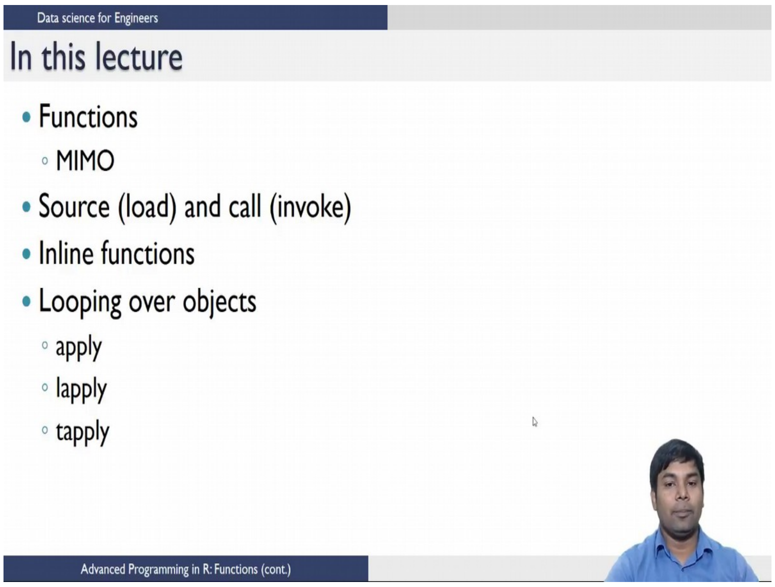
Data Science for Engineers

Prof. Raghunathan Rengaswamy
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture – 09
Advanced programming in R: Functions

Welcome to lecture 8 in the R module of the course Data Science for Engineers. In the previous lectures we have seen how to create functions, how to execute them, but we have limited ourselves to the single output.

(Refer Slide Time: 00:33)

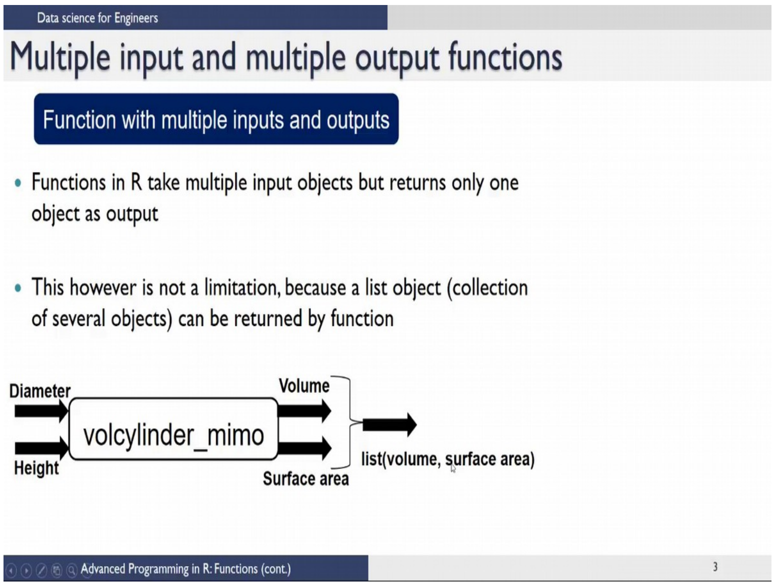

Data science for Engineers
Advanced Programming in R: Functions (cont.)

In this lecture

- Functions
  - MIMO
- Source (load) and call (invoke)
- Inline functions
- Looping over objects
  - apply
  - lapply
  - tapply

In this lecture we are going to see 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.

