

```
sqrt(x) # square root
```

```
## [1] 2.236068
```

x is a vector with a single element. Using the "[]" we can see the elements of "x"

```
x[1]
```

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

Second element is missing. "NA" is the code for missing value.

```
x[2]
```

```
## [1] NA
```

Assign a value to the second element of "x"

```
x[2] <- 10
```

Arithmetic

```
x[1] * x[2]
```

```
## [1] 50
```

check how long the vector is

```
length(x)
```

```
## [1] 2
```

Getting help. How does the function "length" work?

```
?length
help(length)
```

See the class of x. Objects can have different classes depending on the type of data they store.

```
class(x)
```

```
## [1] "numeric"
```

Remove objects from the environment

```
rm(x)
rm(y)
```

Assign a vector of numbers to an object. Use the "c" function to concatenate or combine things.

```
x <- c(1,2,3,4,5)
class(x)
```

```
## [1] "numeric"
```

```
x <- 1:5
class(x)
```

```
## [1] "integer"
```

> CHALLENGE 1

Assign a character to a vector. Characters are expressed between quotes. In the case below, if you add a character to your vector, R automatically "reads" all the numbers as characters. You can only have a single class in a vector.

```
x <- c(1,2,3,4,5,"DNA")
class(x)
```

```
## [1] "character"
```

```
x <- c("A","B","C")
class(x)
```

```
## [1] "character"
```

```
str(x)
```

```
##  chr [1:3] "A" "B" "C"
```

```
x[2]
```

```
## [1] "B"
```

> CHALLENGE 2

Let's load an example data (*iris*) and start exploring R!

The *iris* data was presented by Ronald Fisher in 1936 in the manuscript *The use of multiple measurements in taxonomic problems*. These quantify the morphologic variation of the flower in three species of iris.

The data is composed of measurements in centimeters of the variables sepal length and width and petal length and width, for 50 flowers from each of 3 species of iris. The species are *Iris setosa*, *I. versicolor*, and *I. virginica*.

load the iris data

```
data(iris)
```

Iris is a data.frame. Data frames are like excell spread sheets. They are composed of columns and rows and they can store different data types, such as character and numeric, in the same object. *dfs* are different from matrices, which only allow one type of data. You can name the columns and the rows just like in a spread sheet.

```
class(iris)
```

```
## [1] "data.frame"
```

What are the dimensions of this *df*? 150 rows and 5 columns

```
dim(iris)
```

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

```
nrow(iris)
```

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

```
ncol(iris)
```

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

structure of the *df*

```
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 ...
```

Use "head" to see the first lines of the data.frame

```
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
```

See the column names

```
colnames(iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
## [5] "Species"
```

See row names. There are no row names so the names are referred to as numbers

```
rownames(iris)
```

```
##   [1] "1"   "2"   "3"   "4"   "5"   "6"   "7"   "8"   "9"   "10"  "11"
##  [12] "12"  "13"  "14"  "15"  "16"  "17"  "18"  "19"  "20"  "21"  "22"
##  [23] "23"  "24"  "25"  "26"  "27"  "28"  "29"  "30"  "31"  "32"  "33"
##  [34] "34"  "35"  "36"  "37"  "38"  "39"  "40"  "41"  "42"  "43"  "44"
##  [45] "45"  "46"  "47"  "48"  "49"  "50"  "51"  "52"  "53"  "54"  "55"
##  [56] "56"  "57"  "58"  "59"  "60"  "61"  "62"  "63"  "64"  "65"  "66"
##  [67] "67"  "68"  "69"  "70"  "71"  "72"  "73"  "74"  "75"  "76"  "77"
##  [78] "78"  "79"  "80"  "81"  "82"  "83"  "84"  "85"  "86"  "87"  "88"
##  [89] "89"  "90"  "91"  "92"  "93"  "94"  "95"  "96"  "97"  "98"  "99"
## [100] "100" "101" "102" "103" "104" "105" "106" "107" "108" "109" "110"
## [111] "111" "112" "113" "114" "115" "116" "117" "118" "119" "120" "121"
## [122] "122" "123" "124" "125" "126" "127" "128" "129" "130" "131" "132"
## [133] "133" "134" "135" "136" "137" "138" "139" "140" "141" "142" "143"
## [144] "144" "145" "146" "147" "148" "149" "150"
```

data.frame indexing.

