**R - Data Frames**

A data frame is a table or a two-dimensional array-like structure in which each column contains values of one variable and each row contains one set of values from each column.

Following are the 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 same number of data items.

Create Data Frame

# Create the data frame.

emp.data <- data.frame(

emp\_id = c (1:5),

emp\_name = c("Rick","Dan","Michelle","Ryan","Gary"),

salary = c(623.3,515.2,611.0,729.0,843.25),

stringsAsFactors = FALSE

)

# Print the data frame.

print(emp.data)

In R, the stringsAsFactors argument is used when creating a data frame to specify whether character vectors should be converted into factors or remain as character strings. By default, prior to R 4.0.0, the stringsAsFactors argument was set to TRUE, which meant that R would automatically convert character variables into factors when creating a data frame. In R 4.0.0 and later, the default behavior has changed to FALSE

When we execute the above code, it produces the following result

OUTPUT:

emp\_id emp\_name salary

1 1 Rick 623.30

2 2 Dan 515.20

3 3 Michelle 611.00

4 4 Ryan 729.00

5 5 Gary 843.25

Get the Structure of the Data Frame

The structure of the data frame can be seen by using **str()** function.

# Create the data frame.

emp.data <- data.frame(

emp\_id = c (1:5),

emp\_name = c("Rick","Dan","Michelle","Ryan","Gary"),

salary = c(623.3,515.2,611.0,729.0,843.25),

stringsAsFactors = FALSE

)

# Get the structure of the data frame.

str(emp.data)

When we execute the above code, it produces the following result

OUTPUT:

'data.frame' : 5 obs. of 3 variables:

$ emp\_id : int 1 2 3 4 5

$ emp\_name : chr "Rick" "Dan" "Michelle" "Ryan" ...

$ salary : num 623 515 611 729 843

Extract Data from Data Frame

Extract specific column from a data frame using column name.

# Create the data frame.

emp.data <- data.frame(

emp\_id = c (1:5),

emp\_name = c("Rick","Dan","Michelle","Ryan","Gary"),

salary = c(623.3,515.2,611.0,729.0,843.25),

stringsAsFactors = FALSE

)

# Extract Specific columns.

result <- data.frame(emp.data$emp\_name,emp.data$salary)

print(result)

In R, the $ operator is used to **access** specific columns (variables) of a **data frame** or **list** by their **name**.

When we execute the above code, it produces the following result –

emp.data.emp\_name emp.data.salary

1 Rick 623.30

2 Dan 515.20

3 Michelle 611.00

4 Ryan 729.00

5 Gary 843.25

Extract the first two rows and then all columns

# Create the data frame.

emp.data <- data.frame(

emp\_id = c (1:5),

emp\_name = c("Rick","Dan","Michelle","Ryan","Gary"),

salary = c(623.3,515.2,611.0,729.0,843.25),

stringsAsFactors = FALSE

)

# Extract first two rows.

result <- emp.data[1:2,]

print(result)

When we execute the above code, it produces the following result

emp\_id emp\_name salary

1 1 Rick 623.3

2 2 Dan 515.2

Extract 3rd and 5th row with 2nd and 4th column

# Create the data frame.

emp.data <- data.frame(

emp\_id = c (1:5),

emp\_name = c("Rick","Dan","Michelle","Ryan","Gary"),

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

)

# Extract 3rd and 5th row with 2nd and 4th column.

result <- emp.data[c(3,5),c(2,4)]

print(result)

When we execute the above code, it produces the following result

OUTPUT:

emp\_name start\_date

3 Michelle 2014-11-15

5 Gary 2015-03-27

Expand Data Frame

A data frame can be expanded by adding columns and rows.

Add Column

Just add the column vector using a new column name.

# Create the data frame.

emp.data <- data.frame(

emp\_id = c (1:5),

emp\_name = c("Rick","Dan","Michelle","Ryan","Gary"),

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

)

# Add the "dept" coulmn.

emp.data$dept <- c("IT","Operations","IT","HR","Finance")

v <- emp.data

print(v)

When we execute the above code, it produces the following result

Output:

emp\_id emp\_name salary start\_date dept

1 1 Rick 623.30 2012-01-01 IT

2 2 Dan 515.20 2013-09-23 Operations

3 3 Michelle 611.00 2014-11-15 IT

4 4 Ryan 729.00 2014-05-11 HR

5 5 Gary 843.25 2015-03-27 Finance

Add Row

To add more rows permanently to an existing data frame, we need to bring in the new rows in the same structure as the existing data frame and use the **rbind()** function.

In the example below we create a data frame with new rows and merge it with the existing data frame to create the final data frame.

