R Notes

Mehmet Tevfik DORAK
http://www.dorak.info/r

These notes are aimed at absolute beginners of learners of R program who want to "use R" rather than write codes for R (those are developers and they know where to go). If you just want to use the existing capabilities of R programme and currently (as of 2019) more than 16,000 packages associated with it, go ahead. If you want to be a developer to add to the existing capabilities, this site is not for you (but only after a while; if you are an absolute beginner, you may still benefit from the basics in these notes). Please go through these notes and associated presentations at http://www.dorak.info/r at your own pace. Don't skip the links provided to see what options are available to develop your R skills. Make sure you run the R scripts provided (as prompted) and start writing your own right away. Good luck!

The **R Project for Statistical Computing** (or just **R** for short) is a powerful data analysis tool. It is both a programming language and a computational and graphical environment.

Selected links relevant to R and learning R

- R Packages: https://rdrr.io
- Quick-R: http://www.statmethods.net (BOOK: R in Action)
- Instant R: http://www.instantr.com (BOOKs: InstantR (PDF) > Using R for Statistics; GitHub)
- Cookbook for R: http://www.cookbook-r.com (BOOK: R Graphics Cookbook)
- R Cookbook: http://shop.oreilly.com/product/9780596809164.do (PDF)
- Learn R (Data Science Made Simple): http://www.datasciencemadesimple.com/learn-r-what-is-r
- How To In R: http://howtoinr.weebly.com
- Statistical Tools for High-throughput Data Analysis (STHDA) – Basic R Training: http://www.sthda.com/english
- Kickstarting R: https://cran.r-project.org/doc/contrib/Lemon-kickstart
- R Tutorials on R-bloggers: http://rtutorialseries.blogspot.co.uk/search/label/tutorial
  Introduction Tutorials:
  Part 1: http://www.r-bloggers.com/r-tutorial-series-introduction-to-the-r-project-for-statistical-computing-part-1
  Part 2: http://rtutorialseries.blogspot.co.uk/2009/10/r-tutorial-series-introduction-to-r.html
- R Tutorials @ EndMemo: http://www.endmemo.com/program/R
- R Tutorials @ ListenData: https://www.listendata.com/p/r-programming-tutorials.html
- R Tutorials @ TutorialPoint: https://www.tutorialspoint.com/r (Quick Guide)
- R Tutorials @ DataMentor: https://www.datamentor.io/r-programming
- R Tutorials @ R-Chart: http://www.r-chart.com (emphasis on graphics)
- R Tutorials @ UC Business Analytics R Programming Guide: http://uc-r.github.io
- R Tutorials by Kelly Black (with case studies): https://www.cyclismo.org/tutorial/R/index.html
- R Tutorials @ DataScienceMadeSimple: http://www.datasciencemadesimple.com/learn-r-what-is-r
- R Tutorials @ R Tutorial: http://www.r-tutor.com (Statistics: http://www.r-tutor.com/elementary-statistics)
- R Tutorials @ R-Statistics: http://r-statistics.co/R-Tutorial.html
- YaRrr! The Pirate's Guide to R: https://bookdown.org/ndphillips/YaRrr
- UCLA Statistics/R pages: http://statistics.ats.ucla.edu/stat/r
- Practical R courses at Data Novia
- R Blog (Open Source Automation): http://theautomatic.net/category/r
- Cheat Sheets: R-Graph Gallery; RStudio; DataCamp (see also: R Reference Card & v.2)
- R in Action (live online book): https://livebook.manning.com/#!/book/r-in-action-second-edition/about-this-book
- Practical Data Science with R (live online book): https://livebook.manning.com/#!/book/practical-data-science-with-r
- R for Data Science (live online book): https://r4ds.had.co.nz
- Learn R (Online R Books): http://ucanalytics.com/blogs/learn-r-12-books-and-online-resources
- Learning R- The Ultimate Introduction (incl. Machine Learning): http://blog.ephorie.de/learning-r-the-ultimate-introduction-incl-machine-learning

- CRAN Task Views: https://cran.r-project.org/web/views

R syntax and related tips

- R is a case-sensitive language: `str()` is different `Str()`
- Variable names cannot start with a number
- Spacing is variable, one-space or two-spaces will not matter. Even splitting a line and having an indentation in the second line is fine.
- Single quote (‘) or double quote (“) can be used for the same purpose as long as they are used consistently
- Beware of WORD replacing straight quotes (" … ") with smart quotes (“ … ”). Turn this feature off at AutoCorrect options, and never use WORD as a text editor
- String variables must be written within quotes, numeric variables are without quotes
- Dollar sign (`$`) signifies the column name within a dataset/dataframe (like `iris$Species`)
- Every function ends with a pair of brackets (even if they are empty as in `getwd()`)
- Each function may have multiple arguments/options/parameters separated by commas (and usually with default values) within brackets
- A function may be typed (in a script) with lots of space, including hard return, between its elements and even comments (beginning with `#`) inserted between elements. For example, the following command:

```
boxplot(iris[1:4],
boxwex =0.2,
pch=       14)
# you can even have a comment at the end of first or second line above
```

  is equivalent to:

```
boxplot(iris[1:4], boxwex = 0.2, pch = 14)
```

- In a user-defined function, elements between braces ({…}) are the body of the function (equivalent to indented bits in a Python function)
- Scripts can be saved as text files with file extension (.R) and run from R after highlighting the script and pressing CTRL + **R** (or right click and "**R**un line or selection")
- R function names with more than one word have a dot between the words (like `install.packages`())
- Double column (::) separates a function name from its package name (like `fBasics::basicStats`). This is important if the same function name is used in multiple packages
- CTRL + **L**: C**l**ears the console (without unloading any variable, library, object etc)
- When generating a set of numbers, setting a seed number generator with `set.seed()` is not necessary, but it will make your results reproducible
- R uses forward slices in directory addresses. It is not "C:\R", but "C:/R"
- If you see double colons like in the following: `DescTools::Cstat()`, it means the `Cstat()` function is from the package named `DescTools`.
- If you want to print an object after creating it (and without asking for a print on a separate line), put the assignment line within brackets: `(x <- (3 * 4))`. This function will create the "x" object and will print its value.
- Escape sequences in R: http://127.0.0.1:20926/library/base/html/Quotes.html (or type `?Quotes` in R). The most commonly used ones are **\n** for new line (linefeed) and **\r** for carriage return. For example in a ggplot graphic the if you want the title to appear in two lines: `ggtitle("This is the Title \n This is the Subtitle")`
- SEE ALSO: Good Practices in R Programming (saved as DOs and DON'Ts in R language.PDF); Common Uncommon Notations that Confuse New R Coders (saved as Common Uncommon Notations that Confuse New R Coders.PDF).

