How to Add Code & Results to Your Document
Introduction to R Lab

Benjamin Lacroix

January 28, 2020

1 Overview

R is a free and open sourc software that can be used in its own. But with RStudio things can be a bit easier. To use R, you need to have it installed on your machine form the R[1] website. This tutorial aims at providing you with:

- Quick introduction to R and Studio
- Introduction to a very important and useful feature which is *R Sweave*.

2 Quick Intro to R

R is a command line based environment. In other words, you have to type commands rather than using Wizards, which allows you to do interactive and iterative exploration and analysis of data.If you open RStudio, you will notice that it is divided to four sub windows, bottom left window which is the console window, the editor window at the top left, the history window/environment on the top-right and bottom right is for plots, files and packages. We will explore this in depth during this module.

You can either type commands in the console, or in the script window. It is good to use the scrip window, type your commands and save. I recomment you for each lab to have a clean script file where you type your commands in and run them in order. At the end of the lab, you shold be able to re-run the whole script and obtain the same result.
For this module open a new script and copy the commands from this document to run them.
To execute the command, you can simply select it and click the run button on the top righ corner of the script window, or simly press *ctrl+enter* while the cursor is at the line of that command.

The very first step, is to set your working directory, or your current directory where you have your files, datasets, etc...:

```
currdir <- getwd()
setwd(currdir)
```

2.1 Vectors in R

Vectors are considered the first building block in R, easy to define, populate and subset. The main requirement is that vectors must have the same data type i.e. vectors of integers, vectors of characters, and so on. Consider the following statements (you can type them in the console window, or in the script window in RStudio):

[1] https://www.r-project.org/

```r
# create some vectors
ints <- c(1,2,3,4,5)
ints1 <- c(1:10)
ints2 <- seq(1,10,by=2)
doubles <- seq(from=0,to=10,by=2.5)
chars <- character()
chars[1] <- "e"
chars <- c(chars,"s")

# print vectors components
ints;ints1;ints2; chars;doubles
```

Check the output

```
[1] 1 2 3 4 5
 [1]  1  2  3  4  5  6  7  8  9 10
[1] 1 3 5 7 9
[1] "e" "s"
[1]  0.0  2.5  5.0  7.5 10.0
```

You can check the class type of the vectors that you have defined above by using the function *class()*. You can also perform several operations on the vectors:

```r
class(ints)# check the class of the vector ints
ints[length(ints)]# check the last element of the vector
class(chars)
class(doubles)
ints==ints1 # check if elements are equal
chars==ints # this will issue a warning, why?
```

You can combine vectors, add components, ets... Look at this simple example:

```r
x <- 1:10
y <- 1:10
x + y

##  [1]  2  4  6  8 10 12 14 16 18 20

x*10

##  [1]  10  20  30  40  50  60  70  80  90 100
```

2.2 Matrices

You can think of matrices as a two-dimensional vector. To define a matrix we use the function matrix as follows:

```r
# 4 x 4 matrix
myMat <- matrix (data=1:16,ncol = 4)
```

The argument data specifies which numbers should be in the matrix. Use either **ncol** to specify the number of columns or **nrow** to specify the number of rows. If you now type *myMat* in the console you should get the following results:

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    5    9   13
## [2,]    2    6   10   14
## [3,]    3    7   11   15
## [4,]    4    8   12   16
```

Some times you may need to access specific elements in the matrix:

```
myMat[1,2]# the element at row 1, column 2

## [1] 5

myMat[2,4]# the element at row 2, column 4

## [1] 14

# check matrix dimension
dim(myMat)

## [1] 4 4

# sum the elements at the diagonal of the matrix
sum(diag(myMat))

## [1] 34
```

Task: Create a vector of 40 elements (i.e. 1:40) and put it in a 10 x 4 matrix

```
mymat <- matrix(data=1:40, nrow=10)
```

Remove the second row and the third column

```
mymat <- mymat[-2,]
mymat <- mymat[,-3]
print(mymat)

##       [,1] [,2] [,3]
##  [1,]    1   11   31
##  [2,]    3   13   33
##  [3,]    4   14   34
##  [4,]    5   15   35
##  [5,]    6   16   36
##  [6,]    7   17   37
##  [7,]    8   18   38
##  [8,]    9   19   39
##  [9,]   10   20   40
```

The apply() function is a very useful function that allows you to run an operation over each row or each column. Let's assume you want to calculate the mean of each row:

```
apply(myMat, 1, mean)

## [1]  7  8  9 10
```

And of each column:

```r
apply(myMat, 2, mean)

## [1]  2.5  6.5 10.5 14.5
```

Note the the second argument on which the dimension the function is applied, 1 = row, 2 = column.

2.3 Data Frames

Data Frames are very useful data structure in R, although similar to matrices, they are much more flexible in terms of allow access to elements, columns, features, etc... Lets start by defining some vectors and then put them in a data frame:

```r
stIds <- c(1,2,3,4,5)
stNames<- c("St1","St2","St3","St4","St5")
stGrades<-c("Excellent","Good","Bad","Really Bad","Really Really Bad")

# create a data frame to combine the above vectors
df <- data.frame(student_id=stIds,student_name=stNames,student_grades=stGrades)

# print the data frame without row numbers
print(df,row.names = FALSE )
```

The output of the above code is a data frame that combines all vectors:

```
 student_id student_name    student_grades
          1          St1         Excellent
          2          St2              Good
          3          St3               Bad
          4          St4        Really Bad
          5          St5 Really Really Bad
```

There are so many operations that we can do with a data frame, here are some and we will cover more through the entire course/ not only this module:

```r
names(df) # pint out the column names of the data frame
head(df)  # show the first few rows

# subsetting: is very important operation on data frame

# hide the student name
df[,c(1,3)]

# show records where students grades is "Bad"
df[df$studeng_grades=="Bad",]
```

You can also carry some mathematical operations on some of the columns of the data frame (numeric ones):

```r
# add a new column to the data frame
df$gpa <- NULL
# generate some numbers between 40 and 90 and add it to the gpa column
df$gpa <- runif(1:nrow(df),40,90)

# print data frame contents
print(df,row.names = FALSE )
```

Now if you execute the above code, you will then get the following results:

```
##  student_id student_name    student_grades      gpa
##           1          St1         Excellent 74.69263
##           2          St2              Good 43.29756
##           3          St3               Bad 71.12232
##           4          St4        Really Bad 85.89199
##           5          St5 Really Really Bad 89.34994
```

Now, lets do some basic math operations on the *gpa* column in our data frame

```
avgGPA <- mean(df$gpa)
lowestGPA <- min(df$gpa)
maxGPA <- max(df$gpa)
```

You can print the results as follows:

```
cat("Average GPA is: \t",avgGPA)
cat("Min GPA is: \t",lowestGPA)
cat("Max GPA is: \t",maxGPA)
```

You will get results similar to this:

```
Average GPA is:   72.87089
Min GPA is:   43.29756
Max GPA is:   89.34994
```

Notice one thing about our table (data frame), the *gpa* is not consistent with the *student_grades* column, so lets use the *ifelse* to correct this. We want to change data in our data frame so that students who have *gpa* $\geq$ 80 should get an excellent grade, for gpa's between 70 and 79 this will be very good, 50 to 69 will be considered good, and 40 to 49 is bad grade, else it will be very bad. Check the code below which will implement this scenario

```
# first lets have a backup copy of our data frame
dfBackup <- df
# correct students grades
df$student_grades <- ifelse(df$gpa>=80,"Excellent",
                            ifelse(df$gpa>=70,"Very Good",
                                   ifelse(df$gpa>=50,"Good",
                                          ifelse(df$gpa>=40,"Bad","Very Bad"))))

# print the original and modified data frame
print(df,row.names = FALSE )
print(dfBackup,row.names=FALSE)
```

Check the two data frames and see compare the results

```
##  student_id student_name    student_grades      gpa
##           1          St1         Excellent 74.69263
##           2          St2              Good 43.29756
##           3          St3               Bad 71.12232
##           4          St4        Really Bad 85.89199
##           5          St5 Really Really Bad 89.34994
```

```
 student_id student_name      gpa student_grades
          1          St1 74.69263      Very Good
          2          St2 43.29756            Bad
          3          St3 71.12232      Very Good
          4          St4 85.89199      Excellent
          5          St5 89.34994      Excellent
```

We have already seen how can we add a new column to an existing data frame (recall the *gpa*), below, we want to add row to the data frame:

```
newStudent <- c(6,"ST6","Good",55.5)
df <- rbind(df,newStudent)
```

Check the data frame now, you should have one extra record in it:

```
##  student_id student_name              gpa student_grades
##           1          St1 74.6926316386089      Very Good
##           2          St2  43.297563304659            Bad
##           3          St3 71.1223162803799      Very Good
##           4          St4 85.8919937373139      Excellent
##           5          St5 89.3499384843744      Excellent
##           6          ST6             Good           55.5
```

Lets now save the data frames we created into *.csv* file as below. Notice that within my current directory I have a folder called data, where I put the files in:

```
write.csv(dfBackup,"dfBackup.csv",row.names = FALSE)
write.csv(df,"df.csv",row.names = FALSE)
```

2.4 Functions

The beautiful thing about R is that there is almost a package for every task you could think of when it comes to data analytics and machine learning. Functions are the building block and carrying out these tasks. We have already use some of these functions:

```
x <- runif(10,10,100)
mean(x)
median(x)
xRounded <- round(x,digits = 2)
x;xRounded
```

A very useful function in R is the *help()* function, which you can use to get more information:

```
help("median")
```

You will notice that executing the above help function will give you all the related information to the function *median* on the bottom-righ window of RStudio.

Some times you may need to build your functions. In fact it is good practice to encapsulate your code within functions. Lets write a simple function that takes a vector of number and square its components:

```r
# create a vector of doubles
myNumbers <- seq(from=-1,to=1,by=.1)

# function definition
squareVector <- function (x) {
    return (x*x)
}

# call function
squaredX <- squareVector(myNumbers)
```

An easy way to check that our function is doing the right calculation is to plot the results. The code below will generate a figure similar to Figure 1:

```r
plot(squaredX,type='b',xlab = 'x', ylab = 'x*x', frame=FALSE,col='blue')
```

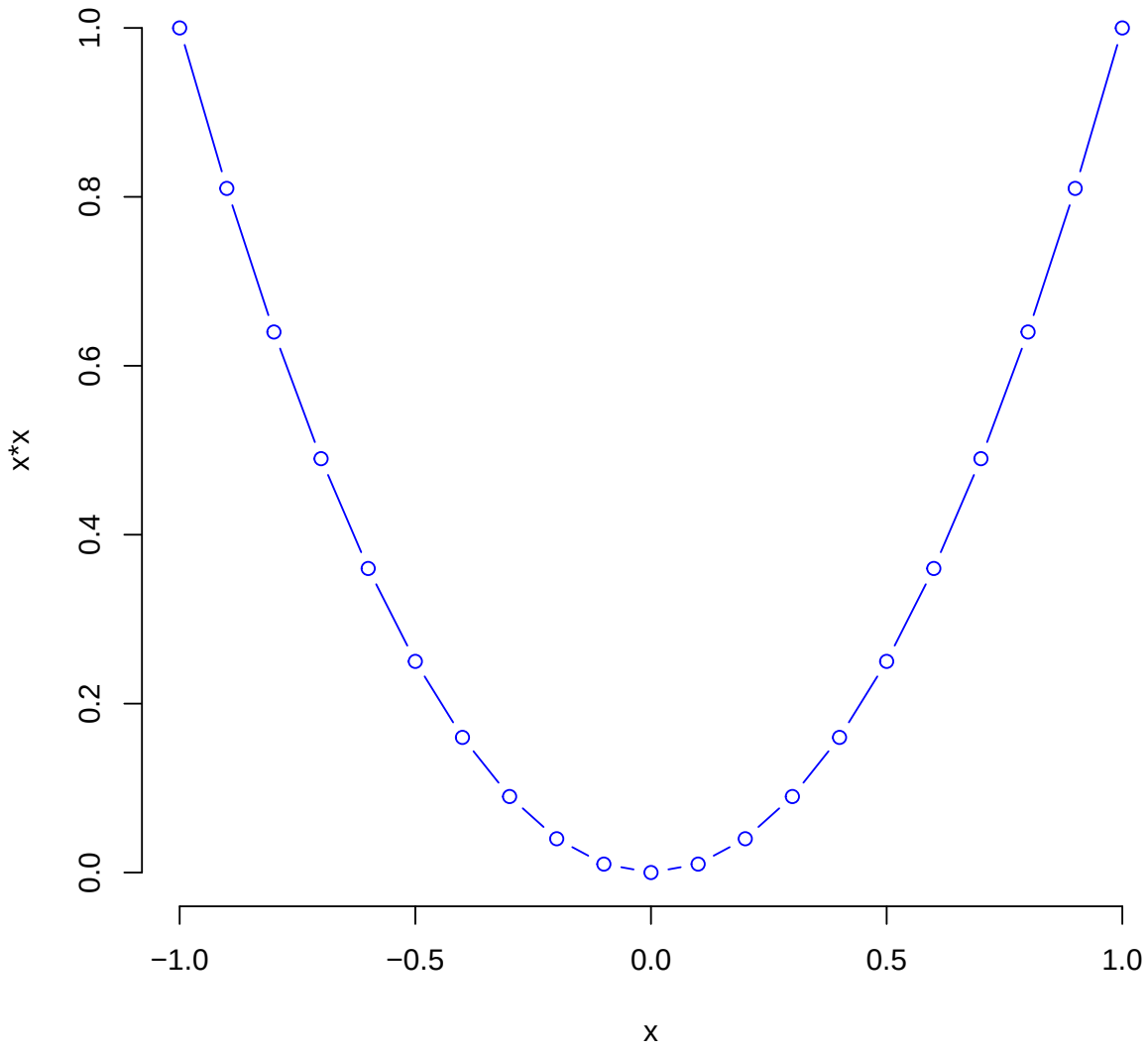

Figure 1: Simple Plot of x^2 Function

2.5 Good R Resources

There are plenty of resources in the web:

- Very short and quick intro to R[2]
- Excellent resource with videos on R, RStudio and Data Analytics[3]

[2] https://cran.r-project.org/doc/contrib/Torfs+Brauer-Short-R-Intro.pdf
[3] https://www.r-bloggers.com/in-depth-introduction-to-machine-learning-in-15-hours-of-expert-videos/

3 Re-produce this document

This document has been produced using RStudio. Notice that it contains normal text, code, and output all together in one document. In order to reproduce the document you need to have the following software installed in your machine:

- RStudio https://www.rstudio.com/
- Latex https://www.latex-project.org/
- the R package knitr (possibly other packages will be required) https://cran.r-project.org/web/packages/knitr/index.html

You also need to understand Latex basics. Check the latex template available on the module pages at Moodle.

4 Tables

4.1 Tables in Latex

The following code is used to generate a table in latex:

```
\begin{table}
\centering
\begin{tabular}{ c c c }
 Student ID & Age & Course \\
 \hline
 0110 & 22 & CS \\
 0001 & 27 & DS
\end{tabular}
\end{table}
```

The result is a table as in Table 1:

Table 1: Just a table

Student ID	Age	Course
0110	22	CS
0001	27	DS

4.2 Using R Package

R has an interesting package called *xtable*. First you need to make sure that this package is installed in your system if not then you can install it using: **install.packages('xtable')**.

Lets use the package to read the file called *year10.csv*. The file is also available on Moodle. Assume that the file in the current working directory, so first we need to read it:

```
# read the file into the variable cwFile
data("mtcars")
```

You can check number of rows and columns of the file:

```r
cat("Number of Rows: ", nrow(mtcars))

## Number of Rows:  32

cat("\nNumber of Columns: ",ncol(mtcars))

##
## Number of Columns:  11
```

Now, we want to generate the latex to present the data in a table using the R package 'xtable':

```r
library(xtable) # load xtable
xtable(mtcars)
```

If you copy the latex code above, and paste it outside the Chunks block in the Rnw document, then you will get the following table:

Table 2: The 'mtcars' dataset

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.00	6.00	160.00	110.00	3.90	2.62	16.46	0.00	1.00	4.00	4.00
Mazda RX4 Wag	21.00	6.00	160.00	110.00	3.90	2.88	17.02	0.00	1.00	4.00	4.00
Datsun 710	22.80	4.00	108.00	93.00	3.85	2.32	18.61	1.00	1.00	4.00	1.00
Hornet 4 Drive	21.40	6.00	258.00	110.00	3.08	3.21	19.44	1.00	0.00	3.00	1.00
Hornet Sportabout	18.70	8.00	360.00	175.00	3.15	3.44	17.02	0.00	0.00	3.00	2.00
Valiant	18.10	6.00	225.00	105.00	2.76	3.46	20.22	1.00	0.00	3.00	1.00
Duster 360	14.30	8.00	360.00	245.00	3.21	3.57	15.84	0.00	0.00	3.00	4.00
Merc 240D	24.40	4.00	146.70	62.00	3.69	3.19	20.00	1.00	0.00	4.00	2.00
Merc 230	22.80	4.00	140.80	95.00	3.92	3.15	22.90	1.00	0.00	4.00	2.00
Merc 280	19.20	6.00	167.60	123.00	3.92	3.44	18.30	1.00	0.00	4.00	4.00
Merc 280C	17.80	6.00	167.60	123.00	3.92	3.44	18.90	1.00	0.00	4.00	4.00
Merc 450SE	16.40	8.00	275.80	180.00	3.07	4.07	17.40	0.00	0.00	3.00	3.00
Merc 450SL	17.30	8.00	275.80	180.00	3.07	3.73	17.60	0.00	0.00	3.00	3.00
Merc 450SLC	15.20	8.00	275.80	180.00	3.07	3.78	18.00	0.00	0.00	3.00	3.00
Cadillac Fleetwood	10.40	8.00	472.00	205.00	2.93	5.25	17.98	0.00	0.00	3.00	4.00
Lincoln Continental	10.40	8.00	460.00	215.00	3.00	5.42	17.82	0.00	0.00	3.00	4.00
Chrysler Imperial	14.70	8.00	440.00	230.00	3.23	5.34	17.42	0.00	0.00	3.00	4.00
Fiat 128	32.40	4.00	78.70	66.00	4.08	2.20	19.47	1.00	1.00	4.00	1.00
Honda Civic	30.40	4.00	75.70	52.00	4.93	1.61	18.52	1.00	1.00	4.00	2.00
Toyota Corolla	33.90	4.00	71.10	65.00	4.22	1.83	19.90	1.00	1.00	4.00	1.00
Toyota Corona	21.50	4.00	120.10	97.00	3.70	2.46	20.01	1.00	0.00	3.00	1.00
Dodge Challenger	15.50	8.00	318.00	150.00	2.76	3.52	16.87	0.00	0.00	3.00	2.00
AMC Javelin	15.20	8.00	304.00	150.00	3.15	3.44	17.30	0.00	0.00	3.00	2.00
Camaro Z28	13.30	8.00	350.00	245.00	3.73	3.84	15.41	0.00	0.00	3.00	4.00
Pontiac Firebird	19.20	8.00	400.00	175.00	3.08	3.85	17.05	0.00	0.00	3.00	2.00
Fiat X1-9	27.30	4.00	79.00	66.00	4.08	1.94	18.90	1.00	1.00	4.00	1.00
Porsche 914-2	26.00	4.00	120.30	91.00	4.43	2.14	16.70	0.00	1.00	5.00	2.00
Lotus Europa	30.40	4.00	95.10	113.00	3.77	1.51	16.90	1.00	1.00	5.00	2.00
Ford Pantera L	15.80	8.00	351.00	264.00	4.22	3.17	14.50	0.00	1.00	5.00	4.00
Ferrari Dino	19.70	6.00	145.00	175.00	3.62	2.77	15.50	0.00	1.00	5.00	6.00
Maserati Bora	15.00	8.00	301.00	335.00	3.54	3.57	14.60	0.00	1.00	5.00	8.00
Volvo 142E	21.40	4.00	121.00	109.00	4.11	2.78	18.60	1.00	1.00	4.00	2.00

5 Tutorial

In order to be able to generate a PDF document (sweaved) in RStudio, you need to create a new file with the extension **.Rnw**. To do so, go to main menu→ File→New File→R Sweave. Once you created a file, then you could deal with it as a LaTeXdocument. If at any point you want to add a code, then you need to insert code, from the main menu go to Code → Insert Chunks and then start coding.
Tasks:

- Download the Lab01-template.Rnw from the campus moodle page.
- Change the author to your name.
- change the date to the deadline of the coursework (it's the 23/04/2020).
- use the 'summary' function to show the summary of the dataset.
- Create a new section titled "Data selection".
- For each of the following tasks, create a subsection using the command \subsection{}.
- Show cars with automatic transmission.
- Show the first 5 rows as a table in the latex format using xtable().
- Calculate the mean miles per gallon for every car (mpg).
- Calculate the mean mpg for automatic cars and manual cars. Comment.