

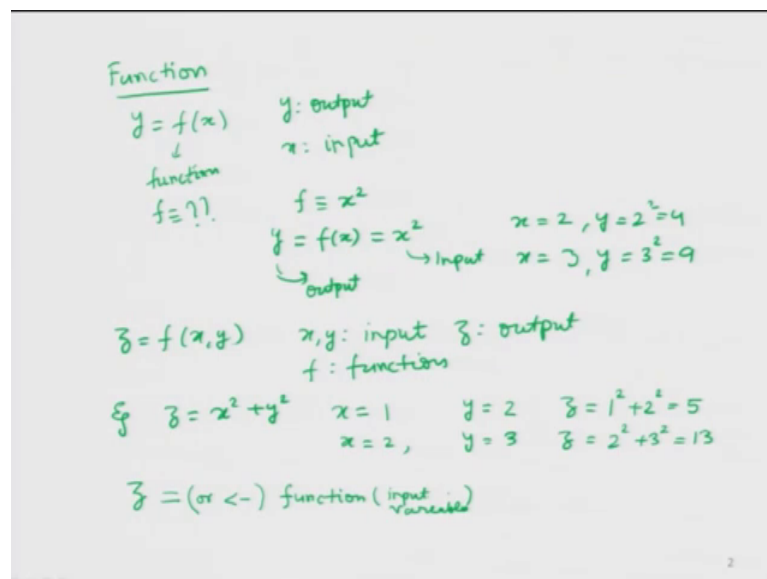
Introduction to R Software
Prof. Shalabh
Department of Mathematics and Statistics
Indian Institute of Technology, Kanpur

Lecture – 06
Functions and Matrices

Welcome to the next lecture on introduction to R software. You may kindly recall that in the earlier lectures we had learnt about how to do the basic mathematical operations in R software like as addition, subtraction, multiplication, division as well as this modulo operations and now we will move to some other aspects of R, which are again the basic fundamentals of R. And in this lecture I will try to tell you something about functions and a little bit about matrix theory and how to use matrix in R. And in the finish next lecture I will try to complete few other things about the Matrix theory.

So, first basic question comes here what is a function.

(Refer Slide Time: 01:04)



Simple high school algebra and high school level concept. Whenever we write y is equal to f of x we call that this is a function. So, here f means function and what is the meaning of this language y equal to f x. We try to say that y is something like output and x is something like input this means whenever I am trying to define a function first I have to specify what is my here this f. For example, just for the sake of understanding if I say f has a form x square; that means, the function will become like y is equal to f of x which

is equal to here x square. This means what? That as soon as I give here the values of x as input values, I will get here a particular value of here y as an output for example, if I take say x equal to here 2 then y will become here 2 square which is equal to here 4 if I try to take here x is equal to 3 then y becomes 3 square and this becomes here 9.

Similarly, incase if I try to say define here another function suppose z which is a function of x and y what does this mean. That simply that here there are 2 variables x and y which are your input variables and z here is your output variable. And f denotes here a function that is the form of the mathematical relationship of x and y with respect to z . So, for example, if I try to say that z is equal to x square plus y square this means that incase if I give here some values of x and some values of y then I will get here some value output as here z . So, for example, if I try to take here x equal to 1 y equal to 2 then I will get here z as 1 square plus 2 square which is equal to here 5 and similarly if I try to take here x equal to 2 y equal to 3 then I will get here z is equal to 2 square plus 3 square is equal to here 13.

So, now what we have to understand that how are we trying to write down this function. I am trying to define here a function here as here z which is equal to or equivalently in your R language I can write down here as a say less than and hyphen sign and then I am trying to define here a f . So, that is try to write down here is completely as a function and then inside this argument I am trying to write down the input variables and so on and then I try to calculate the value of output given here as say z . This is precisely what we do in R. There is only a particular type of syntax which is exactly based on this concept and we try to define a function.

