Scientific Programming using R

Module Number: 02

Module Name: Data Structures, Looping and Branching

Version Code:RPDS2
Released Date:2-Jan-2019

Data Structures, Looping and Branching

AIM:

To equip students with Data Structures. Looping and Branching in R

Objectives:

The Objectives of this module are:

- Understand Different Data Types.
- Understand Matrices and Data frames.
- Understand Branching Statement.
- Understand Looping statements.

Outcome:

At the end of this module, you are expected to:

- Work with different data types.
- Write R script using matrices and data frames.
- Write R script using conditional statements.
- Write R script using looping statements.

Contents

(Continued) Contents

11. Filtering on matrix
12. Generating a Covariance Matrix
13. Applying function to row and column of the matrix
14. Data frame – creating, coercion
15. Combining data frames
16. Applying functions: lapply( ) and sapply( ) on data frames
17. Control statements
18. If statement
19. If-else statement
20. Switch statement

(Continued) Contents

Introduction to Different Data Types

- Data Types are names specifies/tells what kind of information a variable holds
- R supports following Data Types:
  - Vectors
  - Lists
  - Matrices
  - Arrays
  - Factors
  - Data Frames
  - Coercion

Vectors and Atomic Vectors

- Vector is a simplest/smallest value in R
- Following is the example of vector
  - 23
- Vector contains element of the same data type.
- The data types can be integer, double, character, complex or any other type.
- If we combine more than one value using c() is called atomic vector
- Atomic vector is linear vectors of single primitive data type
- Here c() function is concatenate function
- Following line of code creates a atomic vector with name “a” using 3 elements
  - a <- c(5,10, 15)

(Continued) Vectors and Atomic Vectors

- Kinds of atomic vectors
  - Logical - may contain TRUE, FALSE, NA.
  - Integer.
  - Double - real is a deprecated alias.
  - Complex - as in complex numbers; write as 0+0i.
  - Character - “string”.
  - raw - bit streams; printed in hex by default.

Coercion

- Converting one data type to other data type is called coercion
- In R, we have two types of coercions, that occur automatically
  - Coercion with formal objects
  - Coercion with built-in types
- Below code is the example of coercion:

```
v1 = c(10,20,30,40)
typeof(v1)          #at this stage type is double

v1[2] = 'shiva'
typeof(v1)          #at this stage type is character
```

(Continued) Coercion

- R provides some built-in function to convert data types
- Some of those functions as follows
  - as.logical(x)
    - Converts x into logical values
  - as.integer(x)
    - Converts x into integer values
  - as.double(x)
    - Converts x into double values
  - as.complex(x)
    - Converts x into complex values

Lists and List Indexing

- Lists are the R objects which contain elements of different types
- Data types may contain one or more following types
  - numbers, strings, vectors and another list inside it.
- We use **list()** function to create a list in R
- Below code creates a simple lists:
  - l1 = list(10,20,30)
  - l2 = list(c(c(1,2,3), c(10,20,30)))
  - l3 = list('shiva', 'kumar')
  - l4 = list(c(1,2,3), c(10,20,30))

Lists and List Indexing

- List element can be accessed using integer number starts with 1 are called indexes.
- List elements also can be accessed using names other than index. This is the easy way to access list elements.
- We use names() function to give names to the list elements.
- Below line explains how to assign names to the list elements.
  - l1 = list(c(1,2,3), c(10,"20",30))
  - names(l1) = c('1st','2nd')

(Continued) Lists and List Indexing

- Accessing list element using index as follows:
  - l1[[1]] #1st element
  - l1[[2]]#2nd element
- To access elements using names, we must use '$' symbol with list variable
- Accessing list element using names
  - l1$`1st`
  - l1$`2nd`

Adding and Deleting the elements of lists

- We can add, delete and modify elements of a list at any time
- Below code explains how to add and delete elements from lists:

```
l1 = list(10,20,30)
l2 = list('shiva','kumar','sanjay')
print(l1) #before adding element
l1[4] = 100
print(l1) #after adding element
l1[[2]] = 200
print(l1) #after modifying element
```

