Lab 1: Introduction to R

Question Sheet with Solutions

Dr Mercedes Torres Torres

Introduction

This question sheet presents a series of basic exercises designed to help you familiarize yourself with R, particularly its syntax and its wide catalogue of already-implemented functions.

In this lab session, you will learn to:

- Use your first R Markdown Notebook
- Carry out simple arithmetic calculations
- Import new packages and use operations from these packages
- Define and use vectors
- Define and use matrices
- Plot simple functions

To begin, start your R editor (RStudio or your editor of choice) and work through the Lab 1 Instructions first. Then and Lab1-Questions.pdf and start working these exercises.

If you want to use R Notebooks to solve your questions, use the file Lab1-Questions.Rmd. If you want to use simple scripts, open your own .r file in RStudio.

DISCLAMER: The answers in this document present one possible implementation of the solution of the problems. However, there are other implementations that can also provide the correct solution.

NOTE: In this lab, you cannot use any control statements (*if-else*) or loops (*for* or *while*).

1 R Notebooks

Before you start with the exercises, take a look at this file in the .Rmd format. Familiarise yourself with the contents introduced in the Instruction Sheet.

1. Look at the YAML section: do you understand the options shown?
2. Look at how the text is organised in this document:
   2.1. How are headers signalled? How can create different levels of headers?
   They are signalled with the hashtags symbol. The more hashtags, the smaller the header.
   2.2. How do you bold text? How do you italicise text?
   You bold text using two asterisks. Italics text uses either lower dash or one asterisk.
   2.3. How can you write in-line mathematical expressions?
   Using the dollar sign. For new-line math, use two dollar signs.

3. Look at the code chunks in this document:

   3.1. How can you run a code chunk?

   By pressing the green triangle at the top of each chunk.

   3.2. What rules are applied inside these chunks?

   R rules! And the rules establised when defining the chunk (inside the brackets).

   3.3. How are graphs shown? (You will have to solve the Simple Plotting tasks before being able to see this)

   They are shown in-line when running a chunk. When knitting the are shown right below their asssociated instruction.

   3.4. What are some of the options that can be used for the chunks?

   Include warnings or not. Run the chunk or not. Include results or not. Graphs options (caption, size, name, etc.)

2 R Notebooks

Before you start with the exercises, take a look at this file in the .Rmd format. Familiarise yourself with the contents introduced in the Instruction Sheet.

1. Look at the YAML section: do you understand the options shown?
2. Look at how the text is organised in this document:

   2.1. How are headers signalled? How can create different levels of headers?

   2.2. How do you bold text? How do you italicise text?

   2.3. How can you write in-line mathematica expressions?
3. Look at the code chunks in this document:

   3.1. How can you run a code chunk?

   3.2. What rules are applied inside these chunks?

   3.3. How are graphs shown? (You will have to solve the Simple Plotting tasks before being able to see this)

3 Built-in mathematical functions

In this section, you will learn how to carry out simple arithmetic operations in R. *Hint:* you can find more information about arithmetic functions in the Instructions or using the help function in R (??arithmetic)

1. Define x1 as 3342, x2 as 12 and x3 as 3.

```
x1 = 3342
x2 = 12
x3 = 3
```

2. Use the correct operations and functions to calculate:

   a. x1 added to x2

```r
x1+x2
```

```
## [1] 3354
```

b. x1 multiplied by x2

```r
x1*x2
```

```
## [1] 40104
```

c. x1 divided by x2

```r
x1/x2
```

```
## [1] 278.5
```

d. x2 to the power of x3

```r
x1^x2
```

```
## [1] 1.941232e+42
```

e. The module of c.

```r
x1%%x2
```

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

f. The log to the base 10 of x1

```r
log10(x1)
```

```
## [1] 3.524006
```

g. The log to the base 10 of x3

```r
log10(x3)
```

```
## [1] 0.4771213
```

h. The log to the base 2 of x3

```r
log(x3, 2)
```

```
## [1] 1.584963
```

i. The square root of x2
    i. By using a built-in function in R

```r
sqrt(x2)
```

```
## [1] 3.464102
```

```
 ii. By using the exponent function
```

```
x2^0.5
```

```
## [1] 3.464102
```

```
x2^(1/2)
```

```
## [1] 3.464102
```

j. The summation of the results of a. and b.

```
a = x1+x2
b = x1 * x2
a + b
```

```
## [1] 43458
```

k. The mean between a. and b.

```
mean(c(a,b))
```

```
## [1] 21729
```

```
(a+b)/2
```

```
## [1] 21729
```

l. Truncate the result of d.

```
trunc(x1^x2)
```

```
## [1] 1.941232e+42
```

3. Round the constant π with:

a. No decimal places

```
round(pi)
```

```
## [1] 3
```

b. One decimal place

```
round(pi,1)
```

```
## [1] 3.1
```

c. Three decimal places

```
round(pi,3)
```

```
## [1] 3.142
```

4 Additional mathematical functions (installing packages)

In addition to the already built-in functions present in R, you can also install external packages. You can find a vast array of packages in the CRAN repository (https://cran.r-project.org/)

