

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

Basics of Calculations
Lecture - 15
Matrix Operation - Mathematical and Other Operations

Hello friends, welcome to the course Foundations of R Software. Now, you can recall that in the last couple of lectures, we started discussion on the matrix operations and we considered different types of matrix operations. And, now in this lecture, we will continue with some more Matrix Operations. So, now in this lecture just for the sake of quick review I will give you an example of the addition and subtraction of the two matrices.

And after that I will discuss say multiplication of matrices, finding inverse of matrix, and then some more operations like as finding out the characteristic roots or the eigenvalues. One thing I would like to make it here clear that the first thing is that whenever you are trying to do any matrix operation, you have to follow the rules of that particular matrix operation; for example, here we are going to consider the matrix multiplication today. So, you have to follow the rules of the matrix multiplication.

Similarly, when we are trying to finding the inverse of a matrix, then you need to follow the rules of finding of the unique inverse of a matrix. For example, the matrix has to be positive definite, etc. I am not going to discuss about all those basic fundamentals about the matrix theory, but I believe that you know them. And, also I have taken the examples here which consists of only two matrices, but these operation can be extended to any number of matrices provided you fulfil the rules of the matrix operation.

So, now we begin our course and we try to consider first the addition and subtraction of the of two matrices just for the sake of quick review, ok.

(Refer Slide Time: 02:10)

Addition and subtraction of matrices

Example:

```
> x = matrix(nrow=4, ncol=2, data=1:8, byrow=T)
> x
```

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

X: 4x2

```
> 4*x
```

	[,1]	[,2]
[1,]	4	8
[2,]	12	16
[3,]	20	24
[4,]	28	32

4x2 → 4x2

So, now, let me try to take here one more example that is the same example that I took a couple of times earlier, that I am trying to create here a matrix of order 4 by 2 in which the data values are 1 to 8 and the data is arranged by row. So, you can see here this is your here matrix in which the data is arranged like 1, 2, 3, 4, 5, 6, 7, 8 like this. Now, in case if you try to multiply this matrix by a scalar, then you know that each of the element of this matrix gets multiplied.

For example, you can see here if I try to multiply it by here 4, then every element is going to be multiplied by here 4. And, then I have here one more matrix here like this 4 into x. Now, I try to add and subtract x and 4x. So, you can see here this x matrix has got a order 4 by 2 and then 4 into x matrix this also has got the same order 4 by 2.

(Refer Slide Time: 03:14)

Addition and subtraction of matrices

Example:

```
> x + 4*x
```

	[,1]	[,2]
[1,]	5	10
[2,]	15	20
[3,]	25	30
[4,]	35	40

```
> 4*x - x
```

	[,1]	[,2]
[1,]	3	6
[2,]	9	12
[3,]	15	18
[4,]	21	24

So, there is no problem in doing the addition and subtraction operations. So, you can see here, if you try to add here x and $4x$ then you get here this outcome. And, if you try to see this is the same rule has been applied that the corresponding elements at a respective places, they are added together for example, the element at the first row and first column in the two matrices x and $4x$ respectively, they are added.

So, this becomes here 1 plus 4; and it is here 5. And, similarly here the element in the second row and second column which is here 4 in x and 16 in $4x$, they are added together and you get here an outcome 4 plus 16 which is here 20 and so on. So, you can see here this is how the addition of two matrices is obtained. Now, similarly if you try to subtract these two matrices and I try to subtract here four into x minus x .

So, you can see here, now that the same rule is followed here also and in; for example, if you try to see here in the $4x$ matrix at the second row and first column the value here is 12. And the element in the matrix x at the same address is here 3. And, if you try to subtract here 12 minus 3, you get here 9, right. And, similarly in case if you try to see here that the element in the fourth row and second column here is 8; and the element in the fourth row and second column of $4x$ is 32.

So, if you try to subtract 32 minus 8 you get here 24, right. So, this is how this operation goes on. So, let me circle here. So, that you can identify that which operation is corresponding to actually what; and the same thing is if you try to see here is obtained here also. So, now you can see here that it is not a very difficult thing to get such operations done in the R software.