Most common sources of error messages

- Typos (comma instead of dot; pound sign or ampersand instead of dollar sign; leaving out the dot in multi-word function names like read.delimit() or as.factor(); case sensitivity (including WORD changing the initial to a capital letter)
- Parentheses (type, unequal opening and closing parentheses)

- Quotes; including WORD changing your straight quotes (" … ") with smart quotes (“ … ”); inconsistent use of single and double quotes
- Missing commas (e.g., between function arguments)
- Your grouping variable is NOT a factor (but numerical or string/character variable; check with class())
- Missing data is causing trouble
- The function you use exists in two different libraries in use (loaded)
- The library you intend to use is not loaded
- In path definition, R uses forward slash like C:/R (not back slash as in Windows Explorer like C:\R)

The assignment operator <−

You will make lots of assignments using the operator **<−** (aRrow). Don’t use = for the same purpose. Even if it works, it may cause some confusion.

Notice that RStudio automatically surrounds `<-` with spaces, which demonstrates a useful code formatting practice. It is good practice to use spaces on both sides of all operators.

RStudio offers many handy keyboard shortcuts. Also, Alt+Shift+K brings up a keyboard shortcut reference card.

If you get stuck with some syntax (usually, mismatched parentheses or quotes), the R Console will change from the > at the beginning of the line (which means it is waiting for a new command) to the + at the beginning of the line (which means it is waiting for you to finish a command). To get out, hit the **Escape** key.

R language

The screen prompt, >, is the R prompt that waits for commands. The output following the entry of a command is prefixed by [1], which indicates that this is item 1 in a vector of output, and makes the output results easier to be identified. Braces { } are used to group together one or more expressions (like indented blocks in Python). R is a strictly case-sensitive language.

Running R online

If you cannot install R, you can get at least some of the functionality from different providers of online R:

- https://www.tutorialspoint.com/execute_r_online.php
- https://rdrr.io/snippets
- http://www.roncloud.com
- http://www.compileonline.com/execute_r_online.php
- http://www.r-fiddle.org
- http://pbil.univ-lyon1.fr/Rweb/Rweb.general.html
- https://app.displayr.com (for making graphs online from data frames; NOT free)

Start-up screen

On the start-up screen, a function or a command can be typed, or R can be used to perform basic calculations. R uses the usual symbols for addition (+), subtraction (-), multiplication (*), division (/), and exponentiation (^). R uses %% for taking the modulus and %/% for integer division. Comments in R begin with the character # (everything after # up to the end of the line will be ignored by R). If you do not want the lines beginning with # to be interpreted as comment, you can disable this feature by specifying `comment.char = " "` . To clear the standard information from the screen, use CTRL + L.

What to do at the start of a session

- You may want to start a new session on R by the following command: `rm(list=ls())` , which removes (rm) all variables already in the memory.

- If you are going to download lots of packages, start with: `chooseCRANmirror()` OR select "Set CRAN Mirror..." in the Packages tab in the menu, and select the closest mirror to you for all downloads in the current session.

- Get your working directory (default folder):
  `getwd()` # which returns something like: "C:/Users/MD/Documents".
  Note that R uses forward slashes (as in web addresses but not in Windows locations)

- If you need to keep things simple, change your working directory to something simple:
  `setwd("C:/R")` which will change the working directory only for the current session.

- Check the version of R if you like:
  `R.Version()`
  # OR: Help tab and About

- If you want to make sure that all of your installed packages are up-to-date, run `update.packages()` and select a CRAN mirror to get all missing updates.

If you must run R on a mobile device

There are applications for running R on your iOS or Android device with limited functionality. You can explore that, for instance, when waiting for the bus. If you must use a mobile device, try R online, for example: DataCamp Light (GitHub).

Demos and Help

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.

As a first step with R, start the R help browser by typing `help.start()` in the R command window. For help on any function, e.g. the “mean” function, type `?mean`. Try also the Help tab in the menu on top of R console.

Help manuals

A series of manuals are available in the **Help** section of the R menu (Help > Manuals (in PDF)). For example, the manual **An Introduction to R** contains an introduction to the R language and environment (and includes a Sample R Session in Appendix A). More advanced users may like the **Writing R Extensions** manual, which focusses on creation of R packages. See also R-FAQs.

The Working Directory (R-bloggers)

One of the initial things that you want to do when you launch R is to set its working directory. This is the default location on your hard disk that R will look to read and write files. The working directory is comparable to what is called the "default folder" in many other applications. It is important to select a location that is easy to find and remember, so you can access your files when you need them.

To display the current working directory, use the function `getwd()`.

```
1. > getwd()
2. [1] "/Users/Admin/Documents/R"
```

To change the current working directory, use the function `setwd('PATH')`, replacing PATH with the directory path that you would like to use.

```
1. > setwd('/Users/Admin/Documents/R/newProject')
```

Use `getwd()` again to verify that the change took place.

```
1. > getwd()
2. [1] "/Users/Admin/Documents/R/newProject"
```

This change is only for the current R session.

Note that you have the option to set the working directory at any time. Do this when you want to access files in a new location, such as when you are working on multiple projects at the same time or at the start of a new project. (See also: http://stat545-ubc.github.io/block002_hello-r-workspace-wd-project.html.)

While reading a file into R, if you are not sure about the location, you can use the `file.choose()` function to browse the computer. For example, to read a CSV file (with a header row):

```
> read.csv(file.choose(), header = TRUE)
```

Packages (Quick-R)

Packages are collections of **R** functions, data, and compiled code in a well-defined format. The directory where packages are stored is called the **library**. **Base R** comes with a standard set of packages (like the graphics package `lattice`). Others are available for download and installation. Once installed, they have to be loaded into the session to be used:

Package/library management

```
> chooseCRANmirror()        # To choose the CRAN mirror (or via Packages tab in the menu)
> install.packages("packagename")      # to install a specific package
> install.packages(c("packagename1", "packagename2", "packagename3"))
                            # to install multiple packages
                            # If you add dependencies = TRUE  as an option to
                            install.packages()function, you will also get all the dependencies
                            (additional libraries needed) installed if you do not already have them
> require("packagename")    # checks if the required package installed, and installs if not
> installed.packages()      # details of all packages installed in the specified libraries
> library("packagename")    # to load a specific package in the workspace
                            # no need for quotes, but it is good practice
> search()                  # see packages currently loaded
> library()                 # list of currently installed packages
> (.packages())             # list of currently loaded packages; you need to have brackets as shown
> detach(package:lattice, unload = TRUE)     # to unload a loaded package
> remove.packages(pkgs, lib)  # removes an installed specific package (see link)
> .libPaths()               # get library location (there is a dot in the beginning)
```

Alternatively, for a group of related packages installation (FROM: http://phytools.org/eqg/Exercise_3.2):
Use the package "ctv" (i.e., CRAN Task View) to automatically install & update all the packages for R "phylogenetic analysis" that are available and listed in the Task View.

```
> install.packages('ctv')                              # install CRAN Task View
> library('ctv') install.views('Phylogenetics')    # install all phylogenetics-related packages
> update.views('Phylogenetics')
```

Before you run `install.packages`, go to your R packages directory (`C:/Users/mtd/R/win-library/3.5`) and delete any zero-length files that may be present. These are artefacts of failed install attempts and will break `install.packages()` if present.

