BIG DATA TOOLS FOR MANAGERS

Unit-3 : Introduction to R

by
Ankit Velani
Adjunct Faculty, Dept. of MBA,
Siddaganga Institute of Technology,Tumkur

Session-2 : R data types

- Advanced data structures (Continue from Part-1)
  - Vectors
  - Lists
  - Matrices
  - Data Frames

Vector

- Vector is the essential building block for handling multiple items in R.
- It’s a list of items that are of the same type.
- Combine function **c()** used to combine multiple values of same type.
- ie.

```
fruits ← c(“Apple”, “Banana”, “Orange”)

num  ← c(1,2,3,4,5,6)

num ← 1:100 #Integer values in a sequence

dec  ← 1.5 : 6.5 #Numeric values in a sequence
```

Vector

- Length function which helps to find out how many items a vector has.

```
length(fruits)

length(num)
```

Vector

- Combining two vectors

```
num_all <- c(num, dec)
print(num_all)

alpha = c(“A”, “B”, “C”, “D”)
fruits = c(“Apple”, “Banana”, “Orange”)

data = c(alpha, fruits, num_all)
print(data)
```

Vector

- **Sequence**

  common useful functions to create continuous number generation.

seq(from=, to=, by=)
seq(from=, to=, length.out=)

Example:

seq(from=1, to=10, by=2)
seq(from=1, to=10, length.out=20)

Vector

- **Repeat**

common useful functions to repeat the certain values in vector.

rep(x=, each=, times=)

Example:

rep(x=c(1,2,3), each=2)

o/p: 112233

rep(x=c(1,2), each=2, times=4)

O/p :1122 1122 1122 1122

Vector

- **Repeat**

common useful functions to repeat the certain values in vector.

rep(x=, each=, times=)

Example:

rep(x=c(1,2,3), each=2)

o/p: 112233

rep(x=c(1,2), each=2, times=4)

O/p :1122 1122 1122 1122

- **times** provide no of times entire vector elements to repeat
- **each** provide no of times each vector elements to gets repeat.

Vector

- **Sort**

Sort function used to sort vector elements in increasing or decreasing order.

sort(x=, decreasing=)

x = is vector

decreasing is TRUE/FALSE

<u>Example</u>: Sort element in increasing order

```
sort(x= c(2.5, -1, -10, 3.44), decreasing=FALSE)
```

Vector

- **Accessing Vector Elements**

Index to be used to access the vector elements, index starts from 1 to length of vector

Syntax:

```
vector_name[index]
```

Example :

```
num <- c(“A”, “B”, “C”, “D”, “E”)
num[1] #1st Element
num[5] #5th Element
```

Data Frames

- A data frame is R’s most natural way of presenting two-dimensional dataset with collection of observation with one or more variables.
- Data frame is one of the most important and frequently used in R for data analysis.
- **data.frame()** function helps to create data frame in R

Data Frames

- Creation of data frame

```
my_data <- data.frame(
        name    = c(“A”,”B”,”C”,”D”),
        age     = c(40,45,70,60),
        gender = c(“F”,”M”,”F”,”M”)
)
```

name, age, gender are the vectors

Output:

Name	Age	gender
A	40	F
B	45	M
C	70	F
D	60	M

Data Frames

- Accessing elements

data-frame_object[row_range , col_range]

index index

Example:

my_data[] #display all the rows & cols

my_data[1,] #first row with all the columns

my_data[1,3] # first row with only 3rd columns

my_data[1, 2:3] # first row with 2 & 3rd columns

my_data[2:3, 1] # 2 & 3rd row with 1st columns

Data Frames

- Accessing elements with Variable Name

  data-frame_object**$variable_name**

  my_data$name
  my_date$age

- Accessing elements with condition on row index

  data-frame_object**[condition,]**

  my_data[my_data$gender==“M”,]
  my_data[my_data$gender==“F”, 2]

Matrices

- Matrices are the R objects wherein the elements are organized in a 2-D rectangular shape. In a matrix, it contains elements of the same type.

- A column is a vertical representation of data, while a row is a horizontal representation of data.

Syntax:

```
matrix(x=  ,nrow=  ,ncol=  ,byrow= )
```

Matrices

Syntax:

```
matrix(x=  ,nrow=  ,ncol=  ,byrow= )
```

matrix()	: Name of the functions in R
nrow	: is number of rows to be created
ncol	: is number of columns to be created
byrow	: TRUE or FALSE data filling direction in matrix

Matrices

Example:

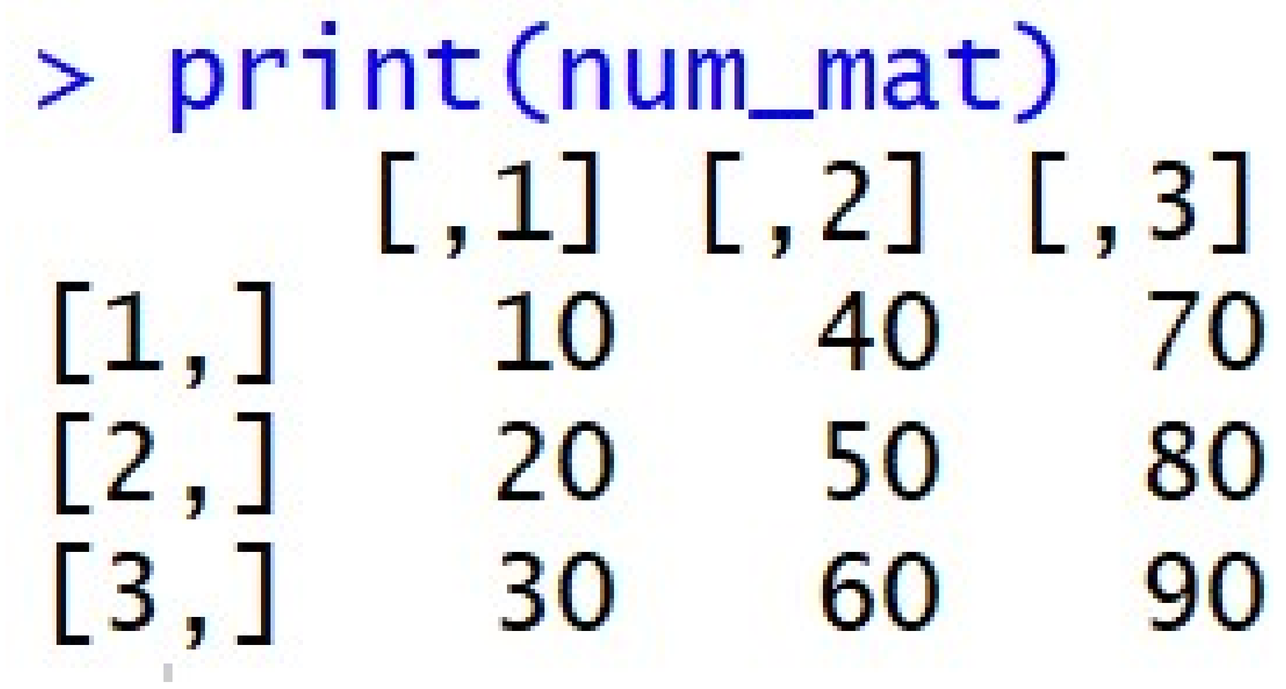
```
data = c(10, 20, 30, 40, 50, 60, 70, 80, 90)

num_mat  = matrix(data, nrow=3, ncol=3)
print(num_mat)
```