(Refer Slide Time: 05:14)

Multiplication of matrices

Multiplication of matrices can be executed with the operator `%*%`

`x = matrix(nrow=4, ncol=2, data=1:8, byrow=T)`

`y = matrix(nrow=2, ncol=4, data=11:18, byrow=T)`

> x	> y
<pre> [,1] [,2] [1,] 1 2 [2,] 3 4 [3,] 5 6 [4,] 7 8 </pre>	<pre> [,1] [,2] [,3] [,4] [1,] 11 12 13 14 [2,] 15 16 17 18 </pre>

$1 \times 11 + 2 \times 15$

$x \%*\% y$

Now, after this I try to consider here the multiplication of matrices. So, you know that in case of multiplication of matrices there is a condition that if I try to take here suppose two matrices x and y . And, suppose the order of the matrix x is m cross n . And, then in case if I want to multiply another matrix y in x then the order of y should be like this n cross something either n cross n or say n cross p such that these two values; that means, the number of columns in the first matrix and the number of rows in the second matrix they have to be the same, right.

So, following this rule of matrix operation, we will try to conduct here all the calculations. So, the first thing is that if you want to multiply the matrices the operator in R software to multiply the matrices is like this percentage star percentage, right. And, if you try to recall if you try to use here only the star sign, only the elements in the corresponding places are multiplied.

But, you know that when you are trying to do the matrix multiplication, you try to follow a different rule, the respective elements at those positions are not simply multiplied. So, following this rule, let me try to show you show you an example here. So, I try to take here a matrix x of order 4 by 2. So, this is of order 4 by 2. And, then I try to take here a matrix y which is of order 2 by 4.

So, if you try to see here this matrix here y is like here 2 by 4. So, if I try to multiply x into y that should not be problem. And the data values in the matrix x are the number from 1 to 8. And the data values in the matrix y are the numbers from 11 to 18, and they are arranged by row. So, if you try to see this is here the matrix x and this is here the matrix y .

So, now, in case if you try to multiply it here you know that if I try to multiply x into this y with the matrix operation, what happened? That this row gets multiplied with this column. So, that will be something like 1 into 11 plus 2 into 15, right. So, that is the way the matrix operations are conducted.

(Refer Slide Time: 07:45)

Multiplication of matrices

```

> x%*%y
      [,1] [,2] [,3] [,4]
[1,]   41  44  47  50
[2,]   93 100 107 114
[3,]  145 156 167 178
[4,]  197 212 227 242
  
```

x 4×2 y 2×4 \rightarrow 4×4
 $y \% \% x$
 2×4 4×2 \rightarrow 2×2

```

> y%*%x
      [,1] [,2]
[1,]  210  260
[2,]  274  340
  
```

```

> x = matrix(c(41, 44, 47, 50, 93, 100, 107, 114, 145, 156, 167, 178, 197, 212, 227, 242), nrow=4, byrow=T)
> y = matrix(c(1, 2, 3, 4, 5, 6, 7, 8), nrow=2, byrow=T)
> x
      [,1] [,2] [,3] [,4]
[1,]   41  44  47  50
[2,]   93 100 107 114
[3,]  145 156 167 178
[4,]  197 212 227 242
> y
      [,1] [,2]
[1,]    1    2
[2,]    3    4
> x%*%y
      [,1] [,2] [,3] [,4]
[1,]  41  44  47  50
[2,]  93 100 107 114
> y%*%x
      [,1] [,2]
[1,]  210  260
[2,]  274  340
  
```

And, so if you try to simply multiply here two matrices x and y here, you get here this outcome. Yeah, in case if you try to do the manual calculations also, you can verify it. And, similarly if you try to do here this y into x , then what is the order of the y here? 2 cross 4; and the order of the x is 4 cross 2. So, the resultant is going to be here a matrix of order 2 by 2. And, the same thing will happen here in x and y also. The order of x is 4 by 2 and the order of matrix y is 2 by 4. So, the resultant will be of a matrix of order 4 by 4.

So, that is why you can see here there are 4 rows and 4 columns in the first operation that is x into y . And, in case of y into x you are getting here a matrix of order 2 by 2 following this rule. So, you can see here it is not a very difficult thing if you try to operate the two matrices with respect to multiplication. You simply have to follow the correct operator, right.

(Refer Slide Time: 08:58)

Multiplication of matrices

Another example: Consider the multiplication of X' and X

```

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

x
 transpose x'

```

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

Transpose of x

Now I want to give you here one more example and with which I would try to show you that there is a particular way of multiplying a specific type of matrix in the R software. For example, if I try to take here a matrix here x and I try to find out its transpose. Transpose will be indicated by here x transpose like this. So, in case if you want to multiply these two matrices, first option is that you would simply try to follow the simple rules of matrix multiplication, but we also have a operator or a function which is called as `crossprod` or `doubleprod`.

