Data Structures in R

Panchatcharam M

Associate Professor

Department of Mathematics and Statistics,
IIT Tirupati

Advanced Data Structures

Most frequently used data structure

- data.frame, matrix and list are most common data structures
- data.frame is the most familiar one
- matrix to people familiar with matrix math
- list for programmers

LISTS

Lists

- Arbitrary objects of either the same type of varying types are required in reality
- Lists can handle this
- It stores any number of items of any type
- It can contain numeric, character or mix of the two or data.frames or another list etc

Lists

```
> list(1,2,3)
[[1]]
[1] 1

[[2]]
[1] 2

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

```
> list("a","b","c")
[[1]]
[1] "a"

[[2]]
[1] "b"

[[3]]
[1] "c"
```

```
> list(1,"a",2.3)
[[1]]
[1] 1

[[2]]
[1] "a"

[[3]]
[1] 2.3
```

Lists

Naming

```
> person=list(name="Raja",age=23,degree="MSc",city="Chennai")
> person
$name
[1] "Raja"

$age
[1] 23

$degree
[1] "MSc"

$city
[1] "Chennai"
```

Lists

Multiple columns

```
> rollno=c("MA24M001","MA24M002","MA24M003","MA24M004")
> name=c("Raja","Ravi","Ramya","Raj")
> totstud=4
> studlist=list("rollno"=rollno,"Name"=name,"Total
Students"=totstud)
> print(studlist$Name)
[1] "Raja"  "Ravi"  "Ramya" "Raj"
```

Accessing by Index and Modifying

Accessing

```
> print(studlist[[2]])
[1] "Raja"  "Ravi"  "Ramya" "Raj"
> print(studlist[[2]][2])
[1] "Ravi"
> print(studlist[[1]][2])
[1] "MA24M002“
```

Modifying Component

```
> studlist[[2]][2]="Shefali"
> print(studlist[[2]][2])
[1] "Shefali"
```

Lists

Length

```
> x=list(1,2,3)
> length(x)
[1] 3
```

Existing Item

```
> course=list("MA522M","MA620L","MA614L")
> "MA620L" %in% course
[1] TRUE
```

Add Item

```
> append(course,"MA602M") #Appended but not saved on course
> course=append(course,"MA602M") #Appended and saved on course
> course=append(course,"MA602M”,after=3) #Appended and saved on course
after index 3
```

Remove Item

```
> newcourse = course[-1] #Removes the first course
> newcourse = course[-3] #Removes the third course
```

Lists

Range of Index

```
> course=list("MA522M","MA620L","MA614L","MA103L","MA512L","MA504L")
> (course)[2:5]
```

Looping

```
> for (x in course) {
      print(x)
}
[1] "MA522M"
[1] "MA620L"
[1] "MA614L"
[1] "MA103L"
[1] "MA512L"
[1] "MA504L"
```

Compine Two lists

```
> x=list("a","b","c")
> y=list(1,2,3)
> z=c(x,y)
> for (x in z) { print(x)}
[1] "a"
[1] "b"
[1] "c"
[1] 1
[1] 2
[1] 3
```

List to Vector

```
> course=list("MA522M","MA620L","MA614L","MA103L","MA512L","MA504L")
> vec=unlist(course)
> vec
[1] "MA522M" "MA620L" "MA614L" "MA103L" "MA512L" "MA504L"
```

MATRIX

Matrix

- Two Dimensional Data Set
- Column and Row
- matrix() function

Syntax

```
matrix(data,nrow,ncol,byrow,dimnames)
```

- data: values
- nrow: number of rows
- ncol: number of columns
- byrow: logical clue, if true, value will be assigned by rows
- dimnames: names of rows and column

Matrix

```
> data=1:9
> data
[1] 1 2 3 4 5 6 7 8 9
> nrow=3
> ncol=3
> byrow=TRUE
> A=matrix(data,nrow,ncol,byrow)
> A
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

```
> rownames(A)=c("a","b","c")
> colnames(A)=c("x","y","z")
> print(A)
  x y z
a 1 2 3
b 4 5 6
c 7 8 9
```

