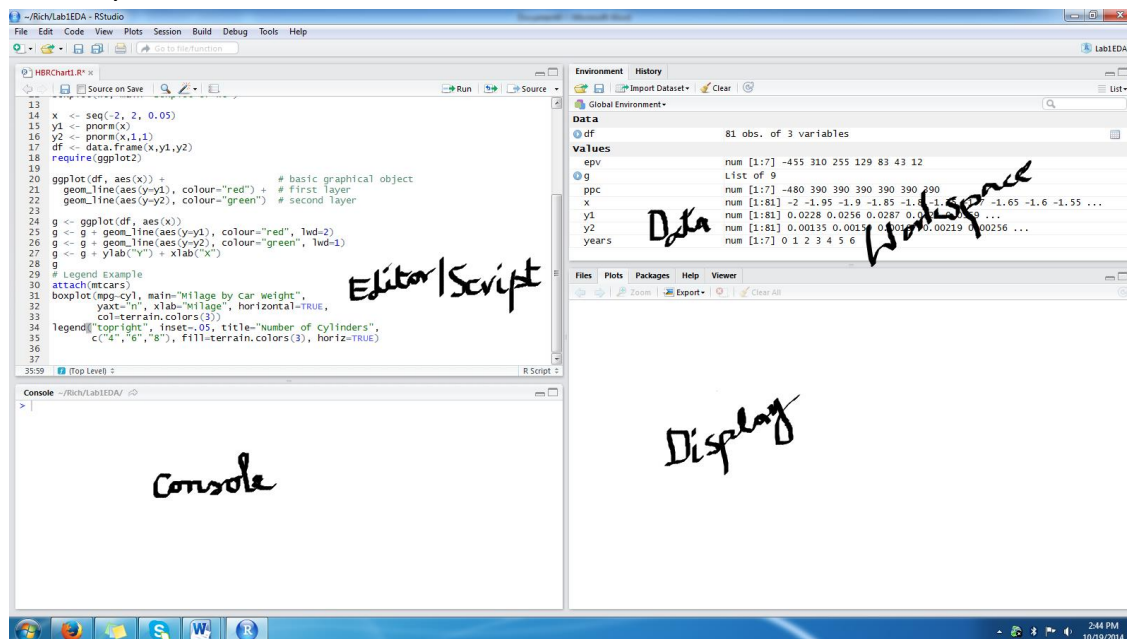


**Pre-requisites:** R and R Studio installed. Preferred versions: R version 3.1.1 or above and R Studio 0.98.1062 or above; any OS windows, Linux or Mac OS.

1. Install R. R is a command line suite of tools that powers R Studio. Download the appropriate version (Windows, Mac or Linux) at: <http://cran.rstudio.com/>
2. For Linux and Windows, you only need the R base. For Mac, download the appropriate .pkg file for your OS, and install it.
3. Once R is installed, navigate to <http://www.rstudio.com/products/rstudio/download/> to download R studio. Pick the appropriate installer for R Studio for your platform; download, and install it.
4. Start RStudio. This will depend on your operating system. It should be available in your Windows Start Menu, your OS X Spotlight, or on your Linux command line. When you start RStudio, you will see a window that looks like shown below but without script window:
5. Start R Studio by clicking on it or by invoking it from the Start menu on Windows.
6. Study the various parts of the R Studio. The editor, the console, the data and the display areas. Also look at the top line menu items. We will explain and learn the menu items when we analyze the data and as the need arises.



7. We will organize our work in separate project folders. File→New Project→New Directory→Empty project→ Lab1EDA
8. You will see only the console window on the left. Add the Script window using the leftmost (+) topline menu icon.

9. You can enter the commands on the console window for executing them one at a time or enter multiple lines in the editor/script window and execute them in “selected regions” of one or more commands entered in the script window. This also allows you transfer R-code from script to console window for execution.

### I. Basic R Commands

1.) Now that you have R and R studio installed, we will go through a list of several basic commands you will find useful.

2.) Variables are assigned using the “varname <- value” syntax, as follows:

```
> foo <- 2
> bar <- 4
> foo + bar
[1] 6
> result <- foo + bar
> result
[1] 6
```

Notice how when you declare these variables, the “Environment” section becomes populated.