(Refer Slide Time: 05:11)

Functions

- Functions are a bunch of commands grouped together in a sensible unit
- Functions take input arguments, do calculations (or make some graphics, call other functions) and produce some output and return a result in a variable. The returned variable can be a complex construct, like a list.

function (input variable)
↳ define

3

So, now if you try to see here what is the function? Functions are simply a bunch of commands which are grouped together in a sensible fashion or a sensible way or a sensible unit. Suppose I want to find out x square plus y square then I have to write it in a separate in some several steps first step will be give the value of x , second step will b give the value of y , next step will be find the value of x square, next step will be find the value of y square and finally, find out the sum of x square and y square. So, there are so many steps which are involved in finding out the submission of x square and y square. So, when I try to group all these commands together that will simply become a function right.

So, functions have the same rule that we will try to write down here a function and then I will inside this one I will try to write down the input variable and then I need to define the form of the function. Whatever is the form of my function those that many calculations will be made based on that I will get the finally, returned output value. So, this is the basic idea about the definition of a function and this functions can be simple quantity or they can be some complex construction also like a list of several commands and so on.

So, what I am going to do here that I will try to explain you that how do we write the function I will try to take few examples, but after that I will request you that please try to create some more functions yourself and practice them. So, now, what is the syntax?

(Refer Slide Time: 07:07)

Functions

Syntax

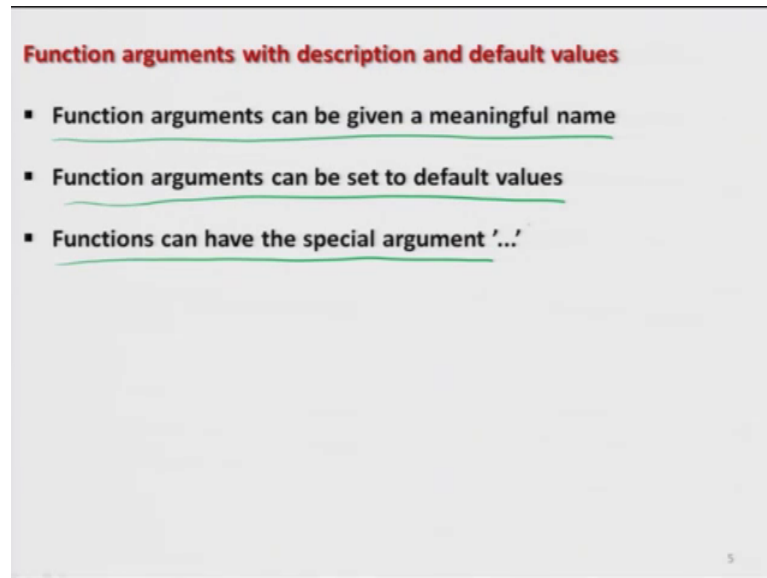
```
Name <- function(Argument1, Argument2, ...) {  
  expression  
}
```

where **expression** is a single command or a group of commands

The syntax is very very simple here you have to give the name which is in output variable and then you have to write function, then inside the brackets you have to write down all the input variables which are separated by comma and then once this bracket is closed over here then you have to make here a curly bracket here and here and in between you have to write down here all the expressions whatever you want to compute and this will become a function. Once I can write down the function then I will tell you how I can do the calculation by calling a particular function.

The advantage of the function is that that whenever we are trying to write down a complicated command or a complicated program then I can divide my entire program into several smaller units and every unit will be written as a function and when I want to operate the entire program I will try to call these functions which will give me the supplementary values and finally, I will try to manipulate with those functions to give us the final outcome. And that is one of the very important aspects of R programming and because of it R has become a very popular language right.

(Refer Slide Time: 08:50)



Function arguments with description and default values

- Function arguments can be given a meaningful name
- Function arguments can be set to default values
- Functions can have the special argument '...'

5

So, now few points to remember. Whenever you are trying to give here an some variable name inside the function arguments always try to give a proper name which has some meaning so that you can identify by their name that whatever you are going to do for example, if you are trying to deal with age and weight. [FL] then incase if you will ask me I will try to give a function name as age and here weight right. And the function arguments can be set to default values as well as you can give your own values also. And beside those things functions k can also have something special type of arguments which is given inside the inverted commas.

