First R Absolute Beginner Meet Up

6/16/2019

Brief introduction of R

https://www.r-project.org/

- Language and environment for statistical computing and graphics.

- Similar to the S language and environment, different implementation.

- Much code written for S runs unaltered under R.

- Open Source!
- R is available as Free Software
- It runs on a wide variety of UNIX platforms and similar systems (including FreeBSD and Linux), Windows and MacOS.

Why is it called R?

Based on the (first) names of the first two R authors Robert Gentleman and Ross Ihaka (Dept. of Statistics, University of Auckland) who developed R from S programming languange developed primarily by John Chambers.

Design of the R System

- The “base” R system downloaded from CRAN (Comprehensive R Archive Network) [1]
- Everything else

[1] network of ftp and web servers around the world that store identical, up-to-date, versions of code and documentation for R.

R functionality is divided into a number of packages

- The “base” R system: base package required to run R, contains most fundamental functions
- The other packages, e.g., ***utils***,***datasets***, ***graphics***, ***splines***, etc.

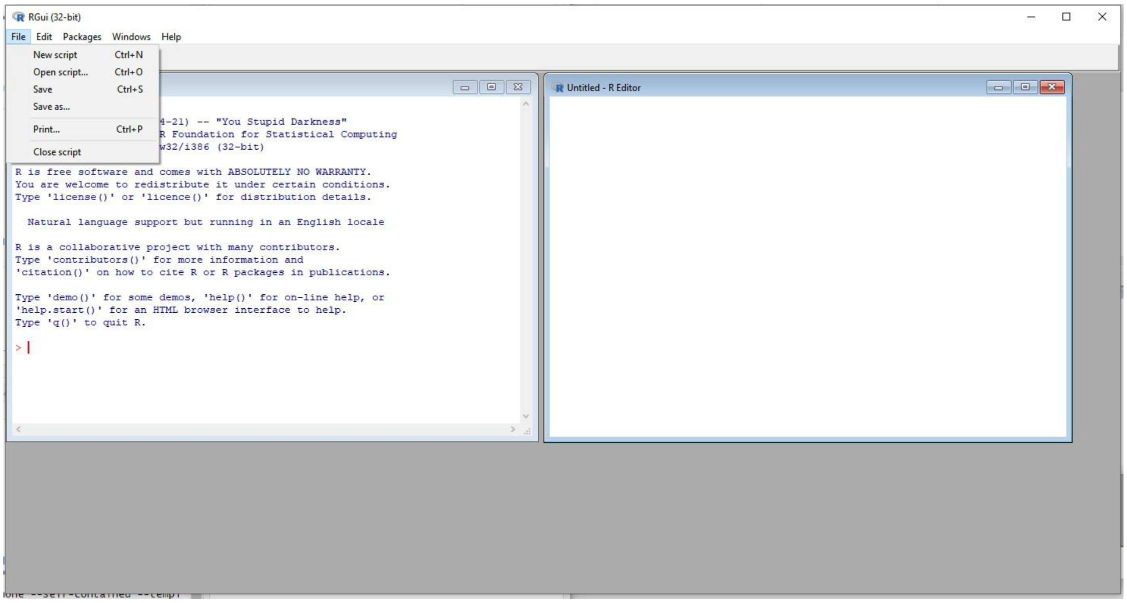
What can you do within R?

- R provides a wide variety of statistical and graphical techniques, and is highly extensible.

- Linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, etc
- Ease with which well-designed publication-quality plots can be produced, the user retains full control.