(Refer Slide Time: 00:53)


Data science for Engineers

Multiple input and multiple output functions

Function with multiple inputs and outputs

- Functions in R take multiple input objects but returns only one object as output
- This however is not a limitation, because a list object (collection of several objects) can be returned by function

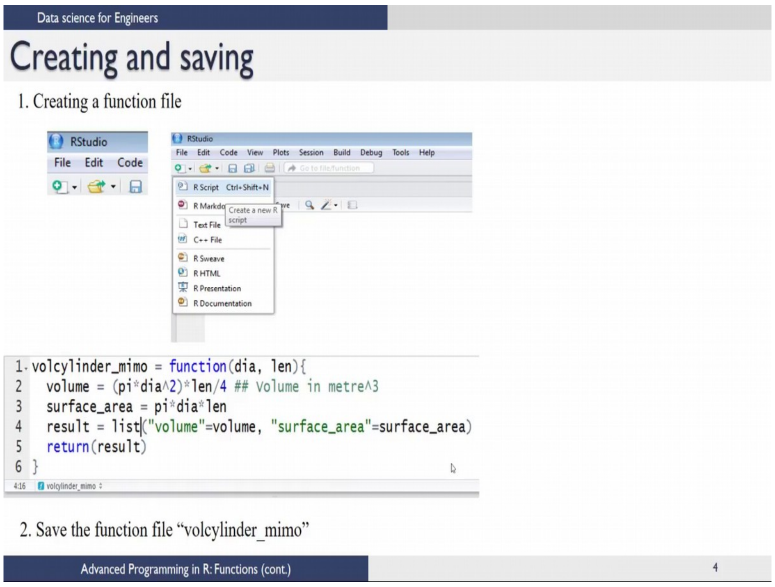
Diameter
Height
volcylinder_mimo
Volume
Surface area
list(volume, surface area)

Advanced Programming in R: Functions (cont.) 3

Let us see the functions with multiple input and multiple outputs. 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. Let us see how we can do that in the next line.

(Refer Slide Time: 01:51)


Data science for Engineers

Creating and saving

1. Creating a function file

```
volcylinder_mimo = function(dia, len){
  volume = (pi*dia^2)*len/4 ## Volume in metre^3
  surface_area = pi*dia*len
  result = list("volume"=volume, "surface_area"=surface_area)
  return(result)
}
```

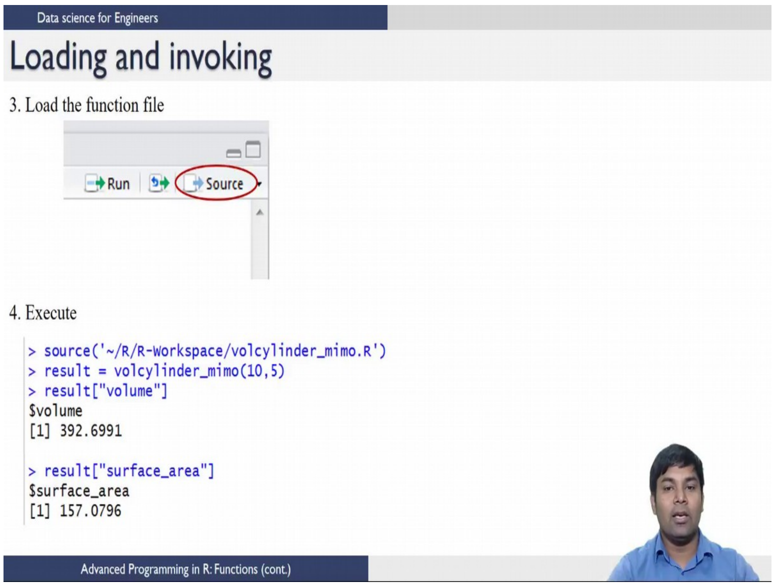
2. Save the function file "volcylinder_mimo"

Advanced Programming in R: Functions (cont.) 4

So, you need to first create an R file which we have seen several times. You can create an R script using a + button R from the file tab once you have opened R script this is the piece of code that does what we need we want to name the function as vol cylinder underscore MIMO because it is a multiple input and multiple output.

And this function takes an arguments diameter and length earlier we have calculated only volume and now we want to calculate the surface also which is given by π times diameter times length. Since R returns only one object first I need to create an object called result which is a list of volume and surface area, I am naming the volume as volume and surface area surface area I will calculate this result and ask the function to return one object the result which contains both volume and surface area.

(Refer Slide Time: 02:53)


Data science for Engineers
Run
Source
Advanced Programming in R: Functions (cont.)