These packages will have a catalogue of additional built-in functions which will increase the functionality of R. Using external packages will help you carry out complex calculations more quickly.

In this section, you will install your first external package: schoolmath. This package contains datasets and functions for simple math operations taugh at school, such as prime calculation. You can find schoolmath here: https://cran.r-project.org/web/packages/schoolmath and its documentation here: https://cran.r-project.org/web/packages/schoolmath/schoolmath.pdf Most R packages' documentation follow the same format: you will find a list of functions implemented in the package, information about what they do, and examples on how to call them.

1. Use the *install.packages()* instruction to install *schoolmath*

```r
install.packages("schoolmath")
```

2. Read the reference manual.
3. Load the *schoolmath* library.

```r
library(schoolmath)
```

4. Define *x1* as 34734, *x2* as 910 and *x3* as 1563.

```r
x1 = 34734
x2 = 910
x3 = 1563
```

5. Decompose *x1*, *x2*, and *x3* into their prime factors.

```r
y1 = prime.factor(x1)
y2 = prime.factor(x2)
y3 = prime.factor(x3)
```

6. Which number has the highest prime factor?

```r
max(c(max(prime.factor(x1)), max(prime.factor(x2)), max(prime.factor(x3))))
```

```
## [1] 827
```

7. What is the least common denominator between *x1*, *x2*, and *x3*?

```r
scm(x1, scm(x2, x3))
```

```
## [1] 1176266910
```

8. What is the greatest common divisor of between *x1* and *x2*?

```
gcd(x1,x2)
```

```
## [1] 14
```

9. What is the greatest common divisor between *x1*, *x2*, and *x3*?

```
gcd(x1,gcd(x2,x3))
```

```
## 1 is a prime!
```

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

5 Vectors

1. Try the following commands and write down what each of them is doing:

```
x <- c(3,6,8) #creates x
x #shows x
x/2 #shows x/2
x^2
sqrt(x)

x[2]  #shows second element of x, 6
x[c(1,3)] #shows elements 1 and 3 of x
x[-3] #deletes elements in position 3

y <- c(2,5,1)
y
x-y
x*y
x[y>1.5] #shows elements of x whose condition indexes in y were true
y[x==6] #shows elements of x whose condition indexes in x were true

4:10 #shows sequence of numbers between 4 and 10 (intervals of 1)
z <- seq(2,3,by=0.1) #saves in z sequence between 2 and 3 (intervals of 0.1)
rep(x,each=4) #repeats values in x 4 times each
```

2. Using *x*, *y*, *z* from the previous exercise, calculate the following:

   a. The maximum of each of the vectors

```
x = c(3,6,8) #creates x
y = c(2,5,1)
z = seq(2,3,by=0.1)
max(x)
```

```
## [1] 8
```

```r
max(y)
```

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

```r
max(z)
```

```
## [1] 3
```

b. The minimum of each of the vectors

```r
min(x)
```

```
## [1] 3
```

```r
min(y)
```

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

```r
min(z)
```

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

c. The mean of each of the vectors

```r
mean(x)
```

```
## [1] 5.666667
```

```r
mean(y)
```

```
## [1] 2.666667
```

```r
mean(z)
```

```
## [1] 2.5
```

d. Which members of x are prime (*hint*: remember the *schoolmath* package)

```r
library(schoolmath)
x[is.prim(x)]
```

```
## [1] 3
```

e. The square of vector y

```r
y^2
```

```
## [1]  4 25  1
```

f. A vector prime with the first 50 prime numbers (*challenge*: can you implement this using only 1 instruction?)

```
prime = primes()[1:50]
```

g. A vector a containing natural numbers from 1 to 50

```
a = 1:50
```

h. A vector b containing natural odd numbers from 1 to 150

```
b = seq(1,150,2)
```

i. A vector o which contains all three vectors concatenated

```
o = c(prime, a, b)
```

j. A vector m which contains values from even numbers in o

```
m = o[o%%2==0]
m1 = o[is.even(o)] #another option
```