So, that we will try to now see with this several example how we are trying to do it.

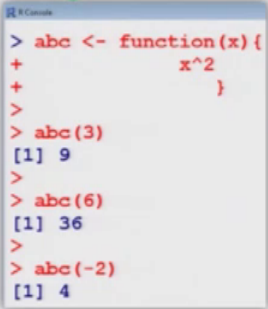
(Refer Slide Time: 09:43)

Functions (Single variable)

The sign `<-` is furthermore used for defining functions:

```
> abc <- function(x) {  
  x^2  
}
```

$y = x^2 \rightarrow \text{Input} = x$
 $abc \quad f = x^2$



```
> abc(3)  
[1] 9  
  
> abc(6)  
[1] 36  
  
> abc(-2)  
[1] 4
```

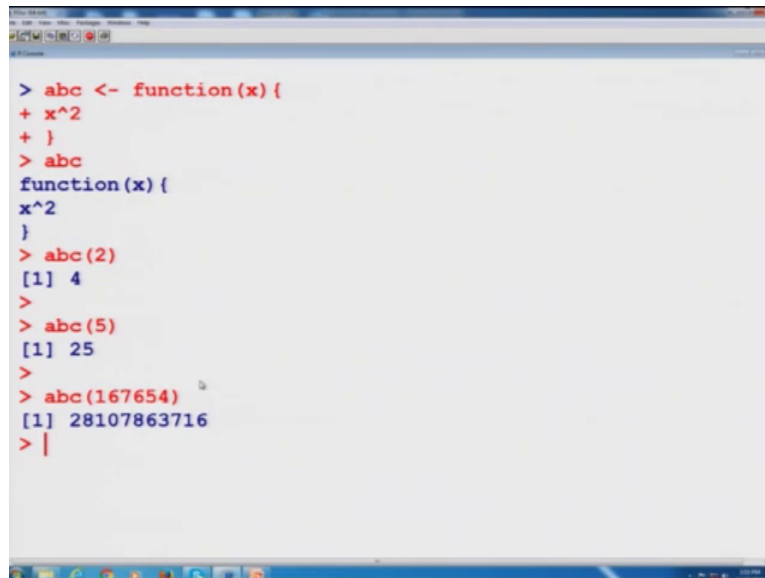
```
> abc <- function(x) {  
+   x^2  
+ }  
>  
> abc(3)  
[1] 9  
>  
> abc(6)  
[1] 36  
>  
> abc(-2)  
[1] 4
```

Suppose, I simply want to compute a function y equal to x square; that means, I would given a value here x and based on that I will I want to write a program that will give me output as a square of my input value. So, what we can do that first of all I have to define a name of the function. So, let us call that suppose I give the function a name `abc` now you can see in this function what are my input variables input here is x . And what is here your f ? This is equivalent to here x square.

So, now you have to observe how I am trying to write down this function on the command prompt first of all I write the name of the function `abc` and after that the equality sign like this or less than hyphen sign and then I am writing exactly the same word `function`. And then here inside this bracket I am trying to write down the input values. So, now, there is only here one input variable x . So, I am writing here only x . In case I have more than one input variable then I will try to write down all the input variables separated by comma. And now I write down this opening curly bracket and I try to write down the form of the function which is here x square and then when I am done when I am when I am finished with all my commands then I will try to close it by this curly bracket right and this will give me the complete function.

So, let us try to first do it on the R and let us try to see how one can do it.