Loading and invoking

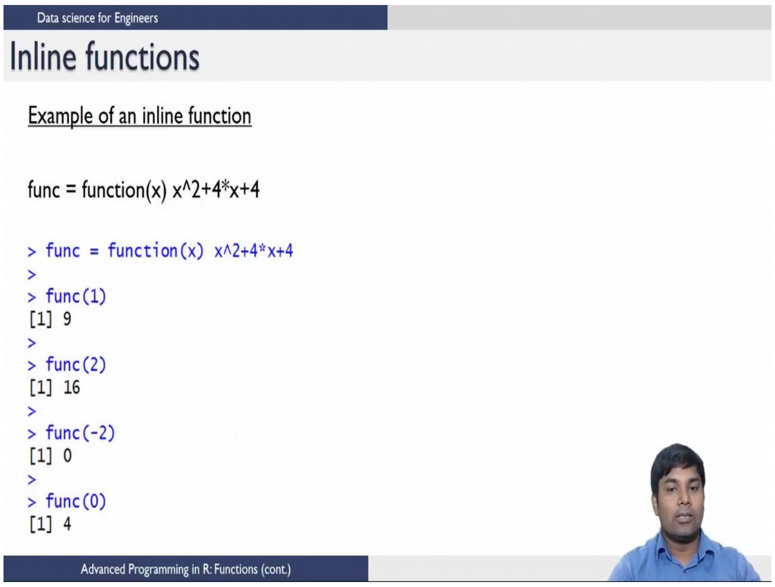
3. Load the function file

4. Execute

```
> source('~/R/R-Workspace/volcylinder_mimo.R')
> result = volcylinder_mimo(10,5)
> result["volume"]
$volume
[1] 392.6991

> result["surface_area"]
$surface_area
[1] 157.0796
```

Once you will write the function you need to load the function to call it loading can be done using the source button, once you source it you are ready to call a function. You can call the function once you load the function. So, I am calling this function result = volcylinder_mimo and I am passing 10 and 5 as my arguments this result will contain two elements the first element volume will contain volume the second element surface area will contain surface area. Once the object result is given out by the function I can access inducer elements by using the techniques what we have teached in the list lecture.

(Refer Slide Time: 03:35)


Data science for Engineers
Advanced Programming in R: Functions (cont.)

Inline functions

Example of an inline function

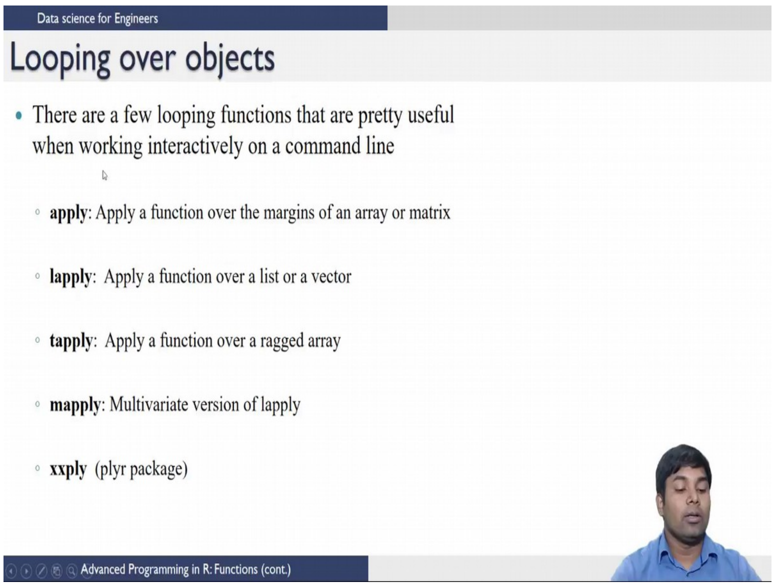
func = function(x) x^2+4*x+4

```
> func = function(x) x^2+4*x+4
>
> func(1)
[1] 9
>
> func(2)
[1] 16
>
> func(-2)
[1] 0
>
> func(0)
[1] 4
```

Sometimes creating an R script file loading it, executing it is a lot of work when you want to just create very small functions such as the ones as shown here x^2+ 4x + 4. I want to evaluate this expression for different x's. So, having a function file and then loading it, invoking it is a lot of work. So, what we can use in this kind of situations is an inline functions to create an inline function you have to use the function command with the argument x and then the expression of the function.