So, we try to understand that, how it works and then I will try to explain you why it is important and why do we need it. So, I try to consider here a matrix of order 4 by 2 in which the data values are from 1 to 8, and they are arranged by rows. The same matrix that we have considered earlier; and you know that in case if you want to find out the transpose of this matrix, then we have a command here `t` and inside the parenthesis you have to write down the name of the matrix. So, this will be, this is here the transpose, transpose of matrix x.

(Refer Slide Time: 10:17)

Multiplication of matrices

```

> t(x) %*% x
      [,1] [,2]
[1,]  84  100
[2,]  100  120
  
```

Handwritten notes: $x: 2 \times 4$, $x': 4 \times 2 \rightarrow 2 \times 2$, $x x': 4 \times 2 \cdot 2 \times 4 \rightarrow 4 \times 4$

```

> x %*% t(x)
      [,1] [,2] [,3] [,4]
[1,]  5  11  17  23
[2,]  11  25  39  53
[3,]  17  39  61  83
[4,]  23  53  83  113
  
```

```

> x = matrix(nrow=4, ncol=2, data=1:8, byrow=T)
> x
      [,1] [,2]
[1,]  1  2
[2,]  3  4
[3,]  5  6
[4,]  7  8
> t(x)
      [,1] [,2] [,3] [,4]
[1,]  1  3  5  7
[2,]  2  4  6  8
> t(x) %*% x
      [,1] [,2]
[1,]  84  100
[2,]  100  120
> x %*% t(x)
      [,1] [,2] [,3] [,4]
[1,]  5  11  17  23
[2,]  11  25  39  53
[3,]  17  39  61  83
[4,]  23  53  83  113
  
```

Now, in case if you want to multiply like transpose of x into x like this, right, then first option is that you simply try to write down here transpose of x matrix multiplication and here x and you will get here this outcome. And, similarly if you want to multiply the x transpose like this then you also you can write down here a matrix multiplication operator into transpose of x, and you will get here this operation.

So, you can see here this x is of order 4 by 2. So, when you try to take here x transpose x the order is going to be like this; 2 cross 4 into 4 cross 2. So, your outcome is going to be here of order 2 by 2. And if you try to make it here xx transpose then it is going to be 4 cross 2 and 2 cross 4, and the outcome is going to be of the order 4 by 4. So, that is what is happening here. And, you know this quantity x transpose x this is very popular in statistics, right.

(Refer Slide Time: 11:22)

```

Multiplication of matrices
Cross Product of a matrix X, X'X, with crossprod()  $x'x$ 

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

> t(x)
      [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8

```

So, that is why people have created a function here which is called as `crossprod` or `double prod`. And, in case if you try to use this function `crossprod` and inside the parenthesis if you write the matrix then you will get the outcome of x transpose x . So, whatever this operation is trying to do here this operation. So, you do not need to write this entire statement, but you can just write here a one simple command.

(Refer Slide Time: 11:59)

```

Multiplication of matrices

> crossprod(x)
      [,1] [,2]
[1,]   84  100
[2,]  100  120

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

> crossprod(x)
      [,1] [,2]
[1,]   84  100
[2,]  100  120

> t(x) %*% x
      [,1] [,2]
[1,]   84  100
[2,]  100  120

```

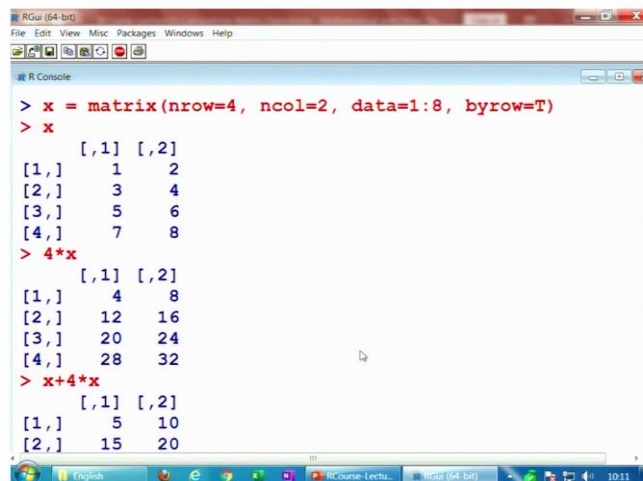
Note: Command `crossprod()` executes the multiplication faster than the conventional method with `t(x) %*% x`

And in case if you try to do it here, you can see here that if I try to write down here `crossprod(x)` then you can see here, this value comes out to be here like this. And, if you want to verify here you can verify in this screen also that I have this matrix here `x`. And, then I try to operate this here command `crossprod`. And then I try to use here the basic command say that `x %*% x` and you can see here that they have got the same outcome.

