

Essentials of Data Science with R Software - 2
Sampling Theory and Linear Regression Analysis
Prof. Shalabh
Department of Mathematics and Statistics
Indian Institute of Technology, Kanpur

Introduction to R Software
Lecture - 06
Operations with Matrices

Hello, welcome to the course Essentials of Data Science with R Software - 2 where we are handling the topics of sampling theory and linear regression analysis. And in this lecture, we are going to talk about the module on Introduction to R Software. So, up to now, in the last couple of lectures, we have discussed the various aspects of R Software, various types of commands, instruction etc., which are going to be useful when we start our statistics part.

So, continuing on the same lines, in this lecture, I am going to talk about the matrix. Matrix are extremely useful in data sciences and in statistics, we feel happy if I can translate something to a format of matrix.

So, it is very important for us to understand how to do the matrix operations, and there are some precautions which we have to take when we are trying to work in R with matrices. So, I will try to give you some examples here and some basic fundamental. So, let us start with our lecture on Matrices, and we will learn different types of operations.

(Refer Slide Time: 01:42)

Matrix

A matrix is a rectangular array with p rows and n columns.

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

We consider only numerical matrices, whose elements are generally real numbers.

$$X = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

array $\left\{ \begin{array}{l} \downarrow \downarrow \text{columns} \\ \leftarrow \text{rows} \\ 2 \times 2 \\ \rightarrow X_{ij} \text{ or } X[i, j] \end{array} \right.$

So, first question comes here what is a matrix? So, matrix you know that for example, I can write down here a matrix something like here 1 2 3 4. So, what is this? This is a sort of array, but it has something that each element has got an address. So, these are my here rows and these are my here columns.

So, you can see here there are here two rows – row 1, row 2 and with there are here two columns – 1 and here 2. So, this is an array of order 2 by 2. So, a matrix is essentially a rectangular array which say p rows and n columns. And whatever are the elements in the matrix, they have got an address. So, they will have two things – one is the location and another is the value.

For example, in this matrix here X if I want to write out this first element here 1, so the position of this element in the matrix is X is in the first row and first column. So, this is denoted by X_{11} or this can be denoted by here X say here 11. So, 11 is giving the address.

So, in case if we want to address or we want to call any element of this matrix say for example, if I want to call an element in the i th row and j th column, then this is denoted by X_{ij} which is actually this is how we type inside a book, so that is why I have written here is a book version. But in say programming, usually we call it by X i comma j and with some bracket, different programs have different types of brackets right. So, there are i rows, i goes from 1 to n and there are p columns, so j goes from 1 to p .

And remember one thing this matrices may be of different types, they may have some real values, complex values, imaginary values. But here we are going to consider only those matrix which have got numerical values whose elements are generally the real numbers, right, so that is our basic assumption, ok.

(Refer Slide Time: 04:17)

Matrix

In R, a 4×2 -matrix X can be created with a following command:

```
> x = matrix(nrow=4, ncol=2, data=c(2,3,4,5,6,7,8,9))
```

$X = \begin{pmatrix} \rightarrow & \downarrow & \downarrow \\ \rightarrow & & \\ \rightarrow & & \\ \rightarrow & & \end{pmatrix}$

$X = \begin{pmatrix} 2 & 6 & 2 & 3 & 4 & 5 \\ 3 & 7 & 6 & 7 & 8 & 9 \\ 4 & 8 & & & & \\ 5 & 9 & & & & \end{pmatrix}$

R Console Output:

```
> x = matrix(nrow=4, ncol=2, data=c(2,3,4,5,6,7,8,9))
> x
     [,1] [,2]
[1,]  2   6
[2,]  3   7
[3,]  4   8
[4,]  5   9
```

So, now the first question comes here how are you going to create a matrix? So, let me first take an example, and then I will try to give you the more details. Suppose, I want to create a matrix of order 4 by 2, order 4 by 2 means 4 is the number of rows, and 2 is the number of columns, right.

And this and then I have to so that means, I have here a matrix here X , where I have defined that there is going to be first row, second row, third row, fourth row, and first column, second column, but what will be the elements? So, for that, I have to define some values.

So, the command in R to create a matrix is `matrix` – `m a t r i x`. And inside this parenthesis, we try to write different types of information which has which are feeded to the matrix command. So, first information in order to create a matrix is we have to specify the number of rows and number of columns.

So, this is denoted by `nrow`; `nrow` will indicate the number of rows and the command `ncol` – `n c o l` that will indicate the number of columns. So, here I am writing `nrow` equal to 4 and `ncol` equal to 2, that means, there are 4 rows and 2 column. Now, the next question is what are the entries inside the data matrix? So, that is indicated by the option here `data`.

Data is equal to, then I try to write down here is the data vector. Remember one thing I will clarify my language once again that here in the matrix theory, there are there is a matrix there is a vector. Vector is to simply a row vector or a column vector where the number of rows or

number of columns in a matrix are simply 1, right means either row or column. So, that is a vector also, that is also a vector. But then what I am saying here this is my data vector. Data vector means all the numerical values are combined with the command `c`. So, the, so this command here `data` will provide the data vector. So, data vector I am taking here `c 2, 3, 4, 5, 6, 7, 8, 9`. And as soon as you execute this command on the R console, you will get here like this outcome.