Once you create this you can call this in the command prompt itself function of one gives takes the value of one as an argument and then x accelerates this expression $1^2 = 1 + 4(1)= 4$, $1 + 4 = 5$, and $5+ 4=9$. Similarly you can get the outputs are different arguments which are passed which are shown in the screen.

(Refer Slide Time: 04:42)


Data science for Engineers

Looping over objects

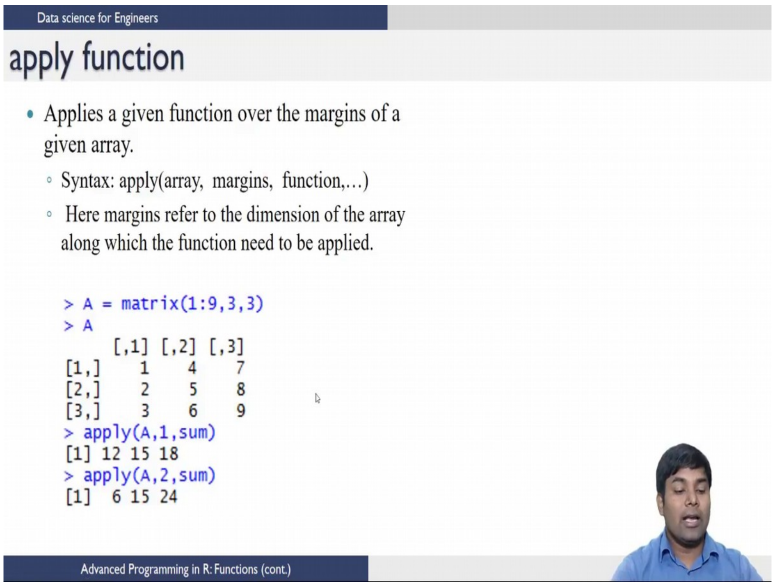
- There are a few looping functions that are pretty useful when working interactively on a command line
  - **apply**: Apply a function over the margins of an array or matrix
  - **lapply**: Apply a function over a list or a vector
  - **tapply**: Apply a function over a ragged array
  - **mapply**: Multivariate version of lapply
  - **xxply** (plyr package)

Advanced Programming in R: Functions (cont.)

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. Let us see what each of these functions does.

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. We will see examples for each of this in the coming slides.

(Refer Slide Time: 05:22)


Data science for Engineers

apply function

- Applies a given function over the margins of a given array.
  - Syntax: apply(array, margins, function,…)
  - Here margins refer to the dimension of the array along which the function need to be applied.

```
> A = matrix(1:9,3,3)
> A
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> apply(A,1,sum)
[1] 12 15 18
> apply(A,2,sum)
[1]  6 15 24
```

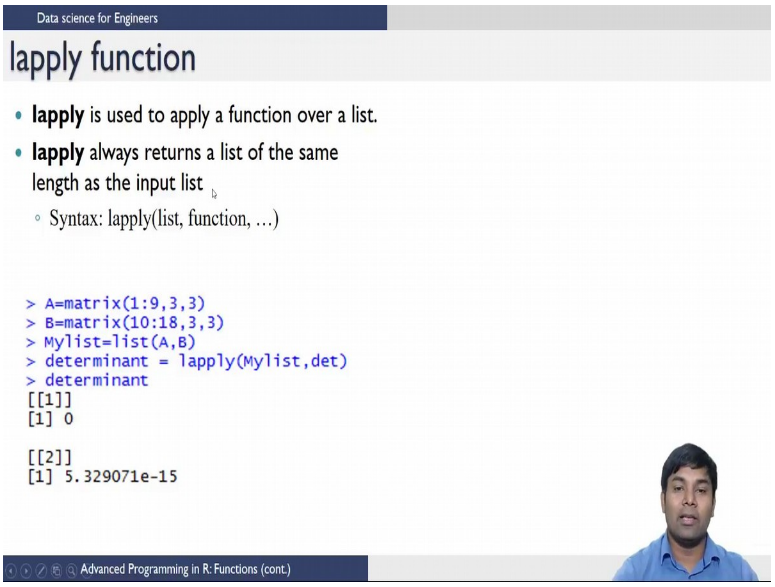
Advanced Programming in R: Functions (cont.)

