Data Structures in R

Vector [1D]
Matrix [2D]
Dataframe [2D]
List
Array [Multi Dimensional]

Data Types in R

Numeric value - num, int
Logical value - logi
String value - chr

Operator hierarchy

Brackets >Exponential >Division >Multiplication > Add/Subtract

%% -> Gives the remainder
%/% -> Gives the quotient, not in decimal

```
> 123.5/5
[1] 24.7
> 123.5%/%5
[1] 24
```

Vectors

Declaring vectors - >

```
> vec1 = c(5,9,8,3,6) # c = concatenate
> str(vec1)
 num [1:5] 5 9 8 3 6
```

Length

```
> length(vec1)
[1] 5
```

Append

```
> vec1 = append(vec1,10) # We need to save to overwrite
> append(vec1,11) # This will not save
[1]  5  9  8  3  6 10 11
> vec1
[1]  5  9  8  3  6 10
```

---

append(vector,value,position)

```
> x = append(x,15,4)
> x
 [1]  5  9  8  3 15  6 10  4  9 12
```

Add multiple values after a specified position

```
> x = append(x,c(324,52,5),7)
> x
 [1]   5   9   8   3  15   6  10 324  52   5   4   9  12
```

```
> x = x[-2] #Deleting
> x
 [1]   5   8   3  15   6  10 324  52   5   4   9  12
```

```
> x[c(1,2)] = c(45,54) #Changing 1st value to 45 and 2nd value to 54
> x
 [1]  45  54   3  15   6  10 324  52   5   4   9  12
```

```
> sum(x)
[1] 539
> prod(x)
[1] 2.387658e+14
> min(x)
[1] 3
> max(x)
[1] 324
```

Cumulative min

Lowest value up to a particular position

```
>
```

```
 [1] 45 45  3  3  3  3  3  3  3  3  3  3
```

Cumulative min

Highest value up to a particular position

```
> cummax(x)
 [1]  45  54  54  54  54  54 324 324 324 324 324 324
```

Vector Operations

```
> a = c(4,7,6,2)
> b = c(2,1,4,9)
> c = c(1,2)
> d = c(3,4,5)
```

When a vector is operated by a scalar quantity

```
> a+5
[1]  9 12 11  7
```

When a vector is operated to another vector of same length

```
> a+b
[1]  6  8 10 11
```

When a vector is operated to another vector of length which is a multiple of the first vector

```
> a+c
[1] 5 9 7 4
```

When a vector is operated to another vector of different length

```
> a+d
[1]  7 11 11  5
Warning message:
In a + d : longer object length is not a multiple of shorter object length
```

Sequencing

```
> x = 8:15
> x
[1]  8  9 10 11 12 13 14 15
> seq(20,30)
 [1] 20 21 22 23 24 25 26 27 28 29 30
> seq(4,22,3)
[1]  4  7 10 13 16 19 22
```

seq(start_value,end_value,step)

seq(start_value,end_value,number of outputs required)

```
> z = seq(4,30,length.out=8)
> z
[1]  4.000000  7.714286 11.428571 15.142857 18.857143 22.571429 26.285714
[8] 30.000000
```

Sort

```
> x = c(3,566,3,36,3,67,21,0,9)
> sort(x)
[1]   0   3   3   3   9  21  36  67 566
```

Repetition

```
> rep(c(5,4,7),5)
 [1] 5 4 7 5 4 7 5 4 7 5 4 7 5 4 7
> rep(3,5)
[1] 3 3 3 3 3
> rep(c(5,4,7),each=5)
 [1] 5 5 5 5 5 4 4 4 4 4 7 7 7 7 7
> rep(c(5,4,7),times=c(2,3,4))
[1] 5 5 4 4 4 7 7 7 7
```

String - Vector Operations

```
> places = c("Mumbai","Pune","Nagpur","Bangalore")
> places
[1] "Mumbai"    "Pune"      "Nagpur"    "Bangalore"
> grep('bai',places)
[1] 1
> grep('bai',places,value=T)
[1] "Mumbai"
>
> grep('i',places,value=T)
[1] "Mumbai"
> grep('e',places,value=T)
[1] "Pune"      "Bangalore"
> grep('mum',places,value=T)
character(0)
> grep('um',places,value=T)
[1] "Mumbai"
> substr(places,3,5)
[1] "mba" "ne"  "gpu" "nga"
> grep(' ',places,value=T)
[1] "Navi Mumbai"
> gsub('bai','baaaaaaaaai',places)
[1] "Mumbaaaaaaaaai"      "Pune"                "Nagpur"
[4] "Bangalore"           "Navi Mumbaaaaaaaaai"
```

Which command ->
Input = Condition | Output = Indexes

```
> x = c(4,8,7,3,2,5,9,1,6,3)
> which(x>=4) #Returns indexes which satisfy the condition
[1] 1 2 3 6 7 9
> length(which(x>=4)) #Returns the count
[1] 6
> x[which(x>=4)] #Returns the values at which index
[1] 4 8 7 5 9 6
```

Logical Operators

```
> a = TRUE
> b = TRUE
> c = FALSE
> d = F

> a & b #and
[1] TRUE
> a | b #or
[1] TRUE
> c | d
[1] FALSE
> c & d
[1] FALSE
> !a #not
[1] FALSE
> !d
[1] TRUE

> x
 [1] 4 8 7 3 2 5 9 1 6 3
> x[which(!x>=4)]
[1] 3 2 1 3
> x[which(x>=4 & x%%2!=0)]
[1] 7 5 9
> which(x>=4 & x%%2!=0)
[1] 3 6 7
> which(x>=4 & !x%%2==0)
[1] 3 6 7
> which(x>=4 & x%%2==1)
[1] 3 6 7
```