We can use the *[ ]* to point to and see specific elements of the *df*. Rows and columns are separated by a comma *[row number here , col number here]*

```
iris[1,] # first row
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
```

```
iris[,1] # first column
```

```
##   [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9 5.4 4.8 4.8 4.3 5.8 5.7 5.4
##  [18] 5.1 5.7 5.1 5.4 5.1 4.6 5.1 4.8 5.0 5.0 5.2 5.2 4.7 4.8 5.4 5.2 5.5
##  [35] 4.9 5.0 5.5 4.9 4.4 5.1 5.0 4.5 4.4 5.0 5.1 4.8 5.1 4.6 5.3 5.0 7.0
```

```
##  [52] 6.4 6.9 5.5 6.5 5.7 6.3 4.9 6.6 5.2 5.0 5.9 6.0 6.1 5.6 6.7 5.6 5.8
##  [69] 6.2 5.6 5.9 6.1 6.3 6.1 6.4 6.6 6.8 6.7 6.0 5.7 5.5 5.5 5.8 6.0 5.4
##  [86] 6.0 6.7 6.3 5.6 5.5 5.5 6.1 5.8 5.0 5.6 5.7 5.7 6.2 5.1 5.7 6.3 5.8
## [103] 7.1 6.3 6.5 7.6 4.9 7.3 6.7 7.2 6.5 6.4 6.8 5.7 5.8 6.4 6.5 7.7 7.7
## [120] 6.0 6.9 5.6 7.7 6.3 6.7 7.2 6.2 6.1 6.4 7.2 7.4 7.9 6.4 6.3 6.1 7.7
## [137] 6.3 6.4 6.0 6.9 6.7 6.9 5.8 6.8 6.7 6.7 6.3 6.5 6.2 5.9
```

```
iris[1, 1] # first column , first row
```

```
## [1] 5.1
```

```
iris[c(1,5), ] # first and fifth rows
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
```

```
iris[1:3, ] # first to third rows
```

```
##   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
```

```
iris[1:3, c(1,3)] # first to third rows, first and third column
```

```
##   Sepal.Length Petal.Length
## 1          5.1          1.4
## 2          4.9          1.4
## 3          4.7          1.3
```

In *dfs* you can also use column name indexing. The *$* indicates the name of the element of the *iris* object. In the case of *df*, it is a column.

```
iris$Sepal.Length
```

```
##   [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9 5.4 4.8 4.8 4.3 5.8 5.7 5.4
##  [18] 5.1 5.7 5.1 5.4 5.1 4.6 5.1 4.8 5.0 5.0 5.2 5.2 4.7 4.8 5.4 5.2 5.5
##  [35] 4.9 5.0 5.5 4.9 4.4 5.1 5.0 4.5 4.4 5.0 5.1 4.8 5.1 4.6 5.3 5.0 7.0
##  [52] 6.4 6.9 5.5 6.5 5.7 6.3 4.9 6.6 5.2 5.0 5.9 6.0 6.1 5.6 6.7 5.6 5.8
##  [69] 6.2 5.6 5.9 6.1 6.3 6.1 6.4 6.6 6.8 6.7 6.0 5.7 5.5 5.5 5.8 6.0 5.4
##  [86] 6.0 6.7 6.3 5.6 5.5 5.5 6.1 5.8 5.0 5.6 5.7 5.7 6.2 5.1 5.7 6.3 5.8
## [103] 7.1 6.3 6.5 7.6 4.9 7.3 6.7 7.2 6.5 6.4 6.8 5.7 5.8 6.4 6.5 7.7 7.7
## [120] 6.0 6.9 5.6 7.7 6.3 6.7 7.2 6.2 6.1 6.4 7.2 7.4 7.9 6.4 6.3 6.1 7.7
## [137] 6.3 6.4 6.0 6.9 6.7 6.9 5.8 6.8 6.7 6.7 6.3 6.5 6.2 5.9
```

Now, more complex pointing.

First, let's see how many species there are in the data set and then use a logical expression to visualize data just for one species.

```
unique(iris$Species) # see uniques
```

```
## [1] setosa     versicolor virginica
## Levels: setosa versicolor virginica
```

```
levels(iris$Species) # or see levels of factor
```

```
## [1] "setosa"     "versicolor" "virginica"
```

```
iris$Species=="virginica" # this shows the rows of Species column that match virginica
```

```
##   [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [12] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [23] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [34] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [45] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [56] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [67] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [78] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [89] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [100] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [111]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [122]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [133]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [144]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

Now we can put this inside the row indexing area of iris to subset the table and assign it to a new object.

