Coding with R

Dr. Niladri Chakraborty

University of the Free State

Feb 2022

1 How to install R

Steps:

- Go to the link `https://cran.r-project.org/bin/windows/base/`
- Download the executable file (.exe).
- Double click. A pop-up window opens up saying 'Do you want to allow .... to this computer?'
- Ssay 'yes'.
- Select language of your preference (default is 'English')
- Click OK and then click 'next' for every next window until 'finish'.

For RStudio,

- Go to the link `https://rstudio.com/products/rstudio/download/`
- Choose the RStudio version you want to install. RStudio Desktop works well for all regular coding.
- Follow the instructions.

2 Let's begin with some elementary codes

- 2+3 (addition)
- 2*3 (multiplication)
- 2-3 (subtraction)
- 2/3 (division)

Write the codes in the R-editor. Then select the lines and press 'ctrl+R'.
Press 'ctrl+L' if you would like to clean the R-console.

3 Vectors and Matrice

Vectors are written as,
x <- c(10.4, 5.6, 3.1, 6.4, 21.7)
or,
x = c(10.4, 5.6, 3.1, 6.4, 21.7)

Elements can be characters: z = c("red","blue").
To print, just type in:
z.
When we assign a specific letter such as 'a' or 'A' to a vector/matrix, we need to continue with the same letter with the same letter case.

Matrices are written as, A = matrix(c(2,1,2,4),nrow = 2) or,
A = matrix(c(2,3,4,1,7,3),ncol = 3).
nrow/ncol = number of rows/columns.

array() function is used for higher dimensional matrices.
A = array(c(4,5,4,4,3,2,7,6,3,2,6,7),dim = c(2,2,3)).

The elementary arithmetic operators for vectors are: +, -, *, / .
Use x^y to indicate x^y.

x[i] returns the i^{th} element of a vector.

4 Some commonly used functions in R

- log() = For logarithm.
  Ex. log(10) # code
  Output is [1] 2.302585
- exp() = For exponential function
  Ex. exp(2)
  7.389056
- pi = It is used to denote π as in trigonometry.
- sin() = For sin(x).
  Ex. sin(pi/6)
  0.5 sin(pi)
  Ex. sin(pi)
  [1] 1.224606e-16
- cos() = For cos(x).
  cos(pi)
  [1] -1.

- tan() = For tan(x).
  tan(pi)
  [1] 0.

- sqrt() = For $\sqrt{x}$. sqrt(-8+3i) does the computation as a complex number.

- max() and min() for maximum and minimum.
  Ex. a = c(2,5,4,7,9)
  > max(a)
  [1] 9.

- range() returns the maximum and minimum of a sequence of number together.
  Ex. > range(a)
  [1] 2 9

- length() returns the number of elements in a sequence of numbers.
  length(a)
  [1] 5

- sum() and prod() returns the sum and product of a sequence of numbers.
  Ex. sum(a)
  [1] 27
  Ex. prod(a)
  [1] 2520

- sort() returns the ordered sequence.
  Ex. sort(a)
  [1] 2 4 5 7 9

- rev() changes the order of a sequence.

- rep() is used to repeat a number/vector/matrix/list.

- replicate() is used to replicate a certain command. (see Q9. below for example)

- pmax() function returns the parallel maxima of two or more input vectors and pmin() does the same for minima. This means the maximum or minimum in each position of the vectors are obtained.
  Ex.
  x = c(2,4,2,3)
  y = c(1,2,3,5)
  z = c(3,1,4,5)

  pmax(x,y,z)
  [1] 3 4 4 5

Note: the 'pmax()' function returns the maximum in each position of {x,y,z}.