# Create the first data frame.

emp.data <- data.frame(

emp\_id = c (1:5),

emp\_name = c("Rick","Dan","Michelle","Ryan","Gary"),

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")),

dept = c("IT","Operations","IT","HR","Finance"),

stringsAsFactors = FALSE

)

# Create the second data frame

emp.newdata <- data.frame(

emp\_id = c (6:8),

emp\_name = c("Rasmi","Pranab","Tusar"),

salary = c(578.0,722.5,632.8),

start\_date = as.Date(c("2013-05-21","2013-07-30","2014-06-17")),

dept = c("IT","Operations","Fianance"),

stringsAsFactors = FALSE

)

# Bind the two data frames.

emp.finaldata <- rbind(emp.data,emp.newdata)

print(emp.finaldata)

When we execute the above code, it produces the following result

Output:

emp\_id emp\_name salary start\_date dept

1 1 Rick 623.30 2012-01-01 IT

2 2 Dan 515.20 2013-09-23 Operations

3 3 Michelle 611.00 2014-11-15 IT

4 4 Ryan 729.00 2014-05-11 HR

5 5 Gary 843.25 2015-03-27 Finance

6 6 Rasmi 578.00 2013-05-21 IT

7 7 Pranab 722.50 2013-07-30 Operations

8 8 Tusar 632.80 2014-06-17 Fianance

# R - Lists

Lists are the R objects which contain elements of different types like − numbers, strings, vectors and another list inside it. A list can also contain a matrix or a function as its elements. List is created using **list()** function.

## Creating a List

Following is an example to create a list containing strings, numbers, vectors and a logical values.

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

# values.

list\_data <- list("Red", "Green", c(21,32,11), TRUE, 51.23, 119.1)

print(list\_data)

When we execute the above code, it produces the following result

Output:

[[1]]

[1] "Red"

[[2]]

[1] "Green"

[[3]]

[1] 21 32 11

[[4]]

[1] TRUE

[[5]]

[1] 51.23

[[6]]

[1] 119.1

## Naming List Elements

The list elements can be given names and they can be accessed using these names.

# 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")

# Show the list.

print(list\_data)

When we execute the above code, it produces the following result

Output:

$`1st\_Quarter`

[1] "Jan" "Feb" "Mar"

$A\_Matrix

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

[1,] 3 5 -2

[2,] 9 1 8

$A\_Inner\_list

$A\_Inner\_list[[1]]

[1] "green"

$A\_Inner\_list[[2]]

[1] 12.3

## Accessing List Elements

Elements of the list can be accessed by the index of the element in the list. In case of named lists it can also be accessed using the names.

We continue to use the list in the above example −

# 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])

# Access the thrid element. As it is also a list, all its elements will be printed.

print(list\_data[3])

# Access the list element using the name of the element.

print(list\_data$A\_Matrix)

When we execute the above code, it produces the following result

Output:

$`1st\_Quarter`

[1] "Jan" "Feb" "Mar"

$A\_Inner\_list

$A\_Inner\_list[[1]]

[1] "green"

$A\_Inner\_list[[2]]

[1] 12.3

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

[1,] 3 5 -2

[2,] 9 1 8

## Manipulating List Elements

We can add, delete and update list elements as shown below. We can add and delete elements only at the end of a list. But we can update any element.

# 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")

# Add element at the end of the list.

list\_data[4] <- "New element"

print(list\_data[4])

# Remove the last element.

list\_data[4] <- NULL

# Print the 4th Element.

print(list\_data[4])

# Update the 3rd Element.

list\_data[3] <- "updated element"

print(list\_data[3])

When we execute the above code, it produces the following result

Output:

[[1]]

[1] "New element"

$<NA>

NULL

$`A Inner list`

[1] "updated element"

## Merging Lists

You can merge many lists into one list by placing all the lists inside one list() function.

# Create two lists.

list1 <- list(1,2,3)

list2 <- list("Sun","Mon","Tue")

# Merge the two lists.

merged.list <- c(list1,list2)

# Print the merged list.

print(merged.list)

When we execute the above code, it produces the following result

Output:

[[1]]

[1] 1

[[2]]

[1] 2

[[3]]

[1] 3

[[4]]

[1] "Sun"

[[5]]

[1] "Mon"

[[6]]

[1] "Tue"

## Converting List to Vector

A list can be converted to a vector so that the elements of the vector can be used for further manipulation. All the arithmetic operations on vectors can be applied after the list is converted into vectors. To do this conversion, we use the **unlist()** function. It takes the list as input and produces a vector.

# Create lists.

list1 <- list(1:5)

print(list1)

list2 <-list(10:14)

print(list2)

# Convert the lists to vectors.

v1 <- unlist(list1)

v2 <- unlist(list2)

print(v1)

print(v2)

# Now add the vectors

result <- v1+v2

print(result)

When we execute the above code, it produces the following result

Output:

[[1]]

[1] 1 2 3 4 5

[[1]]

[1] 10 11 12 13 14

[1] 1 2 3 4 5

[1] 10 11 12 13 14

[1] 11 13 15 17 19