3. We have registered the height in cm and weight in kg for four people. Heights are: 180, 165, 160, 193 (cms). Weights are 87, 58, 65, 100 (kgs)

   a. Create two vectors, *height* and *weight*, with the data.

```
height = c(1.8,1.65,1.6,1.93)
weight= c(87,58,65,100)
```

   b. Create a vector bmi with the Body Mass Index values for the four people. The BMI is calculated with: Weight in kg / (Height in m)^2

```
bmi = weight/height^2
```

   c. Create a vector *bmi_log* with the natural logarithm to the BMI values.

```
bmi_log = log(bmi)
```

   d. Create a vector with the weights for those people who have a BMI larger than 25.

```
v = bmi[bmi>25]
v
```

```
## [1] 26.85185 25.39062 26.84636
```

6 Matrices

4. Create three matrices A, B and C:

$$\mathbf{A} = \begin{bmatrix} 12 & 3 & 4 \\ 9 & 6 & 2 \\ 5 & 17 & 1 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 1 & 1 & 3 \\ 9 & 8 & 5 \\ 2 & 34 & 9 \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} 1 & 3 \\ 9 & 5 \\ 10 & 2 \end{bmatrix}$$

```
A = matrix(c(12,9,5,3,6,17,4,2,1), nrow=3, ncol = 3)
B = matrix(c(1,9,2,1,8,34,3,5,9), nrow=3, ncol= 3)
C = matrix(c(1,9,10,3,5,2), nrow=3, ncol = 2)
```

```
a. Calculate the number of rows of A
```

```
    a = nrow(A)
```

```
b. Calculate the number of columns of C
```

```
    b = ncol(C)
```

```
c. A + B
```

```
    c = A+B
```

```
d. A * B
```

```
    d= A*B
```

```
e. Diagonal matrix of A-B
```

```
    e= diag(diag(A-B))
```

```
f. B transposed
```

```
    f = t(A)
```

```
g. The inverse of A ($A^{-1}$)
```

```
g_aux= diag(rep(1,times=3))
g=solve(A,g_aux)
```

```
h. $|A|$ (the determinant of A)
```

```
h =det(A)
```

```
i. A*B
```

```
i=A*B
```

```
j.  Concatenate A and B by rows. The resulting matrix will have 6 rows and 3 columns
```

```
j=rbind(A,B)
```

```
k. Concatenate B and C by columns. The resulting matrix will have 3 rows and 5 columns
```

```
k = cbind(B,C)
```

```
l. Summation of B's columns
```

```
l=colSums(B)
```

```
m. A%*%C
```

```
m=A%*%C
```

```
n. A*A
```

```
n=A*A # multiplies element by element
```

```
o. A%*%A. What is the difference between this operation and the previous one?
```

```
o= A%*%A # matrix multiplication
```

2. We have collected the marks obtained in five different modules (*m1* to *m5*) from six different students. Answer the following questions **without using any control structures (conditionals, loops, etc.)**:

   a. Create a matrix N that contains their marks. Each student will be a row and each column will be the marks of that module.

      - John: 32, 52, 50, 44, 50
      - Mary: 88, 67, 59, 70, 70
      - Mark: 78, 77, 68, 67, 80
      - June: 89, 90, 81, 89, 87
      - Claire: 61, 65, 50, 78, 50
      - Anthony: 67, 68, 65, 40, 66

```
N = rbind(c(32, 52, 50, 44, 50),
          c(88, 67, 59, 70, 70),
          c(78, 77, 68, 67, 80),
          c(89, 90, 81, 89, 87),
          c(61, 65, 50, 78, 50),
          c(67, 68, 65, 40, 66))
```

b. Create a matrix G that contains their gender information. You might want to codify the gender with numbers to make calculations easier.

