DATA SCIENCE

SECTION.6.

Data Frames

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 or character type.
- Each column should contain same number of data items.

- To **create a DataFrame** in R from **one or more vectors of the same length**, we use the **data.frame()** function.

basic syntax is as follows

df <- data.frame(vector_1, vector_2)

- **Example:**

```
name <- c('Mohamed', 'Ahmed', 'Nour', 'Menna')
age <- c(20,30,40,50)
country <- c('Australia', 'Italy', 'Egypt', 'China')
salary <- c(2000, 1800, 1700, 1000)
Data <- data.frame(name, age, country, salary)
print(Data)
```

```
     name age   country salary
1 Mohamed  20 Australia   2000
2   Ahmed  30     Italy   1800
3    Nour  40     Egypt   1700
4   Menna  50     China   1000
```

we can provide all the vectors directly in the following function:

```
Data <- data.frame(name = c('Mohamed', 'Ahmed', 'Nour', 'Menna'),
                   age = c(20,30,40,50),
                   country = c('Australia', 'Italy', 'Egypt', 'China'),
                   salary = c(2000, 1800, 1700, 1000))
print(Data)
```

```
     name age   country salary
1 Mohamed  20 Australia   2000
2   Ahmed  30     Italy   1800
3    Nour  40     Egypt   1700
4   Menna  50     China   1000
```

Define the column and row names

we use an optional parameter **row.names,** as follows:

```
Data <- data.frame(name = c('Mohamed', 'Ahmed', 'Nour', 'Menna'),
                   age = c(20,30,40,50),
                   country = c('Australia', 'Italy', 'Egypt', 'China'),
                   salary = c(2000, 1800, 1700, 1000),
                   row.names=c('row_1', 'row_2', 'row_3', 'row_4'))
print(Data)
```

```
        name age   country salary
row_1 Mohamed  20 Australia   2000
row_2   Ahmed  30     Italy   1800
row_3    Nour  40     Egypt   1700
row_4   Menna  50     China   1000
```

we can **rename the columns** of a DataFrame after its creation using the **names()** function:

```
names(Data) <- c('col_1', 'col_2', 'col_3', 'col_4')
print(Data)
```

```
> names(Data) <- c('col_1', 'col_2', 'col_3', 'col_4')
> print(Data)
        col_1 col_2     col_3 col_4
row_1 Mohamed    20 Australia  2000
row_2   Ahmed    30     Italy  1800
row_3    Nour    40     Egypt  1700
row_4   Menna    50     China  1000
```

Access Data Frame elements

- We can use single brackets [ ], double brackets [[ ]] or $ to access columns from a data frame.
- Example:

```
Data <- data.frame(name = c('Mohamed', 'Ahmed', 'Nour', 'Menna'),
                   age = c(20,30,40,50),
                   country = c('Australia', 'Italy', 'Egypt', 'China'),
                   salary = c(2000, 1800, 1700, 1000))
print(Data)


Data[3]
Data[["country"]]
Data$country
```

```
> Data[3]
    country
1 Australia
2     Italy
3     Egypt
4     China
>
> Data[["country"]]
[1] "Australia" "Italy"     "Egypt"     "China"
>
> Data$country
[1] "Australia" "Italy"     "Egypt"     "China"
```

Add Rows and Columns

- Use the cbind() function to add additional columns and rbind() to add additional rows in a data frame.

Note: The cells in the new column or row must be of the same length as the existing data frame.

#Add a new row

```
New_row_DF <- rbind(Data, c("Mona", 110,"Italy" ,1100))
New_row_DF
```

```
> New_row_DF <- rbind(Data, c("Mona", 110,"Italy" ,1100))
> New_row_DF
     name age   country salary
1 Mohamed  20 Australia   2000
2   Ahmed  30     Italy   1800
3    Nour  40     Egypt   1700
4   Menna  50     China   1000
5    Mona 110     Italy   1100
```

Add a new column

```
New_col_DF <- cbind(Data, Steps = c(1000, 6000, 5000,2000))
New_col_DF
```

```
> New_col_DF <- cbind(Data, Steps = c(1000, 6000, 5000,2000))
> New_col_DF
     name age   country salary Steps
1 Mohamed  20 Australia   2000  1000
2   Ahmed  30     Italy   1800  6000
3    Nour  40     Egypt   1700  5000
4   Menna  50     China   1000  2000
```