(Refer Slide Time: 12:06)



```
> abc <- function(x){
+ x^2
+ }
> abc
function(x) {
  x^2
}
> abc(2)
[1] 4
>
> abc(5)
[1] 25
>
> abc(167654)
[1] 28107863716
> |
```

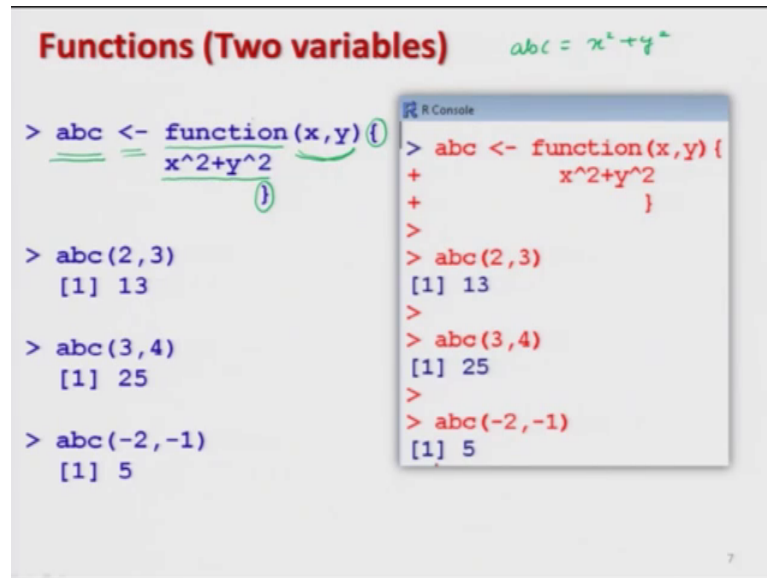
Here we try to see over here I have written here abc less than hyphen sign which is the our equality sign function bracket then inside the bracket input variable x and now on the next line I will try to write down the form of the function which is here x square. And then I will try to write down here this curly bracket which is indicating that I am done with my function. Now there are no more commands to be executed. So, now, you can see here once I get this prompt sign; that means, I have completed my function. So, now, if you try to see here if you try to type abc over here you can see here what is the form of the function right.

Now after this suppose I want to execute this program and suppose I want to input a value x equal to 2 and corresponding to which I want to know what is my outcome. So, what I have to do? I have to do, I have to write down the function name abc and inside this bracket I have to give my input values for example, I am talking here x equal to 2. So, this will be fettle like abc and inside the bracket 2 as soon as I will enter you see that I have got the value here 4 because the square of 2 is 4.

Similarly, in case if I want to know any other value say when x equal to 5 we know the square of 5 is 25. So, let us try to enter and you can see here that I have obtained the same value here and similarly incase if I want to find out any square of another value something like this which are more complicated values you can see here this is here. So, you can see here that by writing a function I can write a small program right. So, now, let

us try to come back to our slide and we continue with our slides one thing I would like to say inform you here that whenever I am trying to write down the function then inside the bracket I have to give the input values input variables. This input variables can be one value or this can be more than one value also right. So, we will try to see it in the next example.

(Refer Slide Time: 14:43)



Functions (Two variables) $abc = x^2 + y^2$

```
> abc <- function(x,y) {  
  x^2+y^2  
}
```

```
> abc(2,3)  
[1] 13  
  
> abc(3,4)  
[1] 25  
  
> abc(-2,-1)  
[1] 5
```

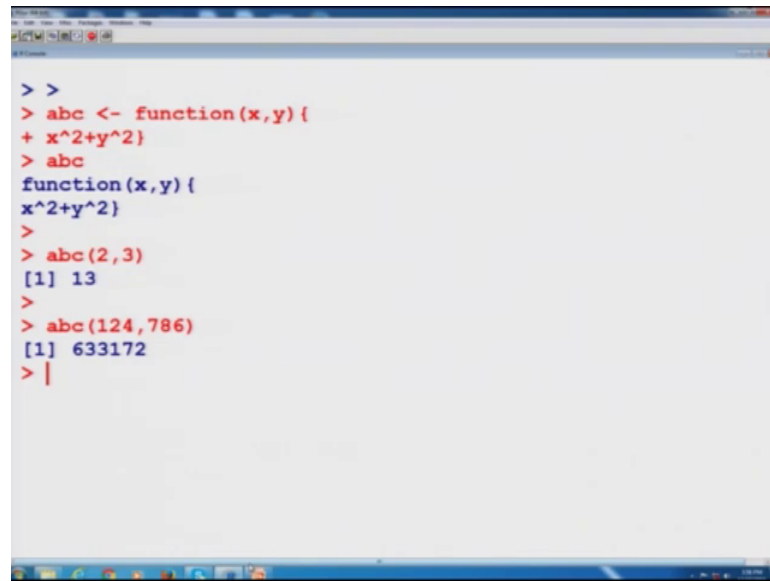
R Console

```
> abc <- function(x,y) {  
+   x^2+y^2  
+ }  
>  
> abc(2,3)  
[1] 13  
>  
> abc(3,4)  
[1] 25  
>  
> abc(-2,-1)  
[1] 5
```