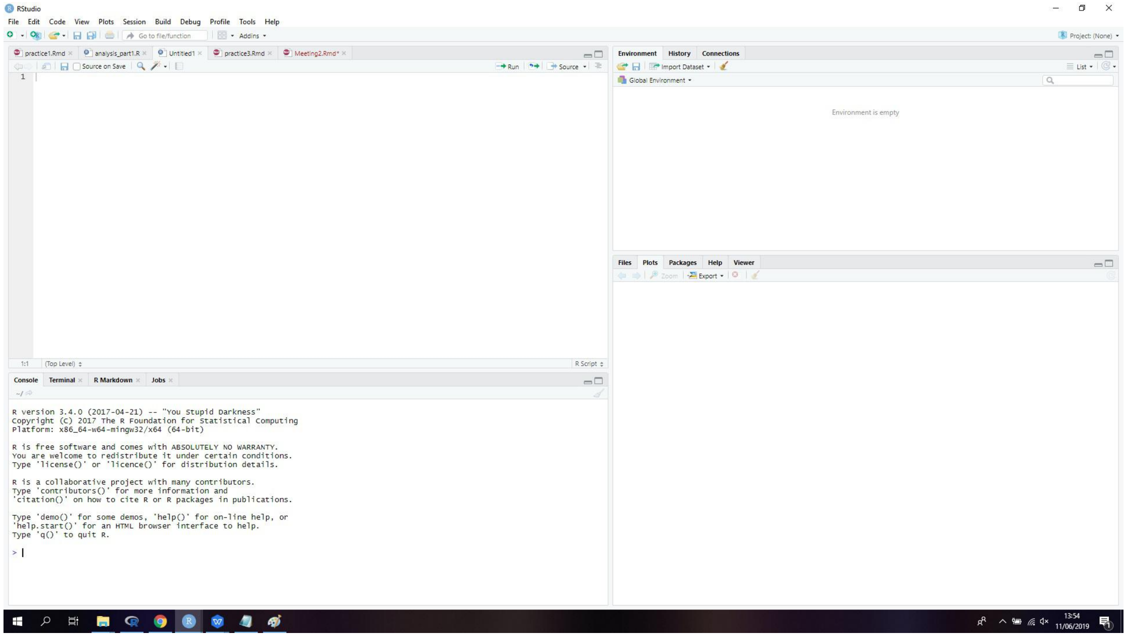
How to use R

R-base software



```
4-21) -- "You Stupid Darkness"
R Foundation for Statistical Computing
w32/i386 (32-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```

R-studio software



```
R version 3.4.0 (2017-04-21) -- "You Stupid Darkness"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> 
```

A few important syntax conventions in R

- R is case sensitive - so be very careful in the use of upper and lower case.
- / - forward slash is used in all path names (as opposed to the backward slash ‘’), or use double backward slash.
- ‘ and “ (single and double quotes) are used interchangeably as long as they are paired.
- () refers to functions and contains the arguments of the corresponding function.
- [] refers to indexing and references row and/or column elements of a data structure.

Let’s start using R!

R makes use of the # sign to add comments, so that you and others can understand what the R code is about. Comments are not run as R code, so they will not influence your result. For example, type in your R script:

```
# 3 + 4
```

and then, within the line click **ctrl + R** or right click and then click **Run line or selection**

What happened?

Try again by typing, and then run the line

```
3 + 4
```

You can also execute R commands straight in the console by typing your command and then click **enter**. This is a good way to experiment with R code, as your submission is not checked for correctness.

Arithmetic in R

In its most basic form, R can be used as a simple calculator.
Consider the following arithmetic operators:

- Addition: +
- Subtraction: -
- Multiplication: *
- Division: /
- Exponentiation: ^

Calculate the following =

- sum 5 with 17
- multiply 2.3 with 0.3
- calculate sum of 3 and 3.6, then, raised to the power of 2

What are the results?

```
5 + 17

## [1] 22

2.3*0.3

## [1] 0.69

(3+3.6)^2

## [1] 43.56
```

Variable assignment

A basic concept in (statistical) programming is called a variable. A variable allows you to store a value (e.g. 4) or an object (e.g. a function description) in R. You can then later use this variable's name to easily access the value or the object that is stored within this variable.

- Example:
  Assign the value 67.02 to x

```
x <- 67.02
x

## [1] 67.02
```

We can also run a command on the assigned variables. Example:

- Assign the value 5 to the variable `orange1`

- Assign the value 8 to the variable `orange2`

- Add `orange1` and `orange2`

```
orange1 <- 5
orange2 <- 8
orange1 + orange2

## [1] 13
```

Next, assign the sum into variable `my_oranges`

```
my_oranges <- orange1 + orange2
my_oranges

## [1] 13
```

What will happen if we assign a text to a variable?

Try to assign 'six' to variable `my_oranges` and print the result

```
my_oranges <- 'six'
my_oranges
```

```
## [1] "six"
```

Data types in R

- Decimals values like 4.5 are called numerics.
- Natural numbers like 4 or -4 are called integers. Integers are also numerics.
- Boolean values (TRUE or FALSE) are called logical.

- Text (or string) values are called characters.

Assign these values to these variables:

42 to `my_numeric`
“some text” to `my_character`
TRUE to `my_logical`

And then check the class of each variables using **class()** command

```
my_numeric <- 42.5

my_character <- "some text"

my_logical <- TRUE

class(my_numeric)

## [1] "numeric"

class(my_character)

## [1] "character"

class(my_logical)

## [1] "logical"
```

Importing Data into R

by Karlijn Willems

Checklist that will make it easier to import the data correctly into R:

- If you work with spreadsheets, the first row is usually reserved for the header, while the first column is used to identify the sampling unit;

- Avoid names, values or fields with blank spaces, otherwise each word will be interpreted as a separate variable, resulting in errors that are related to the number of elements per line in your data set;
- If you want to concatenate words, inserting a . in between to words instead of a space;
- Short names are prefered over longer names;
- Try to avoid using names that contain symbols such as ?, $,%, ^, &, *, (,),-,#, ?,,,<,>, /, |, , [,] ,{, and };
- Delete any comments that you have made in your Excel file to avoid extra columns or NA’s to be added to your file; and
- Make sure that any missing values in your data set are indicated with NA.

Preparing your R workspace

Removing previous data and values

You might have an environment that is still filled with data and values, therefore you can delete all that with the following code:

```
rm(list=ls())
```

`rm()` function allows you to remove objects from an environment.
In this case, you specify that you want to consider a list for this function, which is the outcome of the `ls()` function. This last function returns you a vector of character strings that gives the names of the objects in the specified environment. Since this function has no argument, it is assumed that you mean the data sets and functions that you as a user have defined.

Setting work directory

You can also check and set your working directory. This helps to connect you to your dataset during the whole R session.

to check the directory you can type:

