Creating Vectors: Using c() function

```
x<- c(1,2,3,4,5)#creates a vector named x
>x #prints the vector x
```

Output

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

---

```
# Using the colon(:) operator
x<-7:-10
x
```

**Output**

- [1] 7 6 5 4 3 2 1 0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -10

---

```
# Using the seq() function
x=seq(1,6,by=0.5)
x
class(x)
```

**Output**

[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0
[1] “numeric”

**Example:**

```
x<-seq(1,4,length.out=6)
x
class(x)
```

**Output**

- [1] 1.0 1.6 2.2 2.8 3.4 4.0 [1] "numeric"

---

A vector that contains numeric elements is known as a numeric vector.

**Example:**

```
x<-56.6
y<-c(10.1, 10.2, 33.2)
x
y
class(x)
class(y)
```

**Output**

[1] 56.6 [1] 10.1 10.2 33.2 [1] "numeric" [1] "numeric"

---

Integer vector

A vector that contains integer elements is known as an integer vector.

**Example:**

```
>x=as.integer(7)
> y=8L
> z=c(1,2,3,4,5)
> z=as.integer(z)
> p=c(1L,2L,3L,4L,5L)
>class(x)
class(y)
class(z)
class(p)
```

**Output**

[1] "integer" [1] "integer" [1] "integer" [1] "integer"

---

Character vector

A vector that contains character elements is known as a character vector.

**Example:**

```
> x='Computer'
>y="Science"
>z=65
>z=as.character(z)
>x
>y
>z
char_vec<-c(1,2,3,4,5)
char_vec<-as.character(char_vec)
char_vec1<-c("Learn","R","Programming")
char_vec
class(x)
class(y)
class(z)
class(char_vec)
class(char_vec1)
```

**Output**

[1] "Computer" [1] "science" [1]"65" [1] "1" "2" "3" "4" "5"

---

Logical vector

**Example:**

```
d<-as.integer(5)
e<-as.integer(6)
f<-as.integer(7)
g<-d>e
h<-e<f
g
h
log_vec<-c(d<e, d<f, e<d,e<f,f<d,f<e)
log_vec
class(g)
  class(h)
class(log_vec)
```

**Output**

[1] FALSE [1] TRUE [1] TRUE TRUE FALSE TRUE FALSE FALSE [1] "logical" [1] "logical" [1] "logical"

---

1) Indexing with integer vector

**Example:**

```
seq_vec<-seq(1,4,length.out=6)
seq_vec
seq_vec[2]
```

**Output**

[1] 1.0 1.6 2.2 2.8 3.4 4.0 [1] 1.6

---

2) Indexing with a character vector

**Example:**

```
char_vec<-c("Ram"=25,"Arun"=23,"Raja"=30)
char_vec
char_vec["Arun"]
```

**Output**

RamArunRaja 25 23 30Arun 23

---

**3) Indexing with a logical vector**

```
a<-c(1,2,3,4,5,6)
a[c(TRUE,FALSE,TRUE,TRUE,FALSE,TRUE)]
```

**Output**

[1] 1 3 4 6

---

Vector Operations

1.Combining vectors

```
x<-c(1,2,3,4,5,6)
y<-c("aaa","bbb","ccc","ddd","eee","fff")
z<-c(x,y)
z
```

```
[1] "1"   "2"   "3"   "4"   "5"   "6"   "aaa" "bbb" "ccc" "ddd" "eee" "fff"
```

---

2) Arithmetic operations

```
a<-c(1,3,5,7)
b<-c(2,4,6,8)
a+b
a-b
a/b
a%%b
```

```
[1]  3  7 11 15
[1] -1 -1 -1 -1
[1] 0.5000000 0.7500000 0.8333333 0.8750000
[1] 1 3 5 7
```

---

**Vector Element Recycling**

```
x=c(3,8,4,5,0,11)
 y=c(4,11)
# y becomes c(4,11,4,11,4,11)
 z=x+y
z
z=x-y
z
```

```
[1] 7 19 8 16 4 22
[1] -1 -3 0 -6 -4 0
```