```
iris[iris$Species == "virginica",]
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 101          6.3         3.3          6.0         2.5 virginica
## 102          5.8         2.7          5.1         1.9 virginica
## 103          7.1         3.0          5.9         2.1 virginica
## 104          6.3         2.9          5.6         1.8 virginica
## 105          6.5         3.0          5.8         2.2 virginica
## 106          7.6         3.0          6.6         2.1 virginica
## 107          4.9         2.5          4.5         1.7 virginica
## 108          7.3         2.9          6.3         1.8 virginica
## 109          6.7         2.5          5.8         1.8 virginica
## 110          7.2         3.6          6.1         2.5 virginica
## 111          6.5         3.2          5.1         2.0 virginica
## 112          6.4         2.7          5.3         1.9 virginica
## 113          6.8         3.0          5.5         2.1 virginica
## 114          5.7         2.5          5.0         2.0 virginica
## 115          5.8         2.8          5.1         2.4 virginica
## 116          6.4         3.2          5.3         2.3 virginica
## 117          6.5         3.0          5.5         1.8 virginica
## 118          7.7         3.8          6.7         2.2 virginica
## 119          7.7         2.6          6.9         2.3 virginica
## 120          6.0         2.2          5.0         1.5 virginica
## 121          6.9         3.2          5.7         2.3 virginica
## 122          5.6         2.8          4.9         2.0 virginica
## 123          7.7         2.8          6.7         2.0 virginica
## 124          6.3         2.7          4.9         1.8 virginica
## 125          6.7         3.3          5.7         2.1 virginica
## 126          7.2         3.2          6.0         1.8 virginica
## 127          6.2         2.8          4.8         1.8 virginica
## 128          6.1         3.0          4.9         1.8 virginica
## 129          6.4         2.8          5.6         2.1 virginica
## 130          7.2         3.0          5.8         1.6 virginica
```

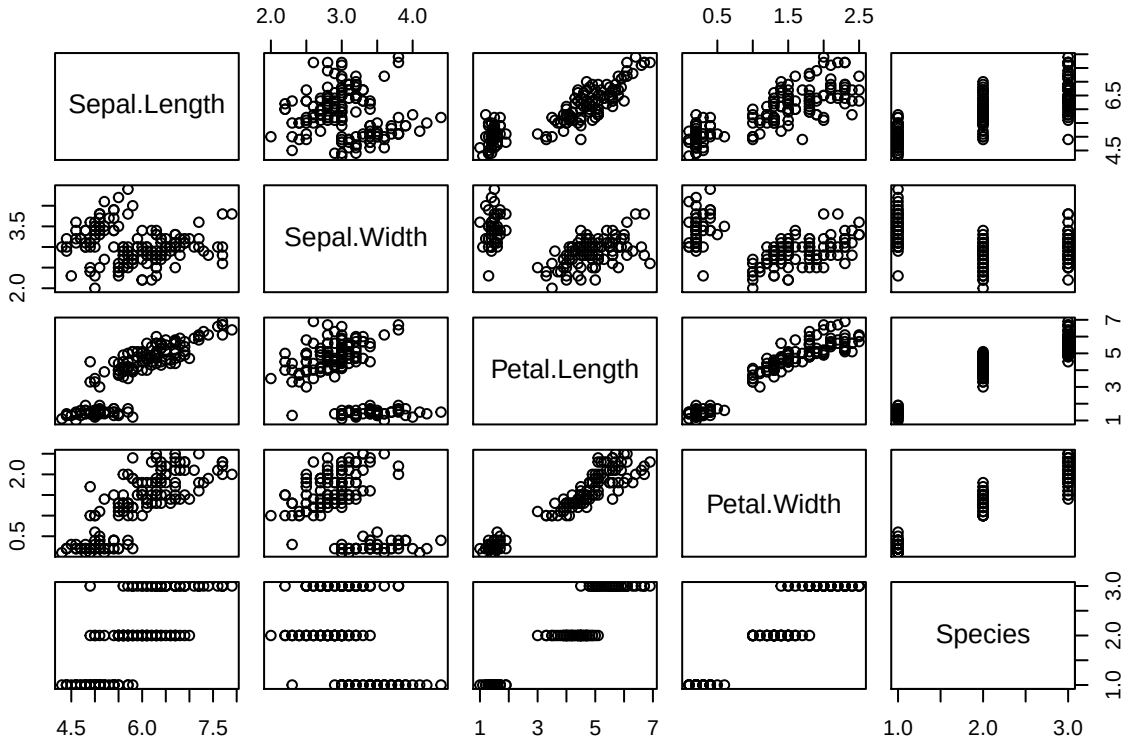
```
## 131          7.4         2.8          6.1         1.9 virginica
## 132          7.9         3.8          6.4         2.0 virginica
## 133          6.4         2.8          5.6         2.2 virginica
## 134          6.3         2.8          5.1         1.5 virginica
## 135          6.1         2.6          5.6         1.4 virginica
## 136          7.7         3.0          6.1         2.3 virginica
## 137          6.3         3.4          5.6         2.4 virginica
## 138          6.4         3.1          5.5         1.8 virginica
## 139          6.0         3.0          4.8         1.8 virginica
## 140          6.9         3.1          5.4         2.1 virginica
## 141          6.7         3.1          5.6         2.4 virginica
## 142          6.9         3.1          5.1         2.3 virginica
## 143          5.8         2.7          5.1         1.9 virginica
## 144          6.8         3.2          5.9         2.3 virginica
## 145          6.7         3.3          5.7         2.5 virginica
## 146          6.7         3.0          5.2         2.3 virginica
## 147          6.3         2.5          5.0         1.9 virginica
## 148          6.5         3.0          5.2         2.0 virginica
## 149          6.2         3.4          5.4         2.3 virginica
## 150          5.9         3.0          5.1         1.8 virginica
```