(Continued) Adding and Deleting the elements of lists

- Below code explains how to add and delete elements from lists:

```
print(l2) #before deleting element
l2[[1]] = NULL
print(l2) #after deleting element
l2[[2]] = 'nagalli'
print(l2) #after modifying element
```

Attributes

- Attributes() is a R built-in function to access and modify attributes of an R Object
- R Object may be list, matrix or any other R object
- Below code explains how to retrieve and delete attributes from a list:

```
l1 = list(c(1,2,3), c(10,"20",30))
names(l1) = c('1st','2nd')
attributes(l1)


attributes(l1) <- NULL #this deletes existing attributes
attributes(l1)   #after modifying
```

Factors

- Factors are used to categorise the data and store it as levels.
- Levels are nothing but groups with the same values.
- It is very useful if data has a unique set of values.
- We factor() function to create factors.
- Syntax as follows:

```
factor(var1)
```

where var1 is the vector variable

Factors

- Following code explains how to create factors:

```
genders <- c('Male', 'Female', 'Female', 'Female', 'Male')
print(genders)
genders <- factor(genders)
print(genders) #after applying factor function
attributes(genders)
attributes(genders)$`levels`
```

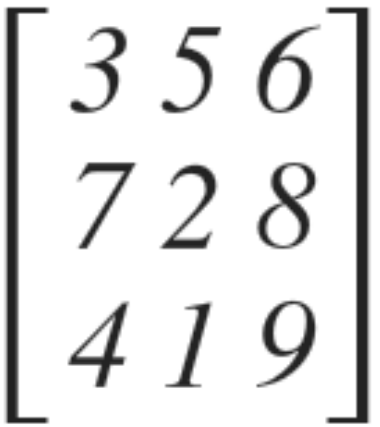
Factors

- We can generate a factor using gl() function
- gl() function as 3 parameters:
  - n is an integer giving the number of levels.
  - k is an integer giving the number of replications.
  - label is a vector of labels for the resulting factor levels.
- Below code snippet generates a factor with 2 levels and 50 data values

```
gender = gl(2,25, labels = c('Male', 'Female'))
print(gender)
```

Matrices and Arrays

- Matrix is a two-dimensional data type, in which data/information arranged in rows and columns as shown in right side image/picture
- We use matrix() function in R to define a matrix
- A prototype of matrix() function as follows
  - matrix(data, nrow, ncol, byrow, dimnames).



(Continued) Matrices and Arrays

Below code defines two matrices with dimension 3x3

```
m1 = matrix(data=c(1:9), nrow = 3)
m2 = matrix(data=c(10:18), nrow = 3)
m1
m2
```

The output of this code given in note section

Matrix Indexing

Below code show how to select elements from a matrix

```
m1 = matrix(data=c(1:9), nrow = 3)
m1[1,2]          #this prints 1st row 2nd column value
m1[2,2]          #this prints 2nd row 2nd column value
m1[1,]           #this prints 1st row all values
m1[,1]           #this prints 1st column all values
m1[1,2:3]        #this prints 1st row 2nd, 3rd column values
m1[1,1:3]        #this prints 1st row 1st, 2nd, 3rd column values
m1[1,2:2]        #this prints 1st row 2nd column value
```

Matrices Calculation

Below performs matrix addition, subtraction, multiplication and division

```
m1 = matrix(data=c(1:9), nrow = 3)
m2 = matrix(data=c(10:18), nrow = 3)
m1 + m2
m1 - m2
m1 * m2
m1 / m2
```

The output of this code given in note section

Generating Covariance Matrix

- Covariance is a measure of how changes in one variable are associated with changes in a second variable. Specifically, covariance measures the degree to which two variables are linearly associated.
- We use cov() function to generate a covariance matrix.
- Covariance matrix is helpful in examining the linear association ship between two variables.
- Understanding a covariance matrix will be much helpful in principal component analysis.
- cor() compute the variance of x and the covariance or correlation of x and y if these are vectors.
- If x and y are matrices then the covariance's (or correlations) between the columns of x and the columns of y are computed.