```
getwd()

## [1] "D:/Temporary work folder/[40] R ladies"
```

if you want to change your directory, then type:

```
setwd('D:\\Temporary work folder\\[40] R ladies')
```

Reading data into R

Next, you'd want to read your data into R. You can read csv, txt, html, and other common files into R. We will discuss for today, how to read txt and csv into R.

If you have a `.txt` or a tab-delimited text file, you can import it using the `read.table()`. There are many variants on how to read a table, especially due to variant use of comma:

- `read.csv()`: for reading "comma separated value" files (".csv").
- `read.csv2()`: variant used in countries that use a comma "," as decimal point and a semicolon ";" as field separators.
- `read.delim()`: for reading "tab-separated value" files (".txt"). By default, point (".") is used as decimal points.
- `read.delim2()`: for reading "tab-separated value" files (".txt"). By default, comma (",") is used as decimal points.

Some format on the functions are as follow:

```
# Read tabular data into R read.table(file, header = FALSE, sep = "", dec =
".")
```

```
# Read "comma separated value" files (".csv") read.csv(file, header = TRUE,
sep = ",", dec = ".", ...)
```

```
# Or use read.csv2: variant used in countries that # use a comma as decimal
point and a semicolon as field separator. read.csv2(file, header = TRUE, sep
= ";", dec = ",", ...)
```

```
# Read TAB delimited files read.delim(file, header = TRUE, sep = "\t", dec =
".", ...) read.delim2(file, header = TRUE, sep = "\t", dec = ",", ...)
```

- `file`: the path to the file containing the data to be imported into R.
- `sep`: the field separator character. " is used for tab-delimited file.
- `header`: logical value. If TRUE, read.table() assumes that your file has a header row, so row 1 is the name of each column. If that's not the case, you can add the argument header = FALSE.
- `dec`: the character used in the file for decimal points.

Let's check the following data set:

`TextPrices` data set

Description Prices and number of pages for a sample of college textbooks

`Format`: A dataset with 30 observations on the following 2 variables.
`Pages`: Number of pages in the textbook
`Price`: Price of the textbook (in dollars)

Details

Two undergraduate students at Cal Poly - San Luis Obispo took a random sample of 30 textbooks from the campus bookstore in the fall of 2006. They recorded the price and number of pages in each book, in order to investigate the question of whether number of pages can be used to predict price.

Let’s try to read these files:

```
#pass in the file name and the extension because you have set your
#working directory to the folder in which your data set is located

df<-read.delim("TextPrices.txt", header = T)
head(df)
```

```
##   X Pages  Price
## 1 1   600  95.00
## 2 2    91  19.95
## 3 3   200  51.50
## 4 4   400 128.50
## 5 5   521  96.00
## 6 6   315  48.50
```

```
#or you can direct to a specific folder where your data set is
df<-read.delim("D:\\R Practice\\TextPrices.txt", header = T)
df<-read.csv("D:\\R Practice\\TextPrices.csv", header = T)
```

It is also possible to read a file from the internet

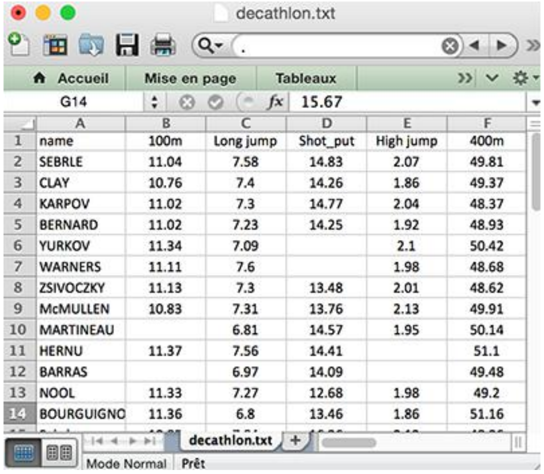
```
my_data <- read.delim("http://www.sthda.com/upload/boxplot_format.txt")
head(my_data)
```