Here is an example for apply function. 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.

Let us create a matrix A with the elements 1 to 9 which is of 3 by 3 size. You can do that using this command and when you print A you can see this. Now, I want to evaluate these sums across the rows and the sums across the columns. You can use the apply function to do so, the syntax for this is take the matrix A and then apply this apply function across the rows of matrix A and function you need to apply is sum.

So, now what does is it will sum up this first row 7 + 4 + 1 it is 12 some of the elements in the second row and prints here, and some of the elements in the third row and prints the output. You can do the same for the columns by specifying the margin as 2 which says, apply the sum function on A across the margin 2 which is the columns. This command will prints the sums of the elements in the columns as 3 + 2 + 1 is 6 and so on.

(Refer Slide Time: 06:50)


Data science for Engineers

lapply function

- **lapply** is used to apply a function over a list.
- **lapply** always returns a list of the same length as the input list
  - Syntax: lapply(list, function, ...)

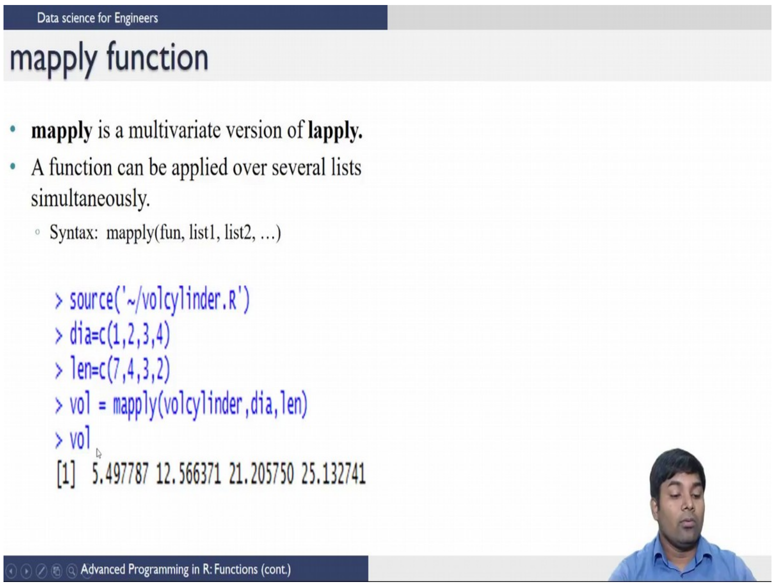
```
> A=matrix(1:9,3,3)
> B=matrix(10:18,3,3)
> Mylist=list(A,B)
> determinant = lapply(Mylist,det)
> determinant
[[1]]
[1] 0

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

Advanced Programming in R: Functions (cont.)

Next we want to lapply function, 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. The syntax for the lapply is as follows. You have to use the command lapply and the list on which the function has to be applied and function which has to be applied on the list. Let us illustrate this lapply using example here create a matrix A which is having the elements from 1 to 9 which is of size 3 by 3 and create a matrix B which are having the elements 10 to 18 and which is of size 3 by 3.

Now, I am creating a list of matrices A and B using the list command. I want to evaluate the determinant of the matrices and then store them in a list. One way to do is calculator determinant of A, calculate determinant of B and calculated it and make them as a list. You can do easily the same operation using the lapply function which is shown here, I want to name that variable has determinant I use a function lapply, I want to apply the determinant function on my list when I do that it will calculate the determinant of A and then store it in the element 1 and calculate the determinant of B and store it in the element 2 of a list.

(Refer Slide Time: 08:19)


Data science for Engineers

mapply function

- **mapply** is a multivariate version of **lapply.**
- A function can be applied over several lists simultaneously.
  - Syntax: mapply(fun, list1, list2, …)

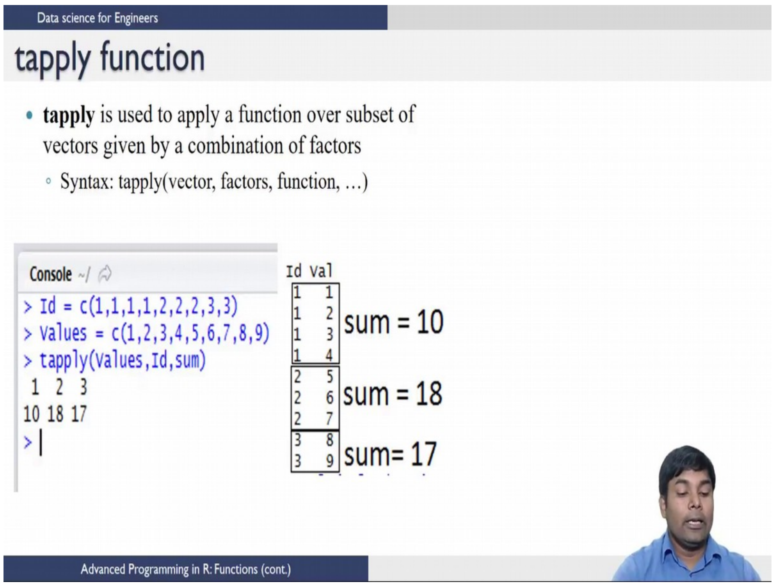
```
> source('~/volcylinder.R')
> dia=c(1,2,3,4)
> len=c(7,4,3,2)
> vol = mapply(volcylinder,dia,len)
> vol
[1]  5.497787 12.566371 21.205750 25.132741
```

Advanced Programming in R: Functions (cont.)

Now, let us move to the mapply. 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. So, we have seen R function vol cylinder. Now, let us suppose I want to calculate the volume for different diameters and different lengths which are having as a list here, I have a list of diameters and I have a list of lengths I want to evaluate the volume for each individual pairs of dia and length.

You can individually take this length and dia and execute the same function, but it is so, tedious mapply helps in simplify this job for us you need to create one variable vol and then apply the function mapply on the vol cylinder function because this is the function first we need to apply the function and then the list 1 and list 2. We have two lists dia and length, and what it does is it will take a pair and then calculate the volume and it will written the volumes were two lists of dia and length that are given.

(Refer Slide Time: 09:40)


Data science for Engineers

tapply function

- **tapply** is used to apply a function over subset of vectors given by a combination of factors
  - Syntax: tapply(vector, factors, function, …)

