VECTORS

DEFINITION- *A vector is sequence of data elements which can store* ***values of similar data type***

(ie. It can only hold elements of same data type.)

- *Members in a vector are known as components.*
- *A vector can be either* ***numeric, character or logical.***
- *These are* ***one- dimensional.***
- *Simplest way to create a vector is to use the command c(), which stands for* ***combine. This allows us to combine a value in a vector.***

For eg-

V1<-c(1,2,3,4,5,6,7,8,9) → Numeric vector

V2<-c("Red", "Blue", "Green") → Character vector

V3<-c(TRUE , FALSE , TRUE , FALSE) → Logical vector

- ***class() command is used to find the TYPE OF VECTOR.***
- *A vector can only hold elements of **same data type** which further means we cannot have a vector that **contains both logical and numeric for that matter.***

*For eg. we have a vector such as **:-***

v<-c(58, "Raman", TRUE)

We'll be getting an error in our output straight away as vectors don't hold elements of different data type.

HOW TO MAKE CHARACTER VECTOR IN R

WE USE COMBINE COMMAND ie. **“c()”** FOR THAT FOR EG :-

Q- Cities I have been to are as follows:-

Boston, Tampa, Mountain View, Minneapolis.

(INPUT)Vectorcode to be used:-

```
Cities<-c(“Boston” , “Tampa” , “Mountain View” , “Minneapolis”)
```

(OUTPUT):-

```
[1] “Boston”        “Tampa”          “Mountain View”          “Minneapolis”
```

Where [1] stands for position of elements.

Q- HOW TO VERIFY IF THE ABOVE GIVEN EXAMPLE IS A VECTOR OR NO ?

USE THE COMMAND:-

is.vector(Cities)

output:- TRUE

(which means yes, it's a vector)

[If it would have not been a vector then in that case the output that we would get :-

Output- FALSE]

Eg. ***v<-c(58, "Raman", TRUE)***

Input- is.vector*(v)*

Output- FALSE

Q- HOW TO FIND TOTAL NO. OF ELEMENTS IN A DATA STRUCTURE?

For that we use the command “ length () ”

Eg-

```
Cities<-c(“Boston” , “Tampa” , “Mountain View” , “Minneapolis”)
```

Use the command-

Input-

```
length(Cities)
```

output you’ll get -

[1]   4   ; where [1] stands for position of output elements.

Other commands, their definition and examples for better understanding: -

- *length () - total no of elements in the data structure.*

Eg 1-

```
x<-c(1,2,3,4,5,6,7,8,9)
```

Input- length(x)

Output –

```
[1]   9
```

- *ls() - defines variables we have used so far in R*

eg 2- [1] “Cities” “ x”

- *rm() - removes that variable from data structure that was already present in R.*

eg 3-

input- rm(x)

and then press ctrl+enter , then the variable x will be removed so now to check whether its removed or not we will again use a command

ie- ls()

so now : input- ls()

output- [1] “Cities”

OTHER IMPORTANT COMMANDS THAT WE USE(WITH EXAMPLES FOR BETTER UNDERSTANDING):-

Q- eg. x<-c(1,2,3,4,5,6,7,1,9,0,3,2,4)

- *sort(x)- to arrange the data structure in ascending order*

  input - sort(x)

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

- *rev(sort(x))- arranges the given data structure in descending order.*

  Input- rev(sort(x))

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

- *order(x)-position of elements arranged in ascending.*
- *Input- order(x)*
- *Output- [1] 10 1 8 2 12 5 11 4 13 3 6 7 9*

- *rev(order(x))- position of elements arranged in descending order.*
  Input- rev(order(x))
  Output- [1] 9 7 6 3 13 4 11 5 12 2 8 1 10

- *Mean- It gives mean of the given data (x).*
  Input- mean(x)
  Output- [1] 3.615385

- *median- It gives median of the given data structure.*
  Input- median(x)
  Output- [1] 3

- *var()- It gives variance of the given data structure.*
  Input- var()
  Output- [1] 6.75641

- *sd(x)- It gives standard deviation of the given data structure.*

  Input- sd(x)

  Output- [1] 2.59931

- *max(x)- It gives maximum value.*
  Input- max(x)
  Output- [1] 9

- *min(x)- It gives minimum value.*
  Input- min(x)
  Output- [1] 0

- *range(x)- It gives minimum and maximum value of the data.*
  Input- range(x)

Output- [1] 0 9

These two are treated as objects:-

- ***NaN****- represents impossible values.*

```
x<-c(1,2,3,NaN,3,4,NaN)
is.nan(x)
[1] FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE
```

- ***NA****- It represents missing/unknown values.*

```
x<-c(1,2,3,NA,4,NA)
is.na(x)
[1] FALSE FALSE FALSE  TRUE FALSE  TRUE
```

SUBSETTING (WE USE [ ] BRACKETS)

EG- *x[c(1,3,4,5)]*

So with the help of this command ie x[c( )] we get our first third fourth and fifth element.

More eg. For better understanding →

x[c(1)] – first element of the vector will be displayed.
x[c(10)] – tenth element of the vector will be displayed.
x[c(-1)] – except the first element all other elements will be displayed.

x[c(-3)] – all elements are displayed except the 3 element.
x[c(1:3)] - 1st , 2nd and 3rd elements are displayed.
x[x>3] - displays elements greater than 3.
x[x<3]- displays elements that are less than 3.
Now " ==" is used to find if some element is present in the given vector or no .
Which(x==6) tells the position in which the element "6" is present.
Which(x==max(x)) tells the position of the maximum element.

NOW THE SYMBOLS THAT WE USE FOR OR & AND :-

- *or- we use* **|**

  eg- x[x>5 | x<7]

- *and- we use* **&**

  eg- x[x>3 & x<7]

--

--

--

THANK YOU…

By- Jasnoor Kaur Chhabra

MAT/19/58