---

**Vector Element Sorting**

```
x=c(56,76,32,12,-6,98,-87,45)
y=sort(x)
y
[1] -87 -6 12 32 45 56 76 [8] 98
# Sort the elements in the reverse order.
y=sort(x,decreasing=TRUE)
 y
```

[1] 98 76 56 45 32 12 -6 [8] -87

```
# Sorting character vectors.
x=c("Red","Blue","yellow","vioelt")
y=sort(x)
print(y)
z=sort(x,decreasing=TRUE)
print(z)
```

[1] "Blue"  "Red"  "vioelt" "yellow"
[1] "yellow" "vioelt" "Red"  "Blue"

---

Logical Index vector

```
a<-c(“aaa",”bbb",”ccc",”ddd",”eee",”fff")
b<-c(TRUE,FALSE,TRUE,TRUE,FALSE,FALSE)
a[b]
```

[1] “aaa” “ccc" “ddd"

---

Duplicate Index

**Example:**

```
q<-c(“aaa",”bbb",”ccc",”ddd",”eee",”fff")
q[c(2,4,4,3)]
```

[1] "bbb" "ddd" "ddd" "ccc"

---

Range Indexes

**Example:**

```
q<-c(“aaa",“bbb",“ccc",“ddd",“eee",“fff")
b<-q[2:5]
b
```

[1] "bbb" "ccc" "ddd" "eee"

---

7) Out-of-order Indexes

**Example:**

```
q<-c(“aaa",”bbb",”ccc",”ddd",”eee",”fff")
q[c(2,1,3,4,5,6)]
```

[1] "bbb" "aaa" "ccc" "ddd" "eee" [6] "fff"

---

8) Named vectors members

```
z=c(“R”, “Programming)
z
names(v) = c("First", "Second")
v
v["First"]
```

```
v[c("Second", "First")]
```

```
xvec<- c(5,6,7,8)
ychar_vec<-c("aaa","bbb","ccc","ddd")
zlogic_vec<-c(TRUE,FALSE,FALSE,TRUE)
o_list<-list(xvec,ychar_vec,zlogic_vec)
print(o_list)
```

```
[[1]]
[1] 5 6 7 8
[[2]]
[1] "aaa" "bbb" "ccc" "ddd"
[[3]]
[1]  TRUE FALSE FALSE  TRUE
```

List Creation

**Example 1:** Creating list with same data type

```
list_1<-list(1,2,3)
list_2<-list("aaa","bbb","ccc")
list_3<-list(c(1,2,3))
list_4<-list(TRUE,FALSE,TRUE)
print(list_1)
print(list_2)
print(list_3)
print(list_4)
```

**Example 2:** Creating the list with different data type

```
list_data<-list("Computer","Science",c(1,2,3,4,5),TRUE,FALSE,54.2,18L)
print(list_data)
```

Let see an example to understand how we can give the names to the list elements.

```
# Creating a list containing a vector, a matrix and a list.
list_data<- list(c("Rama","Arun","Sita"), matrix(c(40,80,60,70,90,80), nrow = 2),
list("BCA","MCA","B.tech"))
# Giving names to the elements in the list.
names(list_data) <- c("Students", "Marks", "Course")
# Show the list.
print(list_data)
```

```
$Students
[1] "Rama" "Arun" "Sita"
$Marks
```

```
     [,1] [,2] [,3]
[1,]   40   60   90
[2,]   80   70   80
$Course
$Course[[1]]
[1] "BCA"
$Course[[2]]
[1] "MCA"
$Course[[3]]
[1] "B.tech"
```

Accessing list elements

1.By using index

```
# Creating a list containing a vector, a matrix and a list.
list_data<- list(c("Rama","Arun","Sita"), matrix(c(40,80,60,70,90,80), nrow = 2),
list("BCA","MCA","B.tech"))
# Accessing the first element of the list.
print(list_data[1])
# Accessing the third element. The third element is also a list, so all its elements
will be printed.
print(list_data[3])
[[1]]
[1] "Rama" "Arun" "Sita"
[[1]]
[[1]][[1]]
[1] "BCA"
[[1]][[2]]
[1] "MCA"
[[1]][[3]]
[1] "B.tech"
```