So, you have to first observe what it is happening. So, you can see here I have given the number of rows 1, 2, 3, 4 which are here 1, 2, 3, and here 4; number of columns are 2 – this is first and second. And you can see here how this number of how these rows and columns are indicated.

Here in the rows after the comma, there is a blank sign; after the comma, there is a blank sign, where is in the column; there is a blank sign, and then comma, and then the column number, then there is a blank sign, then the comma, and then the column number.

And then inside this matrix, the data is given at 2, 3, 4, 5, 6, 7, 8, 9, it is coming like this – 2, then 3, then 4, then 5, and then here 6, then 7, then 8, and then 9, right. So, you can see here that this data is entered in a particular fashion. There is a rule.

There are two option means, whether the data can be entered at 2, 3, 4, 5, 6, 7, 8, 9 or the data can also be entered at 2, 3, 4, 5, 6, 7, 8, 9, right; or in our setup, this can be 2, 3, 4, 5, 6, 7, 8, 9. So, now, you can see here this is the default, but we need a specific function or a command to indicate that how we want to enter our data. But at this moment, I am not using it.

(Refer Slide Time: 08:24)

Matrix

We see:

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

So, this is a general command, but and from this command you can see here I have three parameters which I have used `nrow`, `ncol` and `data`. So, `nrow` is going to denote the number of rows; `ncol` is going to denote the number of columns; and `data` is going to assign the data to the elements of matrix, right.

(Refer Slide Time: 08:47)

Matrix

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

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

```
> x[3,2]
[1] 8
```

X_{ij}

And now you can see here that if I define this matrix and suppose if I want to access a particular element, so as we have seen the element is in the symbols of books this is denoted by X_{ij} . But in the language of computers or the programming language the i th the ij th element of a matrix X is accessed by `x` square bracket and then `i` comma `j`, right.

For example, if I write here x 3, 2, so you can see here, what is here x 3, 2? 3 is the number of row and 2 is the number of column. So, x 3 and then here 2, this is here 8. So, this 8 is here, right. So, this is how we can access a particular element in the vector here x, ok.

(Refer Slide Time: 09:47)

Matrix

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(2,3,4,5,6,7,8,9), byrow = TRUE)
> x
```

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

R Console

```
> x = matrix( nrow=4, ncol=2, data=c(2,3,4,5,6,7,8,9), byrow = TRUE)
> x
```

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

Now, before going further, let me try to show you this thing on the R console also. So, let me try to copy this command over here.

(Refer Slide Time: 10:05)

```
> x = matrix( nrow=4, ncol=2, data=c(2,3,4,5,6,7,8,9) )
> x
```

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

```
> x[3,2]
[1] 7
>
> x(3,2)
Error: could not find function "x"
> x(3,2)
Error: unexpected ',' in "x(3,2)"
> |
```

And now I have copied this command on the R console. And you can see here this is my here x. And in case if I want to call a particular element, suppose if I want to call the element in the third row and second column, it is here 8. And if you try to see, if I try to use here, say another type of bracket, says simple parentheses 3 comma 2, you can see here this will give me error, right.

Even if you try to use one bracket like this – one parenthesis, even then it will give you an error. So, the command here is that you have to use only the simple say square brackets. Right, ok. So, right ok.

So, now, I am coming to an aspect where I am going to introduce one more parameter in the same command. Now, I am adding here one more command here byrow is equal to TRUE. So, byrow is the command which is telling the matrix command that how the values in the data shall be entered. For example, the values inside the data vector are given by here like this. So, how they have to be entered?

So, I am saying here they have to be entered by row and this is true, that means, byrow that means, the pattern has to be follows TRUE. If I change it to FALSE, then it will become column wise means byrow will be FALSE, so automatically that will indicate column wise. So, you can see here if I try to say here byrow equal to TRUE, now the values are 2, 3, they are going by rows, then 3 to 4, then 4 to 5, 4, 5 to 6, 6 to 7, 7 to 8, and 8 to 9, right.

(Refer Slide Time: 12:09)

Matrix

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