String Splitting

```
> strsplit(places[1],3)
[[1]]
[1] "Mumbai"

> strsplit(places[1],"mum")
[[1]]
[1] "Mumbai"

> strsplit(places[1],"Mum")
[[1]]
[1] ""    "bai"

> strsplit(places[1],"")
[[1]]
[1] "M" "u" "m" "b" "a" "i"

> strsplit(places[1],"")[[1]]
[1] "M" "u" "m" "b" "a" "i"
> strsplit(places,"")[[1]]
[1] "M" "u" "m" "b" "a" "i"
> strsplit(places,"")
[[1]]
[1] "M" "u" "m" "b" "a" "i"

[[2]]
[1] "P" "u" "n" "e"

[[3]]
[1] "N" "a" "g" "p" "u" "r"

[[4]]
[1] "B" "a" "n" "g" "a" "l" "o" "r" "e"

[[5]]
 [1] "N" "a" "v" "i" " " "M" "u" "m" "b" "a" "i"

> strsplit(places," ")
[[1]]
[1] "Mumbai"

[[2]]
[1] "Pune"

[[3]]
[1] "Nagpur"

[[4]]
[1] "Bangalore"

[[5]]
```

```
[1] "Navi"   "Mumbai"
```

Paste Command

```
> x1 = c('a','b','c')
> x2 = c(1,2,3)
> paste(x1,collapse = "") #Merge the characters in the vector
[1] "abc"
> paste(x1,x2) #Merge two vectors, default separator = space
[1] "a 1" "b 2" "c 3"
> paste(x1,x2,sep=':') #Defining the separator
[1] "a:1" "b:2" "c:3"
> paste(x1,x2,sep=':',collapse='**')
[1] "a:1**b:2**c:3"
```

Matrix

```
> mat1 = matrix(1:24,nrow=3)
> mat1
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    1    4    7   10   13   16   19   22
[2,]    2    5    8   11   14   17   20   23
[3,]    3    6    9   12   15   18   21   24
```

```
> mat1 = matrix(1:24,nrow=3,byrow=T)
> mat1
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    1    2    3    4    5    6    7    8
[2,]    9   10   11   12   13   14   15   16
[3,]   17   18   19   20   21   22   23   24
```

```
> mat1 = matrix(1:24,nrow=3,byrow=T)
> mat1
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    1    2    3    4    5    6    7    8
[2,]    9   10   11   12   13   14   15   16
[3,]   17   18   19   20   21   22   23   24
> a = c(4,8,7,6)
> b = c(1,4,3,9)
> c = c(4,1,9,2)
> mat2 = rbind(a,b,c)
> mat2
  [,1] [,2] [,3] [,4]
a    4    8    7    6
b    1    4    3    9
c    4    1    9    2
> mat2 = cbind(a,b,c)
> mat2
```

```
     a b c
[1,] 4 1 4
[2,] 8 4 1
[3,] 7 3 9
[4,] 6 9 2
```

Dataframes

```
> names = c('Akhil','Pankaj','Rahul')
> inst = c('IITB','IIMA','VJTI')
> marks = c(99,97,89)
#Defining data frame
> studf = data.frame(names,inst,marks)
> studf
   names inst marks
1  Akhil IITB    99
2 Pankaj IIMA    97
3  Rahul VJTI    89
> str(studf)
'data.frame':      3 obs. of  3 variables:
 $ names: chr  "Akhil" "Pankaj" "Rahul"
 $ inst : chr  "IITB" "IIMA" "VJTI"
 $ marks: num  99 97 89
#Change/rename column names
> studf = data.frame(Student_Name = names,College = inst,Marks=marks)
> studf
  Student_Name College Marks
1        Akhil    IITB    99
2       Pankaj    IIMA    97
3        Rahul    VJTI    89
#To print column names
> colnames(studf)
[1] "Student_Name" "College"      "Marks"
#Changing specific column name after the df is created
> colnames(studf)[3] = 'Grades'
> studf
  Student_Name College Grades
1        Akhil    IITB     99
2       Pankaj    IIMA     97
3        Rahul    VJTI     89
> studf$Student_Name
[1] "Akhil"  "Pankaj" "Rahul"
```

Functions in R

```
TaxCal = function(sal)
{
  if (sal<=20000)
  {
    tax =500
  }

  else{

    if(sal<=50000)
    {
    tax = 0.1*sal
    }
    else
    {
      tax=0.2*sal
    }
  }
  message('Tax = ',tax)
  message('Take Home salary = ',sal-tax)

}
```

```
stu_name = c('Sachin','Dhoni','Virat','Sehwag','Rahane','Rohit','Yuvraj')
stu_id = c('A12','D34','A12','A12','D34','Y45','T23')
stu_df = data.frame(Name = stu_name , ID = stu_id)
```

Sort- Will sort by default in ascending
Order- Order will return the indices of values in ascending

```
> x = c(1,35,64,21,6,4,3,1,2)
> sort(x)
[1]  1  1  2  3  4  6 21 35 64
> order(x)
[1] 1 8 9 7 6 5 4 2 3
> sort(x,decreasing = T)
[1] 64 35 21  6  4  3  2  1  1
> order(x,decreasing = T)
```

```
[1] 3 2 4 5 6 7 9 1 8
```