2.By using names

```
# Creating a list containing a vector, a matrix and a list.
list_data<- list(c("Rama","Arun","Sita"), matrix(c(40,80,60,70,90,80), nrow = 2),
list("BCA","MCA","B.tech"))

# Giving names to the elements in the list.
names(list_data) <- c("Students", "Marks", "Course")

# Accessing the first element of the list
print(list_data["Student"])
```

```
print(list_data$Marks)
print(list_data)

$Students
[1] "Rama" "Arun" "Sita"

$Marks
     [,1] [,2] [,3]
[1,]   40   60   90
[2,]   80   70   80

$Course
$Course[[1]]
[1] "BCA"

$Course[[2]]
[1] "MCA"

$Course[[3]]
[1] "B.tech"
```

## Manipulation of list elements

```
# Creating a list containing a vector, a matrix and a list.
list_data<- list(c("Rama","Arun","Sita"), matrix(c(40,80,60,70,90,80), nrow
= 2),
list("BCA","MCA","B.tech"))

# Giving names to the elements in the list.
names(list_data) <- c("Student", "Marks", "Course")

# Adding element at the end of the list.
list_data[4] <- "Geeta"
print(list_data[4])

# Removing the last element.
list_data[4] <- NULL

# Printing the 4th Element.
print(list_data[4])

# Updating the 3rd Element.
```

```
        list_data[3] <- "Masters of computer applications"
        print(list_data[3])
[[1]]
[1] "Geeta"
$<NA>
NULL
$Course
[1] "Masters of computer applications"
```

---

Converting list to vector

```
        # Creating lists.
        list1=list(10:20)
        print(list1)

        list2=list(5:14)
        print(list2)

        # Converting the lists to vectors.
        v1=unlist(list1)
        v2=unlist(list2)

        print(v1)
        print(v2)

        result=(v1+v2)
        print(result)
[[1]]
 [1] 10 11 12 13 14 15 16 17 18 19 20
[1]  5  6  7  8  9 10 11 12 13 14
[1] 10 11 12 13 14 15 16 17 18 19 20
[1]  5  6  7  8  9 10 11 12 13 14
[1] 15 17 19 21 23 25 27 29 31 33 25
```

---

Merging Lists

**Example**

```
# Creating two lists.
Even_list <- list(2,4,6,8,10)
Odd_list <- list(1,3,5,7,9)

# Merging the two lists.
merged.list <- list(Even_list,Odd_list)
```

```
# Printing the merged list.
print(merged.list)
```

**Output:**

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

[[1]][[2]]
[1] 4

[[1]][[3]]
[1] 6

[[1]][[4]]
[1] 8

[[1]][[5]]
[1] 10

[[2]]
[[2]][[1]]
[1] 1

[[2]][[2]]
[1] 3

[[2]][[3]]
[1] 5

[[2]][[4]]
[1] 7

[[2]][[5]]
[1] 9
```

---

```
# Create two vectors of different lengths.
vector1=c(10,20,30)
vector2=c(5,6,7,8,9,10)

# Take these vectors as input to the array.
```

```
result<- array(c(vector1,vector2),dim = c(3,3,2))
print(result)
```

```
     [,1] [,2] [,3]
[1,]   10    5    8
[2,]   20    6    9
[3,]   30    7   10

, , 2

     [,1] [,2] [,3]
[1,]   10    5    8
[2,]   20    6    9
[3,]   30    7   10
```

---

Naming Columns and Rows

```
        #Creating two vectors of different lengths
        vec1 <-c(1,3,5)
        vec2 <-c(10,11,12,13,14,15)

        #Initializing names for rows, columns and matrices
        col_names<- c("Col1","Col2","Col3")
        row_names<- c("Row1","Row2","Row3")
        matrix_names<- c("Matrix1","Matrix2")