```
Console ~/
> Id = c(1,1,1,1,2,2,2,3,3)
> 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   sum = 10
1   3
1   4
2   5
2   6   sum = 18
2   7
3   8
3   9   sum= 17
```

Advanced Programming in R: Functions (cont.)

Next we move on to the tapply function. tapply is used to apply a function over a subset of vectors given by combination of factors. The syntax for that is tapply a vector, practice and the function unit of length. So, to illustrate tapply let us use this example, I am creating a vector called Id using this concatenation operator which contains four 1's, three 2's and two 3's and I am also creating another vector which is having the values again a concatenation which are having the elements 1 to 9.

Now, if I want to add the values which are having the same ids tapply function can help me. So, what I need to do is tapply I want to add this values the adding is given here as a function which is sum according to the Id they belong to. What it does is it will take the elements corresponding to one Id for example, four 1's and the values 1 2 3 4 and sums them up 4 + 3, 7 and + 2 9 and 1, so sum is 10. Similarly it will take the values corresponding to the Id 2 and then sum it up and values corresponding to the Id 3 and sum it up so that it will print the outputs which are indicating the sums of the elements of category 1, category 2 and category 3. The tapply prints the output as the sums of the elements which are having Id 1, Id 2 and Id 3.

In this lecture we have seen how to write functions which takes multiple inputs and multiple outputs. We have seen inline functions and we have also seen some functions that are useful to loop over the objects. In the next lecture we are going to discuss about the control structures in R.

Thank you.