Matrices

Example:

```
data = c(10, 20, 30, 40, 50, 60, 70, 80, 90)
num_mat  = matrix(data, nrow=3, ncol=3)
print(num_mat)
```

<u>Output:</u>



```
> print(num_mat)
     [,1] [,2] [,3]
[1,]   10   40   70
[2,]   20   50   80
[3,]   30   60   90
```

Matrices

Example:

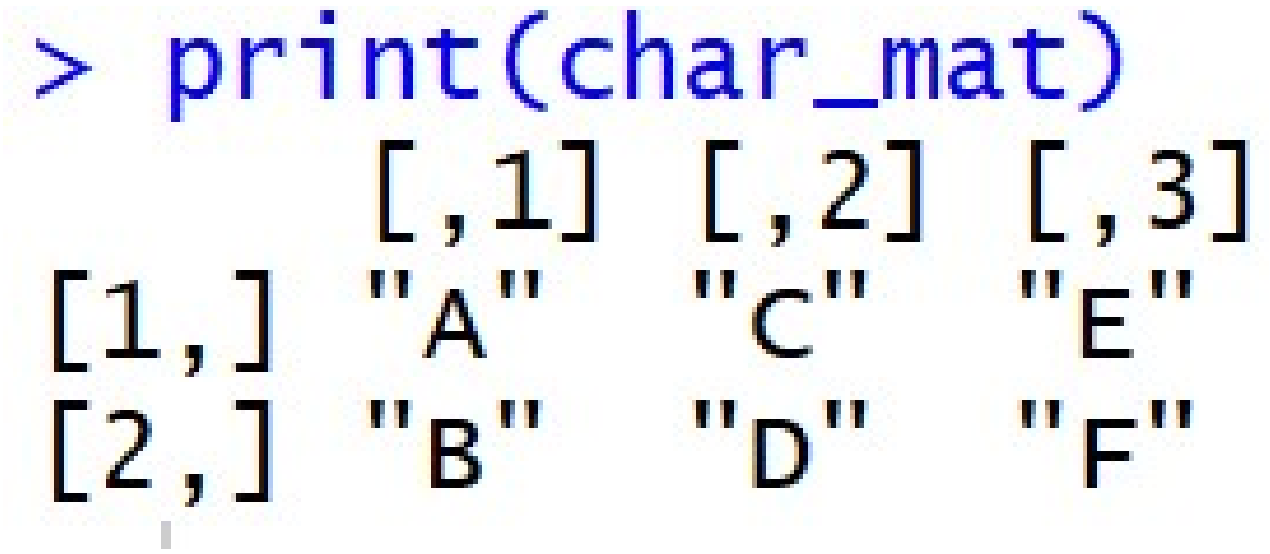
```
data = c("A", "B", "C", "D", "E", "F")

char_mat = matrix(data, nrow=2, ncol=3)
print(char_mat)
```

Matrices

Example:

```
data = c("A", "B", "C", "D", "E", "F")
char_mat = matrix(data, nrow=2, ncol=3)
print(char_mat)
```

Output:



```
> print(char_mat)
     [,1] [,2] [,3]
[1,] "A"  "C"  "E" 
[2,] "B"  "D"  "F" 
```

Matrices

- Filling direction:

R also allow to specify the filling direction for Matrix either by row or columns

byrow = TRUE/FALSE argument can control filling direction in matrix function.

```
m ← matrix(data=c(1,2,3,4,5,6),
                  nrow=3,
                  ncol=2, byrow=FALSE)
```

Matrices

```
m ← matrix(data=c(1,2,3,4,5,6),
                    nrow=3,
                    ncol=2, byrow=FALSE)
o/p:
        1 4
        2 5
        3 6
```

```
m ← matrix(data=c(1,2,3,4,5,6),
                    nrow=3,
                    ncol=2, byrow=TRUE)
o/p:
        1 2
        3 4
        5 6
```

Matrices

- For equal length of Vectors/Matrices/DataFrame can be combined into rows or cols using rbind() and cbind() functions.
- **rbind()** : for combining data by rows
- **cbind()** : for combining data by cols

Ex

```
a ← c(1,2,3)
b ← c(4,5,6)
```

```
rbind(a,b)
1 2 3
4 5 6
```

```
cbind(a,b)
1 4
2 5
3 6
```

Matrices

- dim() function is used to find the dimension of matrices.

```
> dim(m)
```

Output : number of rows, number of cols

Matrices

Accessing Elements

matrix_object[row-range, col-range]

Ex.

m[] # display all the rows & cols

m[1:2] # get the 1st element from row 1 & 2

m[1,1] # get the 1st column from 1st row

m[1, 2] # get the 2nd column from 1st row

m[3, 1:2] # get the 1&2 columns from 3rd row

diag(m) # get the diagonal elements of matrix

Operation on Matrices

A

2 5 2

6 1 4

Transpose **t()** function to get the transpose of matrix

t(A)

<u>O/p:</u>

2 6

5 1

2 4

Operation on Matrices

A
2 5 2
6 1 4

Scaler multiplication of matrix
matrix_obj * scaler value

Ex:
A * 2

Output:
4 10 4
12 2 8

Operation on Matrices

A
2 5 2
6 1 4

B
2 3 2
3 1 2

Element wise arithmetic operation

matrix_object1 + matrix_objct2
matrix_object1 - matrix_objct2
matrix_object1 * matrix_objct2
matrix_object1 / matrix_objct2

Operation on Matrices

A

2 5 2

6 1 4

C

5 -3

-1 1

1 5

Matrix Multiplication

matrix_obj **%*%** matrix_obj

Example:

A %*% C

Lists

- List is powerful data structures in R, which allows to group together any kind of data types & data.

- A single list contains vectors, matrix, logical vector, character, Data frame and list itself.

- list() function used to create a list in R

```
list_1 = list(c(1,2,3,4,5,))
```

Lists

```
fruits_vector <- c("Apple", "Banana", "Orange")

num_vector  <- c(1,2,3,4,5,6,7,8,9,10,11,12)

num_matrix  <- matrix(c(10, 20, 30, 40, 50, 60),
                      nrow=3,
                      ncol=2)

my_data <- data.frame(
  name   = c("A","B","C","D"),
  age    = c(40,45,70,60),
  gender = c("F","M","F","M")
)
```

Lists

```
fruits_vector <- c("Apple", "Banana", "Orange")

num_vector  <- c(1,2,3,4,5,6,7,8,9,10,11,12)

num_matrix  <- matrix(c(10, 20, 30, 40, 50, 60),
                      nrow=3,
                      ncol=2)


my_data <- data.frame(
  name   = c("A","B","C","D"),
  age    = c(40,45,70,60),
  gender = c("F","M","F","M")
)
```

```
x = 100
y = 6+10i
flag = FALSE
```