        #Taking the vectors as input to the array
        res <-
        array(c(vec1,vec2),dim=c(3,3,2),dimnames=list(row_names,col_names,matr
        ix_names))
        print(res)
```

```
Matrix1

     Col1 Col2 Col3
Row1    1   10   13
Row2    3   11   14
Row3    5   12   15

, , Matrix2

     Col1 Col2 Col3
Row1    1   10   13
Row2    3   11   14
```

Row3  5  12  15

Accessing array elements

```
# Create two vectors of different lengths.
vector1 <- c(5,9,3)
vector2 <- c(10,11,12,13,14,15)
column.names<- c("COL1","COL2","COL3")
row.names<- c("ROW1","ROW2","ROW3")
matrix.names<- c("Matrix1","Matrix2")

# Take these vectors as input to the array.
result<- array(c(vector1,vector2),dim = c(3,3,2),dimnames = list(row.names,
column.names, matrix.names))

# Print the third row of the second matrix of the array.
print(result[3,,2])

# Print the element in the 1st row and 3rd column of the 1st matrix.
print(result[1,3,1])

# Print the 2nd Matrix.
print(result[,,2])
```

**Output**

```
COL1 COL2 COL3
  3   12   15
[1] 13
    COL1 COL2 COL3
ROW1    5   10   13
ROW2    9   11   14
ROW3    3   12   15
```

- Manipulating Array Elements

```
#Creating two vectors of different lengths
vec1 <-c(1,3,5)
vec2 <-c(10,11,12,13,14,15)

#Taking the vectors as input to the array1
res1 <- array(c(vec1,vec2),dim=c(3,3,2))
print(res1)

#Creating two vectors of different lengths
vec1 <-c(8,4,7)
```

```
vec2 <-c(16,73,48,46,36,73)

#Taking the vectors as input to the array2
res2 <- array(c(vec1,vec2),dim=c(3,3,2))
print(res2)

#Creating matrices from these arrays
mat1 <- res1[,,2]
mat2 <- res2[,,2]
res3 <- mat1+mat2
print(res3)
```

output

```
     [,1] [,2] [,3]
[1,]    1   10   13
[2,]    3   11   14
[3,]    5   12   15

, , 2

     [,1] [,2] [,3]
[1,]    1   10   13
[2,]    3   11   14
[3,]    5   12   15


     [,1] [,2] [,3]
[1,]    8   16   46
[2,]    4   73   36
[3,]    7   48   73

, , 2

     [,1] [,2] [,3]
[1,]    8   16   46
[2,]    4   73   36
[3,]    7   48   73
```

```
>print(res3)
```

```
     [,1] [,2] [,3]
[1,]    9   26   59
[2,]    7   84   50
[3,]   12   60   88
```

---

Calculations across array elements

```
# Create two vectors of different lengths.
vector1 <- c(5,9,3)
vector2 <- c(10,11,12,13,14,15)

# Take these vectors as input to the array.
new.array<- array(c(vector1,vector2),dim = c(3,3,2))
print(new.array)

# Use apply to calculate the sum of the rows across all the matrices.
result<- apply(new.array, c(1), sum)
print(result)
```

```
[1] 56 68 60
```

---

Let's see an example to understand how the matrix function is used to create a matrix and arrange the elements sequentially by row or column.

```
#Arranging elements sequentially by row.
P <- matrix(c(5:16), nrow = 4, byrow = TRUE)
print(P)

# Arranging elements sequentially by column.
Q <- matrix(c(3:14), nrow = 4, byrow = FALSE)
print(Q)

# Defining the column and row names.
row_names = c("row1", "row2", "row3", "row4")
ccol_names = c("col1", "col2", "col3")

R <- matrix(c(3:14), nrow = 4, byrow = TRUE, dimnames = list(row_names,
col_names))
print(R)
output
```

```
>#Arranging elements sequentially by row.
>P <- matrix(c(5:16), nrow = 4, byrow = TRUE)
>print(P)
     [,1] [,2] [,3]
```

