R language

* Introduction
  + interpreted language
  + made only for statistics calculation
  + Robert Gentleman invented R in 1993
  + by studying S developed by IBM
* Features
  + is Object oriented programming language
* Identifier
  + used to create a function and a variable
  + rules
    - can NOT start an identifier with Number
    - e.g.
      * 1name is invalid
    - can NOT use special characters like space
    - BUT dot(.) is allowed in variable name
    - e.g.
      * first.name = ‘steve’
      * last.name = ‘Jobs’
* Variable
  + syntax:
    - <variable name> = <initial value>
    - <variable name> <- <initial value>
    - <variable name> <<- <initial value>
    - <initial value> -> <variable name>
    - <initial value> ->> <variable name>
  + Rule
    - arrow will always point to the variable name
  + e.g.
    - num = 10
    - num <- 10
    - num <<- 10
    - 10 -> num
    - 10 ->> num
* Types
  + Data Type in R is always inferred
  + automatically decided by R interpreter
  + Array Index in R will always start from One (1)
  + Types
    - Vectors
      * one dimensional array
      * array is a collection of SIMILAR data types
      * broadcast operators
        + e.g.

v1 = c(0, 2, 3, 4, 5)

v2 = c(0, 2, 0, 4, 0)

* + - * + mathematical

v1 + 10

v1 – 10

v1 \* 10

v1 / 10

v1 + v2 # 0 4 3 8 5

v1 – v2

v1 \* v2

v1 / v2

* + - * + logical

v1 & v2 # FALSE TRUE FALSE TRUE FALSE

v1 && v2 # FALSE

v1 | v2 # FALSE TRUE TRUE TRUE TRUE

v1 || v2 # FALSE

* + - * types
        + numeric

can be used for both whole numbers and decimal numbers

e.g.

salary = 4.5

num = 100

* + - * + integer

represents whole number

use suffix L to create vector of integer(s)

e.g.

num = 100L

* + - * + character

represents a string

e.g.

name = ‘steve’

name = “Steve”

* + - * + logical

represents TRUE/FALSE

similar to bool in python

e.g.

canVote = TRUE

isHungry = FALSE

* + - * + raw

contains the raw representations of character(s)

e.g.

address = ‘pune’

address2 = charToRaw(address)

print(address2) # 70 75 6e 65

* + - * + complex
    - Lists
      * collection of vectors
      * create a list without temporary name(s)
        + e.g.

l1 = list(c(10, 20, 30), c(‘p1’, ‘p2’, ‘p3’))

print(l1[[1]]) # 10, 20, 30

print(l1[[1]][3]) # 30

print(l1[[2]]) # p1 p2 p3

print(l1[[2]][2]) # p2

* + - * create a list with temporary name(s)
        + e.g.

l1 = list(names=c(‘p1’, ‘p2’), marks=c(10, 15))

print(l1[[1]]) # p1 p2

print(l1$names) # p1 p2

print(l1$names[2]) # p2

print(l1$marks[2]) # 15

* + - Matrices
      * two dimensional array
      * to create a matrix use matrix()
      * e.g.
        + m1 = matrix(c(1, 2, 3, 4), nrow = 2, ncol = 2, byrow=TRUE, dimnames=list(c(“r1”, “r2”), c(“c1”, “c2”))
        + where

nrow: no of rows

ncol: no of columns

byrow: use column major (FALSE) or row major (TRUE)

dimnames: list of row names and column names

* + - * + print(m1[‘r1’, ‘c1’]) # 1
        + print(m1[‘r2’, ]) # 3 4
      * e.g.
        + m1 = matrix(c(1, 2, 3, 4), nrow=2, ncol=2, byrow=TRUE)
        + print(m1) # 1 2
        + # 3 4
        + print(m1[1, ]) # 1 2
        + print(m1[, 1]) # 1 3
        + print(m1[-1, -1]) # 4
      * recycling of elements
        + if number of elements from the vectors are less than required number of elements then matrix recycles the element(s) from the first position
        + e.g.

m1 = matrix(c(1, 2), nrow=2, ncol=2, byrow=TRUE)

print(m1) # 1 2

# 1 2

* + - Arrays
    - Factors
      * represents categorical data
    - Data Frames
      * two dimensional array
      * constructed by using collection of vectors
      * to create a dataframe use data.frame()
      * note:
        + by default data frame converts the qualitative/categorical data to factors
      * functions
        + summary()

prints the summary of the data frame

if the column is numeric

prints the central tendency values including

min

1st Quantile

Median

3rd Quantile

Mean

max

if the column is a qualitative

the levels along with the number of times the level is getting repeated

* + - * + head()

prints first few lines

* + - * + tail()

prints last few lines

* special values
  + NA: Not Available
* Slicing
  + retrieving less values than the actual values
  + e.g.
    - numbers = c(10, 20, 30, 40, 50, 60, 70, 80, 90, 100)
    - print(numbers) # 10 20 30 40 50 60 70 80 90 100
    - print(numbers[0]) # numeric(0)
    - print(numbers[1]) # 10
    - print(numbers[5]) # 50
    - print(numbers[1:4]) # 10 20 30 40
    - print(numbers[c(1, 2, 5, 6)]) # 10 20 50 60
    - print(numbers[-3]) # 10 20 40 50 60 70 80 90 100
    - print(numbers[c(-5, -3, -7)]) # 10 20 40 60 80 90 100
    - print(numbers[-3:-9]) # 10 20 100
    - print(numbers[c(-1, 0)]) # 20 30 40 50 60 70 80 90 100
  + Invalid cases
    - print(numbers[:])
    - print(numbers[1:])
    - print(numbers[:4])
    - print(numbers[c(-1, 2, 3)])
* loops
  + for..in loop
    - numbers = c(1, 2, 4, 5)
    - for (number in numbers) {
    - print(number)
    - }
* Function
  + block with name which can be used to reuse the code
  + types
    - custom
    - system
      * charToRaw()
        + used to convert character to raw representation
      * c():
        + used to create a vector with multiple values