When you are trying to deal with bigger matrix then it is said that this function `crossprod`, this execute the multiplication faster than the conventional method which is here like this. And the reason is this for example, you know that when you are trying to do such operation like as matrix multiplication and inverse etc., then there are some algorithms which try to conduct such operation.

So, whenever you are trying to use an algorithm then there is always a concept of which algorithm can produce the result faster. So, that is believed here that if you try to use the function `crossprod`, then it will execute the multiplication faster than the conventional method; and that is why it is here. So, now, I try to show you first here these operations on the R console. So, that you get here more confident and then I will try to show you some more operation.

(Refer Slide Time: 13:36)



```
RGui (64-bit)
File Edit View Misc Packages Windows Help
R Console
> x = matrix(nrow=4, ncol=2, data=1:8, byrow=T)
> x
  [,1] [,2]
[1,]  1   2
[2,]  3   4
[3,]  5   6
[4,]  7   8
> 4*x
  [,1] [,2]
[1,]  4   8
[2,] 12  16
[3,] 20  24
[4,] 28  32
> x+4*x
  [,1] [,2]
[1,]  5  10
[2,] 15  20
```

So, let me try to first create here this matrix `x`. So, you can see here this is the `x` matrix here like this. Now, in case if you try to multiply this matrix by here `x`; so, we have this

matrix. And, now in case if you try to write down here x plus 4 into x you can see here, the outcome here is like this.

(Refer Slide Time: 13:57)

```
RGui (64-bit)
File Edit View Misc Packages Windows Help
[1,] 4 8
[2,] 12 16
[3,] 20 24
[4,] 28 32
> x+4*x
      [,1] [,2]
[1,] 5 10
[2,] 15 20
[3,] 25 30
[4,] 35 40
> 4*x-x
      [,1] [,2]
[1,] 3 6
[2,] 9 12
[3,] 15 18
[4,] 21 24
> |
```

And, similarly in case if you try to do here this operation here 4 into x minus x . So, you can see here this is here the operation. So, you can see here it is not a very difficult thing to do. So, now let me try to consider here these two matrices x and y , and then I try to show you here that how you can do the matrix multiplications.

(Refer Slide Time: 14:25)

```
RGui (64-bit)
R Console
> x = matrix(nrow=4, ncol=2, data=1:8, byrow=T)
>
> y = matrix(nrow=2, ncol=4, data=11:18, byrow=T)
> x
      [,1] [,2]
[1,] 1 2
[2,] 3 4
[3,] 5 6
[4,] 7 8
> y
      [,1] [,2] [,3] [,4]
[1,] 11 12 13 14
[2,] 15 16 17 18
> x*y
      [,1] [,2] [,3] [,4]
[1,] 41 44 47 50
[2,] 93 100 107 114
```

So, let me clear this screen and I try to create here these two matrices here x and y. So, you can see here x is like this and y here is like this. And, if I try to write down here x then percentage star percentage y. And you can see here the outcome is obtained here like this.

(Refer Slide Time: 14:42)

```

RGui (64-bit)
File Edit View Misc Packages Windows Help
[3,] 5 6
[4,] 7 8
> y
      [,1] [,2] [,3] [,4]
[1,] 11 12 13 14
[2,] 15 16 17 18
> x%*%y
      [,1] [,2] [,3] [,4]
[1,] 41 44 47 50
[2,] 93 100 107 114
[3,] 145 156 167 178
[4,] 197 212 227 242
> y%*%x
      [,1] [,2]
[1,] 210 260
[2,] 274 340
> |

```

And in case if I try to use the multiplication y into x, that y percentage star percentage x and then it gives you this operation, right. So, you can see here it is not a very difficult thing.