```
[1,]    5    6    7
[2,]    8    9   10
[3,]   11   12   13
[4,]   14   15   16
>
># Arranging elements sequentially by column.
>Q <- matrix(c(3:14), nrow = 4, byrow = FALSE)
>print(Q)
     [,1] [,2] [,3]
[1,]    3    7   11
[2,]    4    8   12
[3,]    5    9   13
[4,]    6   10   14
>
># Defining the column and row names.
>row_names = c("row1", "row2", "row3", "row4")
>ccol_names = c("col1", "col2", "col3")
>
>R <- matrix(c(3:14), nrow = 4, byrow = TRUE, dimnames = list(row_names,
col_names))
>print(R)
     Col1 Col2 Col3
row1    3    4    5
row2    6    7    8
row3    9   10   11
row4   12   13   14
```

---

Accessing matrix elements in R

```
# Defining the column and row names.
row_names = c("row1", "row2", "row3", "row4")
ccol_names = c("col1", "col2", "col3")
#Creating matrix
R <- matrix(c(5:16), nrow = 4, byrow = TRUE, dimnames = list(row_names,
col_names))
print(R)

#Accessing element present on 3rd row and 2nd column
print(R[3,2])

#Accessing element present in 3rd row
print(R[3,])
```

```
        #Accessing element present in 2nd column
        print(R[,2])
        output
># Defining the column and row names.
>row_names = c("row1", "row2", "row3", "row4")
>ccol_names = c("col1", "col2", "col3")
>#Creating matrix
>R <- matrix(c(5:16), nrow = 4, byrow = TRUE, dimnames = list(row_names,
col_names))
>print(R)
     Col1 Col2 Col3
row1    5    6    7
row2    8    9   10
row3   11   12   13
row4   14   15   16
>
>#Accessing element present on 3rd row and 2nd column
>print(R[3,2])
[1] 12
>
>#Accessing element present in 3rd row
>print(R[3,])
Col1 Col2 Col3
  11   12   13
>
>#Accessing element present in 2nd column
>print(R[,2])
row1 row2 row3 row4
   6    9   12   15
```

---

```
R <- matrix(c(5:16), nrow = 4,ncol=3)
S <- matrix(c(1:12), nrow = 4,ncol=3)

#Addition
sum<-R+S
print(sum)

#Subtraction
sub<-R-S
print(sub)
```

```
#Multiplication
mul<-R*S
print(mul)

#Multiplication by constant
mul1<-R*12
print(mul1)

#Division
div<-R/S
print(div)

> x <- matrix(c(5:16), nrow = 4,ncol=3)
> y <- matrix(c(1:12), nrow = 4,ncol=3)
> print(x)
     [,1] [,2] [,3]
[1,]    5    9   13
[2,]    6   10   14
[3,]    7   11   15
[4,]    8   12   16
> print(y)
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
>
> #Addition
> sum<-x+y
> print(sum)
     [,1] [,2] [,3]
[1,]    6   14   22
[2,]    8   16   24
[3,]   10   18   26
[4,]   12   20   28
>
> #Subtraction
> sub<-x-y
```

```
> print(sub)
     [,1] [,2] [,3]
[1,]    4    4    4
[2,]    4    4    4
[3,]    4    4    4
[4,]    4    4    4
>
> #Multiplication
>mul<-x*y
> print(mul)
     [,1] [,2] [,3]
[1,]    5   45  117
[2,]   12   60  140
[3,]   21   77  165
[4,]   32   96  192
>
> #Multiplication by constant
> mul1<-x*5
> print(mul1)
     [,1] [,2] [,3]
[1,]   25   45   65
[2,]   30   50   70
[3,]   35   55   75
[4,]   40   60   80
>
> #Division
> div<-x/y
> print(div)
         [,1]     [,2]     [,3]
[1,] 5.000000 1.800000 1.444444
[2,] 3.000000 1.666667 1.400000
[3,] 2.333333 1.571429 1.363636
[4,] 2.000000 1.500000 1.333333
```

How to create a factor?