Constant and Diagonal Matrices

```
> A=matrix(2,3,3)
> rownames(A)=c("a","b","c")
> colnames(A)=c("x","y","z")
> print(A)
  x y z
a 2 2 2
b 2 2 2
c 2 2 2
```

```
> data=c(1,2,3)
> A=diag(data,3,3)
> rownames(A)=c("a","b","c")
> colnames(A)=c("x","y","z")
> print(A)
  x y z
a 1 0 0
b 0 2 0
c 0 0 3
```

Vectors to Matrix

```
> data1=c(1,2,3)
> data2=c(4,5,6)
> data3=c(7,8,9)
> rbind(data1,data2,data3)
      [,1] [,2] [,3]
data1    1    2    3
data2    4    5    6
data3    7    8    9
```

```
> data1=c(11,12,13)
> data2=c(14,15,16)
> data3=c(17,18,19)
> cbind(data1,data2,data3)
     data1 data2 data3
[1,]    11    14    17
[2,]    12    15    18
[3,]    13    16    19
```

Slicing

```
> data=1:9
> A=matrix(data,3,3)
> print(A[1:2,])#First and Second Row, Row slicing
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
> print(A[1:2,3])#Slice the first row and second row and the 3rd column
[1] 7 8
> print(A[,1:2]) #Column Slicing
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
> print(A[1,1:2]) #Column Slice and then row slicing
[1] 1 4
```

Accessing

```
> data=1:9
> A=matrix(data,3,3)
> print(A[1,2])
[1] 4
> print(A[1,2])
[1] 4
> print(A[1][2]) #Not valid one
[1] NA
> print(A[1])
[1] 1
> print(A[2])
[1] 2
> print(A[9])
[1] 9
```

Modifying

```
> data=1:9
> A=matrix(data,3,3)
> print(A)
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

> A[2,2]=6
> print(A)
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    6    8
[3,]    3    6    9
```

Add a Column/row

```
> data=1:9
> A=matrix(data,3,3,byrow=TRUE)
> print(A)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
> vec=c(10,11,12)
> cbind(A,vec)
           vec
[1,] 1 2 3  10
[2,] 4 5 6  11
[3,] 7 8 9  12
```

```
> rbind(A,vec)
    [,1] [,2] [,3]
       1    2    3
       4    5    6
       7    8    9
vec   10   11   12
```

Deleting a Column/row

```
> data=1:16
> A=matrix(data,4,4,byrow=TRUE)
> print(A)
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
[4,]   13   14   15   16
> A=A[-2,]
> print(A)
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    9   10   11   12
[3,]   13   14   15   16
```

```
> A=A[,-2]
> print(A)
     [,1] [,2] [,3]
[1,]    1    3    4
[2,]    9   11   12
[3,]   13   15   16
```

Item Exists

```
> course=c("MA522M","MA620L","MA614L","MA103L","MA512L","MA504L")
> A=matrix(course,2,3)
> print(A)
     [,1]     [,2]     [,3]    
[1,] "MA522M" "MA614L" "MA512L"
[2,] "MA620L" "MA103L" "MA504L"
> "MA522M" %in% A
[1] TRUE
```

```
> for (x in A) {
print(x)}
[1] "MA522M"
[1] "MA620L"
[1] "MA614L"
[1] "MA103L"
[1] "MA512L"
[1] "MA504L"
```

Dimension

```
> course=c("MA522M","MA620L","MA614L","MA103L","MA512L","MA504L")
> A=matrix(course,2,3)
> dim(A)
[1] 2 3
> length(A)
[1] 6
> nrow(A)
[1] 2
> ncol(A)
[1] 3
```

Matrix Operations

```
> A=matrix(1:9,3)
> print(A)
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> rotate=t(apply(A,2,rev))
> print(rotate)
     [,1] [,2] [,3]
[1,]    3    2    1
[2,]    6    5    4
[3,]    9    8    7
```

