Basics of R programming

Snigdha Cheekoty

8/19/2018

Basics of R programming are discussed in this chapter. The data structures in R are as below. ## Variable

Stores a single value - logical, character, numeric,etc.

```
apple <- 12
apple
```

```
## [1] 12
```

In the above code, we can see that a variable named 'apple' has been assigned a single value of 12. This is *variable assignment*.

Data types

- **numeric** – decimal values
- **integer** – natural/whole numbers
- **character** – string/text
- **logical** – boolean values

- `class()` is a function that returns the type of the R element. For example,

```
a <- 1200
b <- TRUE
c <- "Snigdha"
d <- 23.67888
class(a)
```

```
## [1] "numeric"
```

```
class(b)
```

```
## [1] "logical"
```

```
class(c)
```

```
## [1] "character"
```

```
class(d)
```

```
## [1] "numeric"
```

Vector

One dimensional array that stores data of the *same type*. It can be created using `c()` combine function. For example,

```
boolean_vector <- c(TRUE, TRUE, FALSE)
integer_vector <- c(23,45,89,90)
character_vector <- c("Snigdha","Rohitha","Shashank")
print(boolean_vector)

## [1]  TRUE  TRUE FALSE

print(integer_vector)

## [1] 23 45 89 90

print(character_vector)

## [1] "Snigdha"  "Rohitha"  "Shashank"
```

Naming a vector

We can give names to the vector elements using `names()` function. For example,

```
number <- c(12,13,34)
fruits <- c("apples", "oranges","grapes")
names(number) <- fruits
number

##  apples oranges  grapes
##      12      13      34
```

Arithmetic calculations on vectors

Following calculations(addition, subtraction, multiplication, division, modulo) are possible on the vectors. This results in *element-wise-calculation*,i.e., the calculations happen over the ith element of vector-1 and ith element of vector-2.

```
a <- c(0,1,1)
b <- c(1,1,2)
a+b

## [1] 1 2 3

a-b

## [1] -1  0 -1

a*b

## [1] 0 1 2

a/b
```

```
## [1] 0.0 1.0 0.5
```

```
a%%b
```

```
## [1] 0 0 1
```

- `sum()` function is used to calcualte the total sum of all the vector elements. [*Vector addition*]
- Likewise, `mean()`, `max()`, `min()`, etc., are functions used on all the elements of the vector. For example, vector sum of 'a'-vector `sum(a)` is 2.

Comparison operations in vectors

, <, >=, <=, ==, != are few operators that can be used in such comparison operations.

```
a <- c(0,1,1)
b <- c(1,1,2)
a > b
```

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

```
a < b
```

```
## [1]  TRUE FALSE  TRUE
```

```
a >= b
```

```
## [1] FALSE  TRUE FALSE
```

```
a != b
```

```
## [1]  TRUE FALSE  TRUE
```

```
a == b
```

```
## [1] FALSE  TRUE FALSE
```

Selecting elements of a vector

We can select elements of a vector using square brackets '[ ]'. Unlike many other programming languages, index of the first element of the vector is 1, NOT 0.

```
a <- c(10,20,30,40,50)
a[1] # first element of the vector
```

```
## [1] 10
```

```
a[2] #second element of the vector
```

```
## [1] 20
```

```
a[5] #fifth element of the vector
```

```
## [1] 50
```

Other ways of selecting elements are :

```
a <- c(10,20,30,40,50)
a[c(2,3)]

## [1] 20 30

a[1:3]

## [1] 10 20 30

#The above two statements give the same output- 2nd and 3rd elements
```

Combining several vectors into one vector

Same as combining several individual elements into a vector. (using `c()`)

```
a <- c(0,1,2,3)
b <- c(3,4,5)
c <- c(8,9)
combined <- c(a,b,c)
combined

## [1] 0 1 2 3 3 4 5 8 9
```

Matrix

- Two dimensional form of representing data.
- Contains elements of the same type.
- Horizontal - rows; vertical - columns

Storing data in the matrix

```
new <- matrix(1:9, byrow = TRUE, nrow = 3)
new

##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
```

Where: **1:9** –> collection of elements to be arranged **byrow** –> to be filled by rows **nrow** –> number of rows in the matrix

Naming the columns and rows of a matrix

Using `colnames()` and `rownames()`

```
new <- matrix(1:9, byrow = TRUE, nrow = 3)
colnames(new) <- c("FirstColumn","SecondColumn","ThirdColumn")
rownames(new) <- c("FirstRow","SecondRow","ThirdRow")
new
```

```
##           FirstColumn SecondColumn ThirdColumn
## FirstRow            1            2           3
## SecondRow           4            5           6
## ThirdRow            7            8           9
```

Calculating row and column totals

Using `rowSums()`, elements in the rows are added. Using `colSums()`, elements in the columns are added.

```
rowSums(new)

##  FirstRow SecondRow  ThirdRow
##         6        15        24

colSums(new)

##  FirstColumn SecondColumn  ThirdColumn
##           12           15           18
```

Adding columns and rows to the existing matrix

Using `cbind()` and `rbind()` **cbind()** - adds vectors, matrices to the existing matrix (by column) **rbind()** - adds vectors, matrices to the existing matrix (by row)

```
matrix_A <- matrix(1:9, byrow = TRUE, nrow = 3)
matrix_B <- matrix(21:29, byrow = TRUE, nrow = 3)
vector_A <- c(100,200,300)

#adding by column
all <- cbind(matrix_A,matrix_B, vector_A)
#adding by row
all2 <- rbind(matrix_A, matrix_B, vector_A)
all

##                     vector_A
## [1,] 1 2 3 21 22 23      100
## [2,] 4 5 6 24 25 26      200
## [3,] 7 8 9 27 28 29      300

all2

##          [,1] [,2] [,3]
##             1    2    3
##             4    5    6
##             7    8    9
##            21   22   23
##            24   25   26
##            27   28   29
## vector_A  100  200  300
```

Selecting matrix elements

```
matrix_B <- matrix(21:29, byrow = TRUE, nrow = 3)
matrix_B[1,3]

## [1] 23

matrix_B[1:2,3]

## [1] 23 26

matrix_B[1:2, 1:3]

##      [,1] [,2] [,3]
## [1,]   21   22   23
## [2,]   24   25   26
```