```
# Creating a vector as input.
data <- c("aaa","bbb","ccc","bbb","eee","www","xxx","aaa","ccc","bbb","aaa")

print(data)
print(is.factor(data))

# Applying the factor function.
factor_data<- factor(data)

print(factor_data)
print(is.factor(factor_data))
```

```
>
>print(data)
 [1] "aaa" "bbb" "ccc" "bbb" "eee" "www" "xxx" "aaa" "ccc" "bbb" "aaa"
>print(is.factor(data))
[1] FALSE
>
>
>print(factor_data)
 [1] aaabbb ccc bbbeee www xxx aaa ccc bbbaaa
Levels: aaabbbccceee www xxx
>print(is.factor(factor_data))
[1] TRUE
```

---

Accessing components of factor

```
# Creating a vector as input.
data <- c("aaa","bbb","ccc","bbb","aaa","ddd","bbb","aaa","ddd","ccc","ddd")

# Applying the factor function.
factor_data<- factor(data)

#Printing all elements of factor
print(factor_data)

#Accessing 4th element of factor
print(factor_data[4])
```

```
#Accessing 5th and 7th element
print(factor_data[c(5,7)])

#Accessing all elemcentexcept 4th one
print(factor_data[-4])

#Accessing elements using logical vector
print(factor_data[c(TRUE,FALSE,FALSE,FALSE,TRUE,TRUE,TRUE,FALSE,FA
LSE,FALSE,TRUE)])
> # Creating a vector as input.
> data <- c("aaa","bbb","ccc","bbb","aaa","ddd","bbb","aaa","ddd","ccc","ddd")
>
> # Applying the factor function.
>factor_data<- factor(data)
>
> #Printing all elements of factor
> print(factor_data)
 [1] aaabbb ccc bbbaaadddbbbaaaddd ccc ddd
Levels: aaabbb ccc ddd
>
> #Accessing 4th element of factor
> print(factor_data[4])
[1] bbb
Levels: aaabbb ccc ddd
>
> #Accessing 5th and 7th element
> print(factor_data[c(5,7)])
[1] aaabbb
Levels: aaabbb ccc ddd
>
> #Accessing all elemcent except 4th one
> print(factor_data[-4])
 [1] aaabbb ccc aaadddbbbaaaddd ccc ddd
Levels: aaabbb ccc ddd
>
> #Accessing elements using logical vector
> print(factor_data[c(TRUE,FALSE,FALSE,FALSE,TRUE,TRUE,TRUE,FALSE,FALSE,FALS
[1] aaaaaadddbbbddd
```

Levels: aaabbb ccc ddd

---

```
data <- c("East","West","East","North","North","East","West",
          "West","West","East","North")
# Create the factors
factor_data<- factor(data)
print(factor_data)

# Apply the factor function with required order of the level.
new_order_data<- factor(factor_data,levels = c("East","West","North"))
print(new_order_data)
```

```
> data <- c("East","West","East","North","North","East","West",
+          "West","West","East","North")
> # Create the factors
>factor_data<- factor(data)
> print(factor_data)
 [1] East  West  East  North North East  West  WestWest  East  North
Levels: East North West
>
> # Apply the factor function with required order of the level.
>new_order_data<- factor(factor_data,levels = c("East","West","North"))
> print(new_order_data)
 [1] East  West  East  North North East  West  WestWest  East  North
Levels: East West North
```

---

Generating Factor Levels

```
v <- gl(3, 4, labels = c("Computer", "Science","Rprg"))
print(v)
[1] Computer ComputerComputerComputer Science  ScienceScienceScience  R prg    R prg
[11] R prg    R prg
Levels: Computer Science R prg
```

How to create Data Frame

```
# Create the data frame.
emp.data<- data.frame(
emp_id = c (1:5),
emp_name = c("Raj","Krishna","Ram","Malini","Swathi"),
  salary = c(623.3,515.2,611.0,729.0,843.25),

start_date = as.Date(c("2012-01-01", "2013-09-23", "2014-11-15", "2014-05-11",
                       "2015-03-27")),
stringsAsFactors = FALSE
)
# Print the data frame.
print(emp.data)
emp_idemp_name salary start_date
1      1      Raj 623.30 2012-01-01
2      2  Krishna 515.20 2013-09-23
3      3      Ram 611.00 2014-11-15
4      4   Malini 729.00 2014-05-11
5      5   Swathi 843.25 2015-03-27
```

---

## Getting the structure of R Data Frame