```r
virginica <- iris[iris$Species == "virginica",]
setosa <- iris[iris$Species == unique(iris$Species)[1], ]
versicolor <- iris[iris$Species == unique(iris$Species)[2], ]
virginica <- iris[iris$Species == unique(iris$Species)[3], ]
```

> CHALLENGE 3

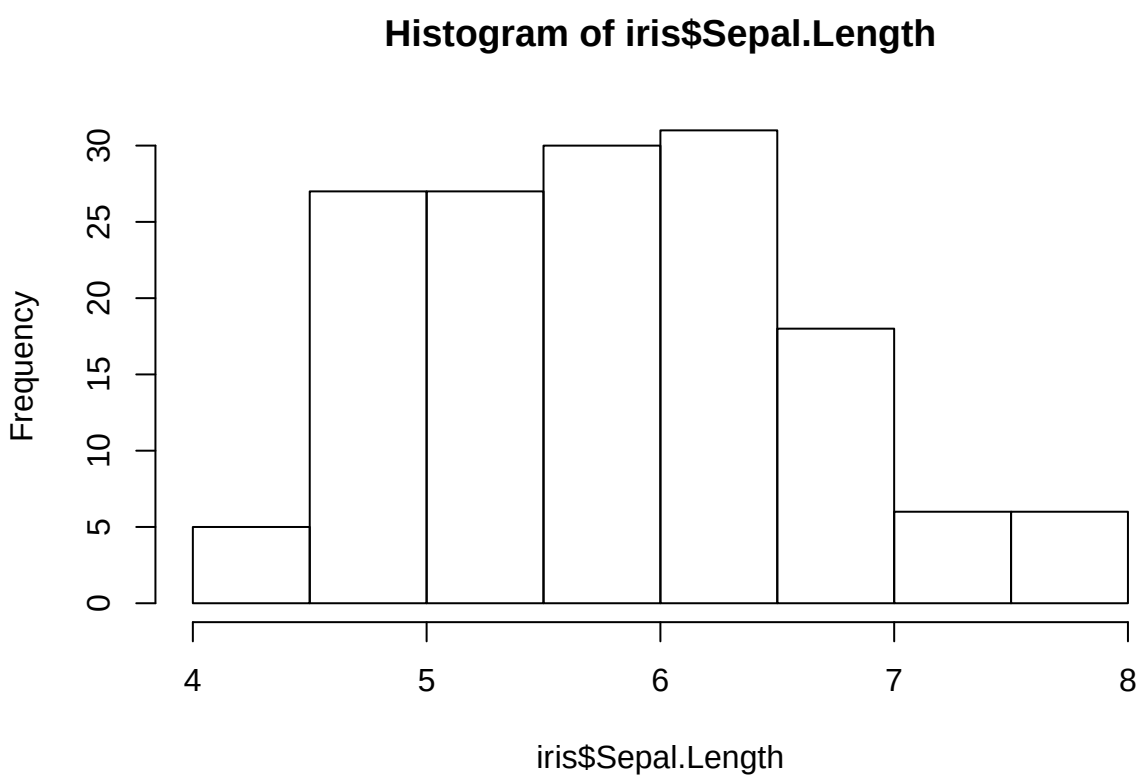
Let's plot the data

