Data Science

Data Exploration

Agenda

01 Objects in R

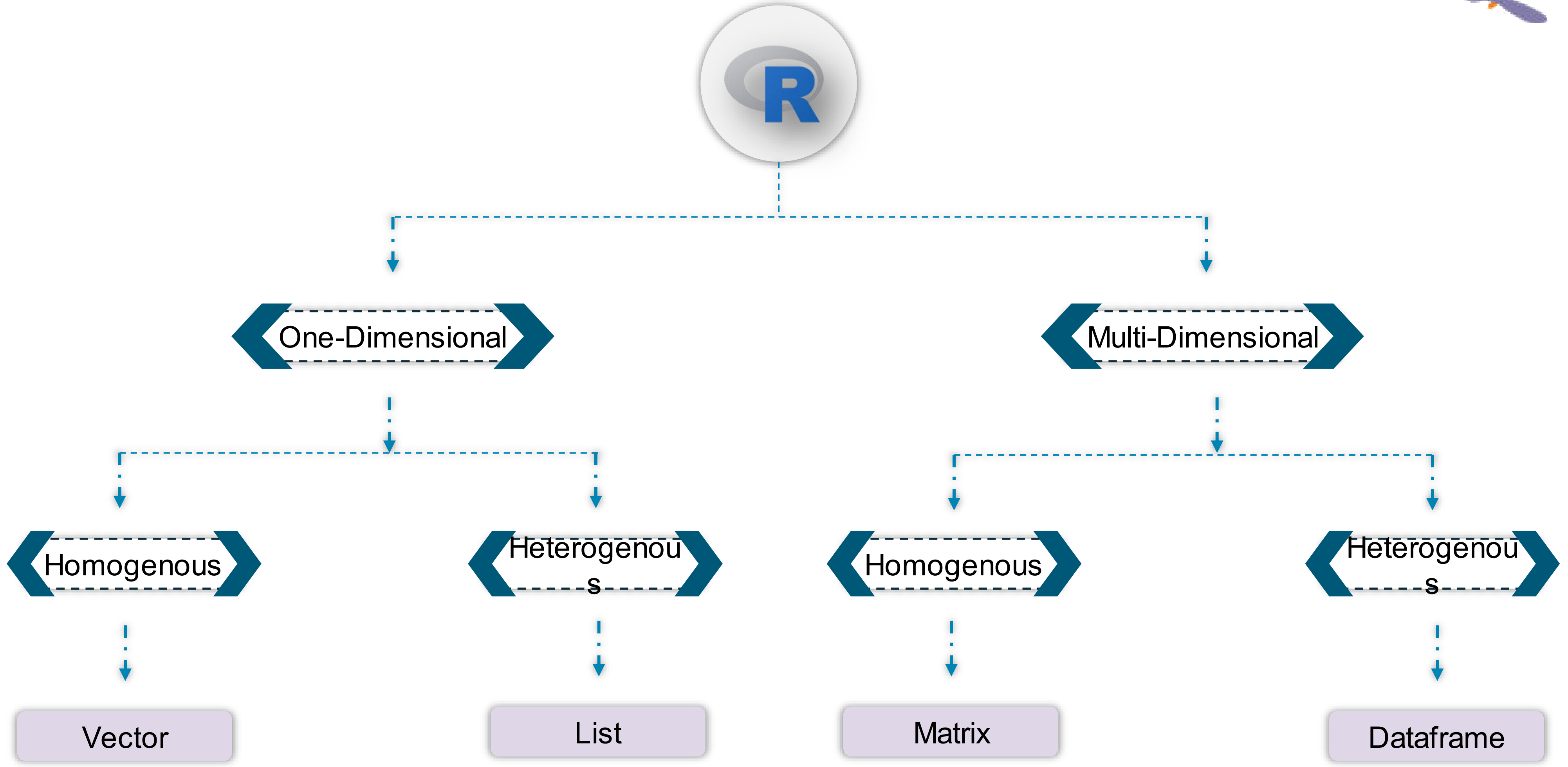
02 Flow Control Statements

03 Inbuilt Functions

04 USER-DEFINED FUNCTIONS

Objects in R

Objects in R



R
One-Dimensional
Multi-Dimensional
Homogenous
Heterogenous
Homogenous
Heterogenous
Vector
List
Matrix
Dataframe

Vectors

Vectors

Vector is a linear object which contains homogenous elements. So, it is a collection of values that all have the same data type

```
c(1,2,3)
```

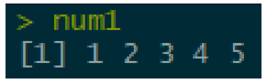
```
c(TRUE,FALSE)
```

Creating a Vector

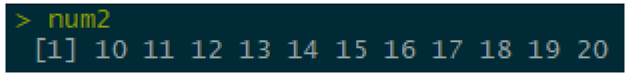
Creating a numeric vector

num1<-c(1,2,3,4,5)

num2<-c(10:20)



```
> num1
[1] 1 2 3 4 5
```

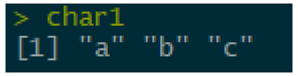


```
> num2
 [1] 10 11 12 13 14 15 16 17 18 19 20
```

Creating a Vector

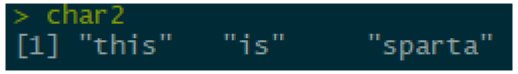
Creating a character vector

char1<-c("a","b","c")



```
> char1
[1] "a" "b" "c"
```

char2<-c("this","is","sparta")



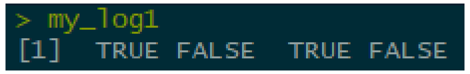
```
> char2
[1] "this"    "is"      "sparta"
```

Creating a Vector

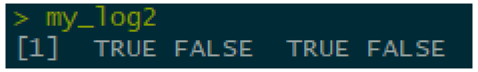
Creating a Logical vector

my_log1<-c(TRUE,FALSE,TRUE,FALSE)

my_log2<-c(T,F,T,F)



```
> my_log1
[1]  TRUE FALSE  TRUE FALSE
```

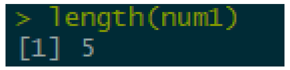


```
> my_log2
[1]  TRUE FALSE  TRUE FALSE
```

Length of Vector

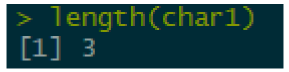
Finding the length of vectors

length(num1)



```
> length(num1)
[1] 5
```

length(char1)

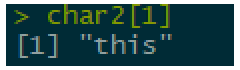


```
> length(char1)
[1] 3
```

Accessing Elements from a Vector

Accessing 2nd element from 'char2'

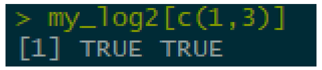
char2[1]



```
> char2[1]
[1] "this"
```

Accessing 1st & 3rd elements from 'my_log2'

my_log2[c(1,3)]

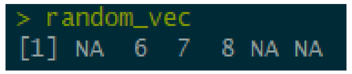


```
> my_log2[c(1,3)]
[1] TRUE TRUE
```

Missing Values in a Vector

Suppose we have a vector:

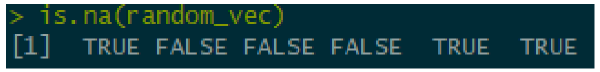
random_vec<-c(NA,6,7,8,NA,NA)



```
> random_vec
[1] NA  6  7  8 NA NA
```

Checking for NA values

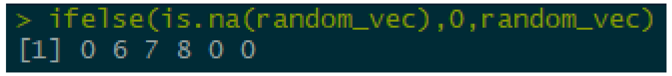
is.na(random_vec)



```
> is.na(random_vec)
[1]  TRUE FALSE FALSE FALSE  TRUE  TRUE
```

Converting NA to 0 using ifelse

ifelse(is.na(random_vec),0,random_vec)



```
> ifelse(is.na(random_vec),0,random_vec)
[1] 0 6 7 8 0 0
```

List

List

List is a linear object which contains heterogenous elements. A list allows you to gather a variety of objects under one name. A list may contain a combination of vectors, matrices, data frames, and even other lists

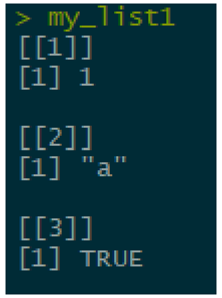
list(101,"Sparta")

list(TRUE,5+2i)

Creating a List

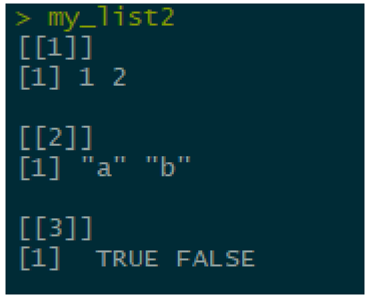
You create a list using the list() function: mylist_data <- list(object1, object2,object3, …)

my_list1<-list(1,"a",TRUE)



```
> my_list1
[[1]]
[1] 1

[[2]]
[1] "a"

[[3]]
[1] TRUE
```

my_list2<-list(c(1,2),c("a","b"),c(TRUE,FALSE))

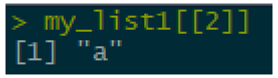


```
> my_list2
[[1]]
[1] 1 2

[[2]]
[1] "a" "b"

[[3]]
[1]  TRUE FALSE
```

Accessing List Elements

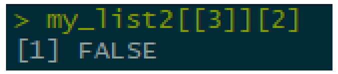
The elements of a list can be retrieved by using double square brackets

my_list1[[2]]



```
> my_list1[[2]]
[1] "a"
```

my_list2[[3]][2]



```
> my_list2[[3]][2]
[1] FALSE
```

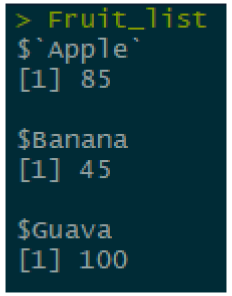
Naming Elements of a List

Giving names to the elements

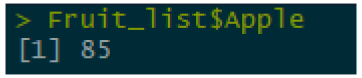
Extracting ‘Apple’