How to Install Packages (Quick-R) via the menu

1. Choose **Install Packages** from the **Packages** menu
2. Select a **CRAN Mirror** (e.g. Norway)
3. Select a package (e.g. `fBasics`)
4. Then use the `library("package")` function to load it for use (e.g. `library("ggplot2")`; quotes are optional)

Alternatively, a package can be installed by typing the following command:

```
> install.packages("packagename")
```

SEE a complete list of contributed packages available from **CRAN**.

(the installation via the CRAN repository should be preferred over manual installation. This way, the dependency of the package will also be downloaded automatically, which otherwise would require manual installation.)

http://www.cran.r-project.org/web/views Packages organized by task

R-Forge Software Map: https://r-forge.r-project.org/softwaremap/trove_list.php
http://www.rseek.org Function finder

R objects

Everything that exists in R like variables, vectors, matrices/arrays, lists and data frames are R objects (matrices contain the same type of data; data frames may contain different types of data). In R, an object is anything that can be assigned to a variable. This includes constants, data structures, functions, and even graphs. An object has a mode (which describes how the object is stored) and a class (which tells generic functions like print how to handle it). To name any R object, use the assignment operator (`<-`). See this note about "objects" in R for details.

R commands/functions

Most of the R commands (or functions) end with a couple of parentheses. The parentheses indicate that you are using a function, but not an object. Within the parentheses, the arguments (or options) of the function are specified (for example, the function `boxplot()` has arguments like `col = "red", pch = 21`, etc). If the function doesn't need any argument or default values are fine, no need to include any argument within the parentheses (it can be left blank as in `quit(), windows(), ls()`).

User-defined functions

https://www.statmethods.net/management/userfunctions.html
https://www.datamentor.io/r-programming/function

Reading files into the R environment

```
read.table()
read.csv()        # For arguments of read.csv(),
                    SEE: https://swcarpentry.github.io/r-novice-inflammation/11-supp-read-write-csv
read.csv2()       # In some European countries the delimiter in a .csv is a semicolon ";" and not a comma.
                    If you are using such files use read.csv2() instead of read.csv().
```

If you are not sure what your working directory is, and you want to read a CSV file into R, use the following command:

```
x <- read.csv(file.choose(), header=TRUE)      # if no header, use FALSE
```

This will allow you to browse your computer and locate the file like you do on Windows Explorer.

Alternatively, you can copy your spreadsheet on the clipboard, and import (paste) into R using the following command:

```
x <- read.delim("clipboard", header=TRUE)      # if no header, use FALSE
```

`read.csv()` automatically interprets nonnumeric data as a factor (categorical variable). if you want your non numerical variable interpreted as strings, not as factors, set the `as.is` parameter to TRUE (`as.is = TRUE`). This indicates that R should not interpret nonnumeric data as a factor.

Reading CSV files into the R environment in countries (including Turkey) using comma as the decimal separator (`read.csv2()`)

In some European and most Asian countries, a comma (8,52) is used as the decimal separator (instead of a period/dot like 8.52). Because of that, Excel in those countries actually uses a semicolon (;) instead of a comma (,) as the delimiter (between the Excel columns) in csv files.
If the standard `read.csv()` function is used to read those csv files, each row appears in a single column with cell contents separated by semicolons.
`read.csv2()` solves this problem and reads semicolons as the separator and reads those types of csv files correctly.
`read.csv2()` is, therefore, equivalent to `read.csv("filename.csv", sep = ";")` (See: https://stackoverflow.com/questions/35360847/r-import-csv-file-all-data-fall-into-one-the-first-column; https://hotware.wordpress.com/2009/12/16/trouble-with-opening-csv-files-with-excel-the-comma-and-semicolon-issue-in-excel-due-to-regional-settings-for-europe)

For further information (similar problem in a different setting well explained):
https://www.ibm.com/support/knowledgecenter/en/SSWU4L/Data/imc_Data/Data_q_a_watson_assistant/How_To_Remove_a_Semicolon_in_a_CSV_Fil197.html
See csv2.read.txt for details.

Reading a PDF file:
https://datascienceplus.com/extracting-tables-from-pdfs-in-r-using-the-tabulizer-package
https://stackoverflow.com/questions/44141160/recognize-pdf-table-using-r

Reading from a ZIP file on the computer:

```
data  <-  read.table(unz("zipfile.zip", "zipfile.dat"), nrows=10, header=T,
quote="\"", sep=",")

myfile  <-  read.csv(zip.file.extract("~/files/test.csv", "myzip.zip")) # Where the file
```

test.csv is actually located in the: ~/files/myzip.zip/test.csv

Read zipped file into R: https://www.r-bloggers.com/read-zipped-file-into-r OR
https://kariert.wordpress.com/2011/05/15/read-zipped-file-into-r

Reading from a ZIP file on the internet:
https://stackoverflow.com/questions/3053833/using-r-to-download-zipped-data-file-extract-and-import-data:
Zip archives are actually more a 'filesystem' with content metadata etc. See help(unzip) for details. To do what you sketch out above you need to:

Create a temp. file name (eg `tempfile()`)
Use `download.file()` to fetch the file into the temp file
Use `unz()` to extract the target file from temp. file
Remove the temp file via `unlink()`

which in code looks like:

```
temp <- tempfile()
download.file("https://www.zip.com/zippedfile.zip", temp)
data <- read.table(unz(temp, "filename.txt"))
unlink(temp)
```

Compressed (.z) or gzipped (.gz) or bzip2ed (.bz2) files can also be downloaded, unzipped and read by R (see this link).

Reading a table from the internet:

```
table <-
read.table("https://www.seeqtl.org/gbrowse2/data/eQTL_Qvalue_cutoff_hapmap3_trans_hg19.txt", header = T)
head(table)
```

	SNP	chr_SNP	bp_SNP	EntrezGeneID	GeneSymbol	Qvalue
1	rs10003238	chr4	47548775	10161	LPAR6	0.0108747976
2	rs10000012	chr4	1357324	1268	CNR1	0.0307436378
3	rs10003238	chr4	47548775	128954	GAB4	0.0007746899
4	rs1000380	chr5	56886188	147945	NLRP4	0.1011226683
5	rs10006566	chr4	170595633	148545	NBPF4	0.1523153909
6	rs10000241	chr4	78196638	169044	COL22A1	0.1912827348

For more on data import, see:

- **Data Import Cheat Sheet**: https://www.rstudio.com/wp-content/uploads/2018/08/data-import-600x464.png
- **DataCamp Tutorial**: https://www.datacamp.com/community/tutorials/r-data-import-tutorial
- **HoningDS Tutorial**: https://honingds.com/blog/import-data-into-r/
- **Data Manipulation in R- Online Course** (Data Novia): https://www.datanovia.com/en/coursecategory/data-manipulation

Graphing in R:

- R Tips (Graphs/Tables): https://www.zoology.ubc.ca/~schluter/R/display
- Producing Simple Graphs with R: https://www.harding.edu/fmccown/r
- Styling Base R Graphics: https://www.jumpingrivers.com/blog/styling-base-r-graphics
- Quick-R (Graphs): https://www.statmethods.net/graphs/index.html
- R Tutorials @ R-Chart: http://www.r-chart.com (emphasis on graphics)
- R Graphics Cookbook_BBC Visual and Data Journalism: https://bbc.github.io/rcookbook
- R Graph Gallery: https://www.r-graph-gallery.com/the-r-graph-gallery (cheatsheets)
- R Graph Gallery (Art from Data): https://www.r-graph-gallery.com/portfolio/data-art
- R Graph Catalog (with scripts): http://shinyapps.stat.ubc.ca/r-graph-catalog (see here)
- R Graphics for Clinical Trials @ Gersonides (with scripts): http://www.gersonides.com/r
- Data Visualization in R (complete course with scripts): http://www.datavis.ca/courses/RGraphics (lecture presentations: Lecture 1; Lecture 2; Lecture 3; Lecture 4)
- DataCamp project: Phyllotaxis: Draw Flowers Using Mathematics (free)
- DataCamp project: Visualizing Inequalities in Life Expectancy (paid)
- Statistical Graphics (StatEducation):
- Statistical charts: https://www.r-graph-gallery.com/author/holtz
- 10 tips for making your R graphics look their best: https://blog.revolutionanalytics.com/2009/01/10-tips-for-making-your-r-graphics-look-their-best.html
- Playing Around with Phyllotactic Spirals: https://chichacha.netlify.com/2019/01/29/playing-around-with-phyllotactic-spirals
- R for Beginners: Some simple code to produce informative graphs, part I: https://dmwiig.net/2017/01/11/r-for-beginners-some-simple-code-to-produce-informative-graphs-part-one
- R for Beginners: Basic graphics code to produce informative graphs, part II (working with big data): https://dmwiig.net/2017/01/16/r-for-beginners-basic-graphics-code-to-produce-informative-graphs-part-two-working-with-big-data
- FlowingData Visualisation by R Tutorials: http://flowingdata.com/category/tutorials
- FlowingData Visualisation by R Courses: https://flowingdata.com/courses
- Get Started. Make Your Own Plot and More: Learn how to plot a graph in R
- Graphical Parameters (Quick-R): https://www.statmethods.net/advgraphs/parameters.html

R can produce plots in a standard format and allows controlling of the plot appearance. Three main packages for data representation are:

- **graphics**: This is already available in the base R and is loaded by default (SEE: Styling Base R Graphics)
- **lattice**: This is already available with the basic installation of R, but needs to be loaded using the `library()` function.
- **ggplot2**: This is not included in the basic installation of R, but can be installed using the function: `install.packages("ggplot2")`.

- **ggplot2** Reference (functions): https://ggplot2.tidyverse.org/reference
- **ggplot2** Cheat Sheet: http://zevross.com/blog/2014/08/04/beautiful-plotting-in-r-a-ggplot2-cheatsheet-3
- R-Statistics: ggplot short tutorial: http://r-statistics.co/ggplot2-Tutorial-With-R.html
- R-Statistics- ggplot2 quickref: http://r-statistics.co/ggplot2-cheatsheet.html
- STHDA: **ggplot2** essentials: http://www.sthda.com/english/wiki/ggplot2-essentials
- STHDA: Histograms with **ggplot2** (tutorial): http://www.sthda.com/english/wiki/ggplot2-histogram-easy-histogram-graph-with-ggplot2-r-package
- STHDA: Dot plots with **ggplot2** (tutorial): http://www.sthda.com/english/wiki/ggplot2-dot-plot-quick-start-guide-r-software-and-data-visualization
- STHDA: **ggplot2** scatter plots: Quick start guide - R software and data visualization: http://www.sthda.com/english/wiki/print.php?id=188
- sape **ggplot2** Quick Reference: http://sape.inf.usi.ch/quick-reference/ggplot2
- Graphics tutorials by Michael Toth (https://michaeltoth.me):

    - Detailed Guide to the Bar Chart in R with **ggplot**: https://michaeltoth.me/detailed-guide-to-the-bar-chart-in-r-with-ggplot.html
    - A Detailed Guide to Plotting Line Graphs in R using **ggplot** geom_line: https://michaeltoth.me/a-detailed-guide-to-plotting-line-graphs-in-r-using-ggplot-geom_line.html
    - A Detailed Guide to the **ggplot** Scatter Plot in R: https://michaeltoth.me/a-detailed-guide-to-the-ggplot-scatter-plot-in-r.html
    - How to Create a Bar Chart Race in R (Animated Bar Chart) - Mapping United States City Population 1790-2010: https://michaeltoth.me/how-to-create-a-bar-chart-race-in-r-mapping-united-states-city-population-1790-2010.html
    - A Detailed Guide to **ggplot** colors: https://michaeltoth.me/a-detailed-guide-to-ggplot-colors.html
    - One Step to Quickly Improve the Readability and Visual Appeal of **ggplot** Graphs: https://michaeltoth.me/one-step-to-quickly-improve-the-readability-and-visual-appeal-of-ggplot-graphs.html
- How to combine Multiple **ggplot** Plots to make Publication-ready Plots: https://datascienceplus.com/how-to-combine-multiple-ggplot-plots-to-make-publication-ready-plots
- Data Visualization with R (live book by Rob Kabakoff): https://rkabacoff.github.io/datavis
- R-Statistics: Top 50 **ggplot2** Visualizations - The Master List (with Full R Code): http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html
- Examples of ggplot()
  https://www.neonscience.org/dc-time-series-plot-ggplot-r
  http://t-redactyl.io/blog/2016/02/creating-plots-in-r-using-ggplot2-part-6-weighted-scatterplots.html (bubble plot)
  https://stackoverflow.com/questions/22598517/how-to-have-multple-labels-in-ggplot2-for-bubble-plot (multiple labels)

R Color Chart: R has 657 built-in color names. The function `colors()` will show all of them. All these color names can be used in plot parameters like `col = "…"`. The function `col2rgb()` can convert all these colors into RGB numbers (try, for example `col2rgb("orchid1")`).

PCH Symbols Chart lists PCH symbols used in R plot. When the PCH is 21-25, the parameter `col = "…"` and `bg = "…"` should be specified. PCH can also be in characters, such as "#", "%", "A", "a", and the character will be plotted.

See also R Graph Gallery cheat sheets for charts related to graphs.