- which.max() and which.min() returns the maximum and minimum. For more than one maximum or minimum, it returns the first instance where the maximum or minimum occurs.
  a = c(2,4,3,2)
  which.min(a)

- order() returns the position of the elements in a vector from minimum to maximum. That means, which position has the lowest number, which position has the second lowest number, and so on.
  For example,
  x = c(2,5,4,3,6,7,1,23,3,4)
  y = order(x)
  y
  [1] 7 1 4 9 3 10 2 5 6 8

  x[y] returns the ordered x such as, [1] 1 2 3 3 4 4 5 6 7 23

- seq(x,y,d) returns a sequence of numbers between x and y at difference of 'd'.
  Ex. seq(2,100,8)
  [1] 2 10 18 26 34 42 50 58 66 74 82 90 98

  Default difference 'd=1'. In that case, one can also write c(x:y).

- dim(A) returns the dimension of a matrix A.
- A[i,j] returns the element i-th row and j-th column.
- A[i,] returns the i-th row element.
- A[,j] returns the j-th column element.
- A[1,] returns the first row.
- A[,3] returns the 3rd column.
- t(B) ## Transpose of B matrix
- A%*%t(B) ## product of A and t(B)
- t(A)%*%B ## product of t(A)and B

5 Excercises

Try to solve the coding problems on your own. Only look at the solution if you dont get the answer right.
Q1. Calculate the sum of square of all the observations in the vector a = c(1,3,4,7,3) from their mean.
Sol.
# the original vector
a = c(1,3,4,7,3)
# deviations from the mean
a-mean(a)
# use sum() function
sum($(a - mean(a))^2$)

Q2. Without using R, think what would be the result?
sum(c(TRUE, TRUE, FALSE, TRUE, FALSE))
Sol. The answer is 3. (run the code) Because R takes TRUE valeus as 1.

Q3. In 'mtcars' data, what is the mean of mpg for cars with 6 cylinders?
Sol. # $ is used to access a variable within a data.frame (or a list)
mpg <- mtcars$mpg
mpg

Then we get the cylinder numbers.

cyl <- mtcars$cyl
cyl

Then we take the mpg values of cars with 6 cylinders.
mpg[cyl==6]

mean(mpg[cyl==6])

Q4. Which car with 6 cylinders has the minimum mpg ?
Sol.
which.max(mpg[cyl==6])
which.min(mpg[cyl==6])

Q5. Arrange the mpg values in **ascending** order.
Sol. mpg <- mtcars$mpg
sort(mpg)

Q6. Arrange the mpg values in **decending** order.
Sol. mpg <- mtcars$mpg
rev(sort(mpg))

Q7. How many cars have 8 cylinders and what is range of mpg for those cars ?
Sol.
# The number of cars with 8 cylinders,
sum(cyl==8)
# **always use'==' inside ()**
# The range of mpg.
range(mpg[cyl==8])

Q8. What is the coefficient of variation for the mpg values for cars with 8 cylinders ?
Sol. To calculate the **coefficient** of variation, we need to calculate the standard deviation and the mean of mpg values for cars with 8 cylinders.
# standard deviation
sqrt(var(mpg[cyl==8]))
Also, one can use,
sd(mpg[cyl==8])
Then obtain the mean as,
mean(mpg[cyl==8])
Then the coefficient of variation can be obtained as
sqrt(var(mpg[cyl==8]))/mean(mpg[cyl==8])
or,
sd(mpg[cyl==8])/mean(mpg[cyl==8])

Q9. Create a vector of coefficients for a 10^{th} degree linear equation, using the sample function, between (1, 50). Sample should be **without replacement**.

Sol. There are 11 coefficient for a 10^{th} degree linear equation. So we draw 11 samples without replacement between (1,50).
coeffs <- sample(1:50,11,replace=T)
coeffs

Q10. Repeat the instruction in Q9 for 11 times. Store them in a matrix A. Then find the solution of the equation, $y = Ax$ where y = c(3.5,4,7,9,1,2,6.8,5,9,4,8)

Sol. We draw a set of vectors of size 11 by using the replicate(n,command) function.
Since we have to draw 11 vectors of size 11, we write the code as

A = replicate(11, sample(1:50,11,replace=T)) # store the set of vectors in A.

Then input y
y = c(3.5,4,7,9,1,2,6.8,5,9,4,8)
and,

solve(A,y) # This is the solution of the linear equation of order 10.