So, now suppose I have here a function say abc which is equal to x square plus y square now I would like to write down this function. So, the rule is simple that I will try to define here a function name then I will try to write down the equality sign then I will try to write down here the function. Now inside this bracket I will try to give all possible input variable whatever are being used in computing the function. Then I would start writing the commands inside this curly brackets and here I try to write down here x hat 2 plus y hat 2 that is x square plus y square and this is my another function.

So, let us try to do it here in the R program itself.

(Refer Slide Time: 15:40)

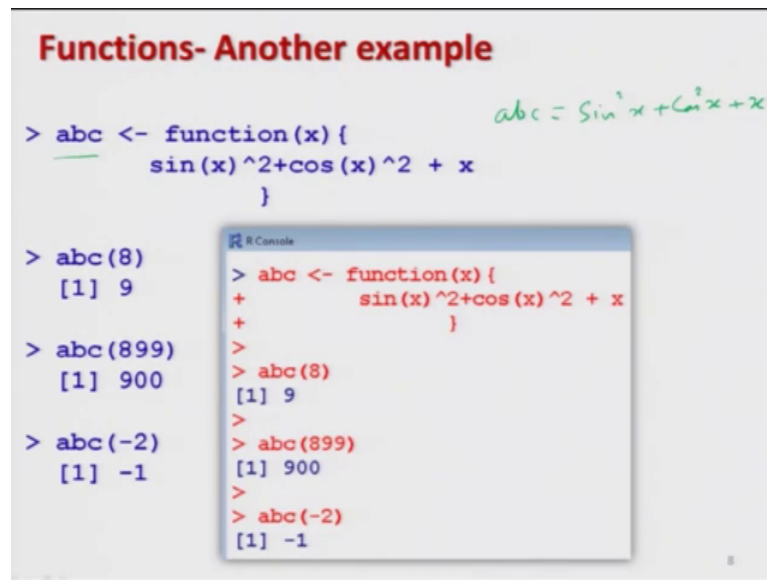


```
> >
> abc <- function(x,y) {
+ x^2+y^2}
> abc
function(x,y) {
x^2+y^2}
>
> abc(2,3)
[1] 13
>
> abc(124,786)
[1] 633172
> |
```

So, now let us try to type this command over here and on the next line I will try to write down here $x^2 + y^2$ and I will try to close it by here a curly bracket right. So, now, you can see here this is my here function a b c. Now I want to find out its value when x is equal to 2 and y is equal to here 3. So, now, I am trying to find out the value of $2^2 + 3^2$ this is nothing, but 13. Similarly incase if I want to find out the square of 1224 plus square of say 786 you can get this values over here.

So, it is simply trying to compute the 124 square plus 786 square right now and then you can see here that I have computed here some more values over here, so you can also try these values in your on your computer to have some more practice and here you can see this is the screen shot whatever I have done it here.