```
##    Nom variable Group
## 1 IND1       10     A
## 2 IND2        7     A
## 3 IND3       20     A
## 4 IND4       14     A
## 5 IND5       14     A
## 6 IND6       12     A
```

Preparing your data before importing into R

To avoid errors during the importation of a file into R, you should make sure that your data is well prepared.

Open your file



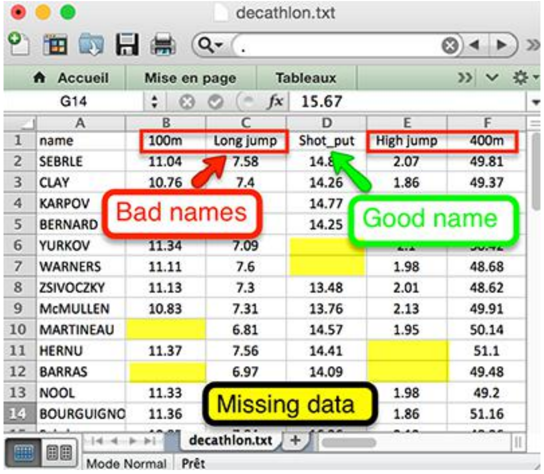
	A	B	C	D	E	F
1	name	100m	Long jump	Shot_put	High jump	400m
2	SEBRLE	11.04	7.58	14.83	2.07	49.81
3	CLAY	10.76	7.4	14.26	1.86	49.37
4	KARPOV	11.02	7.3	14.77	2.04	48.37
5	BERNARD	11.02	7.23	14.25	1.92	48.93
6	YURKOV	11.34	7.09		2.1	50.42
7	WARNERS	11.11	7.6		1.98	48.68
8	ZSIVOCZKY	11.13	7.3	13.48	2.01	48.62
9	McMULLEN	10.83	7.31	13.76	2.13	49.91
10	MARTINEAU		6.81	14.57	1.95	50.14
11	HERNU	11.37	7.56	14.41		51.1
12	BARRAS		6.97	14.09		49.48
13	NOOL	11.33	7.27	12.68	1.98	49.2
14	BOURGUIGNO	11.36	6.8	13.46	1.86	51.16

Prepare your file

Row and column names

- Use the first row as column headers (or column names). Generally, columns represent variables.
- Use the first column as row names. Generally rows represent observations.
- Each row name should be unique, so remove duplicated names.

As we can see below, there are some issues in the data set

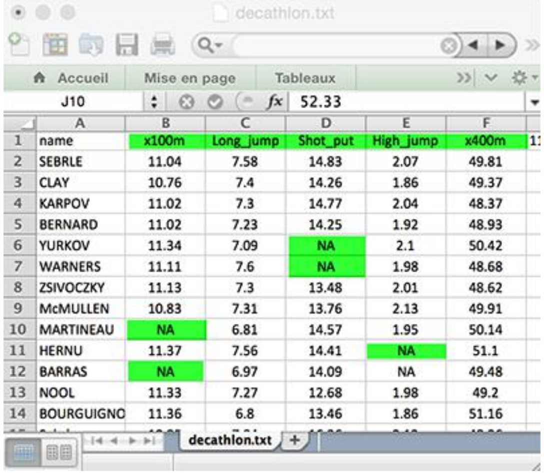


Naming conventions

- Avoid names with blank spaces. Good column names: Long_jump or Long.jump. Bad column name: Long jump.
- Avoid names with special symbols: ?, $, *, +, #, (,), -, /, }, {, |, >, < etc. Only underscore can be used.

- Avoid beginning variable names with a number. Use letter instead. Good column names: sport_100m or x100m. Bad column name: 100m
- Column names must be unique. Duplicated names are not allowed.
- R is case sensitive. This means that Name is different from Name or NAME.
- Avoid blank rows in your data
- Delete any comments in your file
- Replace missing values by NA (for not available)
- If you have a column containing date, use the four digit format. Good format: 01/01/2016. Bad format: 01/01/16

Final file



	A	B	C	D	E	F
1	name	x100m	Long_jump	Shot_put	High_jump	x400m
2	SEBRLE	11.04	7.58	14.83	2.07	49.81
3	CLAY	10.76	7.4	14.26	1.86	49.37
4	KARPOV	11.02	7.3	14.77	2.04	48.37
5	BERNARD	11.02	7.23	14.25	1.92	48.93
6	YURKOV	11.34	7.09	NA	2.1	50.42
7	WARNERS	11.11	7.6	NA	1.98	48.68
8	ZSIVOCZKY	11.13	7.3	13.48	2.01	48.62
9	McMULLEN	10.83	7.31	13.76	2.13	49.91
10	MARTINEAU	NA	6.81	14.57	1.95	50.14
11	HERNU	11.37	7.56	14.41	NA	51.1
12	BARRAS	NA	6.97	14.09	NA	49.48
13	NOOL	11.33	7.27	12.68	1.98	49.2
14	BOURGUIGNO	11.36	6.8	13.46	1.86	51.16

Save your file into `.txt` (tab-delimited text file) or `.csv` (comma separated value file) format.

Now, let's try check on some of these data sets.

`Traffic` data set
Effect of Swedish Speed Limits on Accidents

An experiment was performed in Sweden in 1961–2 to assess the effect of a speed limit on the motorway accident rate. The experiment was conducted on 92 days in each year, matched so that day j in 1962 was comparable to day j in 1961. On some days the speed limit was in effect and enforced, while on other days there was no speed limit and cars tended to be driven faster. The speed limit days tended to be in contiguous blocks.

`Format` This data frame contains the following columns:

`year` 1961 or 1962.

`day` of year.

`limit` was there a speed limit?

y traffic accident count for that day.