```
# Create the data frame.
emp.data<- data.frame(
emp_id = c (1:5),
emp_name = c("Raj","Krishna","Ram","Malini","Swathi"),
  salary = c(623.3,515.2,611.0,729.0,843.25),

start_date = as.Date(c("2012-01-01", "2013-09-23", "2014-11-15", "2014-05-11",
                       "2015-03-27")),
stringsAsFactors = FALSE
)
# Get the structure of the data frame.
str(emp.data)

'data.frame':       5 obs. of  4 variables:
 $ emp_id    : int  1 2 3 4 5
 $ emp_name  : chr  "Raj" "Krishna" "Ram" "Malini" ...
 $ salary    : num  623 515 611 729 843
 $ start_date: Date, format: "2012-01-01" "2013-09-23" "2014-11-15" ...
```

---

## Summary of data in Data Frames

```
# Create the data frame.
emp.data<- data.frame(
emp_id = c (1:5),
emp_name = c("Raj","Krishna","Ram","Malini","Swathi"),
  salary = c(623.3,515.2,611.0,729.0,843.25),

start_date = as.Date(c("2012-01-01", "2013-09-23", "2014-11-15", "2014-05-11",
                       "2015-03-27")),
stringsAsFactors = FALSE
)
# Print the summary.
print(summary(emp.data))

emp_idemp_name            salary        start_date
 Min.   :1   Length:5          Min.   :515.2   Min.   :2012-01-01
 1st Qu.:2   Class :character   1st Qu.:611.0   1st Qu.:2013-09-23
 Median :3   Mode  :character   Median :623.3   Median :2014-05-11
 Mean   :3                      Mean   :664.4   Mean   :2014-01-14
 3rd Qu.:4                      3rd Qu.:729.0   3rd Qu.:2014-11-15
 Max.   :5                      Max.   :843.2   Max.   :2015-03-27
```

## Extracting data from Data Frame

### Extracting the specific columns from a data frame

```
# Creating the data frame.
emp.data<- data.frame(
employee_id = c (1:5),
employee_name= c("Raj","Krishna","Ram","Malini","Swathi"),
sal = c(623.3,515.2,611.0,729.0,843.25),

starting_date = as.Date(c("2012-01-01", "2013-09-23", "2014-11-15", "2014-05-11",
                          "2015-03-27")),
stringsAsFactors = FALSE
)
# Extracting specific columns from a data frame
final <- data.frame(emp.data$employee_id,emp.data$sal)
print(final)

emp.data.employee_idemp.data.sal
1                 1      623.30
2                 2      515.20
```

```
3           3      611.00
4           4      729.00
5           5      843.25
```

Extracting the specific rows from a data frame

```
# Creating the data frame.
emp.data<- data.frame(
employee_id = c (1:5),
employee_name = c("Raj","Krishna","Ram","Malini","Swathi"),
sal = c(623.3,515.2,611.0,729.0,843.25),

starting_date = as.Date(c("2012-01-01", "2013-09-23", "2014-11-15", "2014-05-11",
                          "2015-03-27")),
stringsAsFactors = FALSE
)
# Extracting first row from a data frame
final <- emp.data[1,]
print(final)
output
employee_idemployee_namesalstarting_date
1          1          Raj 623.3    2012-01-01

employee_idemployee_namesalstarting_date
4          4       Malini 729.00    2014-05-11
5          5       Swathi 843.25    2015-03-27
```

Extracting specific rows corresponding to specific columns

```
# Creating the data frame.
emp.data<- data.frame(
employee_id = c (1:5),
employee_name = c("Raj","Krishna","Ram","Malini","Swathi"),
sal = c(623.3,515.2,611.0,729.0,843.25),

starting_date = as.Date(c("2012-01-01", "2013-09-23", "2014-11-15",
"2014-05-11",
                          "2015-03-27")),
```