Fruit_list <- list(Apple = 85, Banana = 45, Guava = 100)

Fruit_list$Apple



```
> Fruit_list
$`Apple`
[1] 85

$Banana
[1] 45

$Guava
[1] 100
```



```
> Fruit_list$Apple
[1] 85
```

Matrix

Matrix

Matrix is a 2-D object which contains homogenous elements

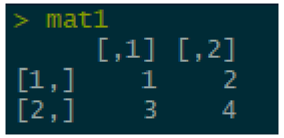
```
matrix(c(1:8),nrow=2)
```

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

Creating a Matrix

Creating a numeric Matrix

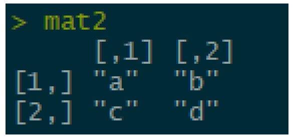
```
mat1<-matrix(c(1,2,3,4),nrow=2,byrow  = T)
```



```
> mat1
     [,1] [,2]
[1,]    1    2
[2,]    3    4
```

Creating a character Matrix

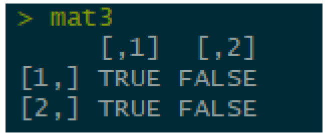
```
mat2<-matrix(c("a","b","c","d"),nrow=2,byrow  = T)
```



```
> mat2
     [,1] [,2]
[1,] "a"  "b"
[2,] "c"  "d"
```

Creating a Logical Matrix

```
mat3<-matrix(c(T,F,T,F),nrow=2,byrow  = T)
```

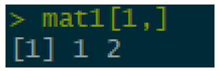


```
> mat3
     [,1]  [,2]
[1,] TRUE FALSE
[2,] TRUE FALSE
```

Accessing Matrix Elements

Getting the first row

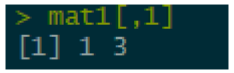
mat1[1,]



```
> mat1[1,]
[1] 1 2
```

Getting the first column

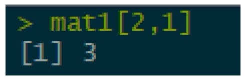
mat1[,1]



```
> mat1[,1]
[1] 1 3
```

Getting element at 2nd row, 1st column

mat1[2,1]



```
> mat1[2,1]
[1] 3
```

Matrix Transpose

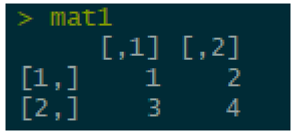
Transpose of a matrix is to convert row into column and column into row

Original Matrix

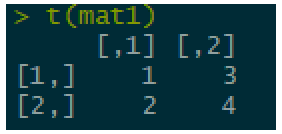
Transposed Matrix

mat1

t(mat1)



```
> mat1
     [,1] [,2]
[1,]    1    2
[2,]    3    4
```

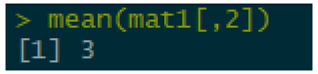


```
> t(mat1)
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

Calculation on Rows & Columns of the Matrix

Calculation of mean on 2nd column of matrix

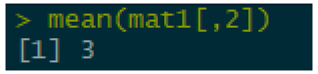
mean(mat1[,2])



```
> mean(mat1[,2])
[1] 3
```

Calculation of mean on 2nd column of matrix

mean(mat1[,2])



```
> mean(mat1[,2])
[1] 3
```

Array

Array

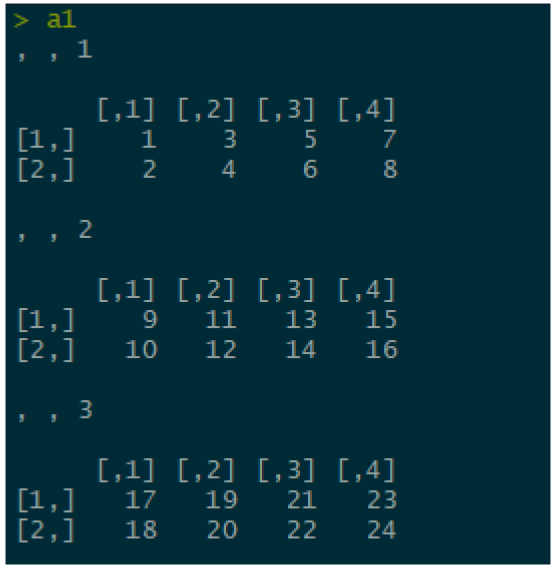
Arrays are homogenous objects which have more than 2 dimensions. It takes vectors as input and uses the values within the dim parameter to form an array

```
array(c(vector1,vector2),dim=(3,3,2))
```

Creating an Array

Creating an array with three dimensions with the numbers 1 to 24

```
a1 <- array(1:24,dim = c(2,4,3))
```

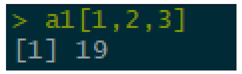


```
> a1
, , 1

     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8

, , 2

     [,1] [,2] [,3] [,4]
[1,]    9   11   13   15
[2,]   10   12   14   16

, , 3

     [,1] [,2] [,3] [,4]
[1,]   17   19   21   23
[2,]   18   20   22   24
```

Accessing Array Elements

Selecting the element at 1st row, 2nd column from the 3rd matrix

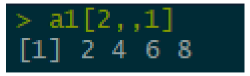
a1[1,2,3]



```
> a1[1,2,3]
[1] 19
```

Selecting the entire 2nd row from the 1st matrix

a1[2,,1]



```
> a1[2,,1]
[1] 2 4 6 8
```

Factor

Factors are objects which are used to categorize the data & store it as levels

```
my_data<-c("Male","Female","Female","Male")
```

```
as.factor(data)
```

```
Levels: Female Male
```

Dataframe

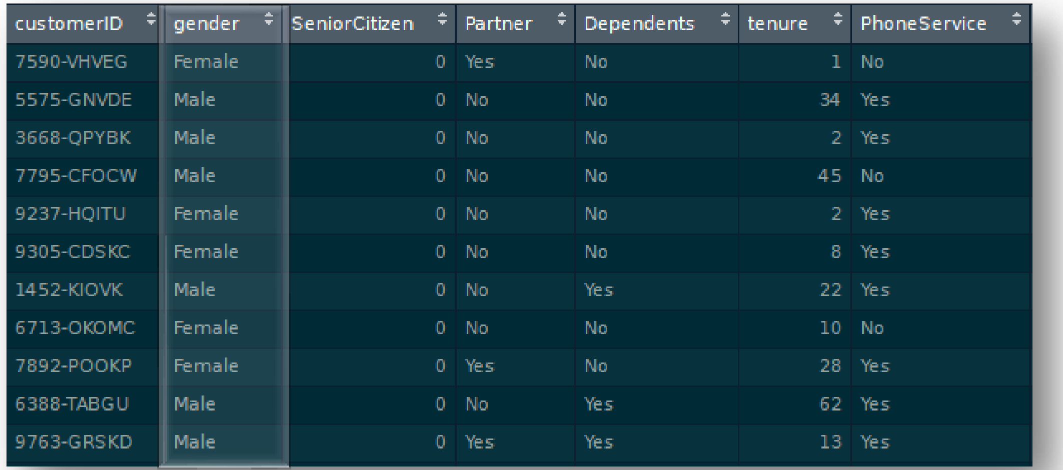
A dataframe is a 2-D table where each column comprises of homogenous elements & each row may contain either homogenous or heterogenous elements