```
df<-read.csv("Traffic2.csv")
#to see the first lines of your data set
head(df)
```

```
##   X year.1961.1962 day_1_92 limit  y
## 1 1           1961        1    no  9
## 2 2           1961        2    no 11
## 3 3           1961        3    no  9
## 4 4           1961        4    no 20
## 5 5           1961        5    no 31
## 6 6           1961        6    no 26
```

```
#to see the first 4 lines of your data set
head(df,n=4)
```

```
##   X year.1961.1962 day_1_92 limit  y
## 1 1           1961        1    no  9
## 2 2           1961        2    no 11
## 3 3           1961        3    no  9
## 4 4           1961        4    no 20
```

```
#to see the last lines of your data set
tail(df)
```

```
##       X year.1961.1962 day_1_92 limit  y
## 179 179           1962       87   yes 24
## 180 180           1962       88   yes 16
## 181 181           1962       89   yes 25
## 182 182           1962       90   yes 14
## 183 183           1962       91   yes 15
## 184 184           1962       92   yes  9
```

```
#to display internal structure of an R object (alternative to summary)
str(df)
```

```
## 'data.frame':    184 obs. of  5 variables:
##  $ X             : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ year.1961.1962: int  1961 1961 1961 1961 1961 1961 1961 1961 1961 1961
 ...
##  $ day_1_92      : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ limit         : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ y             : int  9 11 9 20 31 26 18 19 18 13 ...
```

```
#to invoke a spreadsheet-style data viewer on a matrix-like R object
View(df)
#to perform an operation on a structured blob of data
```

```
#(in this case, to calculate mean value of data set df)
sapply(df, mean, na.rm=TRUE)

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

##              X year.1961.1962       day_1_92          limit              y

##       92.50000     1961.50273       46.69399             NA       21.62360
```

Descriptive statistics

R provides a wide range of functions for obtaining summary statistics. One method of obtaining descriptive statistics is to use the `sapply()` function with a specified summary statistic.

`Allbacks` data set

Description The allbacks data frame gives measurements on the volume and weight of 15 books, some of which are softback (pb) and some of which are hardback (hb). Area of the hardback covers is also included.

`Format` This data frame contains the following columns:

`volume` Book volumes in cubic centimeters

`area` Hard board cover areas in square centimeters

`weight` Book weights in grams

`cover` A factor with levels hb hardback, pb paperback

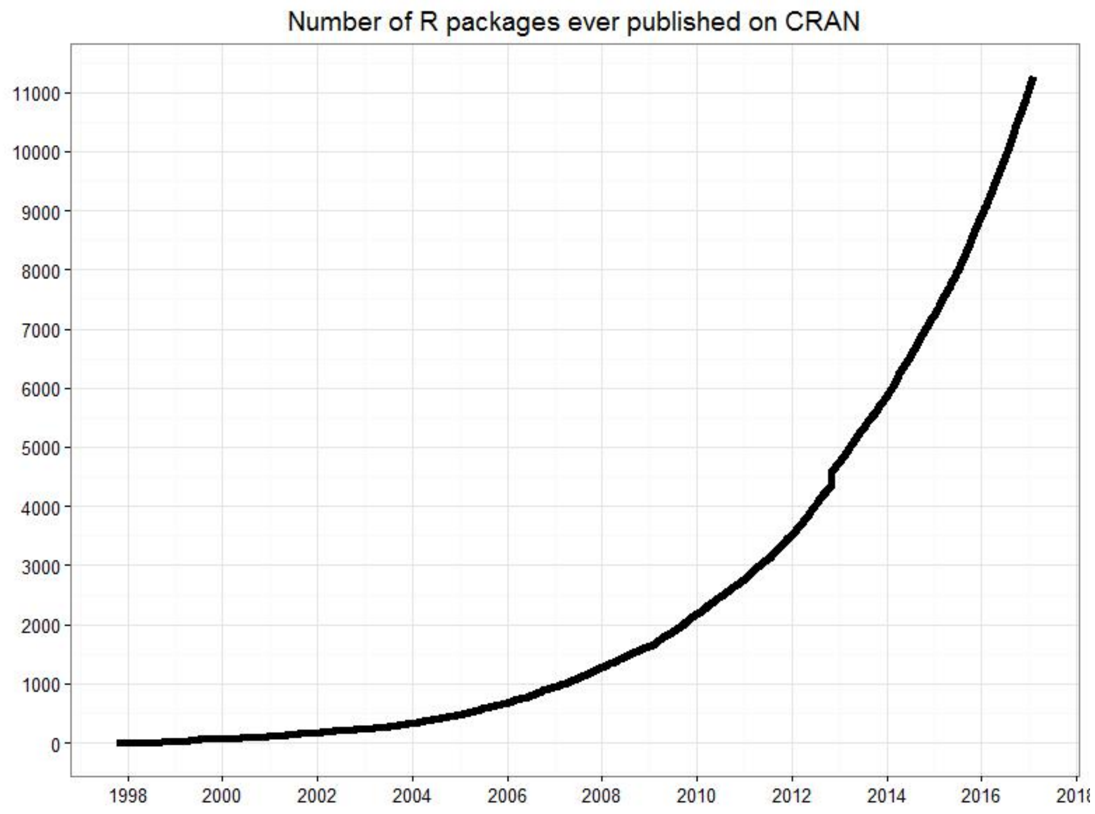
```
# get means for variables in data frame mydata
# excluding missing values

df<-read.csv("allbacks.csv")
head(df)

##   X volume area weight cover
## 1 1    885  382    800    hb
## 2 2   1016  468    950    hb
## 3 3   1125  387   1050    hb
## 4 4    239  371    350    hb
## 5 5    701  371    750    hb
## 6 6    641  367    600    hb
```

Possible functions used in sapply include mean, sd, var, min, max, median, range, and quantile.

