R programming - Tutorial for absolute Beginners

Introduction to R programming

Hi all, This is a tutorial to make you familiarize with R programming just by using a browser.

Practice Exercise

R can be used as a calculator

Here's a simple exercise with an empty code chunk provided for entering the answer.

Write the R code required to add two plus two:

```
2+2
```

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

Write the R code required to multiply three with five:

```
3*5
```

```
## [1] 15
```

Write the R code required to add five with five and divide the answer with two:

```
(5+5)/2
```

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

Basic Data Objects

R programming works with numerous data types including

- Scalars
- Vectors
- Matrices
- Data Frames
- Lists

Scalar

Assign a value 28 to a variable x and then find out the class/data type where it belongs

```
x<-28
class(x)
```

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

Assign the string "R is fantastic" to a variable y and then find out the class/data type where it belongs

```
y<-"R is fantastic"
class(y)
```

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

Assign the value TRUE to a variable z and then find out the class/data type where it belongs

```
z<-TRUE
class(z)
```

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

Thus to add a value to the variable, use <- or =

To print any value we will use like below:

```
x<-42
x
```

```
## [1] 42
```

```
y<-"hello"
y
```

```
## [1] "hello"
```

Vector

A vector is a one-dimensional array. We can create a vector with all basic data-type we learnt before. The simplest way to build a vector in R, is to use the c command.

- Create a character vector a,b,c

```
vec_char <-c("a","b","c")
vec_char
```

```
## [1] "a" "b" "c"
```

- Create a vector with boolean type TRUE,FALSE,TRUE

```
vec_bool <-c(TRUE,FALSE,TRUE)
vec_bool
```

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

In R its possible to slice a vector. in some instances, we are interested in only some of the values in the vector.

```
vec_slice<-c(1,2,3,4,5,6,7,8,9,10)
vec_slice[1:5]
```

```
## [1] 1 2 3 4 5
```

In R you can create adjacent values say 1 to 10 in a much easier way rather than typing everything.

c(1:10). Try this::

```
c(1:10)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

- Create a vector *vec_num* with numbers 1,10,49.

```
vec_num <-c(1,10,49)
vec_num
```

```
## [1]  1 10 49
```

Matrix

A matrix is a 2-dimensional array that has m number of rows and n number of columns. A matrix can also be called as a combination of two or more vectors with the same data type

Note: It is possible to create more than two dimensional arrays with R

How to create a matrix in R Construct a matrix with 5 rows that contain the numbers 1 upto 10

```
matrix_a<- matrix(1:10,byrow=TRUE,nrow=5)
matrix_a
```

```
##      [,1] [,2]
## [1,]    1    2
## [2,]    3    4
## [3,]    5    6
## [4,]    7    8
## [5,]    9   10
```

Print dimension of a matrix

```
dim(matrix_a)
```

```
## [1] 5 2
```

Create a 4 X 3 matrix using ncol and fill the row from top to bottom

```
matrix_c<-matrix(1:12, byrow= FALSE, ncol=3)
matrix_c
```

```
##      [,1] [,2] [,3]
## [1,]    1    5    9
## [2,]    2    6   10
## [3,]    3    7   11
## [4,]    4    8   12
```

Add a column to a Matrix with cbind()

You can add a column to a matrix with cbin() command. cbind() means column binding. cbin() can concatenate as many matrix or columns as specified.

```
matrix_a1 <-cbind(matrix_a,c(1:5))
dim(matrix_a1)
```

```
## [1] 5 3
```

Note: The number of rows of matrices should be equal for cbind to work

Add one row to matrix Example:

```
matrix_c<-matrix(1:12,byrow =FALSE, ncol=3)
add_row <-c(1:3)
matrix_c<-rbind(matrix_a1,add_row)
dim(matrix_c)
```

```
## [1] 6 3
```

Slicing a Matrix

We can select one or many elements from a matrix by using square brackets []. For example:

- matrix_c[1,2] selects elements at first row and second column.
- matrix_c[1,] selects all elements of the first row.
- matrix_c[,1] selects all elements of the first column.

Data Frame

A data frame is a list of vectors which are of equal length. A matrix contains only one type of data, while data frame accepts different data types(numeric, character,factor etc..)

The syntax of creating a dataframe is as :

data.frame(df,stringsAsFactors = TRUE)

what is a factor in R Factors are variables in R which tak on alimited number of different values; Such variables are often referred to as categorical variables.

Example syntax :

factor(x = character(), levels, labels=levels, ordered=is.ordered(x)) x: A vector of data. Need to be string or integer, not decimal Levels: A vector of possible values taken by x

Labels: Add a label to the x data. For eg:- 1 can take the label 'male' while 0 , the label 'female'

Ordered: Determine if the levels should be ordered

We will try one example:

```
gender_vector<-c("Male","Female","Female","Male","Male")
class(gender_vector)
```

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

```
# Convert gender_vector to factor
factor_gender_vector<-factor(gender_vector)
class(factor_gender_vector)
```

```
## [1] "factor"
```

Lets again talk about Data Frames: We can create the first data frame by combining four variables of same length:

```
a <-c(10,20,30,40)
b<- c('book','pen','textbook','pencil_case')
c <-c(TRUE,FALSE,TRUE,FALSE)
d<- c(2.5,8,10,7)
# Create Data frame

df<-data.frame(a,b,c,d)
```

Change column name of data frame

```
names(df) <- c('ID','items','store','price')
df
```

```
##   ID       items store price
## 1 10        book  TRUE   2.5
## 2 20         pen FALSE   8.0
## 3 30    textbook  TRUE  10.0
## 4 40 pencil_case FALSE   7.0
```

Slice a Data Frame It is possible to slice values of a dataframe. We select the rows and columns to return into bracket preceeded by the name of the data frame.

- If we let the left part blank, it will select all rows
- If we leave right part blank, it will select all the coloumns.

Perform the following :

- Select Row 1 in column 2
- select rows 1 to 2
- Select column 1
- Select rows 1 to 3 and columns 3 to 4
- Select ID and store

```
df[1,2]
```

```
## [1] book
## Levels: book pen pencil_case textbook
```

```
df[1:2,]
```

```
##   ID items store price
## 1 10  book  TRUE   2.5
## 2 20   pen FALSE   8.0
```

```
df[,1]
```

```
## [1] 10 20 30 40
```

```
df[1:3,3:4]
```

```
##   store price
## 1  TRUE   2.5
## 2 FALSE   8.0
## 3  TRUE  10.0
```

```
df[, c('ID','store')]
```

```
##   ID store
## 1 10  TRUE
## 2 20 FALSE
## 3 30  TRUE
## 4 40 FALSE
```

Append a column to Data Frame

You can also append a column to a Data Frame.You need to use symbol $ to append a new variable. Add a colum quantity (10,35,40,5) to the dataframe df

```
quantity<-c(10,35,40,5)
df$quantity<-quantity
df
```

```
##   ID       items store price quantity
## 1 10        book  TRUE   2.5       10
## 2 20         pen FALSE   8.0       35
## 3 30    textbook  TRUE  10.0       40
## 4 40 pencil_case FALSE   7.0        5
```

List

A list is a great tool to store many kinds of object in the order expected. We can include matrices, vectors, dataframes or lists. We can imagine list as a bag in which we want to put many different items. When we need to use an item, we open the bag and use it. A list is similar to this: We can store a collection of objects and use them when we need them.

- Create a list containing strings, numbers, vectors and a logical

```
list_data <- list("Red", "Green", c(21,32,11), TRUE, 51.23, 119.1)
print(list_data)
```

```
## [[1]]
## [1] "Red"
##
## [[2]]
## [1] "Green"
##
## [[3]]
## [1] 21 32 11
##
## [[4]]
## [1] TRUE
##
## [[5]]
## [1] 51.23
##
## [[6]]
## [1] 119.1
```

- Create a list containing a vector, a matrix and a list.

```
# Create a list containing a vector, a matrix and a list.
list_data <- list(c("Jan","Feb","Mar"), matrix(c(3,9,5,1,-2,8), nrow = 2),
   list("green",12.3))

# Give names to the elements in the list.
names(list_data) <- c("1st Quarter", "A_Matrix", "A Inner list")

# Access the first element of the list.
print(list_data[1])
```

```
## $`1st Quarter`
## [1] "Jan" "Feb" "Mar"
```

```
# Access the thrid element. As it is also a list, all its elements will be printed.
print(list_data[3])
```

```
## $`A Inner list`
## $`A Inner list`[[1]]
## [1] "green"
##
## $`A Inner list`[[2]]
## [1] 12.3
```

```
# Access the list element using the name of the element.
print(list_data$A_Matrix)
```

```
##      [,1] [,2] [,3]
## [1,]    3    5   -2
## [2,]    9    1    8
```

This tutorial has given you a basic overview of datatypes and Data objects in R programming.