Matrices and Arrays

- An Array is a collection of data similar to a matrix, but have more than two dimensions.
- We use array() function in R to define an array.
- Prototype of array() function as follows
  - array(data, dim)
- It takes vectors as input and uses the values in the dim parameter to create an array.

Matrices & Arrays

- Following R script creates a array with 3 matrices:

```
a1 = array(1:12, dim = c(3,2,2))
print(a1)
```

- Following R script explains how to access array elements:

```
print(a1[,,2])
print(a1[1,1,1])
print(a1[1,1,])
print(a1[,1,])
```

Generating Covariance Matrix

Following code explains how to generate covariance matrix:

```
a <- c(1,2,3)

b <- c(2,3,5)

c <- c(3,5,5)


#create matrix from vectors

M <- cbind(a,b,c)

M_cov <- cov(M)

M_cov
```

Data frame – creating and coercion

- Data frame is a collection of vectors in 2D format.
- Side image is an example of a data frame.
- Simply a tabular data is the data frame.
- Data frame is being used for storing data tables. It is a list of vectors of equal length.
- Characteristics of a data frame:
  - The column names should be non-empty.
  - The row names should be unique.
  - The data stored in a data frame can be of numeric, factor or character type.
  - Each column should contain the same number of data items

	var1	var2	var3	var4
1	1	Hydrogen	H	
2	2	Helium		
3	3	Lithium		
4				

(Continued) Data frame – creating and coercion

- To create data frame we must use data.frame() function
- Following R code snippet creates a data frame with name studs:

```
studs <- data.frame(
  sno = c(100,200,300,400,500),
  sname = c("Rick","Dan","Michelle","Ryan","Gary"),
  marks = c(623.3,515.2,611.0,729.0,843.25),
  stringsAsFactors = FALSE
)
print(studs)
```

(Continued) Data frame – creating and coercion

- We use head() and tail() functions to retrieve top or bottom rows from the data frame
- head()
  - This function default returns top 5 rows from the data set.
  - It has parameter n, if set n value, it returns n rows if n is positive, otherwise, it returns all rows – n from the top if n is negative.
- tail()
  - This function default returns bottom 5 rows from the data set.
  - It has parameter n, if set n value, it returns n rows if n is positive, otherwise, it returns all rows – n from the bottom if n is negative.

(Continued) Data frame – creating and coercion

- We use str() function to find variables information from a data frame, which includes variable names, their data types and their values.
- Following code prints all variables from a stud data frame

  str(stud)
- Below image is the output for the above code:

```
> str(studs)
'data.frame':   5 obs. of  3 variables:
 $ sno  : num  100 200 300 400 500
 $ sname: chr  "Rick" "Dan" "Michelle" "Ryan" ...
 $ marks: num  623 515 611 729 843
>
```

(Continued) Data frame – creating and coercion

- Data frame coercion:
  - A different data types can be converted to data frame using as.data.frame() function command.
- Not all the data types will be supported to coerce the type to data.frame format.
- **For example**, iris numerical variables converted to distance matrix. Distance data types cannot be coerced to the data frame. Hence, first distance coerced to the matrix by the command as.matrix and then coerced to data.frame matrix.

```
Console  Terminal
> distance <- dist(iris[, -5])
> distance_mat <- as.matrix(distance)
> distance_df <- as.data.frame(distance_mat)
> distance_df[1:5, 1:5]
          1         2        3         4         5
1 0.0000000 0.5385165 0.509902 0.6480741 0.1414214
2 0.5385165 0.0000000 0.300000 0.3316625 0.6082763
3 0.5099020 0.3000000 0.000000 0.2449490 0.5099020
4 0.6480741 0.3316625 0.244949 0.0000000 0.6480741
5 0.1414214 0.6082763 0.509902 0.6480741 0.0000000
> class(distance_df)
[1] "data.frame"
> |
```

Combining Data Frames

- We can combine data frames in two ways:
  - Row-wise
    - Data frames must have the same number of a column with the same data types
  - Column-wise
    - Data frames must have a common column
- To combine in row-wise, we use rbind() function.
- To combine in column wise(merging), we use merge() function.