```
data.frame(Name=c("Sam","Bob"),Age=c(32,48))
```

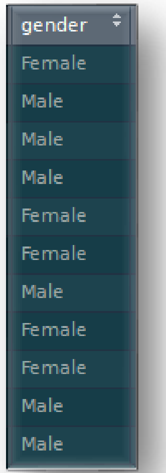
```
Name  Age
Sam   32
Bob   48
```

Extracting Individual Columns

customer_churn$gender



customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService
7590-VHVEG	Female	0	Yes	No	1	No
5575-GNVDE	Male	0	No	No	34	Yes
3668-QPYBK	Male	0	No	No	2	Yes
7795-CFOCW	Male	0	No	No	45	No
9237-HQITU	Female	0	No	No	2	Yes
9305-CDSKC	Female	0	No	No	8	Yes
1452-KIOVK	Male	0	No	Yes	22	Yes
6713-OKOMC	Female	0	No	No	10	No
7892-POOKP	Female	0	Yes	No	28	Yes
6388-TABGU	Male	0	No	Yes	62	Yes
9763-GRSKD	Male	0	Yes	Yes	13	Yes

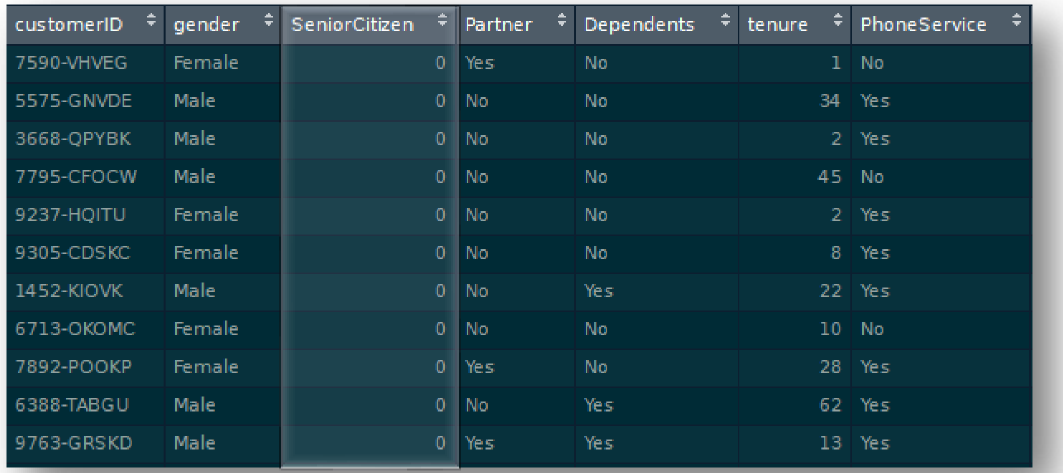


gender
Female
Male
Male
Male
Female
Female
Male
Female
Female
Male
Male

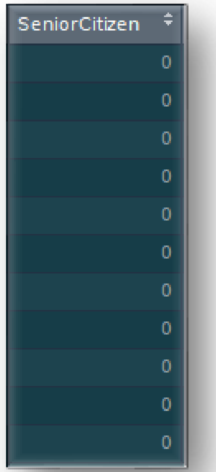
Dataframe_Name$Column_Name

Extracting Individual Columns

customer_churn[,3]



customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService
7590-VHVEG	Female	0	Yes	No	1	No
5575-GNVDE	Male	0	No	No	34	Yes
3668-QPYBK	Male	0	No	No	2	Yes
7795-CFOCW	Male	0	No	No	45	No
9237-HQITU	Female	0	No	No	2	Yes
9305-CDSKC	Female	0	No	No	8	Yes
1452-KIOVK	Male	0	No	Yes	22	Yes
6713-OKOMC	Female	0	No	No	10	No
7892-POOKP	Female	0	Yes	No	28	Yes
6388-TABGU	Male	0	No	Yes	62	Yes
9763-GRSKD	Male	0	Yes	Yes	13	Yes



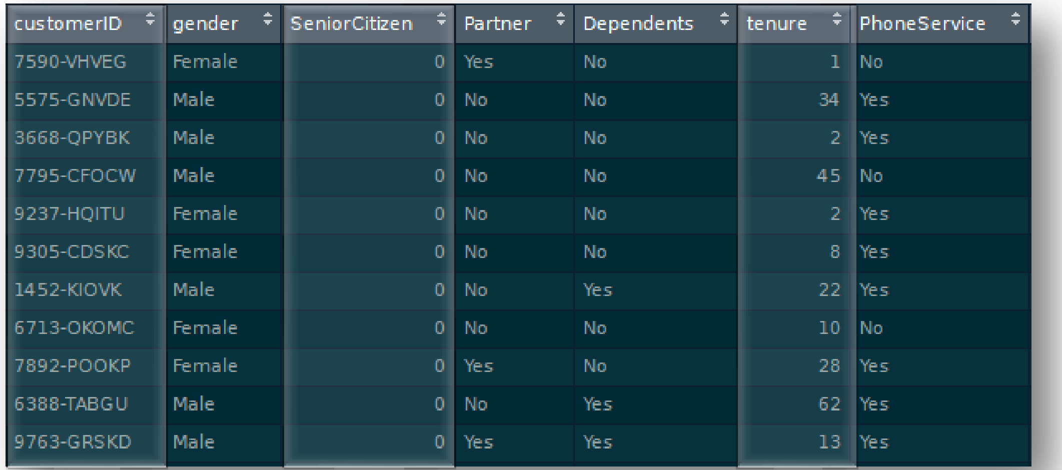
SeniorCitizen
0
0
0
0
0
0
0
0
0
0
0

Dataframe_Name[,col_number]

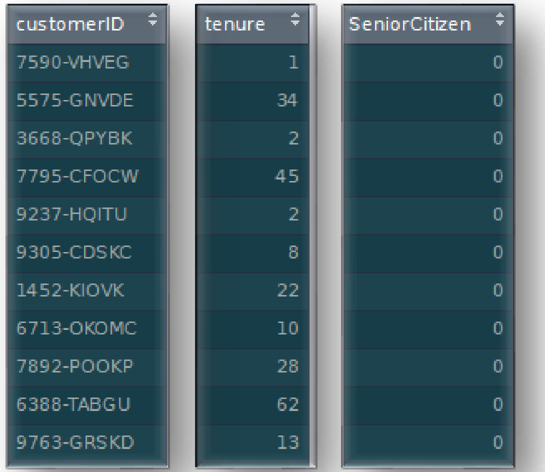
Extract 3rd column

Extracting Multiple Columns

customer_churn[,c(1,3,6)]



customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService
7590-VHVEG	Female	0	Yes	No	1	No
5575-GNVDE	Male	0	No	No	34	Yes
3668-QPYBK	Male	0	No	No	2	Yes
7795-CFOCW	Male	0	No	No	45	No
9237-HQITU	Female	0	No	No	2	Yes
9305-CDSKC	Female	0	No	No	8	Yes
1452-KIOVK	Male	0	No	Yes	22	Yes
6713-OKOMC	Female	0	No	No	10	No
7892-POOKP	Female	0	Yes	No	28	Yes
6388-TABGU	Male	0	No	Yes	62	Yes
9763-GRSKD	Male	0	Yes	Yes	13	Yes



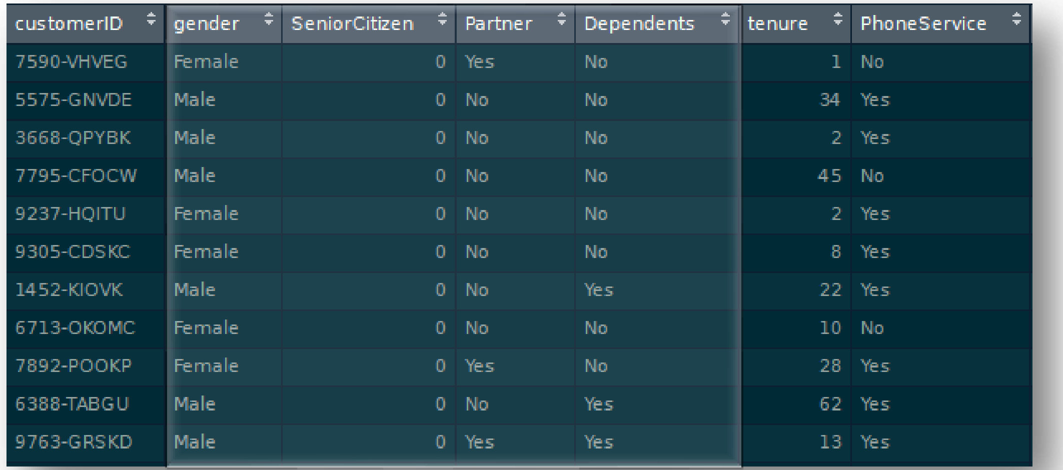
customerID	tenure	SeniorCitizen
7590-VHVEG	1	0
5575-GNVDE	34	0
3668-QPYBK	2	0
7795-CFOCW	45	0
9237-HQITU	2	0
9305-CDSKC	8	0
1452-KIOVK	22	0
6713-OKOMC	10	0
7892-POOKP	28	0
6388-TABGU	62	0
9763-GRSKD	13	0

Dataframe_Name[,c(col_num1, col_num2, col_num3)]

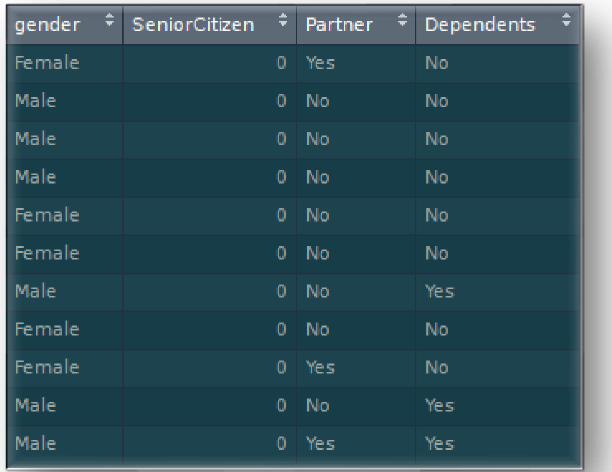
Extract 1st, 3rd & 6th columns

Extracting Continuous Sequence of Columns

customer_churn[,2:5]



customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService
7590-VHVEG	Female	0	Yes	No	1	No
5575-GNVDE	Male	0	No	No	34	Yes
3668-QPYBK	Male	0	No	No	2	Yes
7795-CFOCW	Male	0	No	No	45	No
9237-HQITU	Female	0	No	No	2	Yes
9305-CDSKC	Female	0	No	No	8	Yes
1452-KIOVK	Male	0	No	Yes	22	Yes