```r
plot(iris)
```



OK this is hard to digest, let's explore the data better.

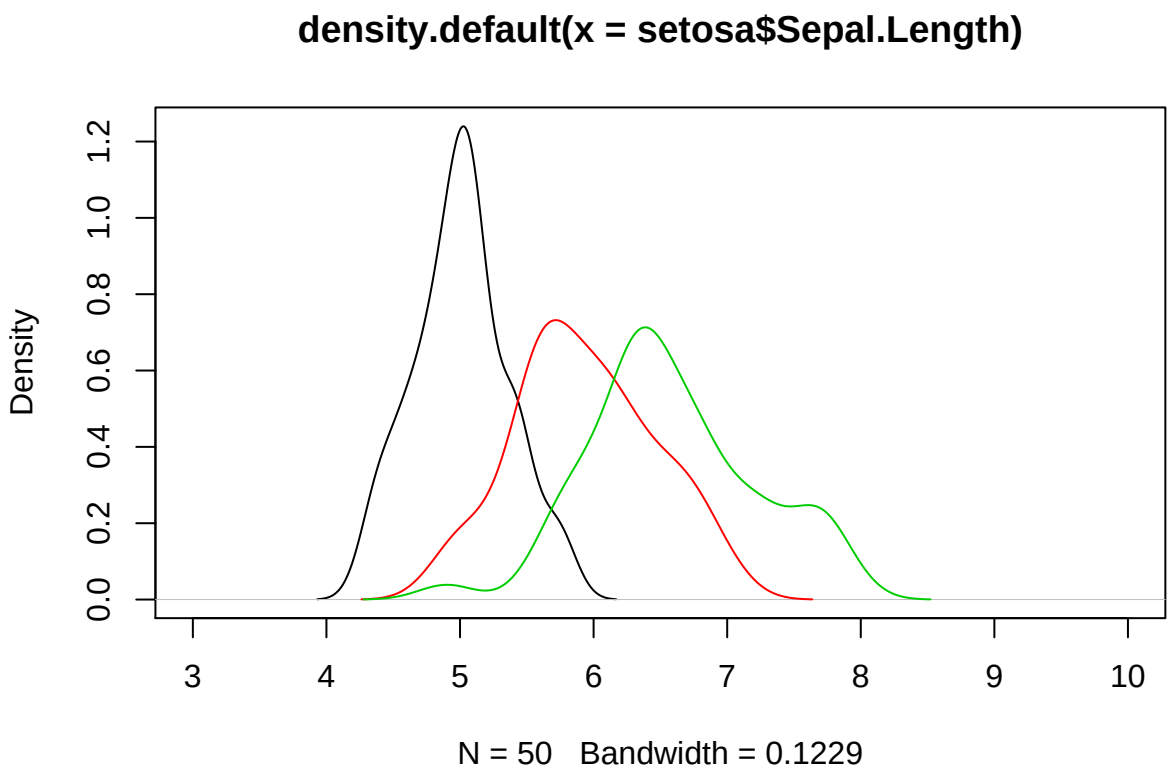
```
hist(iris$Sepal.Length)
```



Let's plot the density of this measurement for each species.

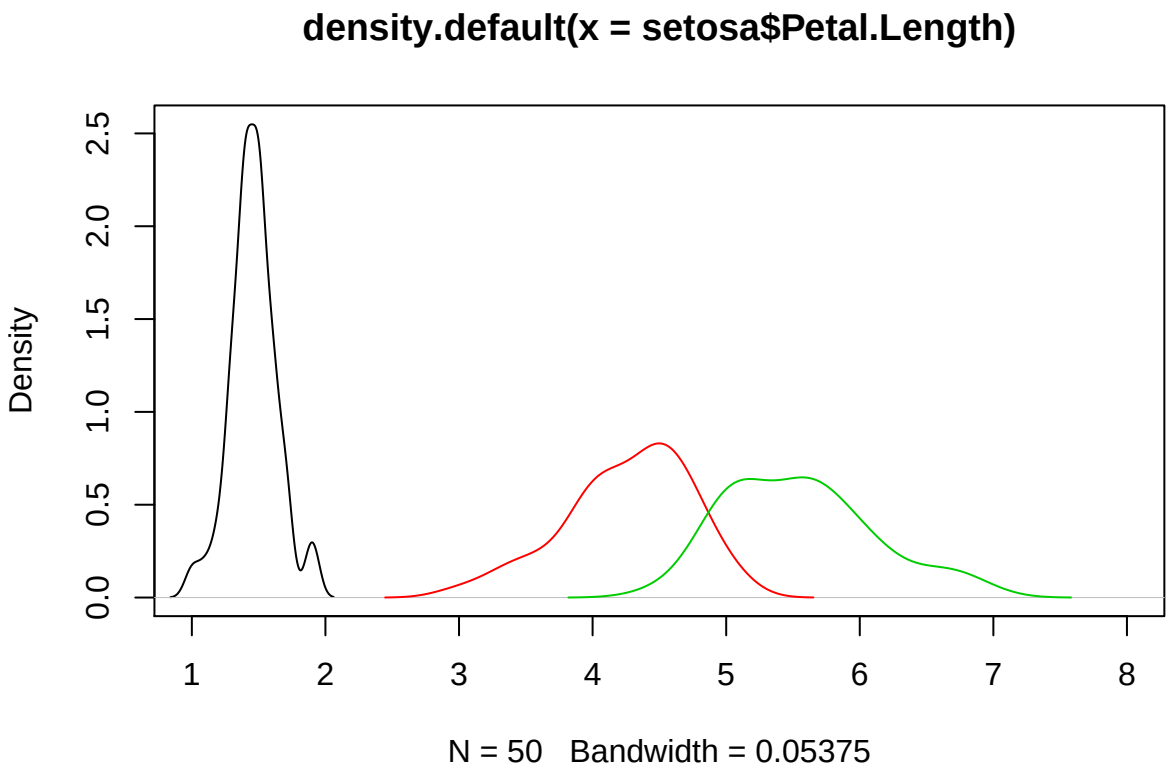
Sepal