```
> x = matrix( nrow=4, ncol=2, data=c(2,3,4,5,6,7,8,9), byrow = FALSE)
> x
```

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

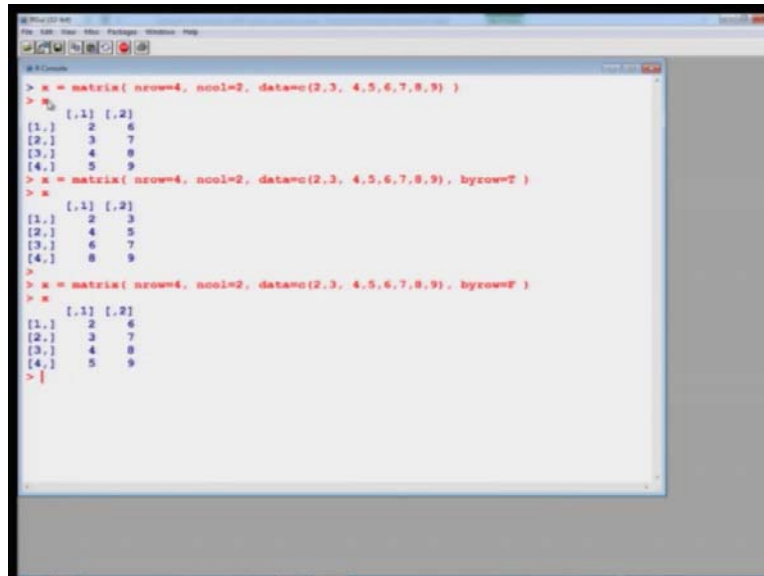
R Console

```
> x = matrix( nrow=4, ncol=2, data=c(2,3,4,5,6,7,8,9), byrow = FALSE)
> x
```

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

And similarly if I make it here byrow equal to FALSE, then the data will not be arranged by rows, but automatically that will be arranged by column. So, this can you can see here 2, then 3, then 4, and then 5, and then here 6, then 7, then 8, and then 9. So, this is how we can control the arrangement of the data values in the matrix. So, let me try to show you here on the R console.

(Refer Slide Time: 12:49)



```
> x = matrix( nrow=4, ncol=2, data=c(2,3, 4,5,6,7,8,9) )
> x
     [,1] [,2]
[1,]  2   6
[2,]  3   7
[3,]  4   8
[4,]  5   9

> x = matrix( nrow=4, ncol=2, data=c(2,3, 4,5,6,7,8,9), byrow=T )
> x
     [,1] [,2]
[1,]  2   3
[2,]  4   5
[3,]  6   7
[4,]  8   9

> x = matrix( nrow=4, ncol=2, data=c(2,3, 4,5,6,7,8,9), byrow=F )
> x
     [,1] [,2]
[1,]  2   6
[2,]  3   7
[3,]  4   8
[4,]  5   9
> }
```

So, you can see here I will try to use here two commands in the same matrix. So, I have created this matrix here x. So, now, you can see here I am trying to add here one more command; byrow is equal to TRUE means I can use now 1 letter only TRUE, and now you can see here this is here x. And you can see here now this arrangement of the data is changed. Earlier it was 2, 3, 4, 5, going in the downward direction; now it is row wise 2, then 3, then 4, then 5.

And in the same command, if I try to make it byrow equal to here FALSE, by F you can see here now the x matrix is change. And this is the same as this one – first one. This and this, they are the same. So, the default entry in the R software for matrix is byrow is equal to FALSE.

The data will always be entered column wise. And if you want to have it row wise, then you have to specify that byrow is equal to TRUE. That is a very important aspect, and that we need to follow. And I will suggest you a simple trick that if whenever you get confused just try to take a matrix of say 2 by 2 or 2 by 3, and then try to first check what is happening, and then you try to enter your data accordingly.

(Refer Slide Time: 14:04)