6713-OKOMC	Female	0	No	No	10	No
7892-POOKP	Female	0	Yes	No	28	Yes
6388-TABGU	Male	0	No	Yes	62	Yes
9763-GRSKD	Male	0	Yes	Yes	13	Yes



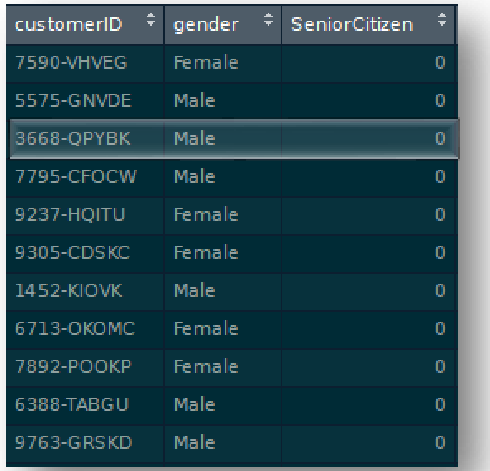
gender	SeniorCitizen	Partner	Dependents
Female	0	Yes	No
Male	0	No	No
Male	0	No	No
Male	0	No	No
Female	0	No	No
Female	0	No	No
Male	0	No	Yes
Female	0	No	No
Female	0	Yes	No
Male	0	No	Yes
Male	0	Yes	Yes

Dataframe_Name[,col_num_x:col_num_y]

Extracting all the columns from ***column number 2*** to **column number 5**

Extracting Rows

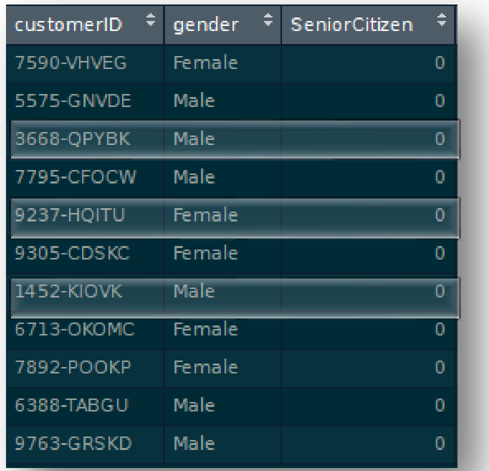
Dataframe_Name[x,]



customerID	gender	SeniorCitizen
7590-VHVEG	Female	0
5575-GNVDE	Male	0
3668-QPYBK	Male	0
7795-CFOCW	Male	0
9237-HQITU	Female	0
9305-CDSKC	Female	0
1452-KIOVK	Male	0
6713-OKOMC	Female	0
7892-POOKP	Female	0
6388-TABGU	Male	0
9763-GRSKD	Male	0

customer_churn[3,]

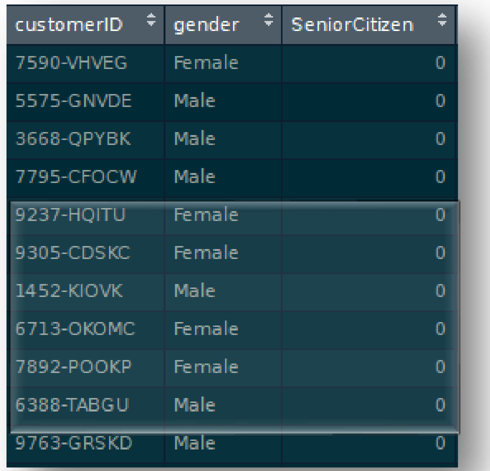
Extracting row number 3



customerID	gender	SeniorCitizen
7590-VHVEG	Female	0
5575-GNVDE	Male	0
3668-QPYBK	Male	0
7795-CFOCW	Male	0
9237-HQITU	Female	0
9305-CDSKC	Female	0
1452-KIOVK	Male	0
6713-OKOMC	Female	0
7892-POOKP	Female	0
6388-TABGU	Male	0
9763-GRSKD	Male	0

customer_churn[c(3,5,7),]

Extracting row numbers 3, 5 & 7



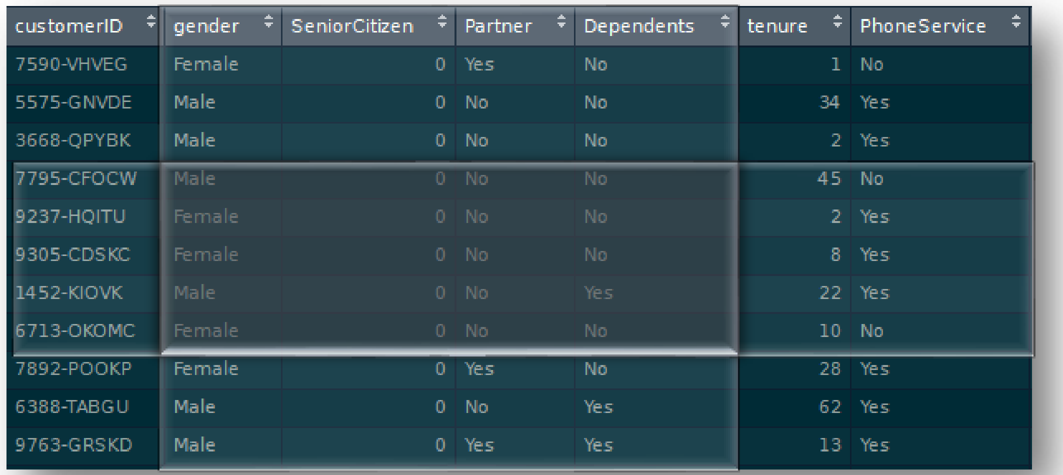
customerID	gender	SeniorCitizen
7590-VHVEG	Female	0
5575-GNVDE	Male	0
3668-QPYBK	Male	0
7795-CFOCW	Male	0
9237-HQITU	Female	0
9305-CDSKC	Female	0
1452-KIOVK	Male	0
6713-OKOMC	Female	0
7892-POOKP	Female	0
6388-TABGU	Male	0
9763-GRSKD	Male	0

customer_churn[5:10,]

Extracting all the rows from 5 to 10

Extracting Rows & Columns Together

Dataframe_Name[x1:xn,y1,yn)]



customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService
7590-VHVEG	Female	0	Yes	No	1	No
5575-GNVDE	Male	0	No	No	34	Yes
3668-QPYBK	Male	0	No	No	2	Yes
7795-CFOCW	Male	0	No	No	45	No
9237-HQITU	Female	0	No	No	2	Yes
9305-CDSKC	Female	0	No	No	8	Yes
1452-KIOVK	Male	0	No	Yes	22	Yes
6713-OKOMC	Female	0	No	No	10	No
7892-POOKP	Female	0	Yes	No	28	Yes
6388-TABGU	Male	0	No	Yes	62	Yes
9763-GRSKD	Male	0	Yes	Yes	13	Yes

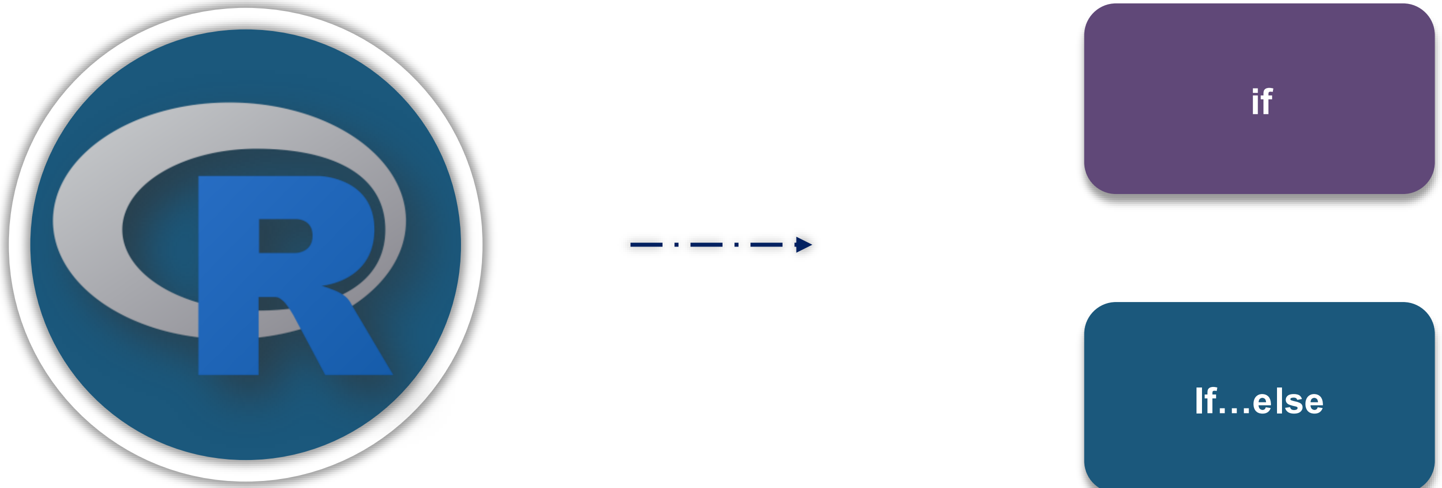
customer_churn[4:8,2:5]

Extracting all the ***rows from 4 to 8*** & all the ***columns from 2 to 5***

Decision Making Statements

Decision Making Statements

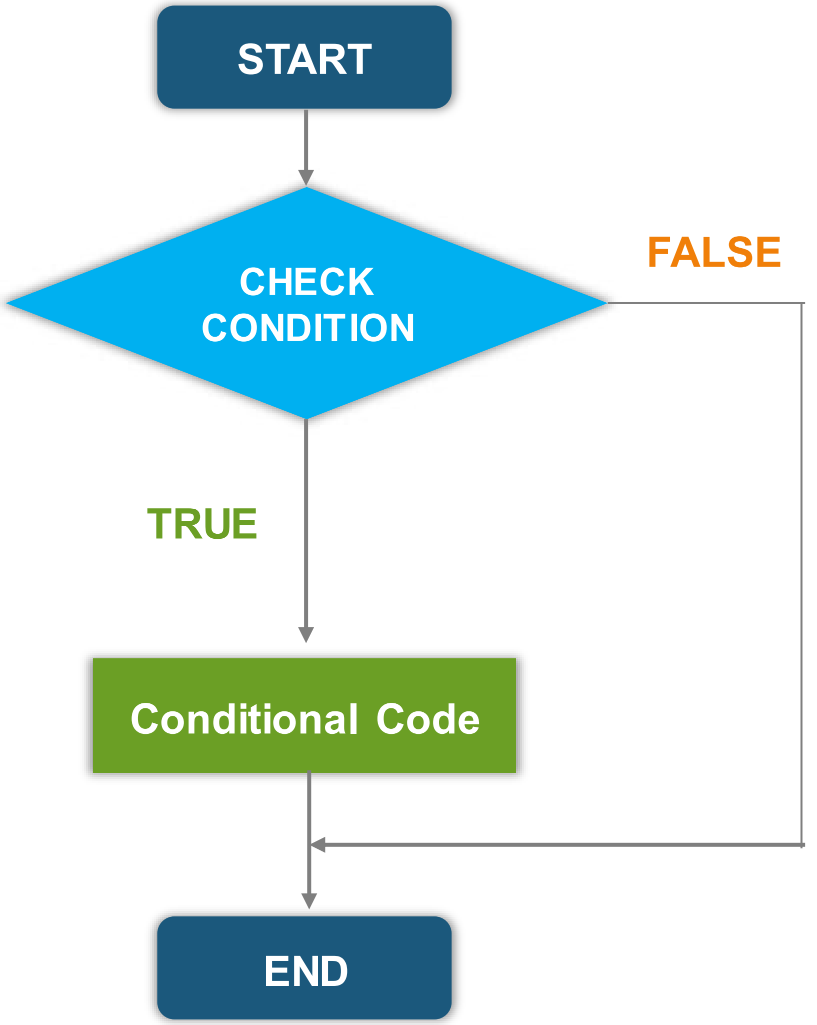
Decision Making Statements help in making a decision on the basis of a condition



If Statement

Syntax

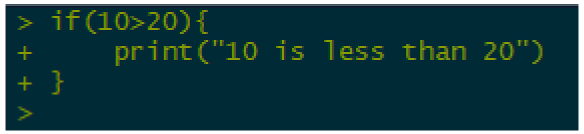
```
if (condition){
      statements...
}
```



If Statement Example

True Condition

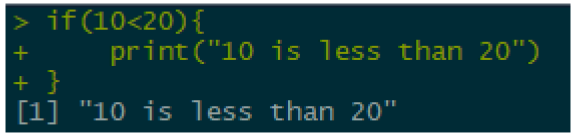
```
if(10>20){
  print("10 is less than
20")
}
```