```
plot(density(setosa$Sepal.Length), xlim=c(3, 10))
lines(density(versicolor$Sepal.Length), col = 2)
lines(density(virginica$Sepal.Length), col = 3)
```



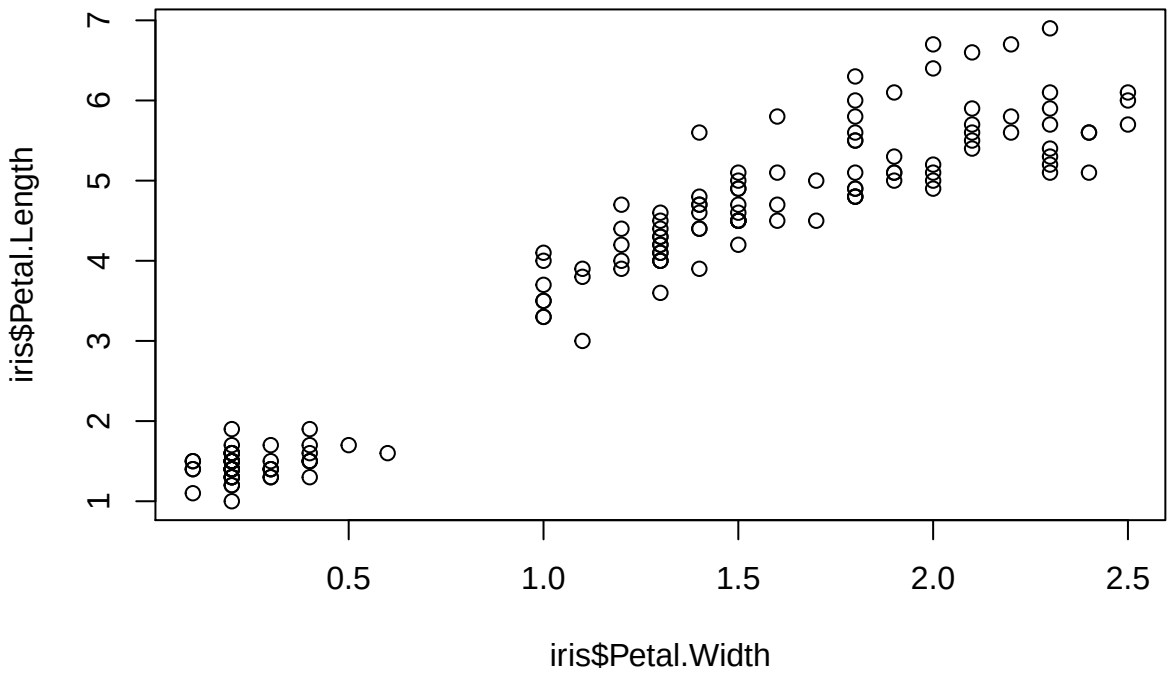
Petal