Combining Data Frames

Following code snippet combines data frame in row wise:

```
studs1 <- data.frame(
  sno = c(100,200),
  sname = c("Shiva","Kumar"),
  stringsAsFactors = FALSE
)

studs2 <- data.frame(
  sno = c(300,400),
  sname = c("nagalli","gopi"),
```

Combining Data Frames

Following code snippet combines data frame in row wise: (Continued…)

```
stringsAsFactors = FALSE
)

studs <- rbind(studs1, studs2)

print(studs)
```

Combining Data Frames

Following code snippet merges data frame in column wise:

```
studs1 <- data.frame(
  sno = c(100,200),
  sname = c("Shiva","Kumar"),
  stringsAsFactors = FALSE
)

studs2 <- data.frame(
  sno = c(100,200),
  marks = c(500,560),
```

Combining Data Frames

Following code snippet merges data frame in column wise: (Continued…)

```
stringsAsFactors = FALSE
)

studs <- merge(studs1, studs2, by = "sno")

print(studs)
```

Applying functions: lapply( ) and sapply( ) on data frames

apply() function:

- The apply() family pertains to the R base package and is populated with functions to manipulate slices of data from matrices, arrays, lists and data frames in a repetitive way. These functions allow crossing the data in a number of ways and avoid explicit use of loop constructs. They act on an input list, matrix or array and apply a named function with one or several optional arguments

The called function could be:

- An aggregating function, like for example the mean, or the sum (that return a number or scalar).
- Other transforming or sub-setting functions.
- Other vectorised functions, which return more complex structures like lists, vectors, matrices and arrays.

(Continued) Applying functions: lapply( ) and sapply( ) on data frames

apply() function:

- Let us start with the godfather of the family, apply(), which operates on wide variety of data types.
- The R base manual tells you that it is called as follows:
- apply(X, MARGIN, FUN.)

(Continued) Applying functions: lapply( ) and sapply( ) on data frames

- Following code generates squares of each element in the array:

```
l1 <- array(1:5, dim = c(1,5))
l2 <- apply(l1, 1, function(x) x*x)
print(l2)
```

- Following code generates sum of rows in the matrix:

```
l1 <- array(1:9, dim = c(3,3))
l2 <- apply(l1, 1, sum)
print(l1)
print(l2)
```

(Continued) Applying functions: lapply( ) and sapply( ) on data frames

lapply() function:

- This function we can use with lists to apply functions to the list emelents
- Does not require dimension(margin)
- Following code generates cubes of each element in the list:

```
l1 <- list(1:4)
l2 <- lapply(l1, function(x) x*x*x)
print(l1)
print(l2)
```

(Continued) Applying functions: lapply( ) and sapply( ) on data frames

sapply() function:

- This is wrapper function to the lapply() function.
- This returns be default vector or matrix
- Following code adds 10 to each element in the list and returns matrix:

```
l1 <- list(1:4)
l2 <- sapply(l1, function(x) x+10)
print(l1)
print(l2)
```

(Continued) Applying functions: lapply( ) and sapply( ) on data frames

sapply() function:

- If we want to return same type of its variable, assign “FALSE” to simplify parameter
- Following code adds 10 to each element in the list and returns list:

```
l1 <- list(1:4)
l2 <- sapply(l1, simplify = FALSE, function(x) x+10)
print(l1)
print(l2)
```

Arithmetic and Relational operators

- Arithmetic Operators are used to performing arithmetic operations
- Below table shows the Arithmetic Operators:

Operator	Description
+	Performs Addition
–	Performs Subtraction
*	Performs Multiplication
/	Performs Division
%%	Performs Modulus
^	Performs Power

Arithmetic and Relational operators

- Relational Operators are used to compare two variables
- Below table shows the Relational Operators:

Operator	Description
>	Greater than
<	Lesser then
>=	Greater than or equal
<=	Lesser then or equal
==	Equals
!=	Not equals

Control Statements

- To control a flow of execution of R code, we use control statements
- Control statements are two types:
  - Conditional statements
  - Looping statements
- Conditional statements:
  - These statements will execute once if the condition is true
- Looping statements:
  - These statements will execute more than once if the condition is false then they stop execution

Control Statements

- R has following conditional statements:
  - If
  - If-else
  - Switch
- R has following looping statements:
  - Repeat
  - While
  - For

If statement

- If statement executes only condition is true
- Syntax as follows:

```
if(condition) {
  // statement(s)
}
```

- Following code prints a is greater than 10, if a is more than 10, otherwise nothing:

```
a = 20
if(a>10) {
  print("a is greater than 10")
}
```

If-else statement

- Similar like if but execute else part if condition is false
- Syntax as follows:

```
if(condition) {
  // statement(s)
} else {
  // statement(s)
}
```

If-else statement

- Following code prints a is greater than 10, if a is more than 10, otherwise it prints smaller or equal to 10:

```
a = 10
if(a>10) {
  print("a is greater than 10")
} else {
  print("a is smaller or equals to 10")
}
```

Switch statement

- It is similar to multiple if-else statements, but its faster than that.
- It checks the value provided and executes the corresponding case.
- Syntax as follows:

```
switch(expression,
    case1,
    case2,
    case3,
    ....)
```

Switch statement

The following rules apply to a switch statement:

- If the value of an expression, is not a character string it is coerced to integer.
- We can have any number of case statements within a switch.
- If an expression evaluates to a character string then that string is matched (exactly) to the names of the elements.
- If there is more than one match, the first matching element is returned.
- No Default argument is available.
- In the case of no match, if there is an unnamed element of its value is returned.

Switch statement

- Following code snippet is example for switch:

```
x <- switch(
  3,
  "first",
  "second",
  "third",
  "fourth"
)
print(x)
```

Switch statement

- Following code snippet is example for switch:

```
x <- switch(
  "shiva",
  "shiva" = "This is shiva",
  "sanjay" = "This is sanjay",
  "kumar" = "This is kumar",
  "Wrong name"
)
print(x)
```

Repeat loop

- This statement executes statement in-finite
- To exit from the loop we must include condition inside it and exit
- Syntax as follows:

```
repeat {
   commands
   if(condition) {
      break
   }
}
```

Repeat loop

- The following code print sum from 1 to 10:

```
i = 1
sum = 0
repeat {
  sum = sum + i
  i = i + 1
  if(i == 11)
    break
}
print(sum)
```

While loop

- While loop executes statements till condition is true
- Syntax as follows:

```
while (condition) {
  statements
}
```

While loop

- Following code prints 10 to 1:

```
n = 10
while(n>0) {
  print(n)
  n = n -1
}
```

For loop

- For loop can be used to iterate items from a list
- This for is different from c programming for
- Syntax as follows:

```
for (value in list) {
  statements
}
```

For loop

- Following code prints 1 to 10:

```
for(i in 1:10) {
  print(i)
}
```

- Following code prints square of each element in the list:

```
l1 = list(2, 4, 5, 10)
for(x in l1) {
  print(x*x)
}
```

Self Assessment Question

1. Select correct statement to create a list

   a. l1 = new list(10,20,30,40)

   b. l1(10,20,30,40)

   c. l1 = list(10,20,30,40)

   d. l1 = c(10,20,30,40)

Answer: l1 = list(10,20,30,40)

Self Assessment Question

2. Which is the data type is used to represent tabular data in R Programming.

a. Data Frame
b. List
c. Vector
d. Matrices

Answer: Data Frame

Self Assessment Question

3. What will be the output of fallowing code?

```
m1 = matrix(1:9, nrow = 3)
print(m1[2,5])
```

a. 3
b. 2
c. 5
d. Error

Answer: Error

Self Assessment Question

4. Select correct statement to create data frame emp

a. emp = data.frame(eno=c(100,200))
b. emp = data.frame(c(100,200))
c. emp = data.frame(eno=100)
d. All of the above