- John: M
- Mary: F
- Mark: M
- June: F
- Claire: F
- Anthony: M

```
G=c(0,1,0,1,1,0)
```

c. How many female students are there?

```
sum(G==1)
```

```
## [1] 3
```

d. What is the overall average of each student?

```
avgs = rowMeans(N)
avgs
```

```
## [1] 45.6 70.8 74.0 87.2 60.8 61.2
```

e. What is the highest average?

```
  max(avgs)
```

```
## [1] 87.2
```

f. How many students have an average between 55 and 75?

```
sum(avgs>=55 & avgs<=75)
```

```
## [1] 4
```

g. What is the difference between the highest average of male students and female students?

```
  abs(max(avgs[G==1])-max(avgs[G==0]))
```

```
## [1] 13.2
```

h. Who performed better in the module m3?

```r
which.max(N[,3])
```

```
## [1] 4
```

i. Which module has the smallest difference between the highest mark and the lowest mark?

```r
which.min(abs(apply(N,2, max)-apply(N,2,min)))
```

```
## [1] 3
```

j. Which module has the most distinctions (70 marks and over)?

```r
colSums(N>=70)
```

```
## [1] 3 2 1 3 3
```

```r
which.max(colSums(N>70))
```

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

k. Which module have the male students failed the most?

```r
M=N[G==0,] # Only male students
which.max(colSums(M<50))
```

```
## [1] 4
```

l. What is the gender of the student with the highest mark in m4?

```r
G[which.max(N[,4])]
```

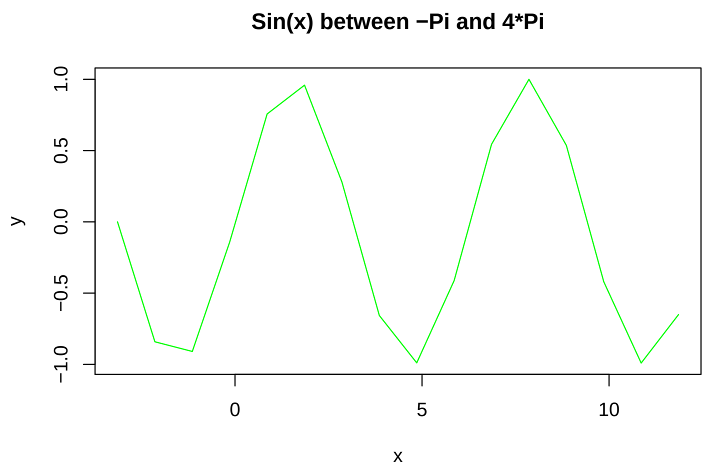
```
## [1] 1
```

7 Simple Plotting

1. Plot the following functions:

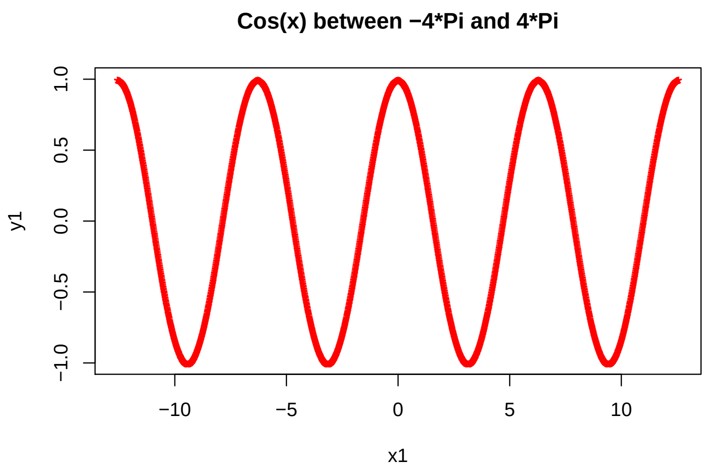
   a. The sin function between -π and 4*π with a green line

```r
x = seq(-pi,4*pi)
y = sin(x)
plot(x,y,col="green",type="l", main = "Sin(x) between -Pi and 4*Pi")
```



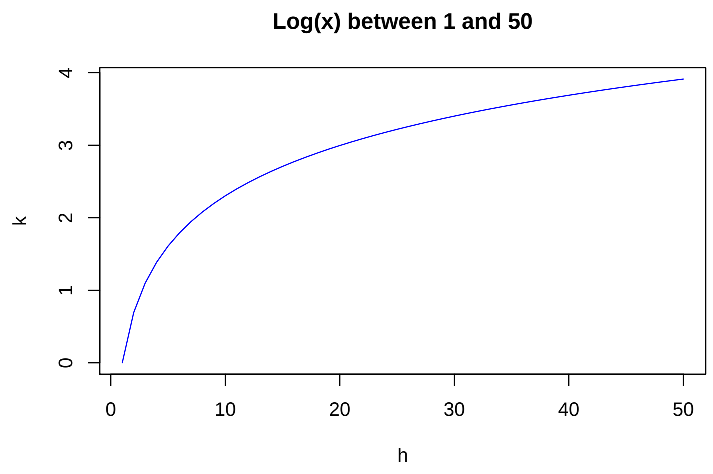
```
b.  The cos function between -4*$\pi$ and 4*$\pi$ with red markers every 0.01 values
```

```
x1 = seq(-4*pi,4*pi,by=0.01)
y1 = cos(x1)
plot(x1,y1,col="red",pch="*", main = "Cos(x) between -4*Pi and 4*Pi")
```