```
stringsAsFactors = FALSE
)
# Extracting 2nd and 3rd row corresponding to the 1st and 4th column
final<- emp.data[c(2,3),c(1,4)]
print(final)
employee_idstarting_date
2          2    2013-09-23
3          3    2014-11-15
```

---

## Modification in Data Frame

**Example: Adding rows and columns**

```
# Creating the data frame.
emp.data<- data.frame(
employee_id = c (1:5),
employee_name = c("Raj","Krishna","Ram","Malini","Swathi"),
sal = c(623.3,515.2,611.0,729.0,843.25),

starting_date = as.Date(c("2012-01-01", "2013-09-23", "2014-11-15",
"2014-05-11",
                          "2015-03-27")),
stringsAsFactors = FALSE
)
print(emp.data)

#Adding row in the data frame
x <- list(6,"Vaishali",547,"2015-09-01")
rbind(emp.data,x)

#Adding column in the data frame
y <- c("Moradabad","Lucknow","Etah","Sambhal","Khurja")
cbind(emp.data,Address=y)
> print(emp.data)
employee_idemployee_namesalstarting_date
1          1          Raj 623.30    2012-01-01
2          2       Krishna 515.20    2013-09-23
3          3          Ram 611.00    2014-11-15
4          4       Malini 729.00    2014-05-11
5          5       Swathi 843.25    2015-03-27
>
> #Adding row in the data frame
```

```
> x <- list(6,"Vaishali",547,"2015-09-01")
>rbind(emp.data,x)
employee_idemployee_namesalstarting_date
1           1          Raj 623.30    2012-01-01
2           2       Krishna 515.20    2013-09-23
3           3          Ram 611.00    2014-11-15
4           4       Malini 729.00    2014-05-11
5           5       Swathi 843.25    2015-03-27
6           6      Vaishali 547.00    2015-09-01
>
> #Adding column in the data frame
> y <- c("Moradabad","Lucknow","Etah","Sambhal","Khurja")
>cbind(emp.data,Address=y)
employee_idemployee_namesalstarting_date   Address
1           1          Raj 623.30    2012-01-01 Moradabad
2           2       Krishna 515.20    2013-09-23   Lucknow
3           3          Ram 611.00    2014-11-15      Etah
4           4       Malini 729.00    2014-05-11   Sambhal
5           5       Swathi 843.25    2015-03-27    Khurja
```

**Example: Delete rows and columns**

```
# Creating the data frame.
emp.data<- data.frame(
employee_id = c (1:5),
employee_name = c("Raj","Krishna","Ram","Malini","Swathi"),
sal = c(623.3,515.2,611.0,729.0,843.25),

starting_date = as.Date(c("2012-01-01", "2013-09-23", "2014-11-15",
"2014-05-11",
                          "2015-03-27")),
stringsAsFactors = FALSE
)
print(emp.data)

#Delete rows from data frame
```

```
emp.data<-emp.data[-1,]
print(emp.data)

#Delete column from the data frame
emp.data$starting_date<-NULL
print(emp.data)
> print(emp.data)
employee_idemployee_namesalstarting_date
1          1          Raj 623.30    2012-01-01
2          2       Krishna 515.20    2013-09-23
3          3          Ram 611.00    2014-11-15
4          4       Malini 729.00    2014-05-11
5          5       Swathi 843.25    2015-03-27
>
> #Delete rows from data frame
>emp.data<-emp.data[-1,]
> print(emp.data)
employee_idemployee_namesalstarting_date
2          2       Krishna 515.20    2013-09-23
3          3          Ram 611.00    2014-11-15
4          4       Malini 729.00    2014-05-11
5          5       Swathi 843.25    2015-03-27
>
> #Delete column from the data frame
>emp.data$starting_date<-NULL
> print(emp.data)
employee_idemployee_namesal
2          2       Krishna 515.20
3          3          Ram 611.00
4          4       Malini 729.00
5          5       Swathi 843.25



>
```