Matrix Rotation

```
> A=matrix(1:9,3)
> B=matrix(11:19,3)
> print(A)
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> print(B)
     [,1] [,2] [,3]
[1,]   11   14   17
[2,]   12   15   18
[3,]   13   16   19
```

```
> print(A+B)
     [,1] [,2] [,3]
[1,]   12   18   24
[2,]   14   20   26
[3,]   16   22   28
> print(A-B)
     [,1] [,2] [,3]
[1,]  -10  -10  -10
[2,]  -10  -10  -10
[3,]  -10  -10  -10
> print(A*B)
     [,1] [,2] [,3]
[1,]   11   56  119
[2,]   24   75  144
[3,]   39   96  171
> print(A/B)
           [,1]      [,2]      [,3]
[1,] 0.09090909 0.2857143 0.4117647
[2,] 0.16666667 0.3333333 0.4444444
[3,] 0.23076923 0.3750000 0.4736842
```

Maximum and Minimum

```
> A=matrix(1:9,3)
> B=matrix(11:19,3)
> min(A)
[1] 1
> max(A)
[1] 9>
which(A==max(A),arr.ind=TRUE)
     row col
[1,]   3   3
> which(A==min(A),arr.ind=TRUE)
     row col
[1,]   1   1
```

Mean, Median, Correlation

```
> data=rnorm(9)
> A=matrix(data,3)
> A
           [,1]      [,2]       [,3]
[1,]  0.3421453 0.3933341 -1.2461415
[2,] -2.2266040 0.4611668 -0.2646993
[3,] -0.1936145 0.5954660  0.1485115
> cor(A)
            [,1]        [,2]       [,3]
[1,]  1.00000000 -0.01137028 -0.4168882
[2,] -0.01137028  1.00000000  0.9136392
[3,] -0.41688818  0.91363916  1.0000000
```

```
> mean(A)
[1] -0.2211595
> median(A)
[1] 0.1485115
```

ARRAYS

Arrays

- Vectors are Single Dimensional
- Matrices are Two Dimensional
- Arrays are Multidimensional
- You can deal with tensors

Syntax

```
array(data,dim=c(nx,ny,nz,...))
```

- data: values
- dim: dimension
- nx: x dimension
- ny: y dimension
- nz: z dimension
- etc

Arrays

```
> data=1:24
> A=array(data,dim=c(2,4,3))
> print(A)
, , 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
```

Arrays

```
> data=1:24
> A=array(data,dim=c(2,4,3))
> print(A)
, , 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
```

```
dim=c(2,4,3)
2: Number of Rows
4: Number of Columns
3: Number of Dimensions
```

Accessing

```
> data=1:24
> A=array(data,dim=c(2,4,3))

> print(A[2,1,3])
[1] 18
```

Syntax

```
array(row_pos,col_pos,matrix_level)
```

Dimension

```
> data=1:24
> A=array(data,dim=c(2,4,3))

> dim(A)
[1] 2 4 3
> length(A)
[1] 24
```

```
> for (x in A) {print(x)}
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
...
[1] 22
[1] 23
[1] 24
> 24 %in% A
[1] TRUE
```

DATA FRAMES

data.frames

- One of the most useful features of R
- Like an excel spreadsheet, it has columns and rows
- Each column is a variable
- Each row is an observation
- Each column is a vector, of same length
- Each column can have different type of data

Data Frames

- Data Displayed in a format as table
- It can have different types of data
- Heterogeneous