Lists

```
fruits_vector <- c("Apple", "Banana", "Orange")

num_vector  <- c(1,2,3,4,5,6,7,8,9,10,11,12)

num_matrix  <- matrix(c(10, 20, 30, 40, 50, 60),
                      nrow=3,
                      ncol=2)


my_data <- data.frame(
  name   = c("A","B","C","D"),
  age    = c(40,45,70,60),
  gender = c("F","M","F","M")
)
```

```
x = 100
y = 6+10i
flag = FALSE
```

Lists

Create a new list which contains vectors, matrix, data frame and basic datatype

```
all_data = list(fruits_vector,
                num_vector,
                num_matrix,
                my_data,
                x,
                y,
                flag)
```

Lists

Accessing List element: list element can be accessible by [[]] double square brackets with index.

all_data

all_data[[1]] # Accessing 1st elements

all_data[[2]] # Accessing 2nd elements

all_data[[2:5]] # Elements from 2 to 5 elements

Access DataFrame from list

```
all_data = list(fruits_vector, num_vector, num_matrix, my_data, x, y, flag)
     index      1              2              3             4      5  6  7
```

```
all_data[[4]]                   #4th element is Data Frame
all_data[[4]][]

all_data[[4]][ 1 , 2 ]     #1st row & 2nd Columns from 4th Element

all_data[[4]][  , 3 ]      # all the rows and 3rd column

all_data[[4]][ 1:3 , 3 ]   # 1 to 3 Rows and 3rd column

all_data[[4]][ 1:3 , 1:2 ]   # 1 to 3 Rows & 1 to 2 columns
```

View Function

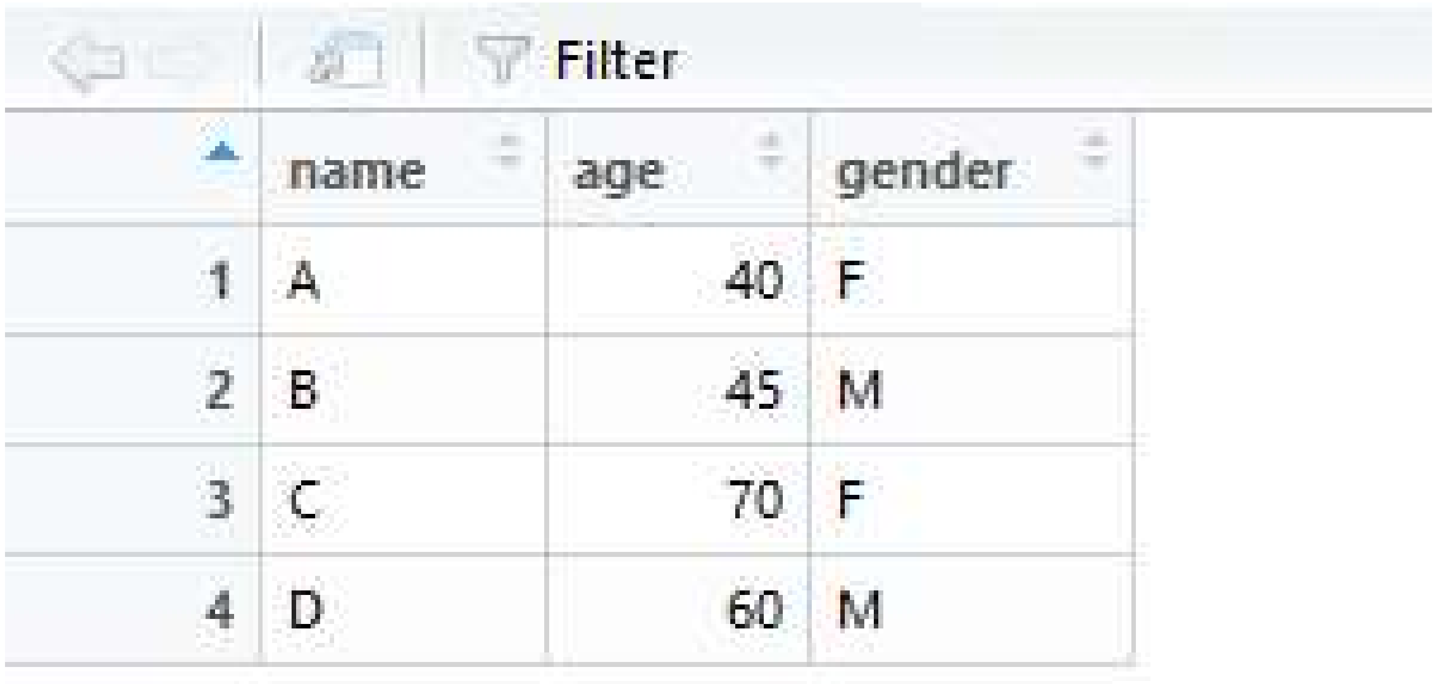
The View() function in R can be used to invoke a spreadsheet-style data viewer within RStudio.