(Refer Slide Time: 16:56)



```
> abc <- function(x) {
  sin(x)^2+cos(x)^2 + x
}
```

abc = Sin² x + Cos² x + x

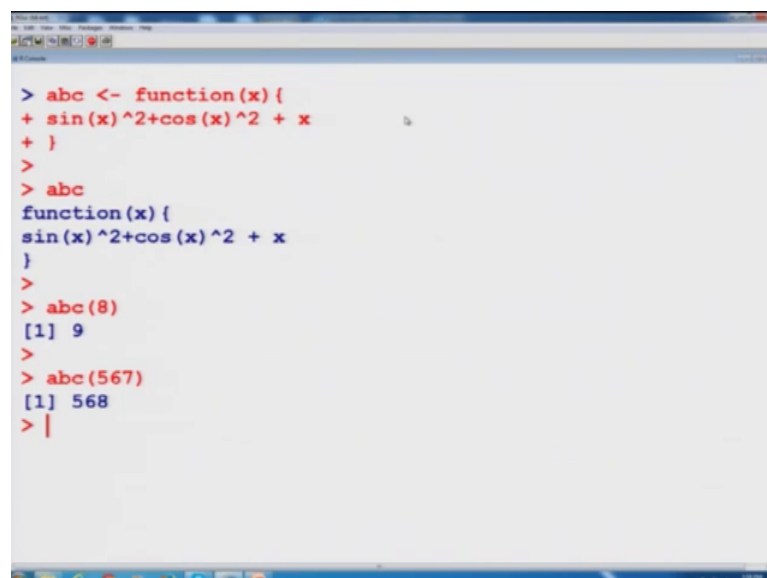
```
> abc(8)
[1] 9
> abc(899)
[1] 900
> abc(-2)
[1] -1
```

```
R Console
> abc <- function(x) {
+   sin(x)^2+cos(x)^2 + x
+ }
>
> abc(8)
[1] 9
>
> abc(899)
[1] 900
>
> abc(-2)
[1] -1
```

Similarly I have made here another example that I am trying to write down here another function here abc which is abc is equal to say sin square x plus cos square x plus here x right we know that sin square x plus cos square x is equal to here 1.

So, let us try to write down this program and can see what really happens over here right.

(Refer Slide Time: 17:31)



```
> abc <- function(x) {
+ sin(x)^2+cos(x)^2 + x
+ }
>
> abc
function(x) {
  sin(x)^2+cos(x)^2 + x
}
>
> abc(8)
[1] 9
>
> abc(567)
[1] 568
> |
```

So, we try to write down this function over here say function x and then I would try to write down here sin x square x plus cos square x plus x and then I will try to close this function and you can see here that this abc now another function which is given by like

this. And now if I want to find out abc of here right that is obviously cos square 8 plus sin square 8 that is 1 plus x that is 1 plus 8 that is equal to 9. And similarly if you want to find out say abc of 567 this comes out to be 568. So, now, you can see here that writing down the program is not difficult at all, but it is a very strong tool in R software that gives us an h over many of the software by writing this function.

(Refer Slide Time: 18:24)

Matrix

Matrices are important objects in any calculation.

A matrix is a rectangular array with p rows and n columns. $x = \begin{pmatrix} & & \\ & & \\ & & \end{pmatrix}$

An element in the i -th row and j -th column is denoted by X_{ij} (book version) or $X[i, j]$ ("program version"), $i = 1, 2, \dots, n, j = 1, 2, \dots, p$.

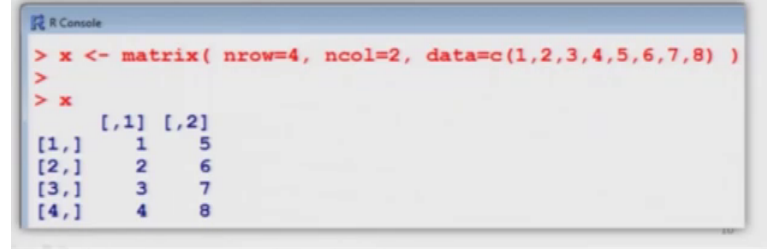
An element of a matrix can also be an object, for example a string. However, in mathematics, we are mostly interested in numerical matrices, whose elements are generally real numbers

The next topic which I am going to take is about matrix theory. Matrix theory we all are aware that matrix is simply a rectangular array which has got certain number of rows say p number of rows and certain number of columns say n number of columns. And this matrix is denoted by something like this or sometime this is denoted as something inside this bracket and it has got certain elements which are denoted by whose address are denoted by the i th row or the j th column.