```
> if(10>20){
+     print("10 is less than 20")
+ }
>
```

False Condition

```
if(10<20){
  print("10 is less than
20")
}
```

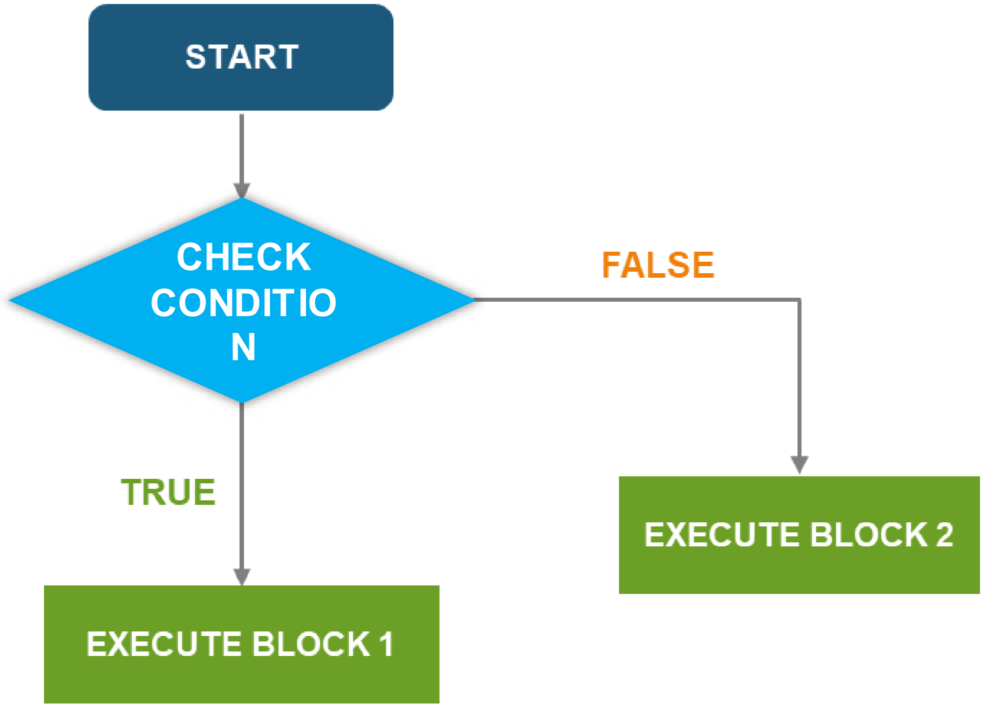


```
> if(10<20){
+     print("10 is less than 20")
+ }
[1] "10 is less than 20"
```

If….else Statement

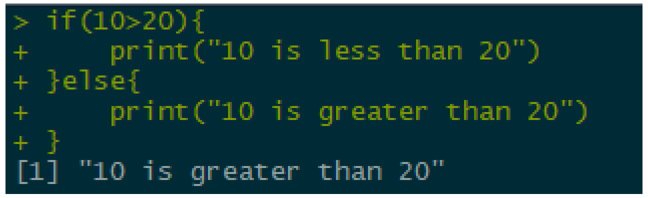
Syntax

```
if (condition){
statements….
}else{
statements…
}
```



If….else Statement Example

```
if(10>20){
  print("10 is less than 20")
}else{
  print("10 is greater than 20")
}
```



```
> if(10>20){
+     print("10 is less than 20")
+ }else{
+     print("10 is greater than 20")
+ }
[1] "10 is greater than 20"
```

Looping Statements

Looping Statements

Looping Statements help in iterating a certain task on the basis of a condition

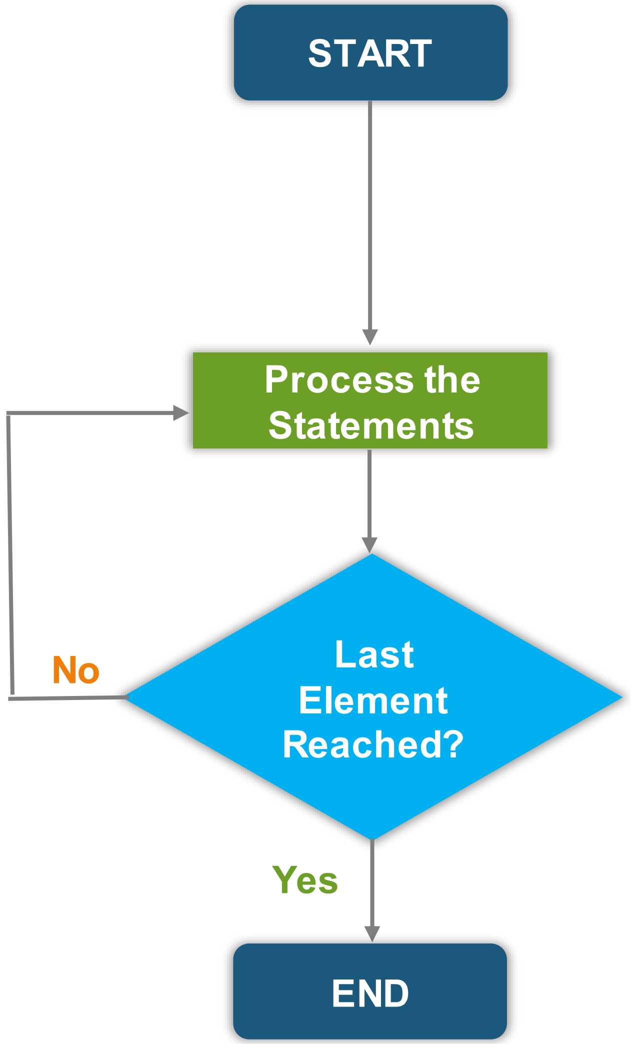
for

while

For Loop

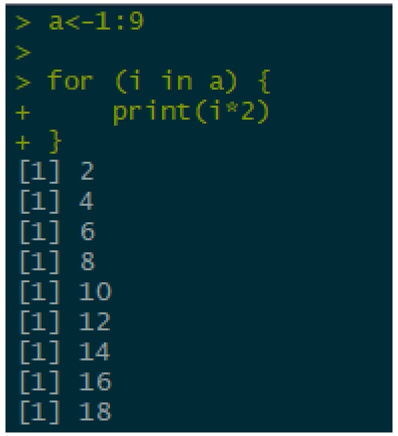
Syntax