Example:

```
View(x)
View(my_data)
View(all_data)
```

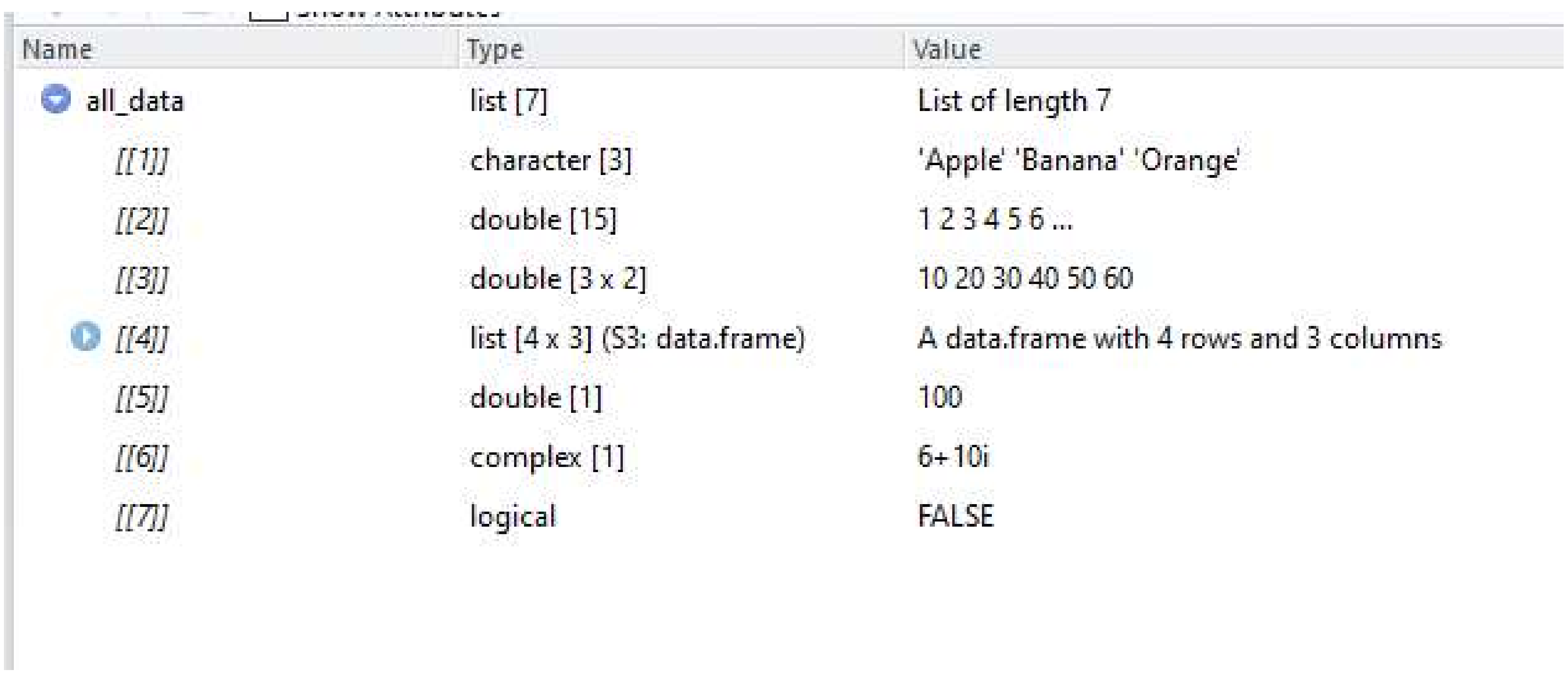
Note :
V is upper case in View function

View(my_data)



	name	age	gender
1	A	40	F
2	B	45	M
3	C	70	F
4	D	60	M

View(all_data)



Name	Type	Value
all_data	list [7]	List of length 7
[[1]]	character [3]	'Apple' 'Banana' 'Orange'
[[2]]	double [15]	1 2 3 4 5 6 ...
[[3]]	double [3 x 2]	10 20 30 40 50 60
[[4]]	list [4 x 3] (S3: data.frame)	A data.frame with 4 rows and 3 columns
[[5]]	double [1]	100
[[6]]	complex [1]	6+10i
[[7]]	logical	FALSE

Recap

- Basic Data Types
  - Logical
  - Numeric
  - Complex
  - Character

Recap

- Advance Data Types / Data Structure
  - Vector (Single Dimension)
  - Matrix (Two Dimension)
  - Data Frame (Two Dimension)
  - List (Mix in terms of dimension)

Recap

- Accessing elements
  - Vector [index]
  - Matrix [row-index, col-index]
  - Data Frame [row-index, col-index]
  - List [[index]]
  - View function to view the data in excel format

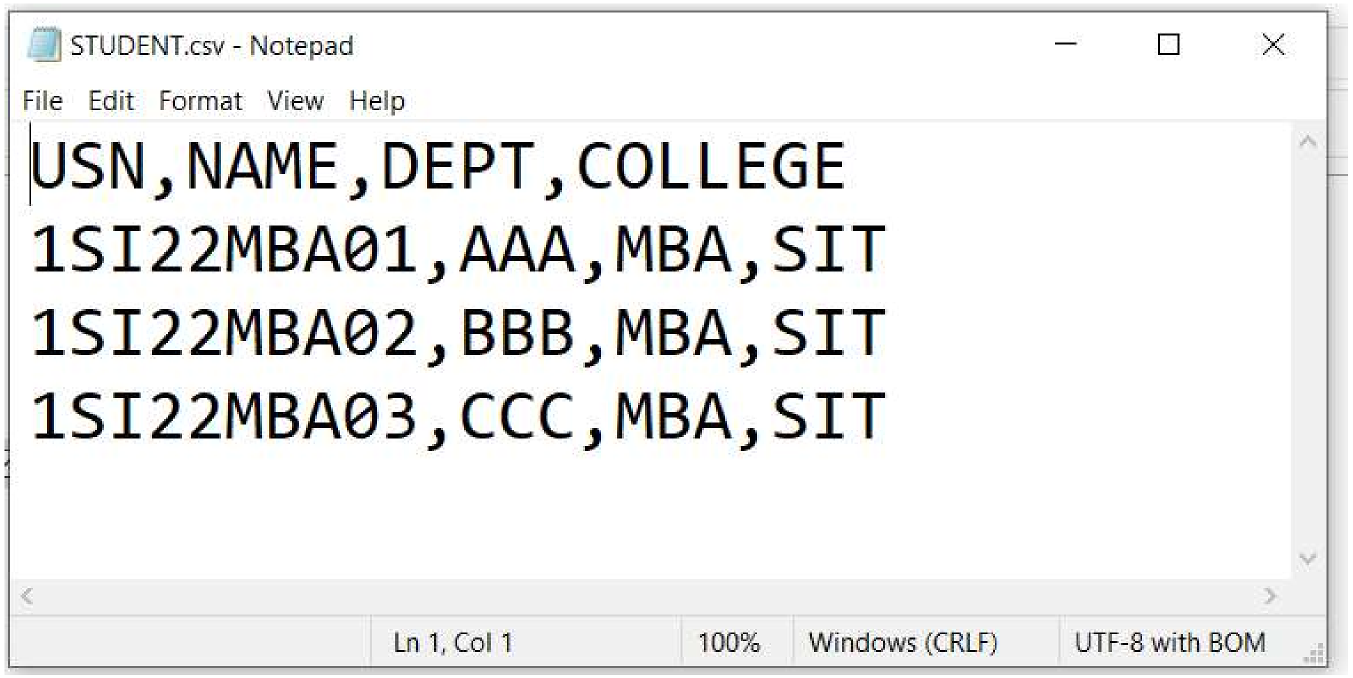
Importing data into R

- Text
- CSV
- Excel
- SPSS
- SAS files
- From the web

Importing CSV file into R

CSV (Comma Separated Values) is a text file in which the values in columns are separated by a comma.

Example:



```
USN,NAME,DEPT,COLLEGE
1SI22MBA01,AAA,MBA,SIT
1SI22MBA02,BBB,MBA,SIT
1SI22MBA03,CCC,MBA,SIT
```

Importing CSV file into R

- read.csv() function to read CSV files

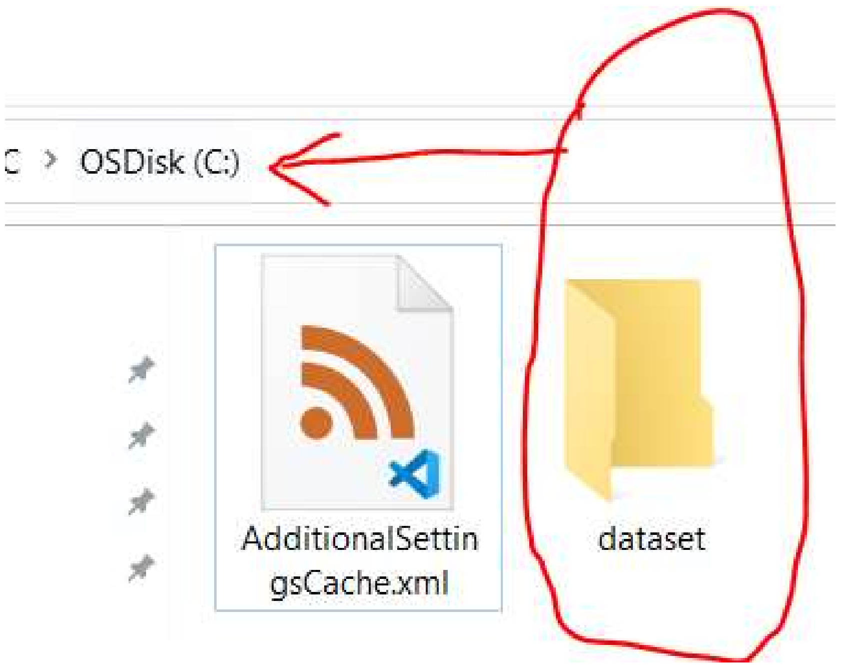
```
read.csv(file_path=" ", header=FALSE, sep=",")
```