```
Matrix: Transpose of a matrix X: X'
```

Consider the matrix

```
> x = matrix( nrow=4, ncol=2, data=c(2,3,4,5,6,7,8,9), byrow = FALSE)
```

```
> x
```

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

```
> xt <- t(x)
```

```
> xt
```

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

Handwritten notes:

$$X = \begin{pmatrix} 2 & 6 \\ 3 & 7 \\ 4 & 8 \\ 5 & 9 \end{pmatrix}$$
$$\text{transpose}(X) = \begin{pmatrix} 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 \end{pmatrix}$$

Now, in matrix theory, there are different types of operations. Sometime these operations are required for the mathematical manipulations, and sometime they are required for arranging the data values. Because whenever in a statistics you want to apply any tool, you have to first enter the data. And different types of commands have different types of requirements.

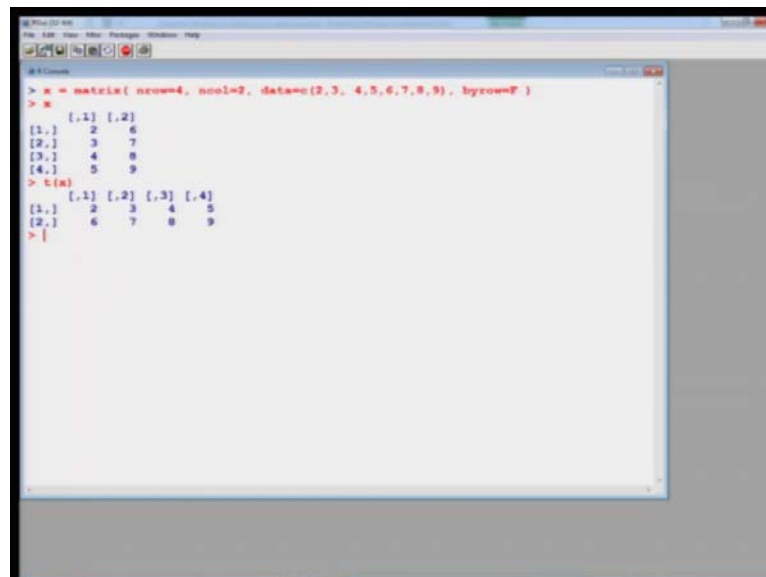
Sometime they want the data input to be in the form of data frame, sometime in the form of matrix, sometimes in the form of a column vector and so on, so that is my idea here to take a couple of commands which are useful for us in the further lectures. So, suppose I again take the same matrix. I am just taking the same matrix for the sake of convenience, so that you do not get confused. So, again the matrix is my same 4 by 2 matrix with the data vector 2, 3, 4, 5, 6, 7, 8, 9, and the data has been arranged by the column wise. So, you can see here, this is my here matrix x.

Now, my objective is to find out the transpose of a matrix. The transpose of a matrix is defined as the matrix in which the rows and columns are interchanged. The row becomes column, and the column become rows. So, you can see here this is my here matrix x which is here like this; x is equal to 2 3 4 5 6 7 8 9.

And if you want to take its transpose, the transpose of x from our mathematical knowledge we know, this will become the first row will become, sorry, first column will become first row. This will become 2 3 4 5; and second column will become second row 6 7 8 9, right.

So, in R the command here is t. So, you have to write down t. And inside the parentheses you have to write down the matrix. So, you can see here if I try to find out the transpose of this matrix here x, the command will come out to be like this here, right, ok. So, now first let me try to show you it on the R console, and then I will move forward. So, ok.

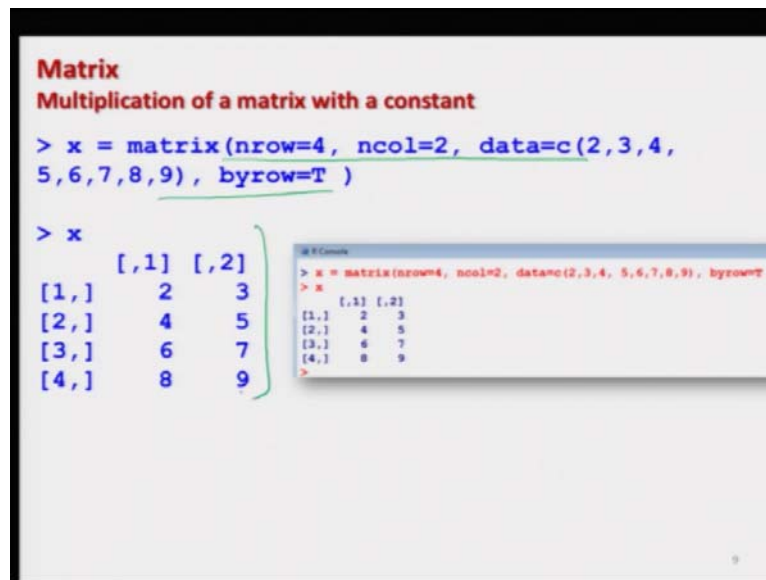
(Refer Slide Time: 16:25)



```
> x = matrix( nrow=4, ncol=2, data=c(2,3, 4,5,6,7,8,9), byrow=F )
> x
     [,1] [,2]
[1,]    2    6
[2,]    3    7
[3,]    4    8
[4,]    5    9
> t(x)
     [,1] [,2] [,3] [,4]
[1,]    2    3    4    5
[2,]    6    7    8    9
> |
```

So, suppose I try to define here this here x here like this, now you can see here I am trying to find out the transpose this is here, the rows and columns they are interchanged. So, you can see that this command is working. And this is a command for finding out the transpose. And I do one thing here that I try to store this value of the transpose of x inside a new variable xt, which I will be using later on to show you different types of manipulation.

(Refer Slide Time: 16:55)



```
Matrix
Multiplication of a matrix with a constant

> x = matrix(nrow=4, ncol=2, data=c(2,3,4,
5,6,7,8,9), byrow=T )

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

The screenshot shows an R console window with the following content:

```
> x = matrix(nrow=4, ncol=2, data=c(2,3,4,
5,6,7,8,9), byrow=T )
> x
      [,1] [,2]
[1,]    2    3
[2,]    4    5
[3,]    6    7
[4,]    8    9
```

So, again means I have created the same matrix, but I want to show you here that when we are trying to multiply a scalar with a matrix, then how it can be done and what happens. And here I would like to have a caution. When we are trying to do the matrix operation, whenever we are trying to multiply two matrices, then there are certain rules of mathematics which have to be followed.

You have to ensure and this is your job to ensure that the orders of the matrix have been defined in such a way such that the matrix multiplication is valid. Means, R will not tell you, R will simply say in case if you are giving the matrices of wrong orders, R will simply tell you that there is an error. And if you try to take 5 matrices and if you want to find out their products, R will not be able to tell you whether you have made a mistake in the second matrix or in the fourth matrix that you that is your responsibility – number 1.