```
for (variable in vector) {
  statements…..
}
```



For Loop Example

```
a<-1:9

for (i in a) {
    print(i*2)
}
```

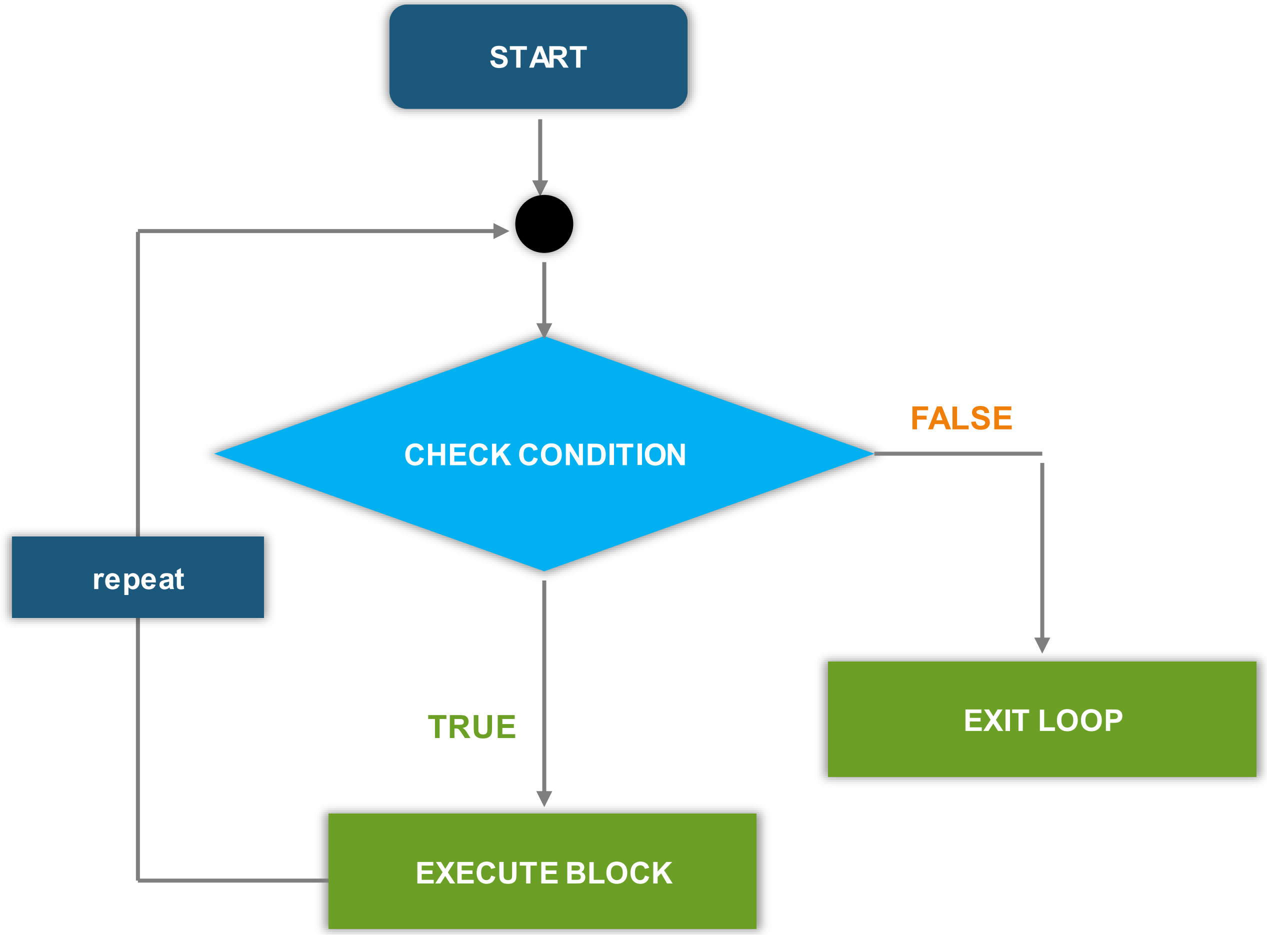


```
> a<-1:9
>
> for (i in a) {
+       print(i*2)
+ }
[1] 2
[1] 4
[1] 6
[1] 8
[1] 10
[1] 12
[1] 14
[1] 16
[1] 18
```

While Loop

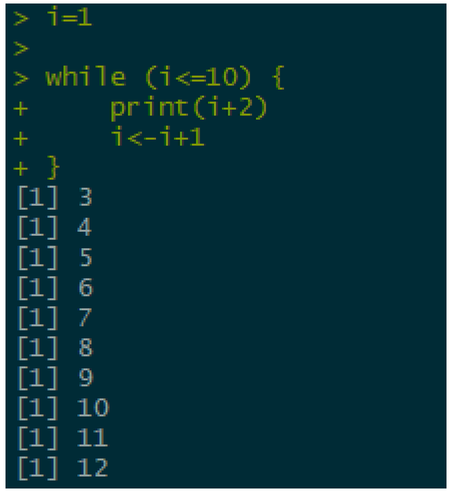
Syntax

```
while(test_expression){
statements……
}
```



While Loop Example

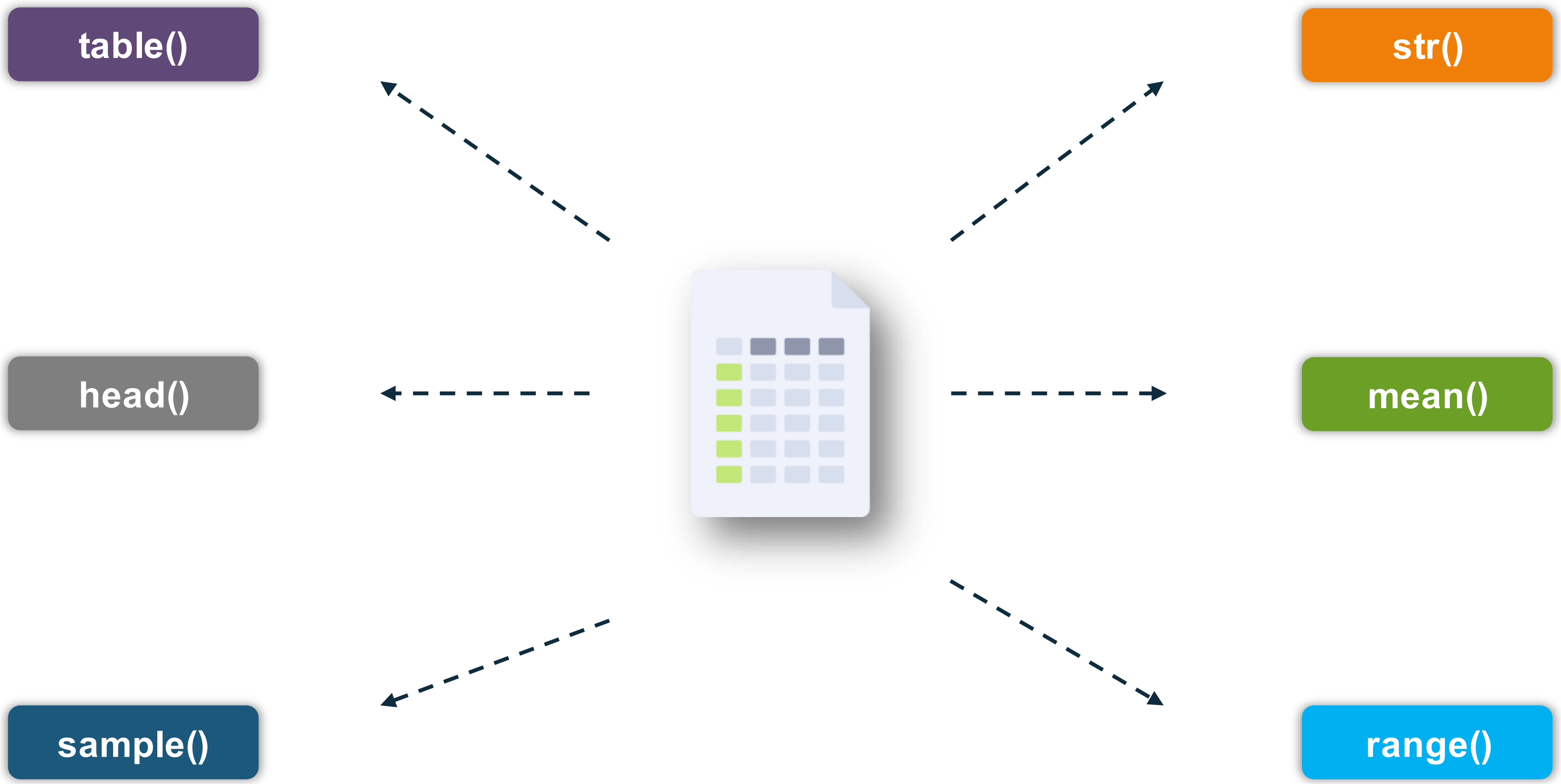
```
i=1