Vehicle Park Data

- Vehicle park data contains the no of vehicles are present on road or market in India from Year 2000 to 2022
- **Columns :**
  - YEAR : Vehicle Sales Year
  - VEHICLE_TYPE : Type of vehicle sold to the market (Truck, Bus, Four & Two wheeler, Others)
  - BRAND : Vehicle brand & Manufacturer
  - VEHICLE_COUNT : No of vehicle sold in market for a year
  - AGE_GROUP : Age group of the vehicle
  - AGE : Vehicle age represent how old vehicle is
  - RTO_REGISTRATION_YEAR : Year on which vehicle go registration

Import Vehicle data into R

- Before importing any datasets in R, first create **dataset** folder in C drive



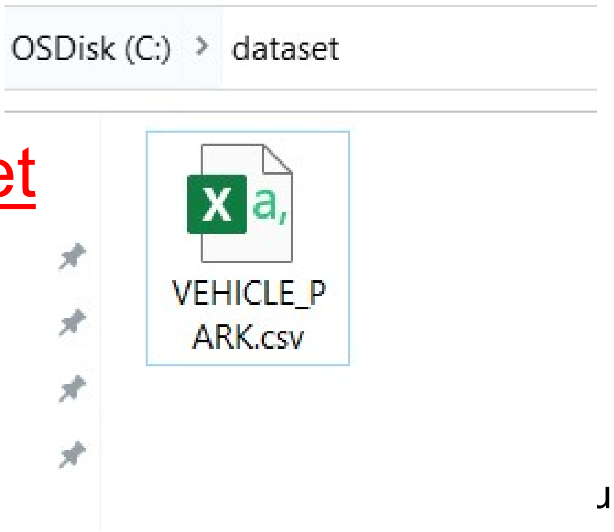
Download VEHICLE_PARK

Download the VEHICLE_PARK.csv file from link-1 or link- 2 and copy paste or save in C:\dataset folder

Link -1 https://raw.githubusercontent.com/sitmbadept/sitmbadept.github.io/main/BDTM/DATASET/VEHICLE_PARK.csv

Link -2 https://drive.google.com/file/d/1EPQhl0wVCZnNP1vx7BE43phOUljzZXGD/view?usp=sharing

Store in C:\dataset



Read VEHICLE_PARK data

<u>Read CSV file in</u>

```
data = read.csv("C:/dataset/VEHICLE_PARK.csv")
```

Structure of Data Frame

str() function gives details information about the data frame.

Syntax:

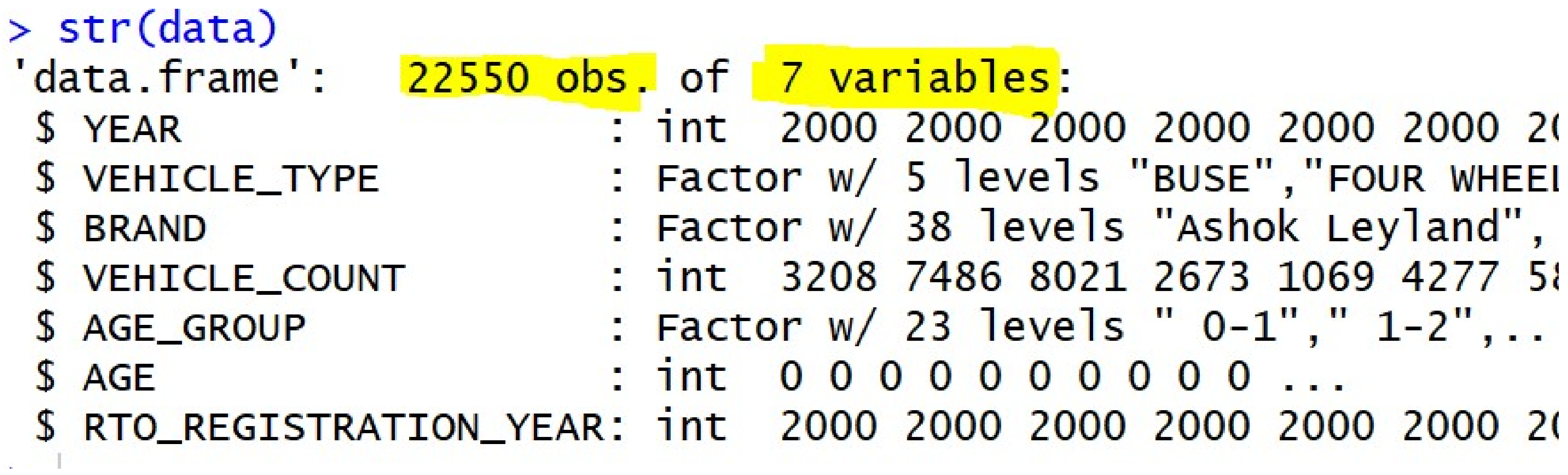
str(data-frame-variable)

Example:

str(data)

Structure of Data Frame

str() function gives details information about the data frame.



```
> str(data)
'data.frame':   22550 obs. of  7 variables:
 $ YEAR                 : int  2000 2000 2000 2000 2000 2000 2(
 $ VEHICLE_TYPE         : Factor w/ 5 levels "BUSE","FOUR WHEEI
 $ BRAND                : Factor w/ 38 levels "Ashok Leyland",
 $ VEHICLE_COUNT        : int  3208 7486 8021 2673 1069 4277 5{
 $ AGE_GROUP            : Factor w/ 23 levels " 0-1"," 1-2",..
 $ AGE                  : int  0 0 0 0 0 0 0 0 0 0 ...
 $ RTO_REGISTRATION_YEAR: int  2000 2000 2000 2000 2000 2000 2(
```

Structure of Data Frame

dim() function gives dimensions of data frame.

nrow() function gives #rows available in data frame

ncol() function gives #columns available in data frame

Syntax:

```
dim(data-frame-variable)
nrow(data-frame-variable)
ncol(data-frame-variable)
```