Syntax

```
data.frame(…, row.names = NULL,
check.rows = FALSE,
           check.names = TRUE,
fix.empty.names = TRUE,
           stringsAsFactors =
default.stringsAsFactors())
default.stringsAsFactors()
```

data.frames

```
> x=1:5
> name=c("MA24M001","MA24M002","MA24M003","MA24M004","MA24M005")
> marks=c(100,98,77,89,67)
> DF=data.frame(x,name,marks)
> DF
  x     name marks
1 1 MA24M001   100
2 2 MA24M002    98
3 3 MA24M003    77
4 4 MA24M004    89
5 5 MA24M005    67
```

data.frames

```
> nrow(DF)
[1] 5
> ncol(DF)
[1] 3
> dim(DF)
[1] 5 3
> names(DF)
[1] "x"     "name"  "marks"
> names(DF)[3]
[1] "marks
> rownames(DF)
[1] "1" "2" "3" "4" "5"
```

data.frames

```
> name=c("MA24M001","MA24M002","MA24M003","MA24M004","MA24M005")
> marks=c(100,98,77,89,67)
> x=1:5
> DF=data.frame(SNo=x,Rollno=name,Marks=marks)
> DF
  SNo   Rollno Marks
1   1 MA24M001   100
2   2 MA24M002    98
3   3 MA24M003    77
4   4 MA24M004    89
5   5 MA24M005    67
```

data.frames

```
> x=1:10
>
y=c("MA24M001","MA24M002","MA24M003","MA24M004","MA24M005","MA24
M006","MA24M007","MA24M008","MA24M009","MA24M010")
> marks=c(100,98,77,89,67,89,45,65,30,23)
> head(DF)
  SNo   Rollno Marks
1   1 MA24M001   100
2   2 MA24M002    98
3   3 MA24M003    77
4   4 MA24M004    89
5   5 MA24M005    67
6   6 MA24M006    89
```

```
> tail(DF)
   SNo           Rollno Marks
5    5         MA24M005    67
6    6         MA24M006    89
7    7         MA24M007    45
8    8         MA24M008    65
9    9         MA24M009    30
10  10 MA24M010)\n10    23
```

data.frames

```
> x=1:10
>
y=c("MA24M001","MA24M002","MA24M003","MA24M004","MA24M005","MA24
M006","MA24M007","MA24M008","MA24M009","MA24M010")
> marks=c(100,98,77,89,67,89,45,65,30,23)
> head(DF,n=3)
 SNo    Rollno Marks
1    1 MA24M001   100
2    2 MA24M002    98
3    3 MA24M003    77
```

```
> tail(DF,n=3)
   SNo    Rollno Marks
8    8 MA24M008    65
9    9 MA24M009    30
10  10 MA24M010    23
```

data.frames

```
> DF$SNo
 [1]  1  2  3  4  5  6  7  8  9 10
> DF$RollNo
NULL
> DF$Rollno
 [1] "MA24M001" "MA24M002" "MA24M003" "MA24M004" "MA24M005"
"MA24M006"
 [7] "MA24M007" "MA24M008" "MA24M009" "MA24M010"
> > DF[1,3]
[1] 100
> DF[1,2]
[1] "MA24M001"
> DF[1,1]
[1] 1
```

data.frames

```
> DF[1,1:3]
  SNo   Rollno Marks
1   1 MA24M001   100
> DF[1:3,1:3]
  SNo   Rollno Marks
1   1 MA24M001   100
2   2 MA24M002    98
3   3 MA24M003    77
```

data.frames

```
> DF[,1:2]
   SNo   Rollno
1    1 MA24M001
2    2 MA24M002
3    3 MA24M003
4    4 MA24M004
5    5 MA24M005
6    6 MA24M006
7    7 MA24M007
8    8 MA24M008
9    9 MA24M009
10  10 MA24M010
> DF[,2:2]
 [1] "MA24M001" "MA24M002" "MA24M003" "MA24M004" "MA24M005" "MA24M006"
 [7] "MA24M007" "MA24M008" "MA24M009" "MA24M010"
> DF[,3]
 [1] 100  98  77  89  67  89  45  65  30  23
```

data.frames

```
> class(DF[,"Rollno"])
[1] "character"
> class(DF[,"marks"])
> class(DF[,"Marks"])
[1] "numeric“
> class(DF["Marks"])
[1] "data.frame"
```