Answer: All above

Self Assessment Question

5. Which one of the given option is used to access the elements in matrix?

a. ( )

b. { }

c. | |

d. [ ]

Answer: []

Self Assessment Question

6. What is coercion?

a. Converting bigger data to smaller data
b. Converting smaller data to bigger data
c. Converting one type to other type
d. None of the above

Answer: Converting one type to other type

Self Assessment Question

7. Using following list, which statement prints 31?

```
l1 = list(10,20,c(30,31),40)
```

a. l1[3][2]

b. l1[[3]][2]

c. l1[3,2]

d. l1[,2]

Answer: l1[[3]][2]

Self Assessment Question

8. Using following list, which statement prints 10?

```
l1 = list(10,20)
names(l1) = c('1st')
```

a. l1['1st']

b. l1$'1st'

c. l1[1]

d. All of the above

Answer: All of the above

Self Assessment Question

9. What is factor?

a. Factor is a group of values with levels
b. Factor is alternative for array of strings
c. Factor is a group of same values without levels
d. None of the above

Answer: Factor is a group of values with levels

Self Assessment Question

10. Which are the correct statements about combing data frames?

a. Data frames must have same number of variables

b. Data frames must have common variable

c. Data frames must have different number of variables

d. Data frames must have different number of variables and one common variable

Answer: a, b & d

Self Assessment Question

11. Which one of the given command is used to calculate element wise matrix multiplication?

a. A * B

b. A %% B

c. A %*% B

d. None of the above

Answer: A * B

Self Assessment Question

12. Which one of the given syntax is used to calculate the covariance matrix of the data set?

a. Sigma

b. Covariance

c. Covar

d. Cov

Answer: Cov

Self Assessment Question

13. Which one of the given options is the output for the R code ***cov(iris)***?

a. Covariance of iris data set

b. Covariance of first four variables

c. Correlation of iris data set

d. Error

Answer: Error

Self Assessment Question

14. In list, elements will be accessed through _____

a. [ ]

b. ( )

c. $

d. { }

Answer: a and c

Self Assessment Question

15. In data frame, variables will be accessed through _____.

a. [ ]

b. ( )

c. $

d. { }

Answer: a and b

Self Assessment Question

16. What is the output of the following code?

```
if(0){
  print("FALSE")
}
```

a. FALSE

b. Nothing

c. TRUE

d. Error

Answer: Nothing

Self Assessment Question

17. What is the output of the following code?

```
switch ('dist',
  'fail' = 'Student failed',
  'pass' = 'Student Passed',
  'distn' = 'Student got Distinction',
  'Student got A+'
 )
```

a. Student failed

b. Student got Distinction

c. Student got A+

d. Error

Answer: Student got A+

Self Assessment Question

18. What is the output of the following code?

```
i = 10
for(i in 1:10) {
  print(i)
}
```

a. Prints 1 to 10

b. Prints only 10

c. Prints nothing

d. Error

Answer: Prints 1 to 10

Self Assessment Question

19. Which one of the given option is the output for the R code Snippet?

```
i = 1;sum = 0
repeat {
  sum = sum + i;  i = i + 1
}
print(sum)
```

a. 10

b. 1

c. 0

d. Infinite loop

Answer: Infinite loop

Self Assessment Question

20. Imagine student data frame has 20 rows, what is the output of the following code?

tail(stud, n = -19)

a. Prints rows from 1 to 19

b. Prints only 1st row

c. Prints only last row

d. None of the above

Answer: Prints only last row

Self Assessment Question

21. Imagine student data frame has 20 rows, what is the output of the following code?

head(stud, n = -19)

a. Prints rows from 2 to 20

b. Prints only 1st row

c. Prints only last row

d. None of the above

Answer: Prints only 1st row

Self Assessment Question

22. Imagine student data frame has 20 rows, which of the following statement prints the 10^{th} row?

a. tail(head(stud, n = 10), n = -9)

b. head(tail(stud, n = -9), n = 1)

c. tail(head(stud, n = -9), n = 10)

d. head(tail(stud, n = 1), n = 9)