3.) You can also create vectors. Vectors (or lists) are created using the “combine” function, called c(). Note that the indices start at 1, not 0:

```
> list <- c(2,4,6,8)
> list[2]
[1] 4
> list[1]
[1] 2
> list[0]
numeric(0)
> list[5]
[1] NA
> |
```

4.) R also supports basic arithmetic and Boolean expressions:

|          |              |
|----------|--------------|
| > 10 / 2 | > 2 == 0     |
| [1] 5    | [1] FALSE    |
| > 10 ^ 2 | > (2+2) == 4 |
| [1] 100  | [1] TRUE     |
| > 4 * 5  | > T == TRUE  |
| [1] 20   | [1] TRUE     |
| > 1 + 6  | > F && T     |
| [1] 7    | [1] FALSE    |
|          | > F    TRUE  |
|          | [1] TRUE     |

- 5.) We can combine all of the above functionality. Try creating a vector called "vect" with several numbers. Multiply the vector by 2 (i.e. vect \* 2) and press Enter. Take a look at the result.
- 6.) In addition to performing operations on each element of a vector like we did before, we can call functions on them. Try calling sqrt(vect) and observe the result.
- 7.) We can also give names to the elements of a vector like so:

```
> vect = c(2,4,6,8)
> names(vect) = c("1st", "2nd", "3rd", "4th")
> vect
1st 2nd 3rd 4th
  2   4   6   8
> vect["2nd"] <- 20
> vect
1st 2nd 3rd 4th
  2  20   6   8
```

You can also use control statements such as if.. else, and for loop; print for debugging, etc.

10. Let us examine R's graphic capability: go to the console window and type

**demo(graphics)**

and observe the visualization possible with R graphics.

11. **Basic plots:** Line graphs, histograms, box plots: **Save the images generated as pictures(.png)**

**Problem 1:** Define two synthetic vectors of data representing sales over 12 months for 2 items. Compare the two using **lines graphs**. (Discussion; Run the code multiple time and see randomness of the second set of data..)

```
sales1<-c(12,14,16,29,30,45,19,20,16, 19, 34, 20)
```

```
sales2<-rpois(12,34) # random numbers, Poisson distribution, mean at 34, 12 numbers
par(bg="cornsilk")
```

```
plot(sales1, col="blue", type="o", ylim=c(0,100), xlab="Month", ylab="Sales" )
title(main="Sales by Month")
```

```
lines(sales2, type="o", pch=22, lty=2, col="red")
```

```
grid(nx=NA, ny=NULL)
```

```
legend("topright", inset=.05, c("Sales1", "Sales2"), fill=c("blue", "red"), horiz=TRUE)
```

**Problem 2:** The sales data is available in a table in a text file. Read it in and draw a side-by-side **histogram** to compare the performance. (Discussion)

```
sales<-read.table(file.choose(), header=T)
sales # to verify that data has been read
barplot(as.matrix(sales), main="Sales Data", ylab= "Total",beside=T, col=rainbow(5))
```

**Problem 3:** Use **boxplot** to compare the two sales data. (Discussion: How will you interpret the graph visualization?)

```
fn<-boxplot(sales,col=c("orange","green"))$stats

text(1.45, fn[3,2], paste("Median =", fn[3,2]), adj=0, cex=.7)
text(0.45, fn[3,1],paste("Median =", fn[3,1]), adj=0, cex=.7)
grid(nx=NA, ny=NULL)
```

**12. Importing data into R studio: from csv, (ODBC relational data source: later), from the web documents.** Data available from sources such as fueleconomy.gov, data.gov, yahoo.finance etc.

<http://www.fueleconomy.gov>

<http://www.fda.gov/aboutfda/transparency/opengovernment/default.htm>

Historical prices at yahoo finance: <http://finance.yahoo.com/g/hp?s=AAPL+Historical+Prices>

**Problem 4: Download csv data from the web and analyze using the methods above.**

Download the historical prices for any two or more stocks of your choice and compare. We will do it for Apple (AAPL) and Facebook (FB) for one year.