When we are going to consider the multiplication with respect to matrices, there are three possibilities. One is a scalar with matrix, and matrix with matrix. Now, once I come to and for that there are two different operations, there are two different types of command. But when we are trying to deal with matrix with matrix, but if I use the symbol of a scalar, it will also give you something. It will not give you error. So, you have to be careful.

So, I will try to show you with an example what I mean. So, but before that let me take the same example. So, now, I have here the same matrix that we have considered up to now the 4 by 2 matrix, and I want to multiply this matrix by a scalar.

(Refer Slide Time: 19:00)

```
Matrix
Multiplication of a matrix with a constant

> x = matrix(nrow=4, ncol=2, data=c(2,3,4,
5,6,7,8,9), byrow=T )
> x
      [,1] [,2]
[1,]  2  3
[2,]  4  5
[3,]  6  7
[4,]  8  9

> 3*x
      [,1] [,2]
[1,]   6   9
[2,]  12  15
[3,]  18  21
[4,]  24  27
```

Suppose, I take this here matrix here x and if I try to multiply by scalar 3. So, in matrix, we know that when we want to multiply this matrix x by 3, then all the elements will be multiplied by 3 like this, right. And you can see here this is here the outcome, ok.

(Refer Slide Time: 19:32)

```
Matrix
Multiplication of a matrix with a constant

# R Console
> x = matrix(nrow=4, ncol=2, data=c(2,3,4, 5,6,7,8,9), byrow=T )
> x
      [,1] [,2]
[1,]    2    3
[2,]    4    5
[3,]    6    7
[4,]    8    9

>
> 3*x
      [,1] [,2]
[1,]    6    9
[2,]   12   15
[3,]   18   21
[4,]   24   27
>
```

Now, and this is the screenshot of the same command ok.

(Refer Slide Time: 19:36)

Matrix
Matrix multiplication: operator %*%

Consider the multiplication of X' with X

```
> xtx = t(x) %*% x
> xtx
      [,1] [,2]
[1,] 120  140
[2,] 140  164
```

R Console

```
> xtx = t(x) %*% x
> xtx
      [,1] [,2]
[1,] 120  140
[2,] 140  164
>
```

Handwritten notes:

- 10×70
- Diagram: $X \begin{matrix} m \times n \end{matrix} \cdot Y \begin{matrix} n \times p \end{matrix} = Z \begin{matrix} p \times q \end{matrix}$

Now, I try to do one thing. Now, I consider the matrix multiplication. We know that if I have two matrices X and Y . And if X is of order m cross here n , then X and Y can be multiplied only if Y has an order of n cross that can be n cross n or n cross p whatever you want, but these two things should match.

In the number of columns in X and the number of rows in Y , they should match. And if there is another matrix here Z , then it has to be of order say p cross q such that these two p 's are the same that is what we have to keep in mind. So, now, I am trying to multiply here two matrices x and trace of x , right.

So, in order to multiply two matrices my command is %*%. Remember for a scalar it was only star, but now for the multiplication of two matrices we have %*%. And if you try to do it here, you will get an outcome here like this, right. I am not going to explain you here how to obtain the matrix multiplication, because I believe that that you know those things.

(Refer Slide Time: 20:57)

```
Matrix  
Matrix multiplication: operator %**%  
  
> y = matrix(nrow=2, ncol=2, data=c(2,3,4,5),  
byrow=T )  
  
> z = matrix(nrow=2, ncol=2, data=c(12,13,  
14,15), byrow=T )  
  
> y  
      [,1] [,2]  
[1,]    2    3  
[2,]    4    5  
  
> z  
      [,1] [,2]  
[1,]   12   13  
[2,]   14   15
```

But before I go further, let me try to show you here two commands. One thing is about multiplication with a scalar and say another is the multiplication with the matrix.

(Refer Slide Time: 21:19)

```
> x = matrix(nrow=4, ncol=2, data=c(2,3,4, 5,6,7,8,9), byrow=T )  
> x  
      [,1] [,2]  
[1,]    2    3  
[2,]    4    5  
[3,]    6    7  
[4,]    8    9  
  
> 3*x  
      [,1] [,2]  
[1,]    6    9  
[2,]   12   15  
[3,]   18   21  
[4,]   24   27  
  
> t(x) %*% x  
      [,1] [,2]  
[1,]  120  140  
[2,]  140  164  
> ]
```

So, let me try to copy here the same matrix, ok, and on the R console I try to paste it. So, now, I have here x, and suppose I try to find out say 3 into see here x. So, you can see here every element is multiplied by here 3, right. Now, in case if I want to consider here a matrix here trace of x, and if I try to multiply it by here x, so you can see here this will come out to be like this right. Now, at the same time means, I want to show you here one thing.