Structure of Data Frame

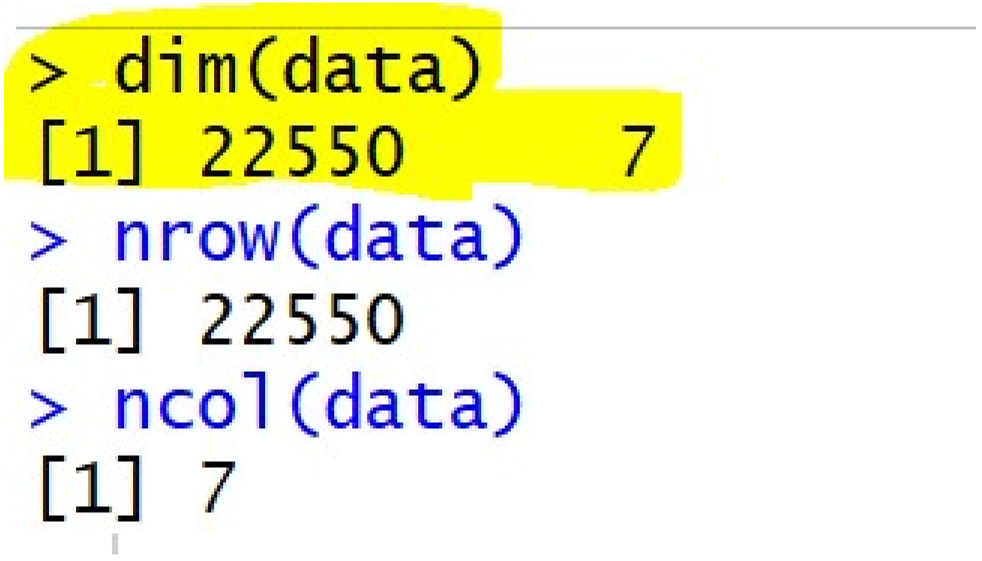
Example:

dim(data)

nrow(data)

ncol(data)

Output:



```
> dim(data)
[1] 22550     7
> nrow(data)
[1] 22550
> ncol(data)
[1] 7
```

Columns of Data Frame

colnames() function gives columns name of the data frame.

Syntax:

```
colnames(data-frame-variable)
```

Example:

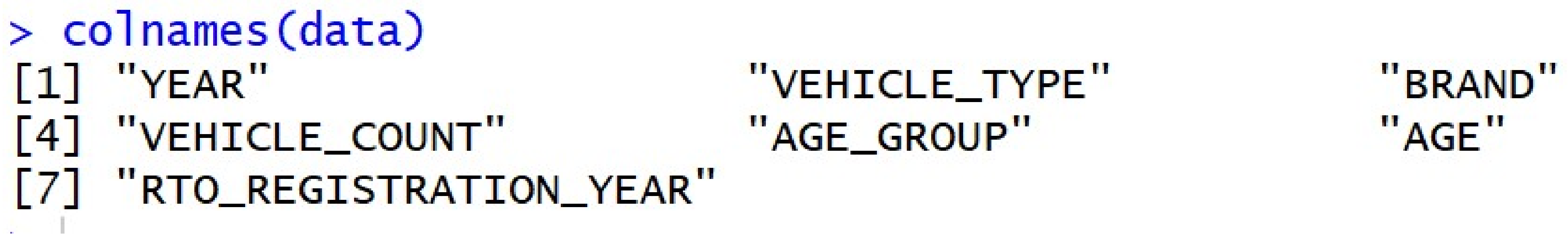
```
colnames(data)
```

Columns of Data Frame

colnames() function gives columns name of the data frame.

Example:

colnames(data)



```
> colnames(data)
[1] "YEAR"                    "VEHICLE_TYPE"          "BRAND"
[4] "VEHICLE_COUNT"           "AGE_GROUP"             "AGE"
[7] "RTO_REGISTRATION_YEAR"
```

Summary

summary() function gives quick summary for the data frame.

<u>Syntax:</u>

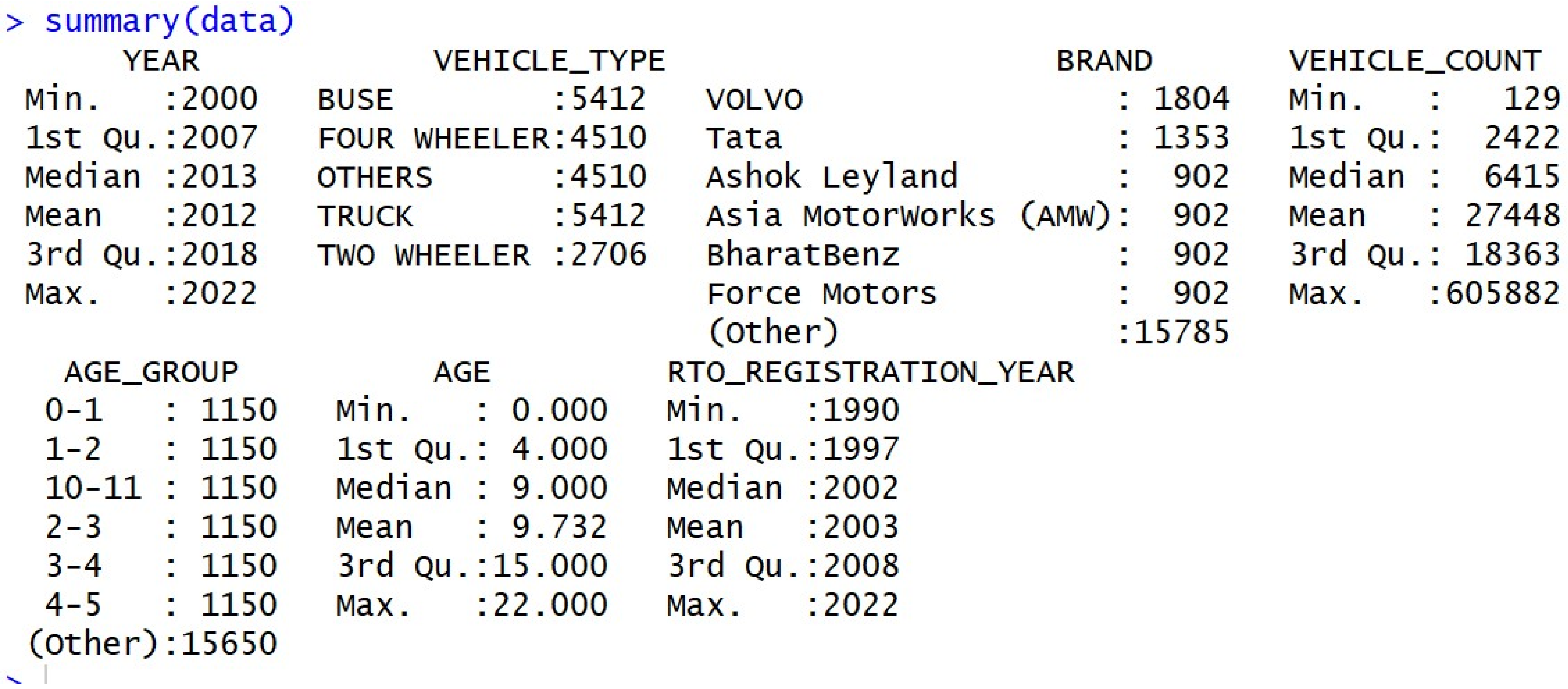
summary(data-frame-variable)

<u>Example:</u>

summary(data)