```
# mean,median,25th and 75th quartiles,min,max

mean(df$area)

## [1] 182.8

max(df$area)

## [1] 468

sd(df$area)

## [1] 203.5868

quantile(df$area)

##    0%   25%   50%   75%  100% 
##   0.0   0.0   0.0 376.5 468.0

median(df$area)

## [1] 0

summary(df$area)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##     0.0     0.0     0.0   182.8   376.5   468.0
```

Using packages

In R, the fundamental unit of shareable code is the package. A package bundles together code, data, documentation, and tests, and is easy to share with others. As of January 2017, there were over 10,000 packages available on the Comprehensive R Archive Network, or CRAN, the public clearing house for R packages. This huge variety of packages is one of the reasons that R is so successful: the chances are that someone has already solved a problem that you're working on, and you can benefit from their work by downloading their package.



How you can use packages:

- Install them from CRAN with `install.packages("x")`

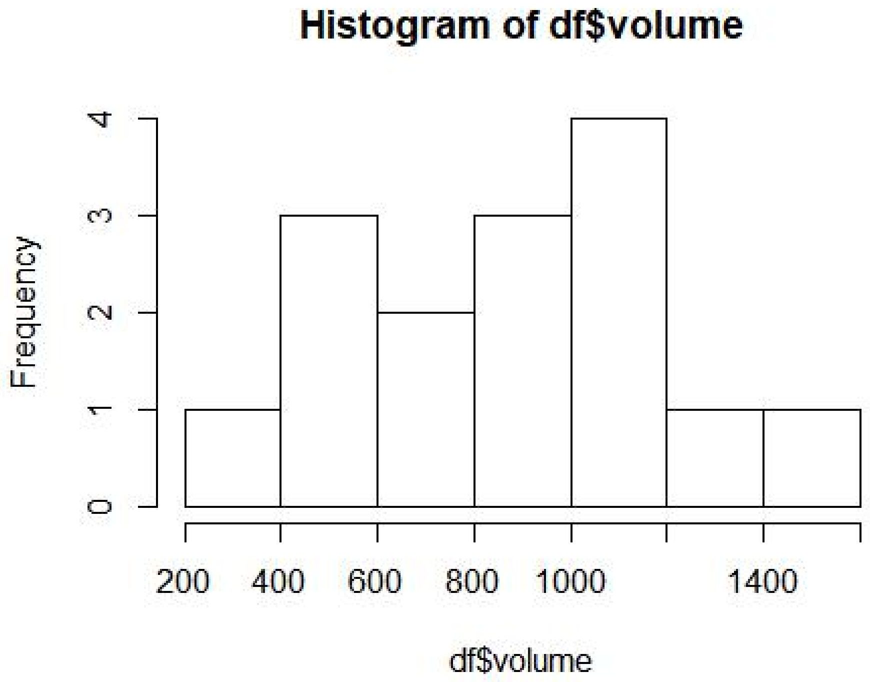
- Use them in R with `library("x")`

- Get help on them with `package?x` and `help(package = "x")`

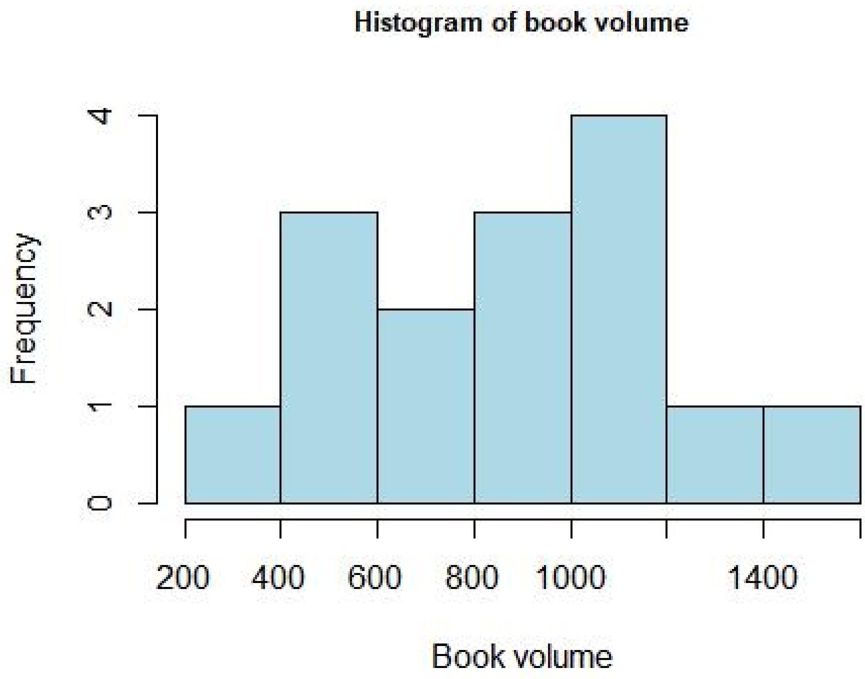
Using base R, we can prepare a histogram. Let's see how a histogram is generated in R and install package `ggplot2` to see how it's done within this package.