```
plot(density(setosa$Petal.Length), xlim=c(1, 8))
lines(density(versicolor$Petal.Length), col = 2)
lines(density(virginica$Petal.Length), col = 3)
```



Petal against Sepal

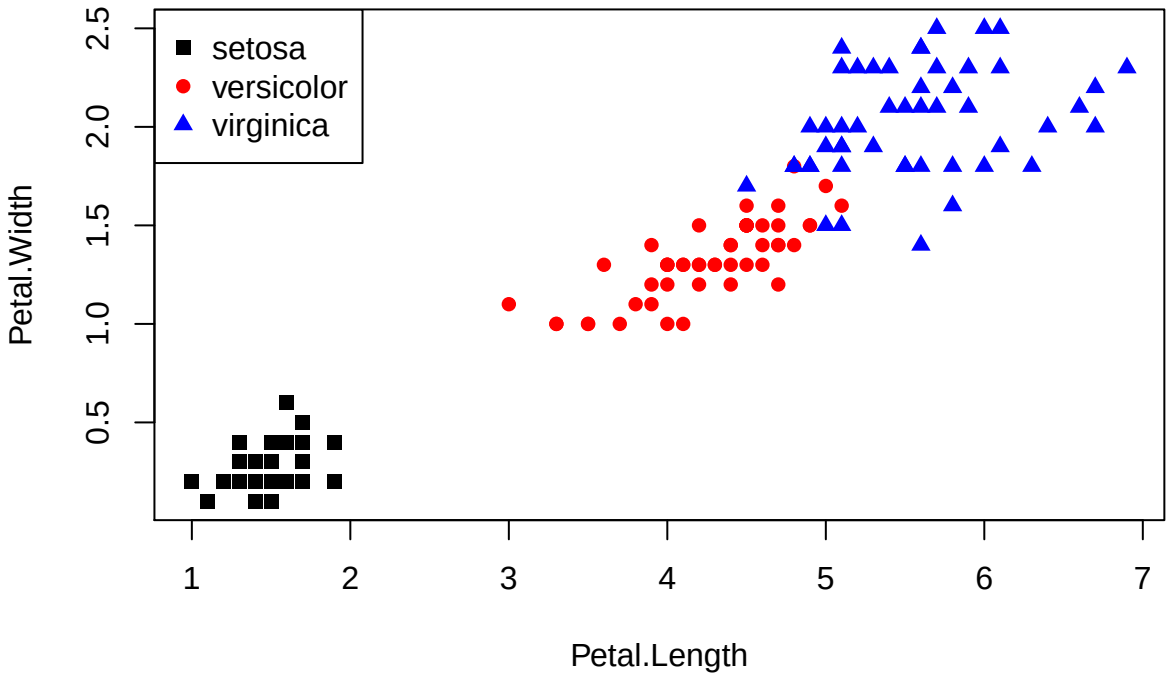
```
plot(iris$Petal.Length ~ iris$Petal.Width)
```



Now with legend and different colors and shapes for each species

```
plot(Petal.Width ~ Petal.Length, col=c("black", "red", "blue")[Species],
     pch=(15:17)[Species], data=iris)

legend("topleft", legend=levels(iris$Species), col=c("black", "red", "blue"), pch=15:17)
```



> CHALLENGE 4

How to read from a table and write to a file.

First let's write the iris data to a file. To do this, it is better to make sure we know where this file will be stored. The place that R writes and also looks for files by default is the working directory.

```
getwd() # to see the path to working directory
```

```
## [1] "/home/marcelo/Dropbox/Desktop"
```

If you want to set up a new working directory use setwd(*put the path to the folder here*). This can be any folder in your system.

Write the iris data to file

Let's first run *?write.table* to see how this function works. The first argument of a function is usually the object you want to process, further arguments should be separated by a comma and can be additional objects to process or options to process the object in different ways.