(Refer Slide Time: 15:02)

```

RGui (64-bit)
File Edit View Misc Packages Windows Help
> x
      [,1] [,2]
[1,] 1 2
[2,] 3 4
[3,] 5 6
[4,] 7 8
> t(x)
      [,1] [,2] [,3] [,4]
[1,] 1 3 5 7
[2,] 2 4 6 8
> t(x)%*%x
      [,1] [,2]
[1,] 84 100
[2,] 100 120
> crossprod(x)
      [,1] [,2]
[1,] 84 100
[2,] 100 120
> |

```

And, similarly if you try to take here x to be here like this and trace of x to be here the transpose of this matrix x comes out to be here like this. And if you try to multiply transpose of x that is like this t x and the percentage star percentage x this comes out to be here like this. And in case if you try to use here the command crossprod, and inside the parentheses you name the matrix and then you can see here, both these outcomes are coming out to be the same.

Well, here I am trying to take a very small matrix that is why you cannot observe the speed of the operation, but I am sure that when you are trying to deal with bigger matrices you will see the difference, right.

(Refer Slide Time: 15:47)

Concatenating matrices

Concatenating matrices row wise: `rbind(x, y)`
 Concatenating matrices column wise: `cbind(x, y)`

```
x = matrix(nrow=3, ncol=2, data=1:6, byrow=T)
y = matrix(nrow=3, ncol=2, data=11:16, byrow=T)
```

> x	> y	> rbind(x, y)
[,1] [,2]	[,1] [,2]	[,1] [,2]
[1,] 1 2	[1,] 11 12	[1,] 1 2
[2,] 3 4	[2,] 13 14	[2,] 3 4
[3,] 5 6	[3,] 15 16	[3,] 5 6
		[4,] 11 12
		[5,] 13 14
		[6,] 15 16

10

So, after this I come to another operation which are related to the matrix. Suppose I want to concatenate the matrices. What is the meaning of concatenate? That means, I want to join, right, I want to bind. So, now, I have here two options say, for example, I can join the matrices row wise or I can join them column wise. So, the R command two combining or concatenating the two matrices row y is rbind and inside the parenthesis you have to write the matrices in the same order in which you want to write.

So, this is r b i n d, all in lower case and similarly if you want to concatenate the matrices column wise then the command cbind c b i n d, all in lower case. And then inside the parentheses you try to write down the matrices in the same order in which you want to

join them. So, we try to understand this operation through this example. So, let me try to create here two matrices x and y.

So, x and y both are of order 3 by 2; and in the matrix x the data values are from 1 to 6. And in the matrix y, the data values are from 11 to 16 which I have taken intentionally so that you can very easily identify that the elements are corresponding to which of the matrices. So, and the elements are arranged row wise. So, if you try to see here this is your here x and this is your here y.

Now, in case if you try to see here, these elements are 1, 2, 3, 4, 5, 6; and these elements are 11, 12, 13, 14, 15, 16 in x and y respectively. So, now, and I try to use here the operation rbind x comma y inside the parenthesis, then you can see here these are the elements of matrix x and these are the elements of matrix y, right. So, if you try to see, what is the outcome of this rbind? It is row wise.

So, all the rows are combined like this. So, that is what you have to see that what exactly R is going to do right. So, these matrices are joint vertically. So, x and then here y and if you have some more matrices possibly, it will come down the y matrices.

(Refer Slide Time: 18:11)

Concatenating matrices
Concatenating matrices row wise: `rbind(x, y)`
Concatenating matrices column wise: `cbind(x, y)`

```
x = matrix(nrow=3, ncol=2, data=1:6, byrow=T)
y = matrix(nrow=3, ncol=2, data=11:16, byrow=T)
```

> x	[,1]	[,2]	> y	[,1]	[,2]
[1,]	1	2	[1,]	11	12
[2,]	3	4	[2,]	13	14
[3,]	5	6	[3,]	15	16


```
> cbind(x, y)
```