So, in the bookish language this the i j th element of the matrix say here x is denoted by x_{ij} or in the program version this is denoted by this - x inside the bracket I would try to write X comma j right. So, and now the element of our matrix can be an object this can be a string, but in mathematics and statistics usually we are interested in the matrices which are containing essentially the numerical values. So, in this lecture also initially we are going to concentrate only on those matrix which have got some numerical values.

(Refer Slide Time: 19:44)

```
In R, a 4 × 2-matrix X can be created with a following command:  
> x <- matrix( nrow=4, ncol=2,  
               data=c(1,2,3,4,5,6,7,8) )  
> x  
      [,1] [,2]  
[1,]    1    5  
[2,]    2    6  
[3,]    3    7  
[4,]    4    8
```



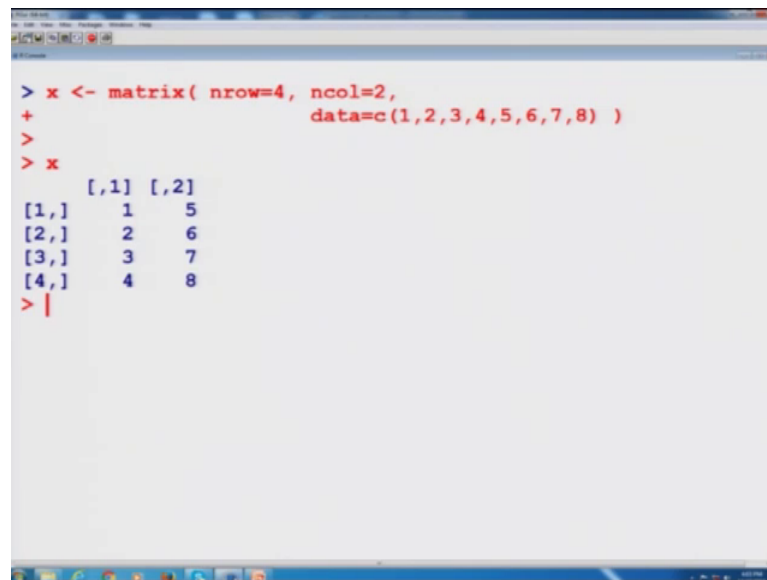
The image shows a screenshot of an R console window. At the top, there is a text box that says "In R, a 4 × 2-matrix X can be created with a following command:". Below this, the R command is shown: `> x <- matrix(nrow=4, ncol=2, data=c(1,2,3,4,5,6,7,8))`. The command is annotated with green circles and arrows: a circle around 'x', a circle around 'nrow=4', a circle around 'ncol=2', and a circle around the closing parenthesis of the data vector. A green arrow points from the 'data=c(1,2,3,4,5,6,7,8)' part of the command to the output matrix. Below the command, the output of the matrix is shown: `> x` followed by a table with two columns labeled [,1] and [,2], and four rows of data: [1,] 1 5, [2,] 2 6, [3,] 3 7, and [4,] 4 8. Below this, there is a screenshot of the R Console window showing the same command and output as above.

So, the first thing is this suppose I want to create a matrix how can I create a matrix from a given set of data. So, first let us try to take some example and later on I will try to show you that what are the complete commands for writing a matrix. Suppose I have got some data 1 2 3 4 5 6 7 8 and I want to arrange this data in a matrix of order 4 by 2 this means that there are going to be 4 number of rows we which are denoted by here n row and there are going to be 2 number of columns which are denoted by here n col.

So, first 2 entry is n row and n col they are trying to tell us the number of rows and number of columns in the matrix. So, as soon as I define a matrix first of all I have to give the matrix a name. So, I have given the matrix a name x and my objective is now to write this set of data 1 to 8 in a matrix of 4 by 2. So, I can write it as say x equal to or say less than hyphen sign then I have to write here matrix m a t r i x and then I have to write down here these 2 brackets. And inside this bracket I have to give all the information that how I want to write my matrix there are going to be 4 rows 2 columns and this is the data which I want to be in my matrix.