(Refer Slide Time: 22:05)

```
R Console  
> x  
  [,1] [,2]  
[1,]  2   3  
[2,]  4   5  
[3,]  6   7  
[4,]  8   9  
> y=3*x  
> y  
  [,1] [,2]  
[1,]  6   9  
[2,] 12  15  
[3,] 18  21  
[4,] 24  27  
> x*y  
  [,1] [,2]  
[1,] 12  27  
[2,] 48  75  
[3,] 108 147  
[4,] 192 243  
> |
```

Suppose, x is here is like this. And suppose I define here another matrix 3 into x, and I store it as your y. So, you can see here. So, this is my here y. Now, I try to use here $x * y$; $x * y$ means $*$ was a symbol for multiplication by scalar. But now you can see here x and y both are matrices, they are of order 4 by 2. What do you think? What it will do? You can see here, this is the outcome.

So, what you have to see here? There is an element wise multiplication. 2 the first element first element of the two matrices they are multiplied 2 into 6 is here 12. Similarly, the first the element in the first row second column in x is 3, the similar address in y is 9. And 3 into 9 is here 27.

So, actually this is we have trying to give you the element wise product. And it is not actually wrong, because in matrix theory we have different types of multiplications. So, this is also a special type of multiplication right. I am not going into those details.

There are different types of multiplication, matrix multiplication, Kronecker product, Hadamard product and so on an inner product, but definitely I am not going to discuss those thing. But I am simply warning you that if you need anything at a greater stage, you should be able to do it, and you should know what are you really doing with R, ok.

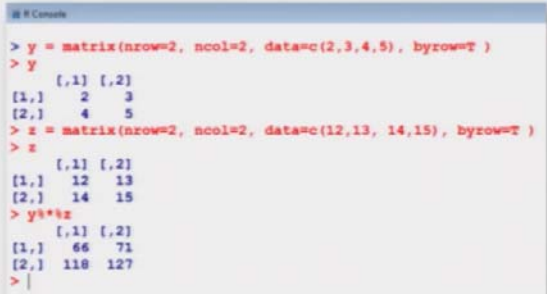
Now, I try to take here say here more example, I try to take here two; I am simply trying to create here two matrices y and z which have which are both are 2 by 2 matrices. And the

elements I have given here as say in y – 2, 3, 4, 5; and in z – 12, 13, 14, 15 just to identify, so that you can easily remember it. So, my y will look like this, and my z will look like this.

(Refer Slide Time: 24:20)

Matrix
Matrix multiplication: operator %%**

```
> y%**z
      [,1] [,2]
[1,]   66   71
[2,]  118  127
```



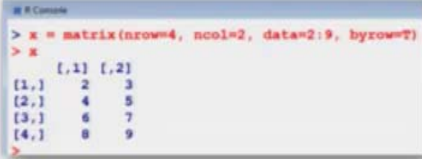
```
> y = matrix(nrow=2, ncol=2, data=c(2,3,4,5), byrow=T)
> y
      [,1] [,2]
[1,]   2   3
[2,]   4   5
> z = matrix(nrow=2, ncol=2, data=c(12,13, 14,15), byrow=T)
> z
      [,1] [,2]
[1,]  12  13
[2,]  14  15
> y%**z
      [,1] [,2]
[1,]   66   71
[2,]  118  127
> |
```

Now, in case if I try to have multiplication product, there is no problem y is 2 by 2, z is 2 by 2. So, you can see here that this product will come like this, and you can see this is here the cross screenshot.

(Refer Slide Time: 24:34)

Matrix
Addition and subtraction of matrices (of the same dimensions) can be executed with the usual operators + and -

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



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


And similarly if you want to add or subtract it, so we have a similar command over here.

(Refer Slide Time: 24:42)

Matrix
Addition and subtraction of matrices (of the same dimensions) can be executed with the usual operators + and -

Create another matrix.

```
> 6*x  
[1,] 12 18  
[2,] 24 30  
[3,] 36 42  
[4,] 48 54
```

R Console

```
> 6*x  
[1,] 12 18  
[2,] 24 30  
[3,] 36 42  
[4,] 48 54  
>
```

$X \pm Y$
 $p \times q \quad p \times q$

Suppose, I and yeah that goes without saying that when you are trying to add or subtract two matrices means once again the rules of matrix theory comes into picture that you can add or subtract the matrices of the same order. For example, if you are trying to add or subtract two matrices X and Y, then both of them should have the same order right that we know.

So, all those rules have to be followed. So, what I am doing here that I have created here a matrix of order 4 by 2, the same earlier matrix and I am trying to multiply it by 6, so that I create another matrix of the same order 4 by 2.

(Refer Slide Time: 25:23)

Matrix
Addition and subtraction of matrices (of the same dimensions!) can be executed with the usual operators + and -

```
> x + 6*x
  [,1] [,2]
[1,]  14  21
[2,]  28  35
[3,]  42  49
[4,]  56  63
```

```
> x - 6*x
  [,1] [,2]
[1,] -10 -15
[2,] -20 -25
[3,] -30 -35
[4,] -40 -45
```

And then I am trying to see here that if you want to add two matrices, then there is no issue then the command is only the plus sign. And if you want to have the subtraction, then this is here the minus sign. It is not difficult because x is of order 4 by 2, 6 into x is also 4 by 2.

So, you can see that both are added. And these are here the screenshot. So, before I go further, let me try to show you them on the R console here, right. So, suppose I try to create here two matrices here y and here z .