while (i<=10) {
  print(i+2)
  i<-i+1
}
```



```
> i=1
>
> while (i<=10) {
+     print(i+2)
+     i<-i+1
+ }
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
[1] 10
[1] 11
[1] 12
```

Inbuilt Functions

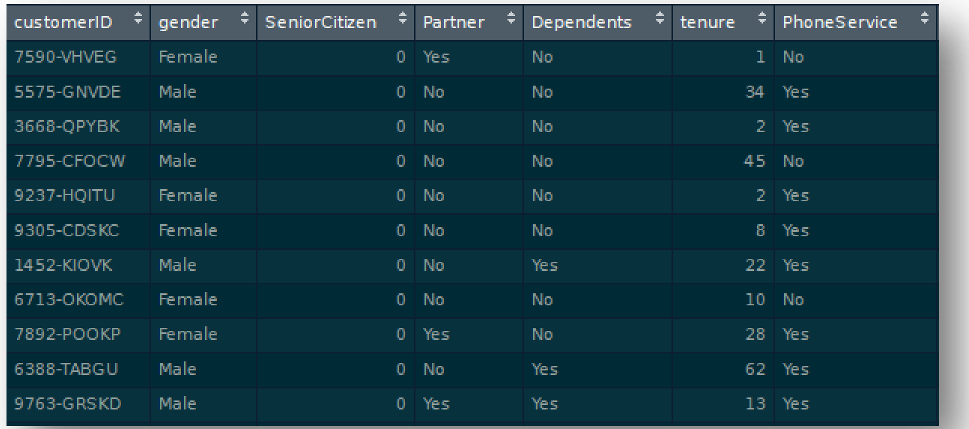
Inbuilt Functions



read.csv()

read.csv() function is used to read a .csv file into R

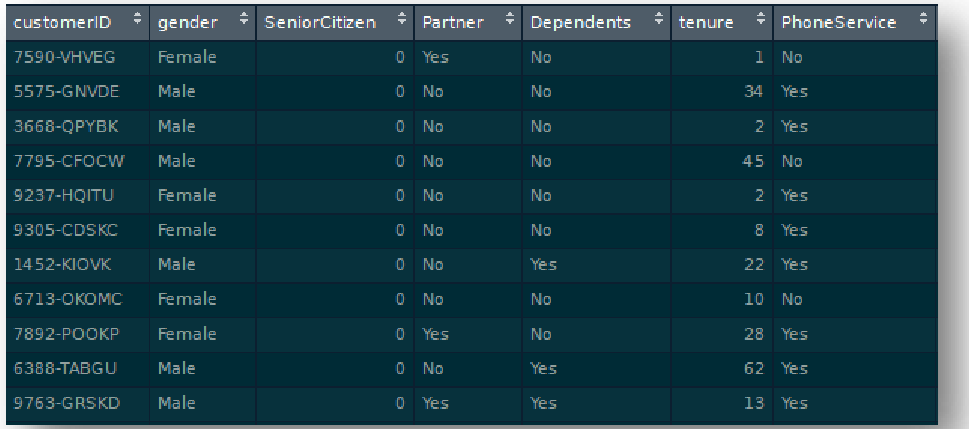
```
read.csv("customer_churn.csv")
```



customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService
7590-VHVEG	Female	0	Yes	No	1	No
5575-GNVDE	Male	0	No	No	34	Yes
3668-QPYBK	Male	0	No	No	2	Yes
7795-CFOCW	Male	0	No	No	45	No
9237-HQITU	Female	0	No	No	2	Yes
9305-CDSKC	Female	0	No	No	8	Yes
1452-KIOVK	Male	0	No	Yes	22	Yes
6713-OKOMC	Female	0	No	No	10	No
7892-POOKP	Female	0	Yes	No	28	Yes
6388-TABGU	Male	0	No	Yes	62	Yes
9763-GRSKD	Male	0	Yes	Yes	13	Yes

str()

str() function gives the structure of an object



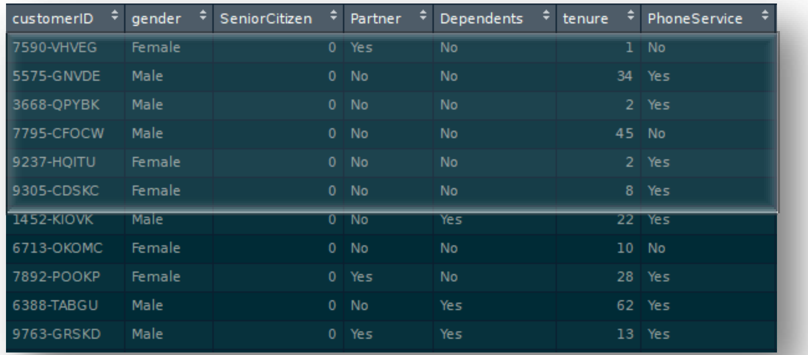
customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService
7590-VHVEG	Female	0	Yes	No	1	No
5575-GNVDE	Male	0	No	No	34	Yes
3668-QPYBK	Male	0	No	No	2	Yes
7795-CFOCW	Male	0	No	No	45	No
9237-HQITU	Female	0	No	No	2	Yes
9305-CDSKC	Female	0	No	No	8	Yes
1452-KIOVK	Male	0	No	Yes	22	Yes
6713-OKOMC	Female	0	No	No	10	No
7892-POOKP	Female	0	Yes	No	28	Yes
6388-TABGU	Male	0	No	Yes	62	Yes
9763-GRSKD	Male	0	Yes	Yes	13	Yes

str(customer_churn)

- Number of rows & columns
- Type/class of columns
- First few values of all columns

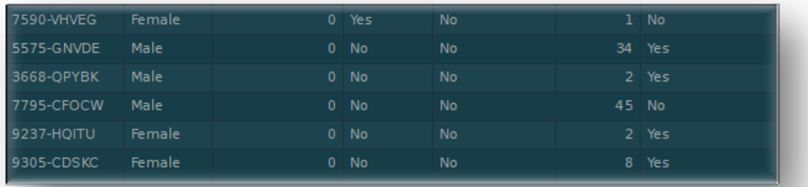
head()

head() function gives the first few records of the dataframe



customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService
7590-VHVEG	Female	0	Yes	No	1	No
5575-GNVDE	Male	0	No	No	34	Yes
3668-QPYBK	Male	0	No	No	2	Yes
7795-CFOCW	Male	0	No	No	45	No
9237-HQITU	Female	0	No	No	2	Yes
9305-CDSKC	Female	0	No	No	8	Yes
1452-KIOVK	Male	0	No	Yes	22	Yes
6713-OKOMC	Female	0	No	No	10	No
7892-POOKP	Female	0	Yes	No	28	Yes
6388-TABGU	Male	0	No	Yes	62	Yes
9763-GRSKD	Male	0	Yes	Yes	13	Yes

head(customer_churn)



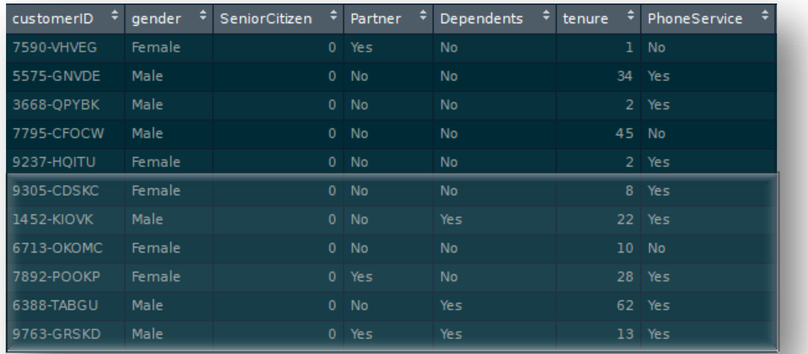
7590-VHVEG	Female	0	Yes	No	1	No
5575-GNVDE	Male	0	No	No	34	Yes
3668-QPYBK	Male	0	No	No	2	Yes
7795-CFOCW	Male	0	No	No	45	No
9237-HQITU	Female	0	No	No	2	Yes
9305-CDSKC	Female	0	No	No	8	Yes

Customer_churn dataframe

First 6 records

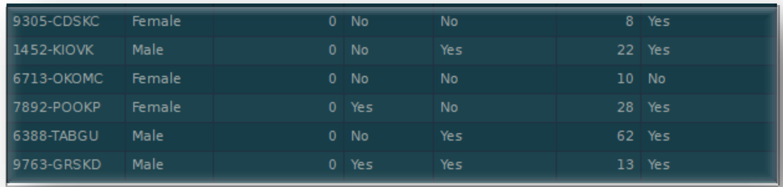
tail()

tail() function gives the last few records of the dataframe



customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService
7590-VHVEG	Female	0	Yes	No	1	No
5575-GNVDE	Male	0	No	No	34	Yes
3668-QPYBK	Male	0	No	No	2	Yes
7795-CFOCW	Male	0	No	No	45	No
9237-HQITU	Female	0	No	No	2	Yes
9305-CDSKC	Female	0	No	No	8	Yes
1452-KIOVK	Male	0	No	Yes	22	Yes
6713-OKOMC	Female	0	No	No	10	No
7892-POOKP	Female	0	Yes	No	28	Yes
6388-TABGU	Male	0	No	Yes	62	Yes
9763-GRSKD	Male	0	Yes	Yes	13	Yes

tail(customer_churn)



9305-CDSKC	Female	0	No	No	8	Yes
1452-KIOVK	Male	0	No	Yes	22	Yes
6713-OKOMC	Female	0	No	No	10	No
7892-POOKP	Female	0	Yes	No	28	Yes
6388-TABGU	Male	0	No	Yes	62	Yes
9763-GRSKD	Male	0	Yes	Yes	13	Yes

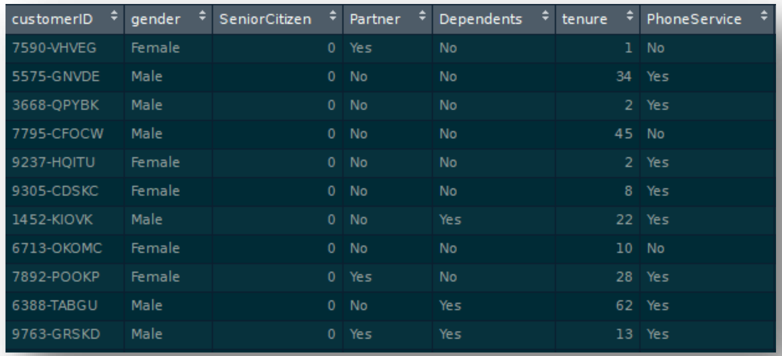
Customer_churn dataframe

Last 6 records

nrow() & ncol()

nrow() displays the number of rows in the dataframe

nrow(customer_churn)

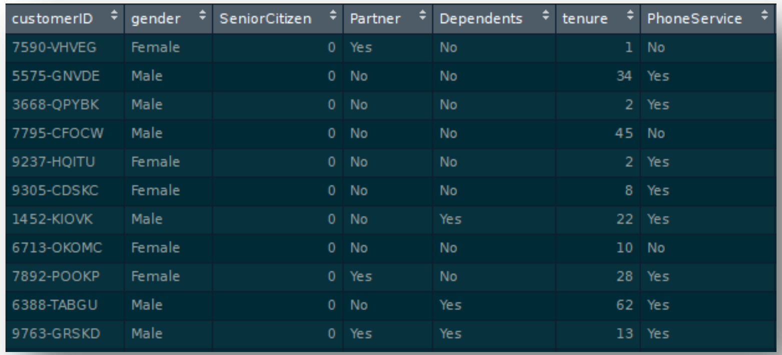


customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService
7590-VHVEG	Female	0	Yes	No	1	No
5575-GNVDE	Male	0	No	No	34	Yes
3668-QPYBK	Male	0	No	No	2	Yes
7795-CFOCW	Male	0	No	No	45	No
9237-HQITU	Female	0	No	No	2	Yes
9305-CDSKC	Female	0	No	No	8	Yes
1452-KIOVK	Male	0	No	Yes	22	Yes
6713-OKOMC	Female	0	No	No	10	No
7892-POOKP	Female	0	Yes	No	28	Yes
6388-TABGU	Male	0	No	Yes	62	Yes
9763-GRSKD	Male	0	Yes	Yes	13	Yes

7043 rows

ncol() displays the number of columns in the dataframe

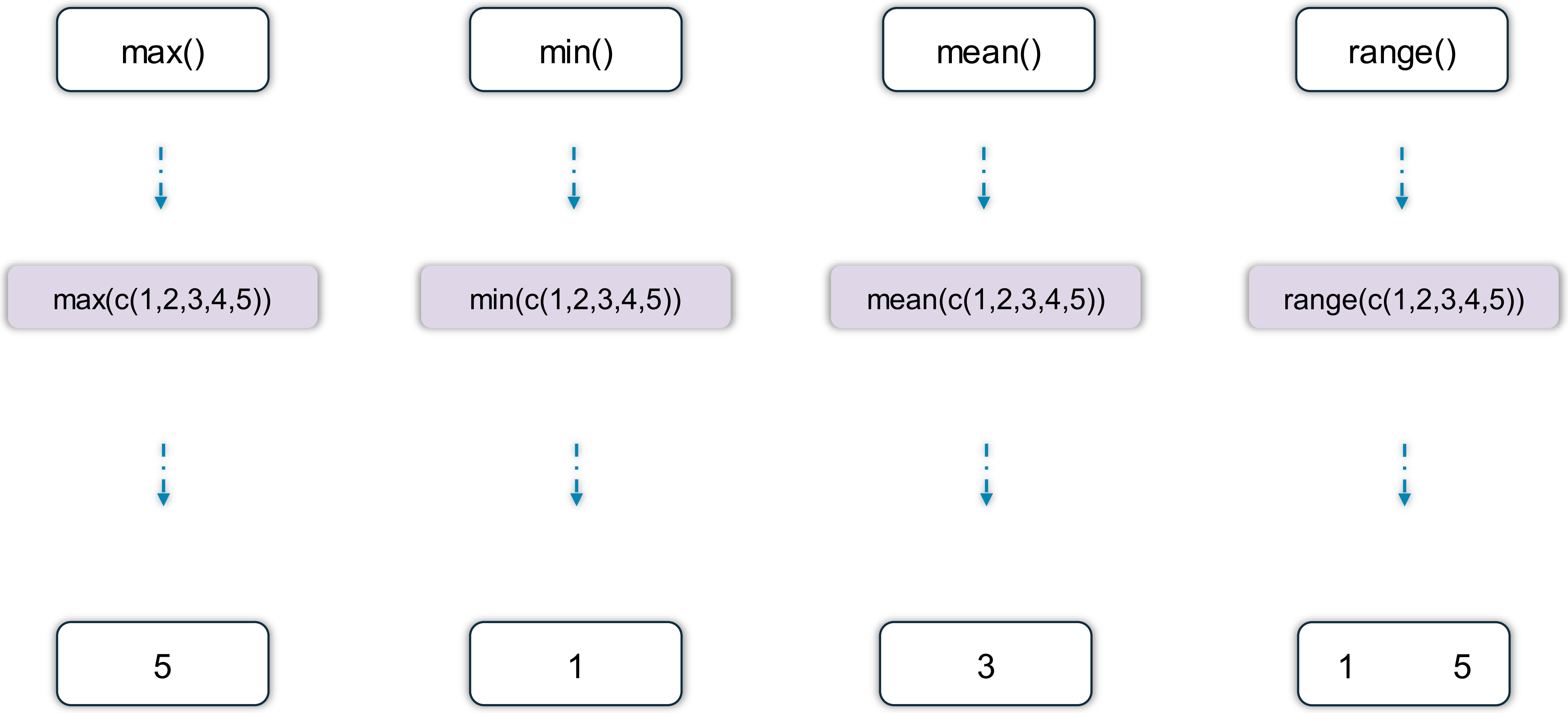
ncol(customer_churn)



customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService
7590-VHVEG	Female	0	Yes	No	1	No
5575-GNVDE	Male	0	No	No	34	Yes
3668-QPYBK	Male	0	No	No	2	Yes
7795-CFOCW	Male	0	No	No	45	No
9237-HQITU	Female	0	No	No	2	Yes
9305-CDSKC	Female	0	No	No	8	Yes
1452-KIOVK	Male	0	No	Yes	22	Yes
6713-OKOMC	Female	0	No	No	10	No
7892-POOKP	Female	0	Yes	No	28	Yes
6388-TABGU	Male	0	No	Yes	62	Yes
9763-GRSKD	Male	0	Yes	Yes	13	Yes

21 columns

Basic Mathematical Functions



max()	min()	mean()	range()
max(c(1,2,3,4,5))	min(c(1,2,3,4,5))	mean(c(1,2,3,4,5))	range(c(1,2,3,4,5))
5	1	3	1 5

sample()

sample() function gives you a random sample of values from the entire data

sample(data,sample_size)

sample(1:100,3)

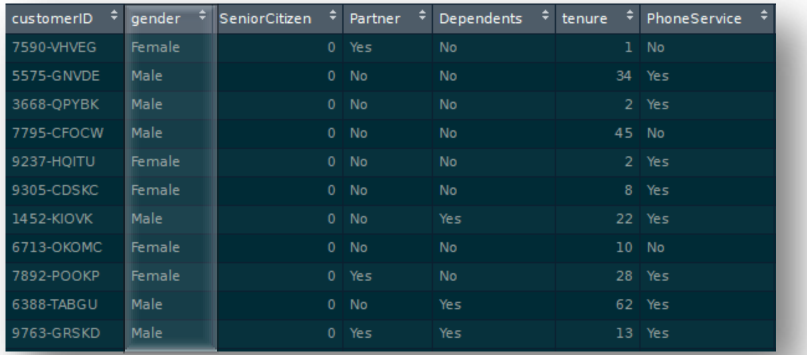
6 12
33

sample(1:100,3)

36 73
9

table()

table() function gives you the count of each level for a categorical column



customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService
7590-VHVEG	Female	0	Yes	No	1	No
5575-GNVDE	Male	0	No	No	34	Yes
3668-QPYBK	Male	0	No	No	2	Yes
7795-CFOCW	Male	0	No	No	45	No
9237-HQITU	Female	0	No	No	2	Yes
9305-CDSKC	Female	0	No	No	8	Yes
1452-KIOVK	Male	0	No	Yes	22	Yes
6713-OKOMC	Female	0	No	No	10	No
7892-POOKP	Female	0	Yes	No	28	Yes
6388-TABGU	Male	0	No	Yes	62	Yes
9763-GRSKD	Male	0	Yes	Yes	13	Yes