So, when I try to do so and when I try to enter it we get here an outcome like this one, right, and the same thing I can show you here also doing with here R that I try to define here a matrix here by this thing and something like this and you see here x comes out to be like this right ok.

(Refer Slide Time: 21:50)



```
> x <- matrix( nrow=4, ncol=2,
+             data=c(1,2,3,4,5,6,7,8) )
>
> x
      [,1] [,2]
[1,]    1    5
[2,]    2    6
[3,]    3    7
[4,]    4    8
> |
```

So, you can now see here that now this is coming out to be a matrix of order 4 by 2, but you have to keep in mind how the data has been arranged 1 2 3 4 and from here 5 then 6 then 7 and then 8 the data is arranged in column wise 1 2 3 4 and then 5 6 7 8. Now there is another issue that if I want to arrange this data in a row; that means, 1 2 3 4 and then 5 6 7 8; that means, I have that I am interested in having a matrix of order 2 by 4 in which the data is going to be arranged in the form of your column right.

So, that also can be done, but before that let me try to explain you what are the different parameters involved in the argument.

(Refer Slide Time: 22:49)

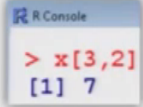
We see:
The parameter `nrow` defines the row number of a matrix.
The parameter `ncol` defines the column number of a matrix.
The parameter `data` assigns specified values to the matrix elements.
The values from the parameters are written column-wise in matrix.

```
> x
      [,1] [,2]
[1,]    1    5
[2,]    2    6
[3,]    3    7
[4,]    4    8
```

$x_{3,2}$

One can access a single element of a matrix with `x[i,j]`:

```
> x[3,2]
[1] 7
```



11

Here this `nrow` is trying to define the number of rows in a matrix; `ncol` is trying to define the number of columns in a matrix and the parameter `data` that is going to assign the specified values to the element in the matrix right. And the default approach is that that the elements have written column wise they just like this. And now suppose this is the matrix that we have obtained in the earlier screen and suppose I want to obtain a particular element, suppose I want to element in the second column and the third row.

So, this is nothing, but your `x[3,2]`. So, this is obtained by writing say here `x` and inside this square brackets now you are not going to use the simple brackets, but you have to use here the square bracket and inside this brackets you have to write down the address the address here is third row and second column. And as soon as you try to write it down you get here the value here 7 for example, here you can see here that suppose I want to write down here `x[3,2]` I get here 7 and similarly if I try to write down here `x[1,1]`, I get here the first element that is here 1.

(Refer Slide Time: 24:12)

```
In case, the data has to be entered row wise, then a 4 × 2-matrix X
can be created with

> x <- matrix( nrow=4, ncol=2,
               data=c(1,2,3,4,5,6,7,8), byrow = TRUE)
> x
      [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
[4,]    7    8
```

Now, suppose if I want to enter the data row wise then I have to write down the same syntax from here to here, what I did earlier, but now I have to write down here by row this is equal to here capital TRUE, that is true right and once you try to do it here then you will get this type of outcome where you can see that the data is arranged like 1 2 then 3 then 3 then 4 then 5 then 6 then 7 then 8 right and let us try to do it here inside the R software also.

So, now you can see here that I have got this type of data over here now you can see here this was the means earlier matrix were arranged column wise, but this is arranged row wise right. So, this is the similar the outcome of when I enter the same data row wise and column wise. So, you can see here that here there was a hidden command this was written by row which was equal to here false. So, this we will try to understand when we try to understand when we try to discuss the complete command of this matrix theory and we will try to discuss some more aspect of this matrix; the idea of giving you some basic knowledge that how to write a matrices in R was given in this lecture.

Now, my request is that please try to have a revision of all these topics particularly with the matrix theory and in the next lecture we will try to discuss some more aspects of matrix theory. And I would also request you that please try to have a quick look in any elementary book on matrix theory whatever you have done in your high school level or class twelve level what whatever you have done that will help you in the next lecture.

Till then, good bye.