Answer: a & b

Self Assessment Question

23. What is the output of the following code?

```
emp = data.frame(ename = c('shiva', 'kumar', 'nagalli'))
class(emp$ename)
```

a. Character

b. Levels

c. Factors

d. None of the above

Answer: Factors

Self Assessment Question

24. Which one of the given options is the output for the R Code snippet?

```
lapply(list(1:5), function(x) x+x-4+4-x)
```

a. 1 2 3 4 5

b. 5 4 3 2 1

c. 15

d. 0

Answer: 1 2 3 4 5

Self Assessment Question

25. Imagine 3x3 matrix m1 has values from 1 to 9, which statement prints sum of all numbers?

a. apply(m1, 1, sum)

b. lapply(list(apply(m1, 1, sum)), sum)

c. sapply(list(apply(m1, 2, sum)), sum)

d. apply(m1, 2, sum)

Answer: b and c

Summary

- R is an open-source statistical computing programming language software, maintained by R Development core team.
- R programming is everything about objects. All variables, functions statements will be used as objects in R.
- R Studio is the important and most familiar IDE and code editor for R programming language.
- RStudio is useful in installing packages and writing code using intellisence suggestions.
- R Statistical packages are hosted in CRAN. Anyone can download by setting a convenient mirror host.
- Vectors store the same kind of information, and we use c() function to create vectors.
- Matrix stores two-dimensional values, we use matrix() function to create matrices.

Assignment

1. Write R script to find given number is prime or not using repeat loop.
2. Write R script to perform multiplication of two matrices using for loop.
3. Write R script to find biggest number from a matrix using for loop.
4. Write R script to find average salary of emp data frame using while loop.
5. Write R script to find 10^{th} emp name who is getting more salary from the top.

Document Links

Topics	URL
Introduction to Different Data Types	https://www.tutorialspoint.com/r/r_data_types.htm
Coercion	https://www.oreilly.com/library/view/r-in-a/9781449358204/ch05s08.html
Factors	https://www.tutorialspoint.com/r/r_factors.htm
Matrices & Arrays	https://www.tutorialspoint.com/r/r_matrices.htm https://www.tutorialspoint.com/r/r_arrays.htm
Data frame – creating & coercion	https://www.tutorialspoint.com/r/r_data_frames.htm
Arithmetic and Relational operators	https://www.tutorialspoint.com/r/r_operators.htm
Switch statement	https://www.tutorialspoint.com/r/r_switch_statement.htm
Repeat loop	https://www.tutorialspoint.com/r/r_repeat_loop.htm
While loop	https://www.tutorialspoint.com/r/r_while_loop.htm
For loop	https://www.tutorialspoint.com/r/r_for_loop.htm

Video Links

Topics	URL
Introduction to Different Data Types	https://www.youtube.com/watch?v=Is25nE5RyG8
Coercion	https://www.youtube.com/watch?v=Lrg3TOeixu8
Factors	https://www.youtube.com/watch?v=xkRBfy8_2MU
Matrices & Arrays	https://www.youtube.com/watch?v=RbjHQNxOYt0 https://www.youtube.com/watch?v=oYhP2WZYpBY
Data frame – creating & coercion	https://www.youtube.com/watch?v=FVSlAk24jTM
Switch statement	https://www.youtube.com/watch?v=ywKAsN5DlHU
Repeat loop	https://www.youtube.com/watch?v=Z_pkI1QkTo0
While loop	https://www.youtube.com/watch?v=4vo_XHc5pxg
For loop	https://www.youtube.com/watch?v=h987LWDvqlQ

E-Book Links

URL
http://diytranscriptomics.com/Reading/files/The%20Art%20of%20R%20Programming.pdf