Data Frames

```
> rollno=c("MA24M001","MA24M002","MA24M003","MA24M004")
> name=c("Raja","Ravi","Ramya","Raj")
> age=c(23,42,32,18)
> Data_Frame=data.frame(rollno,name,age)
> print(Data_Frame)
    rollno  name age
1 MA24M001  Raja  23
2 MA24M002  Ravi  42
3 MA24M003 Ramya  32
4 MA24M004   Raj  18
```

Data Summary

```
> rollno=c("MA24M001","MA24M002","MA24M003","MA24M004")
> name=c("Raja","Ravi","Ramya","Raj")
> age=c(23,42,32,18)
> Data_Frame=data.frame(rollno,name,age)
> print(Data_Frame)
    rollno  name age
1 MA24M001  Raja  23
2 MA24M002  Ravi  42
3 MA24M003 Ramya  32
4 MA24M004   Raj  18
```

```
> summary(Data_Frame)
    rollno              name                age       
 Length:4           Length:4           Min.   :18.00  
 Class :character   Class :character   1st Qu.:21.75  
 Mode  :character   Mode  :character   Median :27.50  
                                       Mean   :28.75  
                                       3rd Qu.:34.50  
                                       Max.   :42.00  
```

Data Access

```
> rollno=c("MA24M001","MA24M002","MA24M003","MA24M004")
> name=c("Raja","Ravi","Ramya","Raj")
> age=c(23,42,32,18)
> Data_Frame=data.frame(rollno,name,age)
> Data_Frame[1]
    rollno
1 MA24M001
2 MA24M002
3 MA24M003
4 MA24M004
> Data_Frame[2]
   name
1  Raja
2  Ravi
3 Ramya
4   Raj
```

```
> Data_Frame$rollno
[1] "MA24M001" "MA24M002" "MA24M003" "MA24M004«
> Data_Frame[["name"]]
[1] "Raja"  "Ravi"  "Ramya" "Raj"
```

Add Row/Column

```
> rollno=c("MA24M001","MA24M002","MA24M003","MA24M004")
> name=c("Raja","Ravi","Ramya","Raj")
> age=c(23,42,32,18)
> Data_Frame=data.frame(rollno,name,age)
> newdata=rbind(Data_Frame,c("MA24M005","Roja",23))
> print(newdata)
    rollno  name age
1 MA24M001  Raja  23
2 MA24M002  Ravi  42
3 MA24M003 Ramya  32
4 MA24M004   Raj  18
5 MA24M005  Roja  23
```

```
> print(newdata)
    rollno  name age marks
1 MA24M001  Raja  23    95
2 MA24M002  Ravi  42    75
3 MA24M003 Ramya  32    45
4 MA24M004   Raj  18    85
```

Remove Row/Column

```
> rollno=c("MA24M001","MA24M002","MA24M003","MA24M004")
> name=c("Raja","Ravi","Ramya","Raj")
> age=c(23,42,32,18)
> Data_Frame=data.frame(rollno,name,age)
> newdata=Data_Frame[-1]
> print(newdata)
   name age
1  Raja  23
2  Ravi  42
3 Ramya  32
4   Raj  18
```

```
> newdata=Data_Frame[-1,]
> print(newdata)
    rollno  name age
2 MA24M002  Ravi  42
3 MA24M003 Ramya  32
4 MA24M004   Raj  18
```

Dimension

```
> rollno=c("MA24M001","MA24M002","MA24M003","MA24M004")
> name=c("Raja","Ravi","Ramya","Raj")
> age=c(23,42,32,18)
> Data_Frame=data.frame(rollno,name,age)
> dim(Data_Frame)
[1] 4 3
> length(Data_Frame)
[1] 3
```