```
#we first check our data set
head(df)

##   X volume area weight cover
## 1 1    885  382    800    hb
## 2 2   1016  468    950    hb
## 3 3   1125  387   1050    hb
## 4 4    239  371    350    hb
## 5 5    701  371    750    hb
## 6 6    641  367    600    hb
```

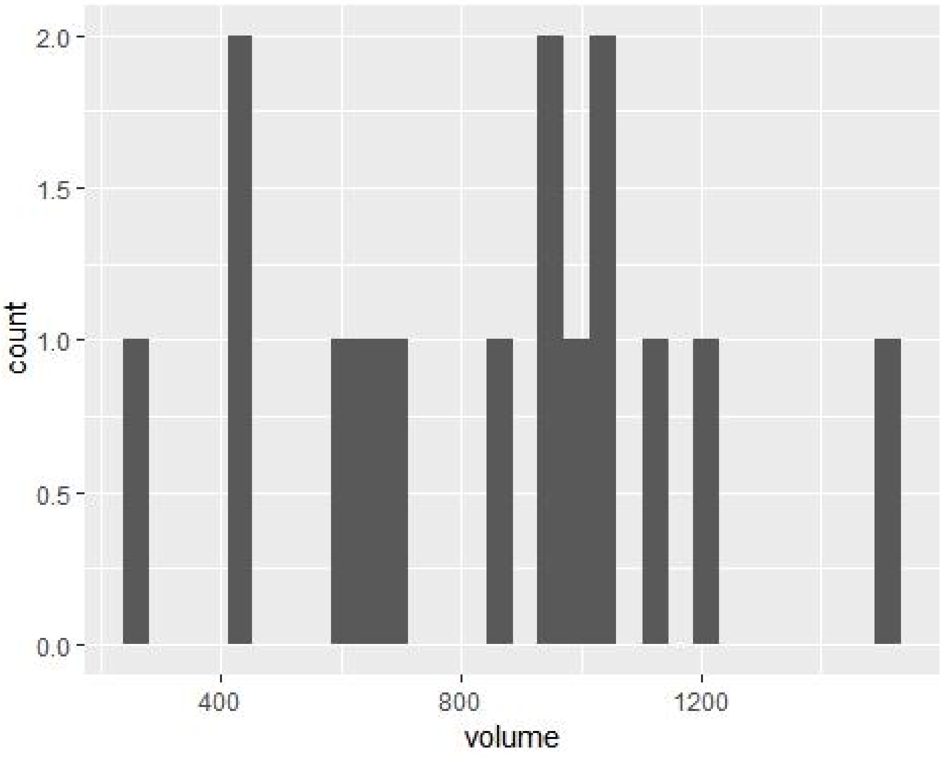
```
#we want to make a histogram for the book volume variable
hist(df$volume)
```



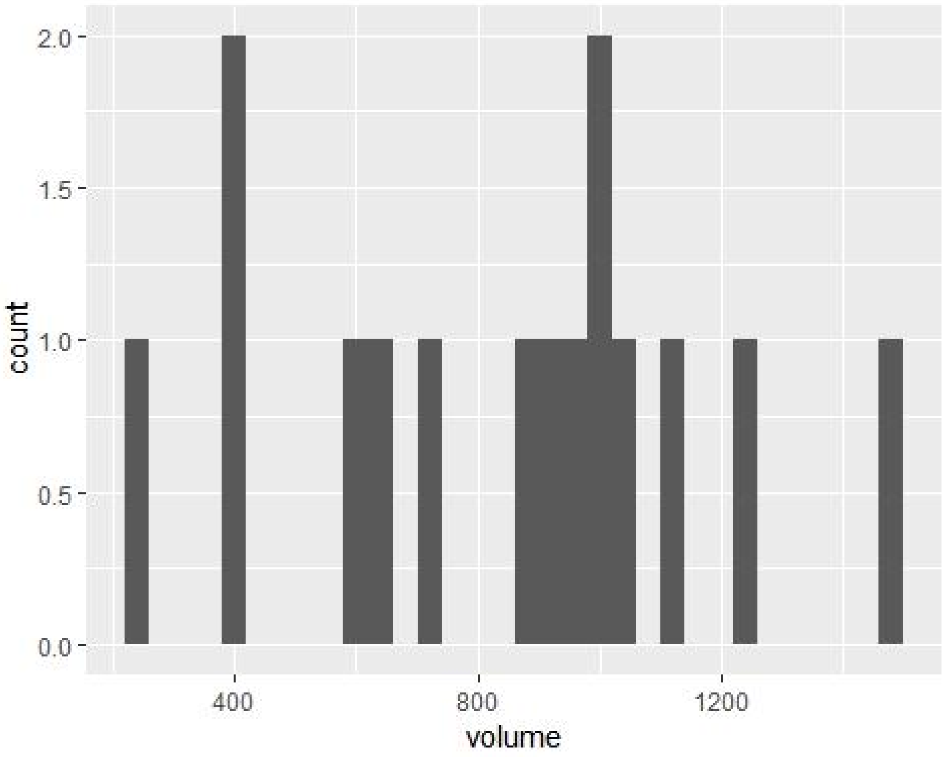
```
hist(df$volume, xlab="Book volume",
     main="Histogram of book volume",cex.main=0.8,col="lightblue")
```



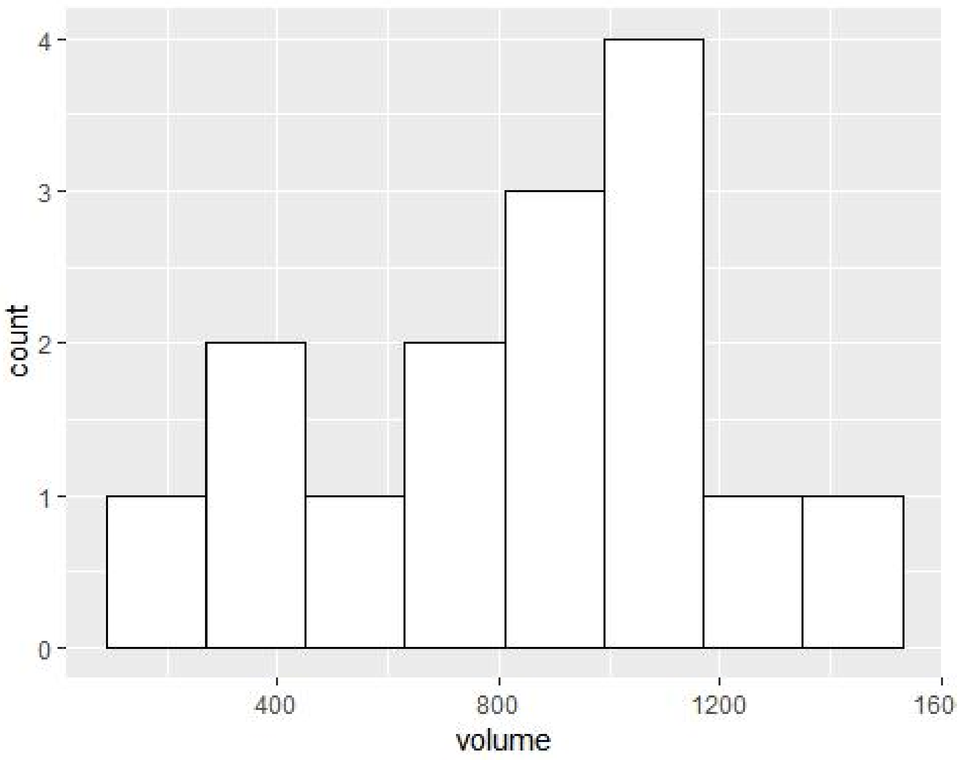
```
#we want to use ggplot2 package