HSV (HSV Color Specification): https://stat.ethz.ch/R-manual/R-devel/library/grDevices/html/hsv.html

R Color Styles: http://bioinfo.umassmed.edu/bootstrappers/bootstrappers-courses/pastCourses/rCourse_2016-04/Additional_Resources/Rcolorstyle.html

Command options for base R plots
Many (but not all -- try them) of the basic plotting commands in base R will accept the same options to control axis limits, labeling, print a title, change the plotting symbol, change the size of the plotting symbols and text, and change the line types. Here are some of the most frequently modified options. Use them inside the parentheses of a plotting command to have their effect. If you are not sure whether a given option works in your case, try it. The worst that could happen is you get an error message, or R ignores you.

```
main = "Eureka"     # add a title above the graph
pch = 16            # set plot symbol to a filled circle
color = "red"       # set the item colour
xlim = c(-10,10)    # set limits of the x-axis (horizontal axis)
ylim = c(0,100)     # set limits of the y-axis (vertical axis)
lty = 2             # set line type to dashed
las = 2             # rotate axis labels to be perpendicular to axis
cex = 1.5           # magnify the plotting symbols 1.5-fold
```

```
cex.lab = 1.5      # magnify the axis labels 1.5-fold
cex.axis = 1.3     # magnify the axis annotation 1.3-fold
xlab = "Body size" # label for the x-axis
ylab = "Frequency" # label for the y-axis
```

Scripts:

```
# To generate a scatterplot:
      a <- rnorm(50) # rnorm(50) generates 50 random numbers in standard normal
              distribution
      b <- rnorm(50) # Another 50 random numbers into vector b
      plot(a, b)     # This plot should have quite a scatter centered around the center
                     # Try to generate a plot with rnorm(10000) to see a centralised
              scatter

# To change the shape of the dot, TRY "pch" argument (pch: plotting character):
      plot(rnorm(1000), pch=21)  # TRY also pch=22, pch=23, pch=24 etc (between 0 and 25)
      OR pch="*", pch="#"
# To read from a table and plot data in two columns
# File name: table.txt; column headers: Body_mass and Brain_mass
      dataset <- read.table("table.txt")   # table.txt is in the working directory
                                           # reads the table as a dataframe called "dataset"
      plot(dataset$Body_mass, dataset$Brain_mass)
                                    #  OR: first use the attach function: attach(dataset)
                                    #  then: plot(Body_mass, Brain_mass)

# To save a graph as an image file format such as PNG, we can use the png() command:
      png("scatterplot.png")
      plot(rnorm(1000))
      dev.off()
# The preceding command will save the graph as scatterplot.png in the current working directory. Similarly, if we
wish to save the graph as JPEG, BMP or TIFF we can use the jpeg(), bmp(), or tiff() commands
respectively.
# SEE: Cookbook for R - Output to a File: http://www.cookbook-r.com/Graphs/Output_to_a_file

# You can specify a number of arguments to adjust the graph as per your needs. The simplest one that we've
already used is the filename. You can also adjust the height and width settings of the graph:
      png("scatterplot.png",
      height=600,
      width=600)
# The default units for height and width are pixels but you can also specify the units in inches, cm or mm:
      png("scatterplot.png", height=4, width=4, units="in")
# The resolution of the saved image can be specified in dots per inch (dpi) using the res argument:
      png("scatterplot.png", res=600)

# To set the plot background color to gray, use the bg argument in the par() command (not with ggplot):
      par(bg="gray")       # TRY: yellow, blue, pink, orchid2; OR: 2, 3, 4 (but not 1!)
      plot(a,b)            # if a and b are vectors of equal lengths, OR plot(rnorm(5000))
# SEE: http://127.0.0.1:19842/library/graphics/html/par.html about par function

# Setting colours for text elements: axis annotations, labels, plot titles, and legends:
# If you want to make the axis value annotations black, the labels of the axes gray, and the plot title dark blue, you
should do the following:
      plot(rnorm(100),
      main = "Plot Title",
      col.axis = "blue",
      col.lab = "red",
      col.main = "darkblue")

# Full command:
```

```
a <- rnorm(50)
b <- rnorm(50)
plot(a, b, pch = 16, col = "orchid1", xlim = c(-3, 3), ylim = c(-3, 3), main
= "MY PLOT", xlab = "X VARIABLE" , ylab = "Y VARIABLE" , col.axis="blue",
col.lab="red", col.main="darkblue")

# Boxplot:
boxplot(iris$Sepal.Width)                    # for the whole dataframe
boxplot(iris$Sepal.Width~iris$Species)       # Three boxplots for each of the three species

boxplot(rnorm(1000), boxwex=0.2, las = 1)  # TRY various values (<1) for boxwex, and 0
                                             and 1 for las (labels are parallel (=0) or perpendicular(=1) to axis)

# Boxplot without outliers:
# TRY the following two boxplot commands:
boxplot(rnorm(100000))
windows()
boxplot(rnorm(100000), outline=FALSE)

# Histogram by a grouping factor using lattice included in Base R:
data(infert)
library(lattice)
histogram(~ parity | case, data = infert)        # parity is a count variable, and case is
                                                   the case-control status variable
histogram(~ spontaneous | case, data = infert)   # spontaneous is a count variable, and
                                                   case is the case-control status variable

# Scatter plot by a grouping factor (in different colours):
library("ggplot2")
data(iris)
qplot(Petal.Width, Petal.Length, color = Species, data = iris)

# Multiple histograms in the same panel:
library("lattice")
data(infert)
histogram( ~ parity + spontaneous + induced, data = infert)

# For a full list of graphical parameters, see (1):
https://www.statmethods.net/advgraphs/parameters.html
# For a full list of graphical parameters, see (2): https://www.statmethods.net/advgraphs/axes.html

# Pie chart using Base R:
v <- c(88, 76, 52, 26, 12)
pie(v)
pie(v, main = "Number of e-mails received", col = rainbow(length(v)),
labels=c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday"))

# Barplot example (with labels and colours):
v <- c(88, 76, 52, 26, 12)
barplot(v)
barplot(v, main = "Number of e-mails received", xlab = "Days", ylab =
"Count", col = rainbow(length(v)), names.arg = c("Monday", "Tuesday",
"Wednesday", "Thursday", "Friday"), border = "blue")

# 3D piechart:
install.packages("plotrix")   # installs the package (called "plotrix" in this case)
library(plotrix)              # calls the package (called "plotrix")
v <- c(2,6,4,9,10,1,12,3,5)   # generates a vector called "v"
pie3D(v)                      # generates a 3D piechart of "v" (D = capital letter!)

# Creating Heat Maps and Contour Plots (Chapter 8 of R Graph Cookbook_PACKTPUB.pdf; p.181)
# Finalizing graphs for publications and presentations (Chapter 10 of R Graph Cookbook_PACKTPUB.pdf; p.223)
```

SEE ALSO: R Graph Essentials_PACKTPUB.pdf

Cookbook for R - Graphs: http://www.cookbook-r.com/Graphs

R for Statistical Analysis:

- Begin with:
  Instant R - An Introduction to R for Statistical Analysis: http://www.instantr.com (& the book: Using R for Statistics_2014.pdf)
- Statistical Analysis with R- Beginner's Guide_PACKT 2010 (and http://rtutorialseries.blogspot.com)
- Instant R Starter_PACKT 2013.pdf
- Quick-R: Statistics (https://www.statmethods.net/stats/index.html (& the book: R in Action_Manning)
- Cookbook for R - Statistical analysis: http://www.cookbook-r.com/Statistical_analysis
- r4stats: statistics: http://r4stats.com/examples/statistics
- R Tutorials @ R Tutorial: http://www.r-tutor.com (Statistics: http://www.r-tutor.com/elementary-statistics)
- R Tutorials @ R-Statistics: http://r-statistics.co/R-Tutorial.html
- Tutorials on Advanced Stats and Machine Learning with R: http://r-statistics.co
- Introduction to Data Science_Jeffrey Stanton_2013.PDF (link)
- Learn to Use R_v3_Computer World.pdf
- Statistics with R- Computing and Graphics_Kjell Konis.pdf (includes a practice run) (link)
- simpleR – Using R for Introductory Statistics.PDF (including a sample R session / Teaching with R; link)
- Introduction to R (StatEducation): http://statseducation.com/Introduction-to-R
- Using R for Introductory Statistics_Verzani_2005 (link)
- VIB Course on Basic Statistics in R with downloadable training material including slides
- Using R for statistical analyses (Garderner's Own):
  http://www.gardenersown.co.uk/Education/Lectures/R/basics.htm
- A Handbook of Statistical Analyses Using R (online book) by Everitt & Hothorn.pdf
- Learning statistics with R (online book): https://learningstatisticswithr.com/book (PDF:
  https://learningstatisticswithr.com/lsr-0.6.pdf)
- Matthias Kohl: Introduction to Statistical Data Analysis with R, Bookboon (2015):
  https://bookboon.com/en/introduction-to-statistical-data-analysis-with-r-ebook
- Analyzing Data in R- From A to Z : http://www.deeplytrivial.com/p/the-to-z-of-r.html

EDx Course- Introduction to Applied Biostatistics: Statistics for Medical Research: Learn data analysis for medical research with practical hands-on examples using R Commander
https://www.edx.org/course/introduction-applied-biostatistics-osakaux-med101x-0

Get Started: Make Your Own Plot and More!
Check out these examples of basic statistics

R packages for undergraduate statistics education @ Citizen-Statistician:
http://citizen-statistician.org/2015/08/09/r-packages-for-undergraduate-stat-ed

OpenIntro Statistics Book (for UG education with R examples):
https://www.openintro.org/stat/textbook.php (see package "openintro")

Statistics with R (online book):
http://zoonek2.free.fr/UNIX/48_R/all.html

Introduction to Statistical Thinking (With R, Without Calculus) by Yakir (2011):
http://pluto.huji.ac.il/~msby/StatThink/index.html

An R Companion for the Handbook of Biological Statistics:
http://rcompanion.org/rcompanion

Bitesize: Using R for Statistical Tests
https://bitesizebio.com/21422/r-you-ready-using-r-for-statistical-tests

An Introduction to Statistical and Data Sciences via R (Modern Dive e-Book):
http://www.moderndive.com/index.html
(see also: http://blog.revolutionanalytics.com/2017/02/moderndive.html)

Statistic on aiR:
http://statistic-on-air.blogspot.com

Linear Regression from Scratch in R:
https://datascienceplus.com/linear-regression-from-scratch-in-r

Survival analysis with R:
https://rviews.rstudio.com/2017/09/25/survival-analysis-with-r

Practical Data Science for Stats:
http://blog.revolutionanalytics.com/2017/09/practical-data-science-for-stats.html >>>
https://peerj.com/collections/50-practicaldatascistats

The Practical R- Primer on Statistical Programming in R:
https://thepracticalr.wordpress.com

Online R Courses:
Udemy: https://www.udemy.com/courses/academics/math-and-science
https://www.udemy.com/data-analysis-with-r
Udemy Basic R Course (incl ggplot & Free): https://www.udemy.com/machlearn1
EDx, Coursera, Stanford Lagunita, FutureLearn and other MOOC providers also offer various R courses
DataCamp and DataQuest are other sources of online R courses and tutorials
Statistics.com: http://www.statistics.com/course-catalog (R-bloggers link)

Stata to R:
A Quick and Easy Way To Turn Your Stata Knowledge Into R Knowledge:
http://dlab.berkeley.edu/blog/quick-and-easy-way-turn-your-stata-knowledge-r-knowledge

Statistics 506: Computational methods and tools in statistics
http://dept.stat.lsa.umich.edu/~kshedden/Courses/Stat506
Stata intro: http://dept.stat.lsa.umich.edu/~kshedden/Courses/Stat506/stata_intro
R intro: http://dept.stat.lsa.umich.edu/~kshedden/Courses/Stat506/r_intro
R tips and common errors: http://dept.stat.lsa.umich.edu/~kshedden/Courses/Stat506/r_common_errors

R Statistics & Programming:
> R for Beginners: Basic R code for common statistical procedures - part I : https://dmwiig.net/2016/12/18/r-for-beginners-basic-r-code-for-common-statistical-procedures-part-i
> R for Beginners: Basic R code for common statistical procedures - part II: https://dmwiig.net/2016/12/27/r-for-beginners-some-simple-r-code-to-do-common-statistical-procedures-part-two

Getting Started in R~Stata- Notes on Exploring Data
http://dss.princeton.edu/training/RStata.pdf (saved as Getting Started in R~Stata- Notes on Exploring Data.PDF)

* * * * * * * * *

Merge the two tables based on common ID field (in R):
FROM: http://manuals.bioinformatics.ucr.edu/home/R_BioCondManual

```
      # search " merge(frame1, frame2, by.x = "frame1col_name", by.y = "frame2col_name",
all = T)"
```

```
# master <- merge(master, target, by.x="id", by.y="id", all.x=TRUE)
# x is the first dataframe ("master") and y is the dataframe being merged
("target")
# To obtain only the common rows, change 'all = TRUE' to 'all = FALSE'
# Example (merging two SNP genotype files to the master 1KG_HLA file):
frame1 <- read.csv(file.choose())    # select the master file (1KG HLA.csv)
frame2 <- read.csv(file.choose())    # select SNP1 genotype file
frame3 <- read.csv(file.choose())    # select SNP2 genotype file
master <- merge(frame1, frame2, by.x = "id", by.y = "id", all = TRUE)
master1kg <- merge(master, frame3, by.x = "id", by.y = "id", all = TRUE)
head (master1kg, 1)  # to confirm that the merging is successful
write.csv(master1kg, "master.csv")  # exports the merged file as a CSV file to
working directory
```

Making a WordCloud in R:
> R-Bloggers: https://www.r-bloggers.com/word-cloud-in-r
> Script (with annotations): http://www.stat.cmu.edu/~hseltman/files/makeWordCloud.R (needs editing)
> http://www.sthda.com/english/wiki/text-mining-and-word-cloud-fundamentals-in-r-5-simple-steps-you-should-know
> http://www.sthda.com/english/wiki/word-cloud-generator-in-r-one-killer-function-to-do-everything-you-need
> R Graph Gallery: https://www.r-graph-gallery.com/wordcloud
> https://www.analyticsvidhya.com/blog/2014/05/build-word-cloud-text-mining-tools
> Changing fonts: https://stackoverflow.com/questions/43460652/change-font-in-wordcloud-package-r
> package("wordcloud"): https://github.com/manjushajoshi/R-code/blob/master/DiggingdatawithR-v2/DiggingdatawithR-v2.R (see also: https://thenewstack.io/data-visualization-basics-r-programming-language), wordcloud function (options): https://www.rdocumentation.org/packages/wordcloud/versions/2.6/topics/wordcloud