Remove Rows and Columns

- using the c() function to remove rows and columns in a Data frame.

```
Data <- data.frame(name = c('Mohamed', 'Ahmed', 'Nour', 'Menna'),
                   age = c(20,30,40,50),
                   country = c('Australia', 'Italy', 'Egypt', 'China'),
                   salary = c(2000, 1800, 1700, 1000))
print(Data)
# Remove the first row and column
Data_New <- Data[-c(1), -c(1)]
Data_New
```

```
> Data_New <- Data[-c(1), -c(1)]
> Data_New
  age country salary
2  30   Italy   1800
3  40   Egypt   1700
4  50   China   1000
```

 # Get column names

names(Data)

```
> names(Data)
[1] "name"    "age"     "country" "salary"
```

to display structure

str(Data)

```
> print(str(Data))
'data.frame':   4 obs. of  4 variables:
 $ name   : chr  "Mohamed" "Ahmed" "Nour" "Menna"
 $ age    : num  20 30 40 50
 $ country: chr  "Australia" "Italy" "Egypt" "China"
 $ salary : num  2000 1800 1700 1000
```

dim() function to find the number of rows and columns

dim(Data)

```
> dim(Data)
[1] 4 4
```

#Dataframe length

length(Data)

```
> length(Data)
[1] 4
```

Display first 6 rows of the data

head(Data)

```
> head(Data)
      name age    country salary
1 Mohamed  20 Australia    2000
2   Ahmed  30      Italy    1800
3    Nour  40      Egypt    1700
4   Menna  50      China    1000
> tail(Data)
      name age    country salary
1 Mohamed  20 Australia    2000
2   Ahmed  30      Italy    1800
3    Nour  40      Egypt    1700
4   Menna  50      China    1000
```

Display last 6 rows of the data

tail(Data)

Specify number of rows for head or tail

head(Data,n=2)

tail(Data,n=2)

```
> head(Data,n=2)
      name age   country salary
1 Mohamed  20 Australia   2000
2   Ahmed  30     Italy   1800
> tail(Data,n=2)
   name age country salary
3  Nour  40   Egypt   1700
4 Menna  50   China   1000
```

Use the summary() function to summarize the data from a Data Frame.

summary(Data)

```
> summary(Data)
    col_1               col_2          col_3               col_4
 Length:4           Min.   :20.0   Length:4           Min.   :1000
 Class :character   1st Qu.:27.5   Class :character   1st Qu.:1525
 Mode  :character   Median :35.0   Mode  :character   Median :1750
                    Mean   :35.0                      Mean   :1625
                    3rd Qu.:42.5                      3rd Qu.:1850
                    Max.   :50.0                      Max.   :2000
```

.

K_mean Clustring

K-Means clustering groups the data on similar groups. The algorithm is as follows:

1. Choose K; then select K random "centroids“ ➔ In our example, K=2

2. Assign records to the cluster with the closest centroid

3. Recalculate the resulting centroids

    Centroid: the mean value of all the records in the cluster

4. Repeat steps 2 & 3 until record assignments no longer change

```
# Calculate K-Means using R built-in kmeans function
Kmean_clustering <- kmeans(Data, centers = 2)
Kmean_clustering
```

Data: numeric matrix, numeric data frame or a numeric vector

Centers: is the number of clusters (k) to be produced

my_data <- Data[, -c(1,3)]

my_data

```
> my_data <- Data[, -c(1,3)]
> my_data
  age salary
1  20   2000
2  30   1800
3  40   1700
4  50   1000
```

Kmean_clustering <- kmeans(my_data, centers = 2)

Kmean_clustering

```
> Kmean_clustering <- kmeans(my_data, centers = 2)
> Kmean_clustering
K-means clustering with 2 clusters of sizes 3, 1

Cluster means:
  age   salary
1  30 1833.333
2  50 1000.000

Clustering vector:
[1] 1 1 1 2

Within cluster sum of squares by cluster:
[1] 46866.67     0.00
 (between_SS / total_SS =  91.7 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

QUESTION

1- To view only the first 10 rows of the data frame 'myTable', which of the following functions should be used?

a. ncol(myTable, 10)

b. nrow(myTable, 10)

c. head(myTable, 10)

d. tail(myTable,10)

2- Which function can be used to create a data frame?

a. data.frame()

b. dframe()

c. df()

d. dataframe()

3- Point out the wrong statement.

a. k-means clustering is a method of vector quantization

b. k-means clustering aims to partition n observations into k clusters

c. k-nearest neighbor is same as k-means

d. none of the mentioned

ANSWER

1. C
2. A
3. C