- We will download the csv file by specifying the URL string in the file reader in R.
- Alternatively you can download using the data import tab of the right top quadrant of R Studio.

```
fb1<-read.csv("http://real-
chart.finance.yahoo.com/table.csv?s=FB&d=10&e=5&f=2014&g=d&a=11&b=12&c=2013&
ignore=.csv")
```

We can do the above. But the links are not available. So I have downloaded the files into csv files in the data directory.

```
fb1<-read.csv(file.choose())
aapl1<-read.csv(file.choose())
```

```

par(bg="cornsilk")
plot(aapl1$Adj.Close, col="blue", type="o", ylim=c(0,100), xlab="Days", ylab="Price" )
lines(fb1$Adj.Close, type="o", pch=22, lty=2, col="red")
legend("topright", inset=.05, c("Apple","Facebook"), fill=c("blue","red"), horiz=TRUE)
Just study the distribution of the adjusted close of the stock price of Apple.
hist(aapl1$Adj.Close, col=rainbow(8))

```

(Analysis)

13. **Problem 5: Data sets available with R:** R community has created a lot of data for others to use. Examine the data sets already available with R. data(), attach(), detach(), head(), summary()

```
data()
```

Observe the data sets available for explorations.

```
attach(mpg)
```

```
head(mpg)
```

```
summary(mpg)
```

#after analysis remove the data from the memory

```
detach(mpg)
```

Also explore newer data sets in

```
library(help=datasets)
```

```
library(datasets)
```

```
head(uspop)
```

```
plot(uspop)
```

Also look at this github site: <http://vincentarelbundock.github.io/Rdatasets/datasets.html>

14. **Problem 6: Accessing external APIs:** eg. Google map lat-long API: “map” command.

The idea here is to plot the results of analysis on a map: geographical or otherwise. List a collection of cities you have visited and plot it on a map.

```
library("ggmap")
```

```
library("maptools")
```

```
library(maps)
```

```
visited <- c("SFO", "Chennai", "London", "Melbourne", "Lima,Peru", "Johannesbury, SA")
```

```
ll.visited <- geocode(visited)
```

```
visit.x <- ll.visited$lon
```

```
visit.y <- ll.visited$lat
```

```
map("world", fill=TRUE, col="white", bg="lightblue", ylim=c(-60, 90), mar=c(0,0,0,0))
```

```
points(visit.x,visit.y, col="red", pch=36)
```

Here is another example using the map of The United States.

```
library("ggmap")
library("maptools")
library(maps)
visited <- c("SFO", "New York", "Buffalo", "Dallas, TX")
ll.visited <- geocode(visited)
visit.x <- ll.visited$lon
visit.y <- ll.visited$lat
map("state", fill=TRUE, col=rainbow(50), bg="lightblue", mar=c(0,0,0,0))
points(visit.x,visit.y, col="yellow", pch=36)
```

We can get very high resolution maps, different types of maps, geographical maps, historical maps, and plot on them any information you like. Check this document:

Maps package: [http://statacumen.com/teach/SC1/SC1\\_16\\_Maps.pdf](http://statacumen.com/teach/SC1/SC1_16_Maps.pdf)

15. **Problem 7:** we will conclude the “base” graphics capabilities of R package with a very old but popular data set available in R: mtcars (motor trends car package). Attach and explore mtcars. Draw scatter plots of the dependent variables (i) 5 variables (ii) 4 variables.

**Repeat the plot with some other rich data set from R package.**

```
splom(mtcars[c(1,3,4,5,6)], main="MTCARS Data")
splom(mtcars[c(1,3,4,6)], main="MTCARS Data")
splom(mtcars[c(1,3,4,6)], col=rainbow(),main="MTCARS Data")
```

Another data set: “rock”

```
splom(rock[c(1,2,3,4)], main="ROCK Data")
```

16. **Problem 8:** Working with **ggplot2** package (), loading a package, installing package. Object-oriented and incremental additions (extensibility) are special features of this package. We can layer the commands to a base plot.

There are numerous worked out examples available in the R vignettes. Search for them and work on them to learn about R.

Data for all these exercises is available at: <http://www.cse.buffalo.edu/~bina/cse487/Labs/lab1>