(Refer Slide Time: 26:14)

```
> x = matrix(nrow=2, ncol=2, data=c(12,13, 14,15), byrow=T)
> y
  [,1] [,2]
[1,]  2   3
[2,]  4   5
> x
  [,1] [,2]
[1,] 12  13
[2,] 14  15
> y*x
  [,1] [,2]
[1,] 66  71
[2,] 118 127
> y*x
  [,1] [,2]
[1,] 24  39
[2,] 56  75
> y+x
  [,1] [,2]
[1,] 14  16
[2,] 18  20
> y-x
  [,1] [,2]
[1,] -10 -10
[2,] -10 -10
> |
```

So, you can see here this is your here y, and this is your here z. So, you can see here if you want to multiply, if you want to have a matrix multiplication say y into z, this is here like this. And yeah if you try to multiply here simply here y and z element wise if you want, then it is y * z, so that you have to keep in mind.

And similarly if you try to see here now both these matrices they are of the same order. So, if I want to have the addition y plus z, you can see here they are added. You can see here y is here 2, and then z is here 12 the first element. So, 12 plus 2 is 14 and so on. So, that is what is happening, right.

And now if I want to make it here y minus z, you can see here the subtraction is also possible. So, addition and subtraction, they are straight forward. They are using the same symbol whatever we have done earlier. And similarly you can try with this command also that straight forward right I do not need to show it here, right.

(Refer Slide Time: 27:31)

```
Matrix  
Matrix Addition:  
  
> y = matrix(nrow=2, ncol=2, data=c(2,3,4,5),  
byrow=T)  
  
> z = matrix(nrow=2, ncol=2, data=c(12,13,  
14,15), byrow=T)
```

> y	[,1] [,2]	> z	[,1] [,2]
[1,]	2 3	[1,]	12 13
[2,]	4 5	[2,]	14 15

18

(Refer Slide Time: 27:33)

Matrix
Matrix Addition and Subtraction:

```
> y+z
      [,1] [,2]
[1,]  14  16
[2,]  18  20

> y-z
      [,1] [,2]
[1,] -10 -10
[2,] -10 -10
```

```
R Console
> y = matrix(nrow=2, ncol=2, data=c(2,3,4,5), byrow=T)
> y
      [,1] [,2]
[1,]  2    3
[2,]  4    5
> x = matrix(nrow=2, ncol=2, data=c(12,13, 14,15), byrow=T)
> x
      [,1] [,2]
[1,] 12   13
[2,] 14   15
> y+z
      [,1] [,2]
[1,] 14   16
[2,] 18   20
> y-z
      [,1] [,2]
[1,] -10  -10
[2,] -10  -10
> |
```

So, this is exactly what I have shown you here on the R console. So, you can just take the same command and practice and verify whether you are getting the same thing or not.

(Refer Slide Time: 27:45)

Matrix
Access to rows, columns or submatrices:

```
> x[3,]
[1] 6 7
x[3,-]
[1] 6 7

> x[,2]
[1] 3 5 7 9
```

```
R Console
> x = matrix(nrow=4, ncol=2, data=2:9, byrow=T)
> x
      [,1] [,2]
[1,]  2    3
[2,]  4    5
[3,]  6    7
[4,]  8    9
> x[3,]
[1] 6 7
> x[,2]
[1] 3 5 7 9
> |
```

Observe the notations.

$$x[,2] = \begin{pmatrix} 3 \\ 5 \\ 7 \\ 9 \end{pmatrix} \quad x[i,j]$$

Now, I come to another aspect that once you have defined the matrix, then suppose you want to call a particular row or a particular column. Up to now, we have done only one thing that I have shown you that how are you going to call a particular element which was say x inside the square bracket i,j. And here I would try to show you something where you have to be extremely careful, try to observe on the slide very carefully.

Suppose, I have defined here the matrix, the same matrix that I defined earlier that 2, 3, 4, 5, 6, 7, 8, 9. And suppose I want to call here third row. Third row is given by here this thing. So, my command here will be I want to call here say 3, this is the address of third row, and then here this will be a blank sign. And you do not have to do anything and then if you write here x. So, this will give you here this value 6, 7. So, you can see here if you do it here you get here 6, 7.

And similarly if I want to call the second column, then second column is given over here, and you can see here the address here is something like square bracket comma and 2, this is the second column. And if you give here x this is the second column here. So, if you try to call this value, it will give you here 3, 5, 7, 9, this column.

So, this will give you here 3, 5, 7, 9, right which is given here. But now you have to be very careful. Means, if I hide the two commands, suppose if I hide this thing and if I hide this thing. And if you will and if you try to look only on this part can you really say whether which of them is row and which of them is column? The answer is no.

So, in case if you want to know by looking at the outcome whether this is an outcome of a row or a column, or whether you have called a row or whether you have called a column unless and until you try to look into this address that what is the location of the blank space, you cannot know.

So, this is what you have to keep in mind. Yes, means, I am telling you means and I am trying to highlight this point because in some software it is possible to know by looking at the outcome that whether the outcome is a row or a column. So, that you have to be extremely careful.