```
table(customer_churn$gender)
```

Female	Male
3488	3555

rbind() & cbind()

rbind()

rbind() function combines vector, matrix or data frame by rows

Name	Marks
Sam	97
Bob	25

student

rbind(student,c("Anne",75))

Name	Marks
Sam	97
Bob	25
Anne	75

student

cbind()

cbind() function combines vector, matrix or data frame by columns

Name	Marks
Sam	97
Bob	25

student

```
cbind(student,Grade=c("A","C"))
```

Name	Marks	Grade
Sam	97	A
Bob	25	C

student

merge()

merge()

merge() function is used to join two data.frames horizontally. The merging is done with respect to one or more common columns

```
merge(employee,department,by="Department")
```

Department	Location
Tech	Chicago
Analytics	New York
Support	Boston

Employee

Name	Salary	Department
Sam	75000	Tech
Bob	105000	Sales
Anne	120000	Analytics

Department

Department	Name	Salary	Location
Analytics	Anne	120000	New York
Tech	Sam	75000	Chicago

Merged Dataframe

User-Defined Functions

User-Defined Functions

User-defined functions are those functions which are defined by the user. These functions are made for code reusability and for saving time and space

Syntax

```
function_name<-function(parameter){

...
...
...

}
```

```
Add_five<-function(x){
x+5
}
```

Quiz

Quiz

What are the conditions necessary to apply “rbind()” function?

a. The row no of all the data sets must be equal

b. There must be primary key in all datasets

c. The column no of all the datasets must be equal

d. The column name or attributes of all datasets must be same

Quiz

What are the conditions necessary to apply “cbind()” function?

a. The row no of all the data sets must be equal

b. There must be primary key in all datasets

c. The column no of all the datasets must be equal

d. All of the above

Quiz

What are the conditions necessary to apply “merge()” function?

a. The row no and column no of all the data sets must be equal

b. There must be primary key in all datasets

c. None of the above

d. All of the above

Quiz

Which of the following is used to read a .csv file?

a. read.csv()

b. read.table()

c. read.txt()

d. None of the above

Quiz

Which of the following commands is used to display the top 6 observations of the data ?

a. summary()

b. head()

c. tail()

d. library()

Quiz

Which of the following commands is used to display the structure of the data ?

a. summary()

b. mean()

c. class()

d. str()

Thank You

India : +91-7847955955

US : 1-800-216-8930 (TOLL FREE)

sales@intellipaat.com

24X7 Chat with our Course Advisor