R

@June 9, 2022

Documentation of Project:

R is a programming language for statistical computing and graphics used to clean, analyze, and graph your data. In this tutorial, I plan to introduce the language, its IDE and usage in various forms. However, as it is not like other programming languages, rules and syntax will be covered. Concepts such as flow of control(loops and if statements) and packages built in R will be discussed. Data sets such as data frames, vectors, matrices and factors will be shown and will end with reading csv files and creating several different plots and graphs.

Script:

15 minute video

Outline -

1. What is R? What is it used for? What IDE can I use (terminal vs. RStudios) ?
2. Important Functions in R
3. Graphs and plots
4. Data Frames, arrays, matrices, vectors
5. Reading csv file, manipulating and plotting

Important Points -

- R can be used as a calculator
- library

- greater-than sign (>) is the prompt symbol which appears in the Console Pane
- : is used to mention a range of number → 10:20 would be [10-20] (both included), could also use 2: (3*2+5) which would be [2-11]
- constants such as ‘pi’ → (1:3) + pi would give {414,5.14,6.14}
- power operator (^) 3^3 gives 27
  Modular arithmetic → 31%%7 gives 3 (remainder)
  integer part → 31%/%7 gives 4
- Workspace known as the global environment that can be used to store the results of calculations, and other objects (variables)
  interest <- 10^20 (sign convention)
  could also use =, but <- is preferred as we are requesting an action rather than stating a relation
- we get around most of our work in R through functions.
  function to quit R is q()
- R is case-sensitive
  x <- 1:10
  MEAN(x) would not work unless you create a user-defined function MEAN() which uses the mean() function
  MEAN <- mean ; now the MEAN(x) function call would work
- A list of all objects in the current workspace can be printed to the screen using the objects() function; a synonym is ls() function
- Remember that if we quit our R session without saving the workspace image, then these objects will disappear. If we save the workspace image, then the workspace will be restored at our next R session.
- runif() ⇒ random number generator
  runif(4,min= , max=)
- Vectors in R →
  numeric vectors is a list of numbers.
  c() function is used to join values into a vector
  a <- c(10,2,5,8,3,4,7,2,7)
  ax <- c(a,x) (vectors can be internally joined too)

- extracting elements from vectors
  ax[3] or a[c(2,5,1)] or ax[1:5]
  negative indices can be used to avoid certain elements → ax[-c(2,5)]
- can easily perform vector arithmetic
  x^3, y <- x + 6
- patterned vectors :
  seq(1,9, by = 2) → 1 3 5 7 9 (if by is not specified, by default 1 will be used)
  rep(3,5) → 3 3 3 3 3
  rep(seq(2,10,by=2),2) → 2 4 6 8 10 2 4 6 8 10
  rep(c(1,3), each = 4) → 1 1 1 1 3 3 3 3
  rep(c(1,2),c(4,9)) → 1 1 1 1 2 2 2 2 2 2 2 2 2
  rep(1:5, rep(2,5)) → 1 1 2 2 3 3 4 4 5 5
- random pattern vectors :
  sample(1:6, size = 5, replace = TRUE)
  die is tossed 5 times to give random values between 1-6
- character vectors :
  all elements of a vector must be of the same type,
  colors <- c(”red”,”blue”,”green”)
  to take substrings, we use substr(x, start, stop) like substr(colors,1,2)
  paste() function to concatenate - paste (colors, “flowers”)
  collapse = “, “ used to make all into one string
- Factors :
  alternative way to store character data
  provide a level for the data
  as.integer(grp) - non repetitive
- Matrices and arrays
  m <- matrix(1:6, nrow = 2, ncol = 3)
  access element of a matrix using indices → m[1,2]
  access element of a array using indices → m[4]
  access row of a matrix using indices → m[1,]
  access column of a matrix using indices → m[,2]
  a <- array(1:24, c(3,4,2))
- use ?q or help(q) to access the help facility for a function q

- Storage of numbers : in the form of binary ⇒ 2^0, 2^1, 2^2,…
- values < - NULL (keeps the variable empty)
  xyz[seq(2, 20, 2)] <- seq(2, 20, 2)
  ## [1] NA 2 NA 4 NA 6 NA 8 NA 10 NA 12 NA 14 NA 16 NA 18 NA 20
  is.na(xyz) ⇒ detects if there are null values
- elementary built in functions:
  var(x), summary(x), length(x), min(x), max(x), pmin(x) {pairwise minima}, pmax(x) {pairwise maxima}, range(x), IQR(x)
- Logical operations in R:
  a <- c(TRUE, FALSE, FALSE, TRUE)
  b <- c(13,6,3,2)
  b[a] gives 13 and 2
  sum(a) gives 2 → number of true values
  !a gives FALSE TRUE TRUE FALSE
  && and || are similar to & and |; they are not vectorized: only one calculation is done
- Relational operators:
  v <- c(4,6,2,9,7)
  v > 5 gives you FALSE TRUE FALSE TRUE TRUE
- Data frames and lists:
  data sets are stored in R as data frames, and we get some with R, such as women.
  summary(women), nrow(women), ncol(women), dim(women), str(women)
  extract from data frames similar to matrices
  columns can be accessed using $ operator
  women$height[women$weight>150]
  with() function allows us to access columns of data frame without using $,
  with(women, weight/height)
  constructing data frames: xy <- data.frame(x,y)
- barplot(WorldPhones51, cex.names = 0.75, cex.axis = 0.75,
  main = "Numbers of Telephones in 1951")
- dotchart(WorldPhones51, xlab = "Numbers of Phones (’000s)")
- groupsizes <- **c**(18, 30, 32, 10, 10)
  labels <- **c**("A", "B", "C", "D", "F")

pie(groupsizes, labels,col = **c**("grey40", "white", "grey", "black", "grey90"))

- boxplot(Sepal.Length ~Species, data = iris, ylab = "Sepal length (cm)", main = "Iris measurements", boxwex = 0.5)
- Flow control:
  for() loop - for (name in vector) { commands }
  factorial function -
  n < - 10
  res < - 1
  for (i in 1:n)
  res < - res * i
  res
- if() statement -
  if (condition) {commands when TRUE}
  if (condition) {commands when TRUE} else {commands when FALSE}
  x <- 3 if (x > 2) y <- 2 * x else y <- 3 * x
- while() loop-
  Fib1 <- 1
  Fib2 <- 1
  Fibonacci <- c(Fib1, Fib2)
  while (Fib2 < 300) {
  Fibonacci <- c(Fibonacci, Fib2)
  oldFib2 <- Fib2
  Fib2 <- Fib1 + Fib2
  Fib1 <- oldFib2
  }
- replicate() function:
  evaluate the statements n times - replicate(n, {statements}

Personal Notes -

2 Introduction to the R language 7

2.1 First steps 7
2.2 Basic features of R 11
2.3 Vectors in R 13
2.4 Data storage in R 22
2.5 Packages, libraries, and repositories 27
2.6 Getting help 28
2.7 Logical vectors and relational operators 34
2.8 Data frames and lists 37
2.9 Data input and output 43

3 Programming statistical graphics 49

3.1 High level plots 50
3.2 Choosing a high level graphic 62
3.3 Low level graphics functions 63
3.4 Other graphics systems 70

4 Programming with R 76

4.1 Flow control 76
4.2 Managing complexity through functions 91
4.3 The `replicate()` function 97
4.4 Miscellaneous programming tips 97
4.5 Some general programming guidelines 100
4.6 Debugging and maintenance 107
4.7 Efficient programming 113