Summary

summary() function gives quick summary for the data frame.



```
> summary(data)
      YEAR             VEHICLE_TYPE                            BRAND         VEHICLE_COUNT
 Min.   :2000   BUSE          :5412   VOLVO                      : 1804   Min.   :   129
 1st Qu.:2007   FOUR WHEELER:4510   Tata                       : 1353   1st Qu.:  2422
 Median :2013   OTHERS        :4510   Ashok Leyland              :  902   Median :  6415
 Mean   :2012   TRUCK         :5412   Asia MotorWorks (AMW):  902   Mean   : 27448
 3rd Qu.:2018   TWO WHEELER :2706   BharatBenz                 :  902   3rd Qu.: 18363
 Max.   :2022                         Force Motors               :  902   Max.   :605882
                                      (Other)                    :15785
   AGE_GROUP           AGE          RTO_REGISTRATION_YEAR
  0-1    : 1150   Min.   : 0.000   Min.   :1990
  1-2    : 1150   1st Qu.: 4.000   1st Qu.:1997
  10-11  : 1150   Median : 9.000   Median :2002
  2-3    : 1150   Mean   : 9.732   Mean   :2003
  3-4    : 1150   3rd Qu.:15.000   3rd Qu.:2008
  4-5    : 1150   Max.   :22.000   Max.   :2022
 (Other):15650
>
```

First/Last few rows

head() function gives top records

tail() function gives last records

Syntax:

```
head(data-frame-variable,  n= )
tail(data-frame-variable,     n= )
```

Example:

```
head(data, n=10)
tail(data, n=20)
```

Slicing and Indexing of DataFrame

Access element using indexes

data_frame_variable[row-index, col-index]

data[ ] #Display all the rows & cols

data[5:7,] #Display rows from 5 to 7 and all the cols

data[c(1, 10, 20),] #Display rows 1, 10, 20 and all cols

data[, c(1,3)] #Display all rows, 1 & 3 columns

data[c(1, 10, 20), c(1,3)]

#Display 1, 10, 30 rows, 1 & 3 columns

Slicing and Indexing of DataFrame

Access Specific Columns using $ and c() function

data$AGE
data$YEAR

# All the rows & selected column
data[, c('BRAND', 'YEAR', 'AGE')]

# First 10 rows & selected column
data[1:10 , c('BRAND', 'YEAR', 'AGE’)]

Recap

- read.csv
- str
- dim
- colnames
- Summary
- head
- tail

Recap

- read.csv
- str
- dim
- colnames
- Summary
- head
- tail

- Using Index (row & column)
- Using $ and c() combine function

Splitting

split() function perform partition on the dataset by specific columns/variables.

Syntax:

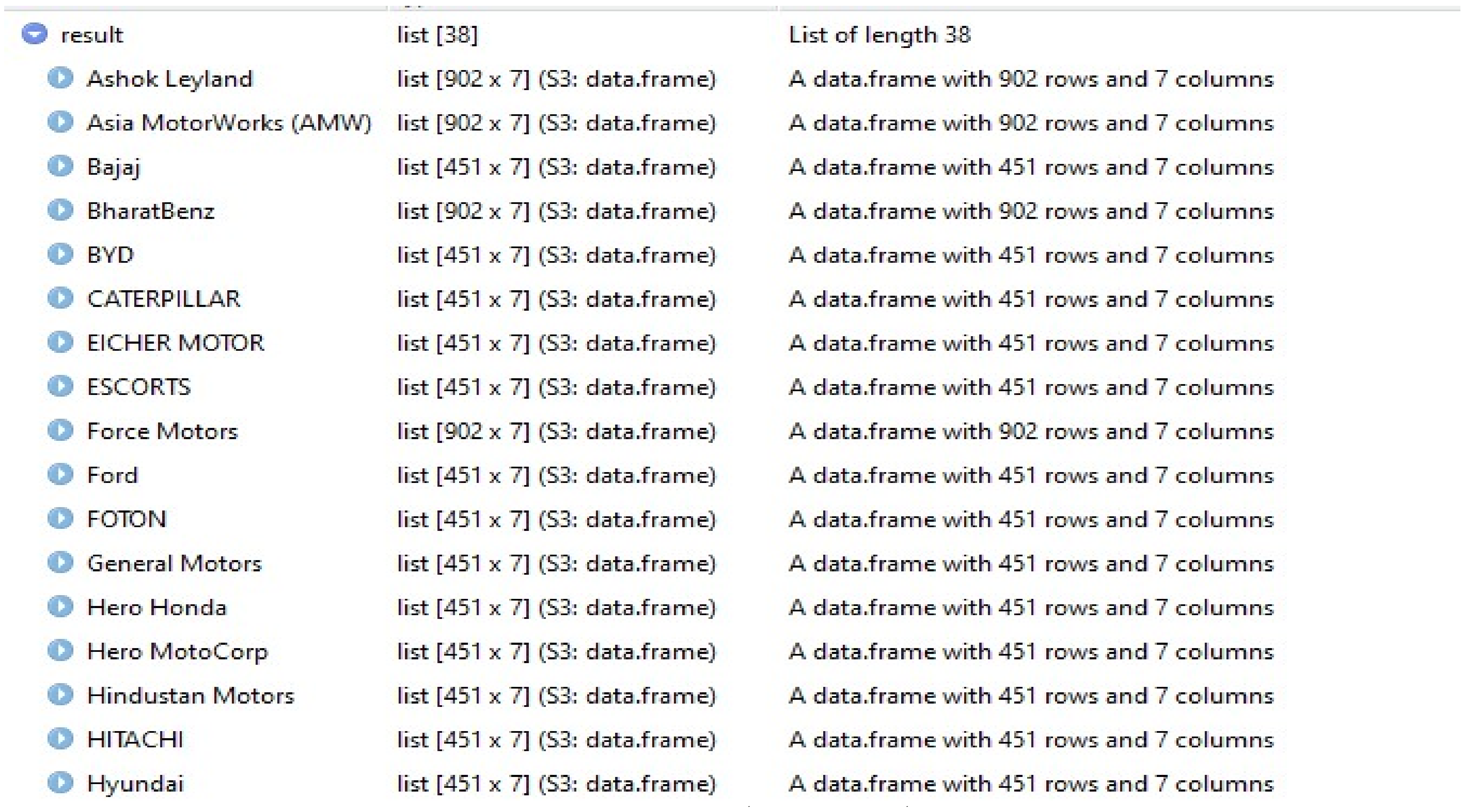
split(data-frame-variable, column-name-for-split**)**

<u>Example: Split dataset by BRAND</u>

```
result = split(data, data$BRAND)
View(result)
```

Splitting

split() function perform partition on the dataset by specific columns/variables.



