Basic_R_Commands

Venkat

2024-01-28

1. CTRL+L is used to clear the console logs
2. To list the alphabets –> letters
3. To list the alphabets in uppercase –> LETTERS
4. To list the months name in an year (fully qualified name)–> month.name
5. To list the months name in an year (short form abc) –> month.abb

Data Types:

1. INTEGER
2. NUMERIC
3. CHARACTER
4. LOGICAL
5. COMPLEX

Type checking and type conversion

```
a <- 10
# Check the datatype -> typeof(object_name)
print(typeof(a))
```

```
## [1] "double"
```

```
# To verify the data --> is_object_name()
print(is.double(a))
```

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

```
#To convert the datatype to another type --> as.data_type(object_name)

print(as.integer(a))
```

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

Objects:

1. vector is an Ordered collection of basic datatypes. All the elements of a vector are of same datatype.

```
# Here, ages is a numerical vector.
ages <- c(21, 23, 12, 26)
```

```
# List --> Ordered collection of objects. It can contain a vector, matrix, complex vector, character ar

ids = c(1,2,3,4)
emp.names = c("venkat", "Sairam", "Venkata Sairam Yanamandra")

emp = list(id = ids, names=emp.names)
print(emp$names) ## Only prints the emp names.
```

```
## [1] "venkat"                    "Sairam"
## [3] "Venkata Sairam Yanamandra"
```

```
print(emp)## Prints the empyloyee details
```

```
## $id
## [1] 1 2 3 4
##
## $names
## [1] "venkat"                    "Sairam"
## [3] "Venkata Sairam Yanamandra"
```

```
print(emp[1]) # prints the employee ids
```

```
## $id
## [1] 1 2 3 4
```

```
print(emp[[1]][1]) ## prints the first employee id.
```

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

```
emp['location'] = "Visakhapatnam" ## Creates a new element location with given value.
```

```
#To concatenate two lists use: c(list2, list1)
list1 <- list(c(1, 2, 3,4), c("a", "b", "c", "d"))

list2 <- c(list1, emp)
print(list2)
```

```
## [[1]]
## [1] 1 2 3 4
##
## [[2]]
## [1] "a" "b" "c" "d"
```

```
##
## $id
## [1] 1 2 3 4
##
## $names
## [1] "venkat"                    "Sairam"
## [3] "Venkata Sairam Yanamandra"
##
## $location
## [1] "Visakhapatnam"
```

Dataframe: used to store the tabular data.

```
list1 <- c(1, 2,3,4)
list2 <- c("a", "b", "c", "d")
data.frame(list1, list2)
```

```
##   list1 list2
## 1     1     a
## 2     2     b
## 3     3     c
## 4     4     d
```

Dataframes can also be created by importing the data from a text file.

```
data <- read.table(file = "input_data.txt", sep=" ", header=TRUE)
```

```
## Warning in read.table(file = "input_data.txt", sep = " ", header = TRUE):
## incomplete final line found by readTableHeader on 'input_data.txt'
```

```
#using subset command, we can filter the dataframe.
print(subset(data, name=="venkat"))
```

```
##   id   name            email salary
## 3  3 venkat venkat@gmail.com 122412
```

```
# printing the columns, rows, rows and columns
data[c(1:2),] # This commands prints the first two rows and all the columns.
```

```
##   id    name                     email salary
## 1  1 Venkata venkatasairam.y@gmail.com  10000
## 2  2     Abc             Abc@gmail.com  21000
```

```
data[, c(1:2)] # prints all the rows and first two columns data
```

```
##   id    name
## 1  1 Venkata
## 2  2     Abc
## 3  3  venkat
## 4  4 praveen
```

```
data[c(2), c(2,3)] # prints row-2 and columns-2,3 values
```

```
##   name         email
## 2  Abc Abc@gmail.com
```

```
# Adding extra row and column to the existing data frame
# use rbind for rows and cbind for columns
data <- rbind(data, c(5, "lohith", "lohith@gmail.com", 21212))

print(data)
```

```
##   id    name                    email salary
## 1  1 Venkata venkatasairam.y@gmail.com  10000
## 2  2     Abc             Abc@gmail.com  21000
## 3  3  venkat          venkat@gmail.com 122412
## 4  4 praveen         praveen@gmail.com  85000
## 5  5  lohith          lohith@gmail.com  21212
```

```
# delete the rows, columns, and rows and columns.
data <- cbind(data, LOCATION=c("vizag", "hyd", "vizag", "vizag", "yanam"))

print(data)
```

```
##   id    name                    email salary LOCATION
## 1  1 Venkata venkatasairam.y@gmail.com  10000    vizag
## 2  2     Abc             Abc@gmail.com  21000      hyd
## 3  3  venkat          venkat@gmail.com 122412    vizag
## 4  4 praveen         praveen@gmail.com  85000    vizag
## 5  5  lohith          lohith@gmail.com  21212    yanam
```