10 Favorite R Packages and the Cool Things You Can Do with Them (Bitesize):
http://bitesizebio.com/23003/my-10-favorite-r-packages-and-the-cool-things-you-can-do-with-them

The Undergraduate Guide to R: https://sites.google.com/site/undergraduateguidetor/manual-files

Fun Data for Teaching R: https://bartomeuslab.com/2016/01/21/fun-data-for-teaching-r

Two minutes video tutorials: http://www.twotorials.com

Top R language resources to improve your data skills:
https://www.computerworld.com/article/2497464/top-r-language-resources-to-improve-your-data-skills.html

****

Useful R Functions: A random selection

If you are going to download lots of packages, start with:
```
chooseCRANmirror()          # OR select "Set CRAN Mirror..." in the Packages menu
```
and select the closest mirror to you for all downloads in the current session

Object:
In R, every data structure is an object which belongs to a class. To find out the class of an object (x):
```
data.class(x)
```
Examples:
```
data.class(iris)
> [1] "data.frame"
data.class(iris$Sepal.Length)
> [1] "numeric"
data.class(iris$Species)
> [1] "factor"
```

Creating a data frame:

```
df <- data.frame()
fix(df)
```

Editing a dataframe:

```
data(iris)
edit(iris)  # Opens a spreadsheet for editing
  #OR:
fix(iris)
```

The difference between edit() and fix() is pretty trivial:
edit() lets you edit an object and returns the new version
fix() lets you edit an object and modifies the original

Converting a horizontal dataframe to vertical (long) format (converting multiple columns in a dataset to a single column):

Dataframe `iris` is in long format. In vertical format, there would be columns for each of the three species and their data would be in those columns rather than having a species variable and having all the data for all species in single columns.
Package: reshape2
Function: melt()
See: https://www.statmethods.net/management/reshape.html
https://www.r-bloggers.com/melt
https://seananderson.ca/2013/10/19/reshape

Copying an Excel file to R via the clipboard:

Copy the spreadsheet on clipboard, and import it to R using the following function:

```
x <- read.delim("clipboard", header=TRUE)     # if no header, use FALSE
```

Inserting multiple plots in one panel (not in ggplot):

```
par(mfrow = c(3,1))
boxplot(iris$Sepal.Length); boxplot(iris$Sepal.Width); boxplot(iris$Petal.Width)
```

  #OR

```
par(mfrow = c(2,2))
boxplot(iris$Sepal.Length); boxplot(iris$Sepal.Width);
boxplot(iris$Petal.Length); boxplot(iris$Petal.Width)
```

Having two plots simultaneously on two different graphics windows:

FROM: https://stackoverflow.com/questions/20135744/how-to-open-new-plot-window-during-plot

```
library("DataExplorer")
data(iris)
plot_histogram(iris)
windows()              # opens a new graphics window while keeping the first open open
plot_density(iris)
```

  #OR in one line:

```
plot_histogram(iris); windows(); plot_density(iris)
```

Making R to pause between plots:

Each plot created in R overwrites the previous one. You can set R to pause between plots so you can view each one before it's overwritten.
Use the global graphics option called "ask" is set to TRUE, R will pause before each new plot:

```
par(ask = TRUE)
```

When you are tired of R pausing between plots, set it to FALSE:

```
par(ask = FALSE)
```
An example is provided in the demo_ggplot2.R script.

Loading an RData file:
```
load("filename.RData")
```
To save a dataframe as RData file:
```
save(dataframename, file = "filename.RData")
```

Creating a variable and printing it (1: the usual way):
```
x <- (3 * 4)
x
# [1] 12
```

Creating a variable and printing it (2: the quick way):
```
(x <- (3 * 4))  # wrapped by brackets
# [1] 12        # printing is automatic
```

file.choose() to select a file from anywhere on the computer:
```
   x <- read.csv(file.choose(), header=TRUE)    # if no header, use FALSE
   dim(x)  # to check the dimensions of the imported CSV file
```

Demos
Type 'demo()' for some demos {like: >demo(glm.vr)}, 'help()' for on-line help, 'help.start()' for an HTML browser interface to help, help.search() for online search {like: help.search("glm")}. Type 'q()' to quit R.

How to Cite:
```
citation()                # For R
citation("DescTools")     # For a specific package
```

Check: R Functions of the Day: https://rfunction.com for more tricks

Generating Factor Levels with gl() function. SEE: https://www.tutorialspoint.com/r/r_factors.htm

Dealing with dataframes:
```
data()                                  # To find out which built-in dataframes come with base R
data(women)                             # Calls the built-in dataframe "women"
write.csv(women, "test.csv")            # Exports the dataframe "women" as a CSV file (test.csv) to the
                                          default working directory
csv <- read.csv("test.csv")             # Imports the dataframe "women" as a CSV file (test.csv) from
                                          the default working directory
csv <- read.csv(file.choose())          # If unsure of the file location, you can browse your computer
csv <- read.csv("c:\temp\women.csv")    # You can also type the full location (in this case, the file
                                          is in c:/temp folder)
```

Working directory:
```
getwd()         # To find out the working directory where exported files will be saved and files to be
                  imported will be looked for by default
setwd()         # To set the working director like setwd("C:\TMP") for the duration of the session
```

List of objects / Removal of objects:
```
ls()            # list current objects
rm(object)      # delete an object
rm(list=ls())   # delete all objects
```

Clearing the console or specific variables:
```
rm(list=ls(all=TRUE))          # To clear R console (everything generated in the current session will
```

be removed)

```
rm(variablename)                    # To clear a specific variable
```

(Just to clear the screen without removing anything g from the memory, press CTRL + L)

Generating numerical vectors:

```
seq(from=1, to=3, by=0.1)           # To have a series of numbers starting from 1 to 3 and increasing by
                                      0.1. This can also be written as: seq(1, 3, 0.1)
runif(4)                            # To generate four random numbers within the default range of 0 to 1
runif(10, min=1, max=100)           # To generate 10 random numbers within the range 1 to 100
                                    # For "runif",
                                      see: http://www.cookbook-r.com/Numbers/Generating_random_numbers
rnorm(n, mean = 0, sd = 1)          # To generate "n" number of random numbers fitting to normal
                                      distribution with default mean=0 and sd=1
rnorm(25, 30, 5)                    # To generate a sample of size 25 from a normal distribution with mean
                                      30 and standard deviation 5
a  <-  rbinom(100, 10, 0.5)         # 100 people tossing a fair coin (third argument = 0.5) 100 times
mean(a)                             #  should be close to 10*0.5 = 5
c(1,2,3,4)                          # to create a vector of these numbers
x <- sample(1:10, 50, replace = TRUE)
                                    # creates a vector of 50 elements drawn from 1 to 10 with replacement
      x
        [1]  2  1  6  9  1  3  5  2 10  2  7  8  8  4  9  2  2  6  9 10  5  6  2  5  7
       [26]  9  1  6 10  1 10  5  1  7  5  2  7  9  2  6  5  1  4  9  2  4  2  4  8  3

y <- rep(1, 100)                    # creates a vector by repeating the element '1' one hundred times
```