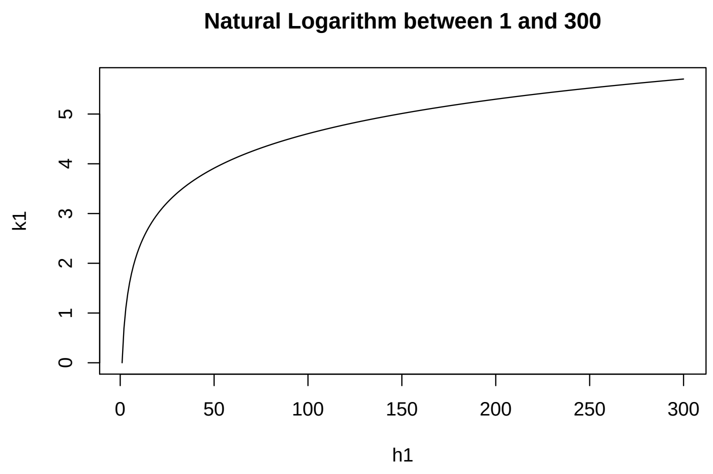
c. The log function between 1 and 50 with a blue line

```
h = seq(1,50)
k = log(h)
plot(h,k,col="blue",type="l", main = "Log(x) between 1 and 50")
```



d. The natural logarithm between 1 and 300 with a black line

```r
h1 = seq(1,300)
k1 = log(h1)
plot(h1,k1,col="black",type="l", main = "Natural Logarithm between 1 and 300" )
```

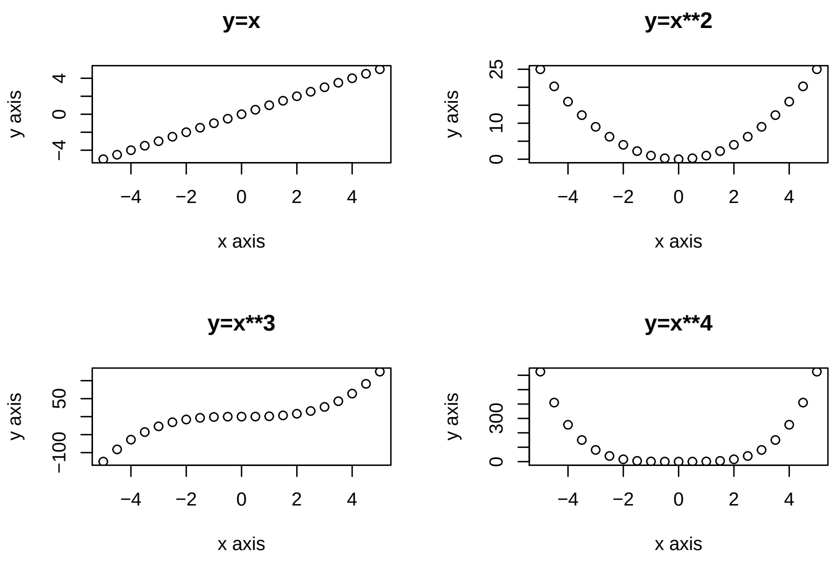


2. Create a 2x2 plot that shows the following functions in each quadrant. Each plot should: a) be shown between the -5 and 5 at intervals of 0.5 and b) Have sensible titles and axes labels:

   a. $y = x$
   b. $y = x^2$
   c. $y = x^3$
   d. $y = x^4$

```r
sinx= seq(-5,5, by=0.5)
siny1 = sinx
siny2= sinx^2
```

```
siny3= sinx^3
siny4= sinx^4
par(mfrow=c(2,2))
plot(sinx,siny1,main= "y=x", xlab = "x axis", ylab = "y axis")
plot(sinx,siny2,main= "y=x**2", xlab = "x axis", ylab = "y axis")
plot(sinx,siny3,main= "y=x**3", xlab = "x axis", ylab = "y axis")
plot(sinx,siny4,main= "y=x**4", xlab = "x axis", ylab = "y axis")
```


y=x
y=x**2
y=x**3
y=x**4
y axis
x axis