Combine two Dataframe

```
> newdata=cbind(Data_Frame,Data_Frame)
> print(newdata)
   rollno  name age  rollno  name age
1 MA24M001  Raja  23 MA24M001  Raja  23
2 MA24M002  Ravi  42 MA24M002  Ravi  42
3 MA24M003 Ramya  32 MA24M003 Ramya  32
4 MA24M004   Raj  18 MA24M004   Raj  18
```

```
> newdata=rbind(Data_Frame,Data_Frame)
> print(newdata)
   rollno  name age
1 MA24M001  Raja  23
2 MA24M002  Ravi  42
3 MA24M003 Ramya  32
4 MA24M004   Raj  18
5 MA24M001  Raja  23
6 MA24M002  Ravi  42
7 MA24M003 Ramya  32
8 MA24M004   Raj  18
```

FUNCTIONS

Functions

- ***function*** **creates a function and assigns it a name**
- **return sends a result back to the caller**
- **Arguments are passed by assignment**
- **Arguments and return types are not declared**

```
func_name= function(arg1, arg2, ...){
# optional doc string
    statements or Body
      return (expression) # from function
}
```

```
> product=function(x,y){
+ return (x*y) }
> product(2,3)
[1] 6
```

Functions

```
mygcd=function(a, b){
  #greatest common divisor
while(a){
  temp=a
  a=b %% a
  b=temp
}
  return(b)
}
print(mygcd(12,20))
```

```
4
```

$$a = q_0 b + r_0$$
$$b = q_1 r_0 + r_1$$

$$b = q_0 a + r_0$$
$$a = q_1 r_0 + r_1$$
$$r_0 = q_2 r_1 + r_2$$

ARGUMENTS

Positional Arguments

```
functionpositional=function(x,y,z){ #Positional Arguments
  print("I am inside the Positional")
  print(paste("My Name is ",x))
  print(paste("My Age is ",y))
  print(paste("My Marks is ",z))
}
x="IITTP"
y=6
z=4.8
```

```
functionpositional(x,y,z)
[1] "I am inside the Positional"
[1] "My Name is  IITTP"
[1] "My Age is  6"
[1] "My Marks is  4.8"
```

```
> functionpositional(y,x,z)
[1] "I am inside the Positional"
[1] "My Name is  6"
[1] "My Age is  IITTP"
[1] "My Marks is  4.8"
```

Keyword Arguments

```r
functionkeyword=function(name=x,age=y,marks=z){ #Positional Arguments
  print("I am inside the Keyword")
  print(paste("My Name is ",name))
  print(paste("My Age is ",age))
  print(paste("My Marks is ",marks))
}
x="IITTP"
y=6
z=4.8
```

```
> functionkeyword(age=y,name=x,marks=z)
[1] "I am inside the Keyword"
[1] "My Name is  IITTP"
[1] "My Age is  6"
[1] "My Marks is  4.8"
```

```
> functionkeyword(name=x,age=y,marks=z)
[1] "I am inside the Keyword"
[1] "My Name is  IITTP"
[1] "My Age is  6"
[1] "My Marks is  4.8"
```

Default Arguments

```
funcdefault=function(a,b,c=10,d=100){
  print(paste(a,b,c,d))
}
```

```
> funcdefault(1,2,3,4)
[1] "1 2 3 4"
> funcdefault(1,2)
[1] "1 2 10 100"
> funcdefault(1,2,3)
[1] "1 2 3 100"
```

Variable number of Arguments

✓ Non-keyword Arguments with variable length

```
varyarg=function(...){
  x=list(...)
  print(sum(...))
}

> varyarg(12,33,4,5,56)
[1] 110
> varyarg(1,2,3)
[1] 6
```

Variable number of Arguments

✓ Non-keyword Arguments with variable length

```
studentdetails=function(name,...){
  print(name)
  print(list(...))
}
studentdetails("placement",28,'A',TRUE)
[1] "placement"
[[1]]
[1] 28

[[2]]
[1] "A"

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

Variable number of Arguments

- ✓ **How about variable number of keyword arguments?**
- ✓ **Explore it**

Inline function

- ✓ **A small function that we need it on the fly**

```
sq = function(x) x^2*4+x/3

print(sq(4))
print(sq(-2))
```

Built-in Functions

```
abs()
sqrt()
round()
exp()
log()
log10()
floor()
ceiling()
trunc()
```

```
cos()
sin()
tan()
```

```
mean()
median()
cor()
var()
sd()
quantile()
sum()
range()
diff()
min()
max()
scale()
```

