functions with multiple inputs and multiple outputs which we call MIMO how to source and call those functions. We will also see about inline functions how to loop over objects using the commands apply, lapply and tapply.

The functions in R
takes multiple input objects, but written only one object as output, this is however, not a limitation because you can create lists of all the outputs which you want to create and once the list is written out you can access the into the elements of the list and get the answers which you want.

Let us consider this example I want to create a function vol cylinder underscore MIMO which takes diameter and height of the cylinder and returns volume and surface area. Since R can written only one object what I have to do is I have to create one object which is a list that contains volume and surface area and return the list.

```
volCylinder=function(dia=5,len=100,r=10)
{
 volume=pi*dia^2*len/4
 surfacearea=pi*dai*len
 result=list("volume"=volume,"surface area"=surfacearea)
 return(result)
 print(r)

}
```

```
> source('D:/volCylinderMIMO.R')
Loading required package: arules
Loading required package: Matrix

Attaching package: ‘arules’

The following objects are masked from ‘package:base’:

    abbreviate, write

> result=volCylinder(10,5)
Warning message:
```

```
package 'arules' was built under R version 3.6.1
> result
$volume
[1] 392.6991

$`surface area`
[1] 157.0796
```

INLINE FUNCTIONS:

```
> func=function(x)x^2+4*x+4
> func(1)
[1] 9
```

So, now, we move on to looping over objects there are few looping functions that are pretty useful when working interactively on a command line few examples are apply,lapply, tapply and so on.Apply function applies a function over the margins of an array R matrix, lapply function applies a function or a list or a vector, tapply function applies a function over a ragged array and mapply is a multivariate version of lapply

What apply function does is applies a given function over a margins of a given array, when you say margins here this refers to the dimension of an array along which the function need to be applied.

```
> A
  a b c
d 1 2 3
e 4 5 6
f 7 8 9
> apply(A,1,sum)
 d  e  f
 6 15 24
> apply(A,2,sum)
 a  b  c
12 15 18
```

lapply function is used to apply a function over a list so that is where you have l here. Lapply will always return a list which is of the same length as the input list.

```
> A=Matrix(1:9,3,3)
> A
3 x 3 Matrix of class "dgeMatrix"
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> B=Matrix(10:18,3,3)
> B
3 x 3 Matrix of class "dgeMatrix"
     [,1] [,2] [,3]
[1,]   10   13   16
[2,]   11   14   17
[3,]   12   15   18
> MYLIST=list(A,B)
> MYLIST
[[1]]
3 x 3 Matrix of class "dgeMatrix"
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

[[2]]
3 x 3 Matrix of class "dgeMatrix"
     [,1] [,2] [,3]
[1,]   10   13   16
[2,]   11   14   17
[3,]   12   15   18

> determinant=lapply(MYLIST,det)
> determinant
[[1]]
[1] 0
```

```
[[2]]
[1] 5.329071e-15
```

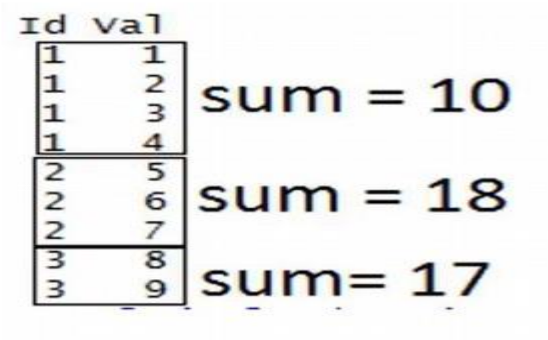
mapply is a multivariate version of lapply. What is does is this function can be applied on several lists simultaneously the syntax is mapply the function you need to apply on list 1 and list 2 together.

```
> source('D:/volCylinder.R')
> dia=c(1,2,3,4)
> len=c(7,3,4,2)
> VOL=mapply(volCylinder,dia,len)
> VOL
[1]  5.497787  9.424778 28.274334 25.132741

>
```

tapply is used to apply a function over a subset of vectors given by combination of factors.

```
> id=c(1,1,1,1,2,2,2,3,3)
> values=c(1,2,3,4,5,6,8,9)
> tapply(values,id,sum)
Error in tapply(values, id, sum) : arguments must have same length
> values=c(1,2,3,4,5,6,7,8,9)
> tapply(values,id,sum)
 1  2  3
10 18 17
```


Id Val
1 1
1 2
1 3
1 4
sum = 10
2 5
2 6
2 7
sum = 18
3 8
3 9
sum= 17