Casting, melting, recasting the data frames

Joining two data frames

```
# left join
data1 <- data[, c(2,3)]
data2 <- data[c(2,4), ]
data2
```

```
##   id    name             email salary LOCATION
## 2  2     Abc     Abc@gmail.com  21000      hyd
## 4  4 praveen praveen@gmail.com  85000    vizag
```

```
#left_join(dataframe_1, dataframe_2, by="column-name")
new_df_left_join <- left_join(data1, data2, by="email")
print(new_df_left_join)
```

```
##    name.x                     email   id  name.y salary LOCATION
## 1 Venkata venkatasairam.y@gmail.com <NA>    <NA>   <NA>     <NA>
## 2     Abc             Abc@gmail.com    2     Abc  21000      hyd
## 3  venkat          venkat@gmail.com <NA>    <NA>   <NA>     <NA>
## 4 praveen         praveen@gmail.com    4 praveen  85000    vizag
## 5  lohith          lohith@gmail.com <NA>    <NA>   <NA>     <NA>
```

```
new_df_right_join <- right_join(data1, data2, by="email")
print(new_df_right_join)
```

```
##    name.x             email id  name.y salary LOCATION
## 1     Abc     Abc@gmail.com  2     Abc  21000      hyd
## 2 praveen praveen@gmail.com  4 praveen  85000    vizag
```

Precedence in R:

1. Brackets
2. Exponent
3. Division
4. Multiplication
5. Addition
6. Subtraction

Matrix operations:

```
matrix(c(1:9), nrow=3, ncol=3, byrow=TRUE)
```

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

Creating a SCalar matrix:

```
#Syntax: matrix(scalar_value, no_of_rows, no_of_columns)
matrix(3, 2, 4)
```

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

Creating a diagonal matrix:

```r
# Syntax: diag((vector_of_diagonal_elements), no_rows , no_of_columns )
diag(c(1,2,3), 3, 3)
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    2    0
## [3,]    0    0    3
```

Matrix Metrics:

1. dim(A) returns the no of rows and columns in the matrix.
2. nrow(A) returns the no of rows in the matrix
3. ncol(A) returns the no of columns in the matrix
4. length(A) returns the number of elements present in the matrix.
5. rownames(A) Strings can be assigned as names of rows.

```r
A = matrix(c(1:9), nrow = 3, ncol=3, byrow = T)
rownames(A) = c("A", "B", "C")
colnames(A) = c("id", "name", "email")
A
```

```
##   id name email
## A  1    2     3
## B  4    5     6
## C  7    8     9
```

```r
## Accessing a particular row of a matrix
A[1, ]
```

```
##    id  name email
##     1     2     3
```

```r
## Accessing the cols of a matrix
A[, 2]
```

```
## A B C
## 2 5 8
```

```r
## ACcessing the particular entry of a matrix
A[1,2]
```

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

```r
## Access eveything except one column
A[, -2]
```

```
##   id email
## A  1     3
## B  4     6
## C  7     9
```

```
## Access everything except one row
A[-1, ]
```

```
##   id name email
## B  4    5     6
## C  7    8     9
```

```
## Use rbind() for concatenating the rows to the existing matrix.

B = matrix(c(10:12),nrow = 1, byrow = T)
B
```

```
##      [,1] [,2] [,3]
## [1,]   10   11   12
```

```
rbind(A, B)
```

```
##   id name email
## A  1    2     3
## B  4    5     6
## C  7    8     9
##   10   11    12
```

```
##Use cbind() for concatenating the columns to the existing matrix.

C = matrix(c(10, 11, 12))

cbind(A, C)
```

```
##   id name email
## A  1    2     3 10
## B  4    5     6 11
## C  7    8     9 12
```

Algebraic operations on matrix:

```
A + A
```

```
##   id name email
## A  2    4     6
## B  8   10    12
## C 14   16    18
```

```
A - A
```

```
##   id name email
## A  0    0     0
## B  0    0     0
## C  0    0     0
```

```r
A * A # Element wise multiplication
```

```
##   id name email
## A  1    4     9
## B 16   25    36
## C 49   64    81
```

```r
A / A # Element wise division
```

```
##   id name email
## A  1    1     1
## B  1    1     1
## C  1    1     1
```

```r
A %*% A # Actual matrix multiplication operation.
```

```
##    id name email
## A  30   36    42
## B  66   81    96
## C 102  126   150
```

##Looping over functions:

```r
## Apply function:
## applies a given function over the margins of a given array.
## Syntax: apply(array, margins, function_to_be_executed)

A
```

```
##   id name email
## A  1    2     3
## B  4    5     6
## C  7    8     9
```

```r
apply(A, 1, sum) # calculates the sum for each row in the matrix.
```

```
##  A  B  C
##  6 15 24
```

```r
apply(A, 2, sum) # calculates the sum for each col in the matrix.
```

```
##    id  name email
##    12    15    18
```

lapply function:

```r
## lapply is used to apply a function over a list.
## lapply always returns a list of the same length as the input.
A
```

```
##   id name email
## A  1    2     3
## B  4    5     6
## C  7    8     9
```

```r
B = matrix(c(10:18), nrow=3, ncol=3, byrow=TRUE)
C = list(A, B)
lapply(C, det)
```

```
## [[1]]
## [1] 6.661338e-16
##
## [[2]]
## [1] 0
```

Sequence function in R:

```r
# Creates an equi-spaced points between 'from' and 'to'
# Syntax: seq(from, to, by, length)
# from: starting number
# to: Ending number
# by: increment or decrement number
# length: number of elements required in the output.

seq(from=1, to=10, by = 2)
```

```
## [1] 1 3 5 7 9
```

```r
seq(from=1, to=15, length=5)
```

```
## [1]  1.0  4.5  8.0 11.5 15.0
```

```r
sum = 0
for( i in seq(1, 100, 10)){
  sum = sum + i
  print(c(i, sum))
}
```

```
## [1] 1 1
## [1] 11 12
## [1] 21 33
## [1] 31 64
## [1]  41 105
## [1]  51 156
```

```
## [1]  61 217
## [1]  71 288
## [1]  81 369
## [1]  91 460
```