```
unique()
order()
aggregate()
```

```
substr()
grep()
sub()
paste()
strsplit()
tolower()
toupper()
```

Single Input Single Output

```
area=function(radius){
  return(pi*radius^2)
}
print(area(2))
[1] 12.56637
```

Multiple Input Multiple Output

```
myrect=function(a,b){
  area=a*b
  peri=2*(a+b)
  result=list("Area"=area,"Perimeter"=peri)
}
result=myrect(2,4)
print(result["Area"])
print(result["Perimeter"])
```

If Arguments missed

```
mycone = function(r, l, h ){
  lat_area = pi*r*l
  print(lat_area)
}

# No error
print(mycone(5, 10))
[1] 157.0796
```

```
mycone = function(r, l, h ){
  vol=1/3*pi*r^2*h
  print(vol)
}

# This'll throw an error
print(mycone(5, 10))
Error in mycone(5, 10) :
argument "h" is missing, with
no default
```

Function as Arguments

```
myslant=function(r,h){
  return(sqrt(r^2+h^2))
}

mycone = function(r, h, func2 ){
  lat_area = pi*r*func2(r,h)
  print(lat_area)
  vol=1/3*pi*r^2*h
  print(vol)
}

# This'll throw an error
mycone(5, 10,myslant)
[1] 175.6204
[1] 261.7994
```

THREE TYPES OF FUNCTIONS

Primitive Functions

- ✓ **Execute the following and see the output**

```
> typeof(sum)
[1] "builtin"
> names(methods:::.BasicFunsList)
```

- ✓ **Some functions call C code directly**
- ✓ **These functions are called primitive functions**
- ✓ **Found in base package**
- ✓ **Harder to write but efficient to use**

Inflix Functions

- **Predefined inflix operators**
  - **%*% Matrix Multiplication**
  - **%in% Matching Operator**
  - **%x% Kronecker Product**
  - **%/% Integer Division**
  - **%o% Outer Product**

```
'%sum%'<-function(a,b){
  return(a+b)
}
> 3 %sum% 5
[1] 8
```

- **Function which the function name comes in between its arguments**
- **R comes with built-in inflix operators**
- **:, ::, :::, $, @, ^, *, /, +, -, >, >=, <, <=, ==, !=, !, &, &&, |, ||, ~, <-, and <<-.**

Replacement Functions

- ❑ **Modify their arguments in place**
- ❑ **Name of the replacement function succeeded by <**

```
"replace<-"<-function(x,value){
  x[3]=value
  x
}
x=c(1,1,3,4,5,5,5)
print(x)
[1] 1 1 3 4 5 5 5
replace(x)=8L
print(x)
[1] 1 1 8 4 5 5 5
```

✓ **Note, you should use value, otherwise, it will return error**

RECURSION

Recursion

- ✓ **Recursion**
  - ✓ **Base Case**
  - ✓ **Non Base case**

```
myfactorial=function(n){
  if(n==0 || n==1){
    return(1)
  }
  else{

return(n*myfactorial(n-1))
  }
}
myfactorial(5)
```

CONVERSION

Conversion Function

- ✓ **Convert data from one type to another type**
  - ✓ **For Data Types**
  - ✓ **For Data Structures**

Function	Example
`as.numeric()`	`as.numeric(c(1,2,3))`
`as.integer()`	`as.integer(c(1.25,2.33,4.4))`
`as.character()`	`as. character(c(1.25,2.33,4.4))`
`as.logical()`	`as.logical(3>4)`
`as.Date`	`as.Date(c("02/03/24","03/04/24"),"%d/%m/%y")`

Function	Example
as.data.frame()	as.data.frame(c(1.25,2.33,4.4))
as.vector()	as. vector(c(1.25,2.33,4.4))
as.matrix()	as. matrix(c(1.25,2.33,4.4))

End of Datastructures in R