Generating a numerical matrix:

```
M1 = matrix( c(2, 4, 3, 1, 5, 7), nrow=3, ncol=2)  # to create a matrix of these
six numbers as three rows and two columns
   or
M2 = matrix( nrow=2, ncol=2, data=c(1,3,5,9), byrow=T )

length(x)  # to get the length of vector x
```

 # To create a simulated dataframe:

```
c1 <- rnorm(100, 1, 3)
c2 <- rnorm(100, 2, 5)
c3 <- rnorm(100, 3, 6)
 a <- data.frame(c1, c2, c3)
dim(a)
>100 3
```

 # To choose a subset of a dataframe based on a value in a column, try this:

```
data(iris)
petal0.4 <- iris[iris$Petal.Width == 0.4, ]
petal0.4
```

it will print the data only for Petal.Width = 0.4 (this subset is now a new object named petal0.4) .

 # OR:

```
data(iris)
iris_subset2 <- subset(iris, Species = "virginica")
iris_subset2
```

it will print the subset of iris for Species = virginica only.

 # OR:

```
iris_subset3 <- iris[iris$Species == "virginica", ]
iris_subset3
```
it will print the subset of iris for Species = virginica only.

We can extract specific rows of that column using square brackets, [] :
```
subset_Puromycin <- Puromycin[10:15, 2:3]

iris_subset <- subset(iris, select = c(Sepal.Length, Petal.Length))
head(iris_subset)
```

 # To find out the dimensions, row and column numbers of a dataframe:
```
dim(dataframename)
nrow(dataframename)
ncol(dataframename)
```

 # attach() and detach()
```
# attach(dataframename)   # The dataframe is now the default dataset for any manipulation and need not be defined by name in following procedures
# detach(dataframename)   # Cancels the attach() and removes the dataframe from the search path
```

 # apply() will apply the same command (like mean()) to all rows (second argument = 1) or columns (second argument = 2) [mean() function only works for numeric data (rows or columns)
```
apply(iris[-5], 1, mean)   # will give means of each row
apply(iris[-5], 2, mean)   # will give means of each column
```

A t-test demonstration:
```
a <- rnorm(25, 30, 5)              # To generate a sample of size 25 from a normal distribution with mean 30 and standard deviation 5
b <- rnorm(25, 25, 7)              # To generate a sample of size 25 from a normal distribution with mean 25 and standard deviation 7
t.test(a,b)                        # To run the t-test
```
Change the sample size (first argument) in a and b to show the statistical power concept
Change the SD (third argument) to show the effect of intra-group variability
Change the mean (second argument) to show the effect of difference in the mean

 # Boxplot by categories in a table:
```
data <- Puromycin
head(Puromycin)
boxplot(data$conc ~ data$state)     # state variable should be a factor
    # OR:
by(data$conc, data$state, boxplot) # this will generate two boxplots for each level of the factor variable "state) and only the last one will be visible
par(mfrow = c(1,2))                # changes the settings for two plots to appears in two columns
by(data$conc, data$state, boxplot) # the two boxplots will be side-by-side

boxplot((ChickWeight[ChickWeight$Time == 21, 1]) ~
(ChickWeight[ChickWeight$Diet]))
boxplot(ChickWeight$weight ~ ChickWeight$Diet,  xlab = "Weight", xlab = "Diet")
```

OPTIONS for boxplot: pch = 24, col = "red", main = "MY PLOT", col.axis="blue", col.lab="red", col.main="darkblue", boxwex = 0.2

 # Adding a line to the boxplot:
```
data(iris)
boxplot(iris[-5])
```

```
abline(h=2.5, col = "blue")
```

 # Correlation:

```
data(women)
data <- women
cor(data$height, data$weight)                    # Provides Pearson's correlation coefficient (r) by default
cor(data$height, data$weight, method = "pearson")
cor(data$height, data$weight, method = "kendall")     # Provides Kendall's tau value when specified
cor(data$height, data$weight, method = "spearman")    # Provides Spearman's rho value when specified
cor.test(data$height, data$weight)               # Provides Person's correlation coefficient (r), its 95% CI and the P value
cor(data)                                        # Generates a matrix of all pairwise correlations in the dataset/dataframe for Pearson test by default
cor(data, method = "spearman")                   # Generates a matrix of all pairwise correlations in the dataset for Spearman test
pairs(data)                                      # Generates plots of pairwise correlations in the whole dataset
```

 # Correlation scatter plot by a factor (in different colours):

```
library("ggplot2")
data(iris)
qplot(Petal.Width, Petal.Length, color = Species, data = iris)
```

 # Correlation matrix with heatmap:

```
library(DataExplorer)
plot_correlation(iris)
```

 # Correlation matrix with colored ellipses (http://www.cookbook-r.com/Graphs/Correlation_matrix)

```
install.packages("ellipse")
library("ellipse")
data(iris)
ctable <- cor(iris[-5])    # if desired, specify the decimal numbers by  round(ctable, 2)
plotcorr(ctable)           # no colors
# Adding color:
colorfun <- colorRamp(c("#CC0000","white","#3366CC"), space="Lab")   # Defining the color palette
plotcorr(ctab, col=rgb(colorfun((ctab+1)/2), maxColorValue=255), mar = c(0.1, 0.1, 0.1, 0.1))      # Same graph with color and reduced margins
```

 # J-shaped curve = 2*(x**2) - 3*x + 4

```
x=seq(1:1000)
y = 2*((seq(1:1000))**2) - 3*(seq(1:1000)) + 4
plot(x,y)
```

 # Simulating a dataframe:

```
set.seed(955)
vvar <- 1:20 + rnorm(20,sd=3)
wvar <- 1:20 + rnorm(20,sd=5)
xvar <- 20:1 + rnorm(20,sd=3)
yvar <- (1:20)/2 + rnorm(20, sd=10)
zvar <- rnorm(20, sd=6)
data <- data.frame(vvar, wvar, xvar, yvar, zvar)
head(data)
#>        vvar       wvar     xvar       yvar      zvar
```

```
#> 1 -4.252354  5.1219288 16.02193 -15.156368 -4.086904
#> 2  1.702318 -1.3234340 15.83817 -24.063902  3.468423
#> 3  4.323054 -2.1570874 19.85517   2.306770 -3.044931
#> 4  1.780628  0.7880138 17.65079   2.564663  1.449081
#> 5 11.537348 -1.3075994 10.93386   9.600835  2.761963
#> 6  6.672130  2.0135190 15.24350  -3.465695  5.749642
```