#install.packages("ggplot2")
library(ggplot2)

## Warning: package 'ggplot2' was built under R version 3.4.4

# Basic histogram
ggplot(df, aes(x=volume)) + geom_histogram()

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



```
# Change the width of bins
ggplot(df, aes(x=volume)) +
  geom_histogram(binwidth=40)
```



```
# Change colors
p<-ggplot(df, aes(x=volume)) +
  geom_histogram(color="black", fill="white",binwidth=180)
p
```



Homework

Check Titanic data set

Survival of Passengers on the Titanic

Description Information on the survival status, sex, age, and passenger class of 1309 passengers in the Titanic disaster of 1912.

`Format` A data frame with 1309 observations on the following 4 variables.

`survived` no or yes.

`sex` female or male

`age` in years (and for some children, fractions of a year); age is missing for 263 of the passengers.

`passengerClass` 1st, 2nd, or 3rd class.

What you should do:

- Check the dataset, are all the columns already filled in? Are all the labels have been properly named? If not, what can you do?
- Answer these:

1. What is the average age of women who survived?
2. Which passenger class survived the most?
3. Can you make a histogram of the age of male who didn't survive?

Please send the answer to: umardhiahsir@gmail.com by 24th of June 2019!

References

- https://vincentarelbundock.github.io/Rdatasets/datasets.html
- https://www.datacamp.com/community/tutorials/r-data-import-tutorial#Getting
- http://www.sthda.com/english/wiki/reading-data-from-txt-csv-files-r-base-functions
- http://www.sthda.com/english/wiki/best-practices-in-preparing-data-files-for-importing-into-r
- https://www.statmethods.net/stats/descriptives.html
- http://r-pkgs.had.co.nz/intro.html
- https://blog.revolutionanalytics.com/2017/01/cran-10000.html
- http://www.sthda.com/english/wiki/ggplot2-histogram-plot-quick-start-guide-r-software-and-data-visualization
- https://www.statmethods.net/stats/descriptives.html
- https://twitter.com/statsgen/status/995456566403854336