```
write.table(iris, file = "iris.txt", quote = F, col.names = T, row.names = F)
```

See if the file was saved. List all the files in my working directory.

```
list.files()
```

```
##  [1] "DSC02779.JPG"        "Files"               "iris.txt"
##  [4] "R_workshop_files"    "R_workshop.md"       "R_workshop.pdf"
##  [7] "R_workshop.R"        "R_workshop.Rmd"      "test.pdf"
## [10] "test.Rmd"            "The_coalescent.html" "The_coalescent.pdf"
## [13] "The_coalescent.R"    "The_coalescent.Rmd"  "Videos_2Coelho"
```

Let's remove the iris object just to see the read.table working.

```
rm(iris) # remove iris from the environment.
objects() # see all objects, iris is not there anymore.
```

```
## [1] "setosa"     "versicolor" "virginica"  "x"
```

```
iris <- read.table(file = "iris.txt", header = T) # read iris back in.
objects() # iris is there!
```

```
## [1] "iris"       "setosa"     "versicolor" "virginica"  "x"
```

```
head(iris) # see first lines of iris jus  t to make sure it worked.
```

```
##   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
```

Creating and adding a new column to the df. We will use four functions to do this: *rep*, *length*, *grep* and *cbind*. Let's first work with each one individually.

```
x <- rep(1, times = 5) # rep will repeat 1 5x
x
```

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

```
y <- length(x)
y
```

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

```
grep("a", c("a", "b", "c", "d", "a"))
```

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

We can nest all of these functions to generate a numeric index of the species names. Coding is a very creative activity. There are boundaries but also a great deal of freedom, since there are different ways to accomplish the same task. When you are coding you have to exercise your imagination.

```
index <- c(rep(1, length(grep("setosa", iris$Species))),
           rep(2, length(grep("versicolor", iris$Species))),
           rep(3, length(grep("virginica", iris$Species))))
index
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [71] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3
## [106] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [141] 3 3 3 3 3 3 3 3 3
```

Now we can use *cbind* to bind the index to the iris df

```
iris <- cbind(iris, index)
head(iris)
```

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

To bind rows use *rbind*

It is also possible to combine iris with index in a list, which is a different type of data structure. Lists are very flexible, it is possible to have a list with different combinations of vectors, matrices, data frames etc.

```
iris.list <- list(iris, index)
str(iris.list)
```

```
## List of 2
##  $ :'data.frame':    150 obs. of  6 variables:
##   ..$ Sepal.Length: num [1:150] 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##   ..$ Sepal.Width : num [1:150] 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##   ..$ Petal.Length: num [1:150] 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##   ..$ Petal.Width : num [1:150] 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 ...
##   ..$ index       : num [1:150] 1 1 1 1 1 1 1 1 1 1 ...
##  $ : num [1:150] 1 1 1 1 1 1 1 1 1 1 ...
```

Lists have a different way of indexing its contents.

```
head(iris.list[[1]])
```

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

```
iris.list[[2]]
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [71] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3
## [106] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [141] 3 3 3 3 3 3 3 3 3 3
```

Naming the elements of a list

```
names(iris.list) <- c("iris.df", "Species.index")
head(iris.list$iris.df)
```

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

> CHALLENGE 5

***For* loops. Loops can be used to repeat chunks of code. *paste* is a function to paste characters, we are using it to paste the sepal length to the species name.**

```
for (i in 1:10) {
  print(paste(iris$Sepal.Length[i],iris$Species[i], sep="_"))
}
```

```
## [1] "5.1_setosa"
## [1] "4.9_setosa"
## [1] "4.7_setosa"
## [1] "4.6_setosa"
## [1] "5_setosa"
## [1] "5.4_setosa"
## [1] "4.6_setosa"
## [1] "5_setosa"
## [1] "4.4_setosa"
## [1] "4.9_setosa"
```

```
  #for(i in 1:nrow(iris)){
#print(paste(iris$Sepal.Length[i]/iris$Sepal.Width[i],iris$Species[i], sep="_"))
#}
```

***Functions*. If you are using the same code to process different input data, it is useful to create a function.**

```
f_to_c <- function(temp_F) {
  temp_C <- (temp_F - 32) * 5 / 9
  return(temp_C)
}

f_to_c(32)
```

```
## [1] 0
```

You can obviously put a function inside a loop and vice-versa.

> CHALLENGE 6