(Refer Slide Time: 30:53)

Matrix
Access to rows, columns or submatrices:

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

x[c(1,3,5), c(7,9)]

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

x[i,j]
rows columns

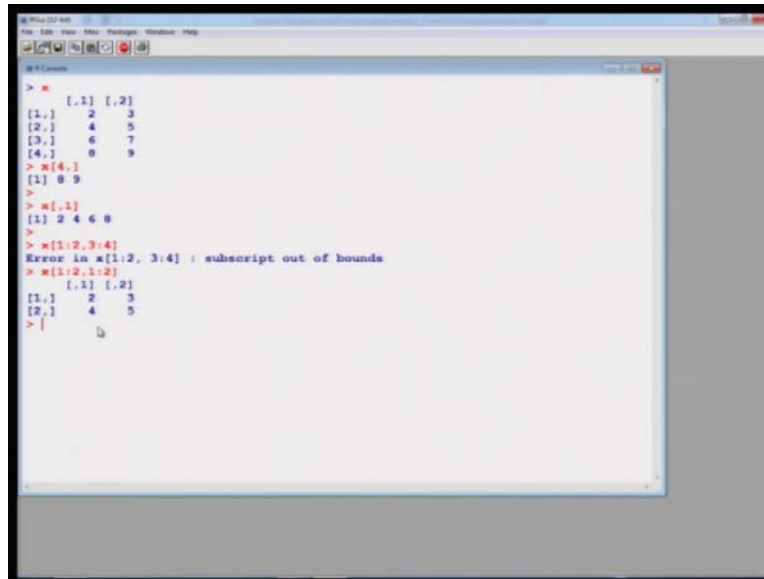
21

And similarly if you want to extract a sub matrix out of a matrix, then also that can be done. You have to simply means you have learnt that the format here is X i, j. So, now, you have to give here the means in the place of i, you have to give the addresses of rows and you have to give the addresses of columns in place of j, and then you can subtract those matrices also.

For example, I want to extract here a matrix say in which I want to extract the first 3 rows and 2 columns. So, you can see here this is the sub matrix of the x matrix which I want to extract. So, I will give here the address 1 to 3 that means, rows 1, 2 and 3; and columns 1 and 2. And now I will get this thing over here.

Now, this is also possible that this rows and columns can be given as a c command also. For example, if there is a matrix where you want first head and 5, 5th row and say if you want say here 7th and 9th column can also this can be done. But my idea here is to just to give you an impression that well these things are possible and you can do it. So, now let me try to take here this simple example, and I try to show you how you can find it out on the R console.

(Refer Slide Time: 32:26)



```
> x
     [,1] [,2]
[1,]  2   3
[2,]  4   5
[3,]  6   7
[4,]  8   9
> x[4,]
[1] 8 9
> x[,1]
[1] 2 4 6 8
> x[1:2,3:4]
Error in x[1:2, 3:4] : subscript out of bounds
> x[1:2,1:2]
     [,1] [,2]
[1,]  2   3
[2,]  4   5
> |
```

So, you can see here this is your here x. Now, if I want to find out here x say 4th row. So, I have to give it here like this. So, you can see here this is my here 4th row which is coming here. And if I want to find out the say first column only, so, I have to give here the address of first column means I will say simply just look at this address, how it has been given this is bracket comma 1 bracket.

So, I simply copied here this thing, and I put it here x, the name of the matrix from where you want to extract. And you will see here that this is 2, 4, 6, 8, like a 2, 4, 6, 8. Just try to see the position of my cursor, right. So, this is how you can obtain this thing. And if I want to see here see extract the sub matrix, I can say here x 1 to 2, and say here and columns here 3 to 4, right.

You can see here, there is an error. Why? Because, you have given the numbers which are out of the bound. So, try to give the numbers based on, then the available number of rows. For example, I can say here that I want to have a sub matrix which is containing of only first two rows and first two columns. So, you can see here this is 2, 3, 4, 5, right.

Let me stop in this lecture. And I have given you a brief idea about the matrix operation. There are many many matrix operations and those things are possible. For example, if you want to find out the inverse of a matrix, then the command is solve. There are different types of products Kronecker product, Hadamard product, inner product and so on. There are different types of operation finding on the eigenvalues, eigen vectors, etc. All those things are possible in R with the matrix theory.

But definitely I am not going to discuss here means, I do not expect at this moment that I will be using them in the further lectures, but you must know it. Yes, if you want to learn the decision sciences, without matrix theory without mathematics, you cannot do anything, yes and if you try to do it as I said you will simply become a compounder not become a doctor.

So, I would request you, you try to take up any book on the matrix theory, try to see that whatever commands you have studied in your undergraduate or class 12, try to see can you really do them on the R console also, what are the commands. And if you do it, that will give you more confidence.

So, try to take up some problems from the books, from the assignment. Try to solve them, attempt them. And in case if you are stuck, try it hard, try once again. And I am sure that you will successfully solve them. So, you attempt and I will see you in the next lecture. Till then, good bye.