result	list [38]	List of length 38
Ashok Leyland	list [902 x 7] (S3: data.frame)	A data.frame with 902 rows and 7 columns
Asia MotorWorks (AMW)	list [902 x 7] (S3: data.frame)	A data.frame with 902 rows and 7 columns
Bajaj	list [451 x 7] (S3: data.frame)	A data.frame with 451 rows and 7 columns
BharatBenz	list [902 x 7] (S3: data.frame)	A data.frame with 902 rows and 7 columns
BYD	list [451 x 7] (S3: data.frame)	A data.frame with 451 rows and 7 columns
CATERPILLAR	list [451 x 7] (S3: data.frame)	A data.frame with 451 rows and 7 columns
EICHER MOTOR	list [451 x 7] (S3: data.frame)	A data.frame with 451 rows and 7 columns
ESCORTS	list [451 x 7] (S3: data.frame)	A data.frame with 451 rows and 7 columns
Force Motors	list [902 x 7] (S3: data.frame)	A data.frame with 902 rows and 7 columns
Ford	list [451 x 7] (S3: data.frame)	A data.frame with 451 rows and 7 columns
FOTON	list [451 x 7] (S3: data.frame)	A data.frame with 451 rows and 7 columns
General Motors	list [451 x 7] (S3: data.frame)	A data.frame with 451 rows and 7 columns
Hero Honda	list [451 x 7] (S3: data.frame)	A data.frame with 451 rows and 7 columns
Hero MotoCorp	list [451 x 7] (S3: data.frame)	A data.frame with 451 rows and 7 columns
Hindustan Motors	list [451 x 7] (S3: data.frame)	A data.frame with 451 rows and 7 columns
HITACHI	list [451 x 7] (S3: data.frame)	A data.frame with 451 rows and 7 columns
Hyundai	list [451 x 7] (S3: data.frame)	A data.frame with 451 rows and 7 columns

Subset

subset() function perform data filter based on given conditions.

Syntax:

subset(data-frame-variable, conditions)

<u>Example: Split dataset by AGE</u>

subset(data, data$AGE == 10)

Subset

subset() function perform data filter based on given conditions.

<u>Example: Split dataset by BRAND & AGE</u>

```
subset(data, data$AGE == 10 & data$BRAND=='VOLVO')
```

Subset

Another option to filter data by applying condition in row index.

Syntax:

data-frame-variable[conditions,   column-index]

Example:

data[data$AGE==10,] # Filter where AGE is 10 and all columns

data[data$AGE==10, 2] # Filter where AGE is 10 and 2nd columns

Sorting / Ordering

order() function perform data sorting on data frame, order function apply on row index.

Syntax:

data-frame-variable[order(col-name, decreasing=TRUE), column-index]

Example:

data[order(data$AGE, decreasing=TRUE),]

Add New Column

Adding new columns to existing DataFrame is quite easy, here are the option

- using cbind() function
- using $ symbol

Add New Column

cbind() function must have same number of rows while creating new columns.

```
CITY = rep(c("TUMKUR"), times=22550)
data= cbind(data, CITY)
View(data)
```

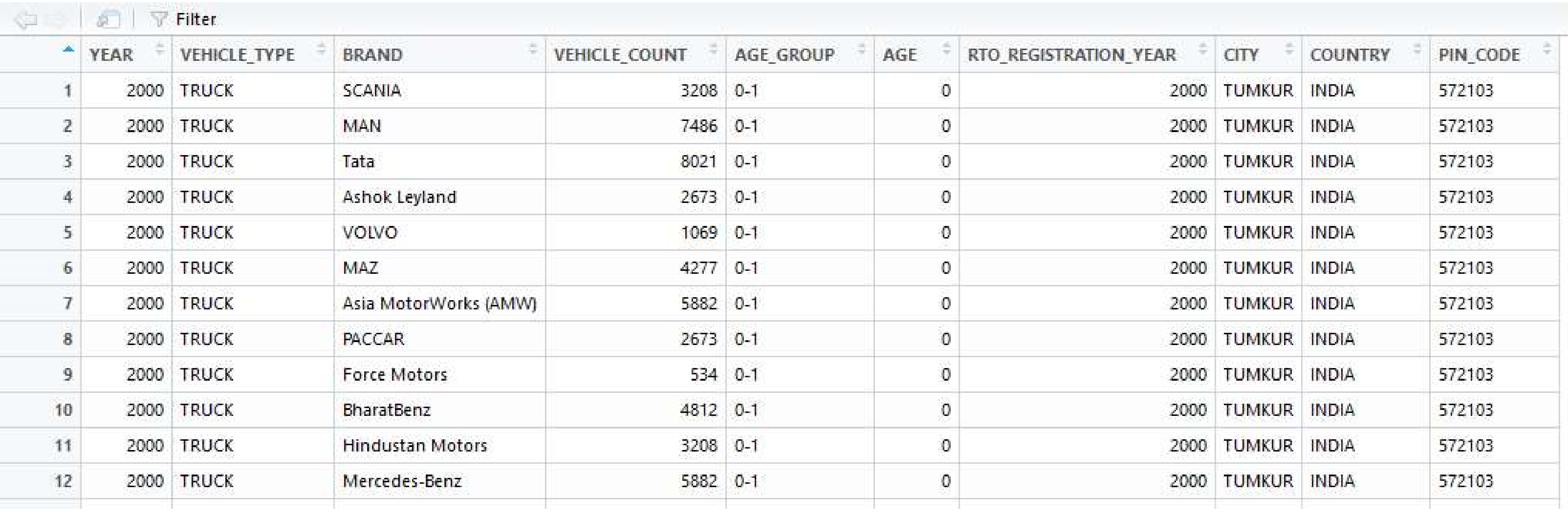
Add New Column

<u>Using $ symbol</u>

data$COUNTRY = "INDIA“
data$PIN_CODE = 572103

View(data)

Data Frame after adding new cols



Filter

	YEAR	VEHICLE_TYPE	BRAND	VEHICLE_COUNT	AGE_GROUP	AGE	RTO_REGISTRATION_YEAR	CITY	COUNTRY	PIN_CODE
1	2000	TRUCK	SCANIA	3208	0-1	0	2000	TUMKUR	INDIA	572103
2	2000	TRUCK	MAN	7486	0-1	0	2000	TUMKUR	INDIA	572103
3	2000	TRUCK	Tata	8021	0-1	0	2000	TUMKUR	INDIA	572103
4	2000	TRUCK	Ashok Leyland	2673	0-1	0	2000	TUMKUR	INDIA	572103
5	2000	TRUCK	VOLVO	1069	0-1	0	2000	TUMKUR	INDIA	572103
6	2000	TRUCK	MAZ	4277	0-1	0	2000	TUMKUR	INDIA	572103
7	2000	TRUCK	Asia MotorWorks (AMW)	5882	0-1	0	2000	TUMKUR	INDIA	572103
8	2000	TRUCK	PACCAR	2673	0-1	0	2000	TUMKUR	INDIA	572103
9	2000	TRUCK	Force Motors	534	0-1	0	2000	TUMKUR	INDIA	572103
10	2000	TRUCK	BharatBenz	4812	0-1	0	2000	TUMKUR	INDIA	572103
11	2000	TRUCK	Hindustan Motors	3208	0-1	0	2000	TUMKUR	INDIA	572103
12	2000	TRUCK	Mercedes-Benz	5882	0-1	0	2000	TUMKUR	INDIA	572103

Vehicle Park (R-Code)

R-code is available for basic exercises

Double click to download →

VehiclePark.R

OR

Download code from give link:

https://raw.githubusercontent.com/sitmbadept/sitmbadept.github.io/main/BDTM/CODES/R/VehiclePark.R