	[,1]	[,2]	[,3]	[,4]
[1,]	1	2	11	12
[2,]	3	4	13	14
[3,]	5	6	15	16

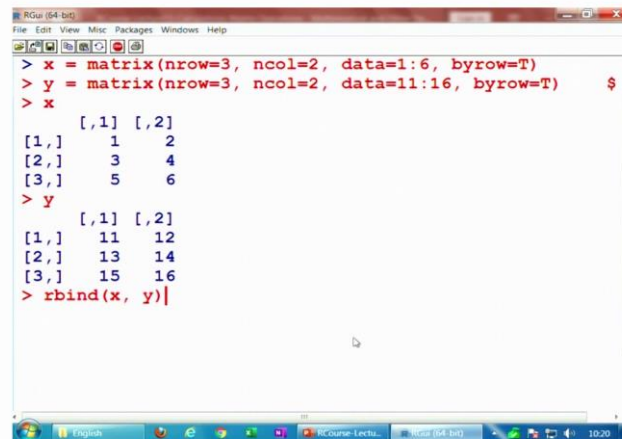
11

And similarly if you try to consider here cbind in the same matrices; so, this is your here matrix x and this is your here matrix y. And now you want to join them column wise. So, you can operate here the command cbind and inside the parenthesis this is x comma y.

So, you can see here this is here x and this is here y. So, in this case you can see that the matrices are joined horizontally which is column wise.

So, once again I would request you that you please try to observe that how R is working when it is trying to join the matrices row wise and column wise. So, let me try to show you first these two operations on the R software. So, that you get more confident and then I will try to show you more operations. So, let me clear the screen.

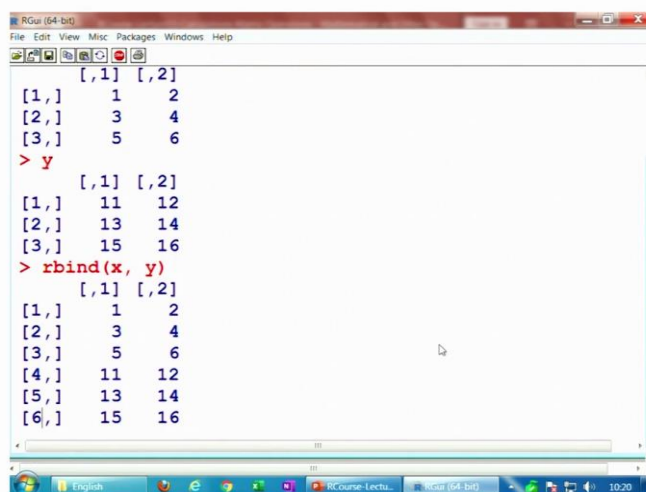
(Refer Slide Time: 19:04)



```
RGui [64-bit]
File Edit View Misc Packages Windows Help
> x = matrix(nrow=3, ncol=2, data=1:6, byrow=T)
> y = matrix(nrow=3, ncol=2, data=11:16, byrow=T)
> x
      [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
> y
      [,1] [,2]
[1,]   11   12
[2,]   13   14
[3,]   15   16
> rbind(x, y)
```

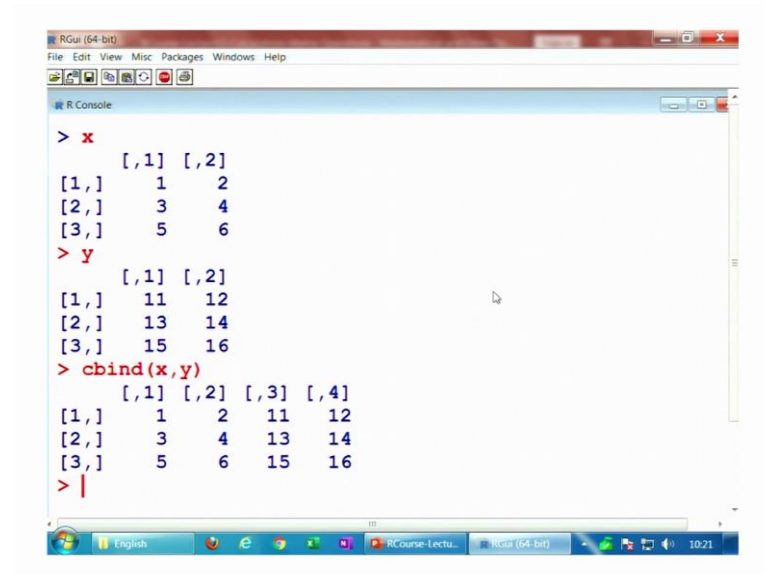
And if you try to see, these are the two matrices x is here like this y here is like this. And if you try to take here rbind and inside the parentheses, if you try to say x y it is here like this you can see.

(Refer Slide Time: 19:14)



```
RGui [64-bit]
File Edit View Misc Packages Windows Help
      [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
> y
      [,1] [,2]
[1,]   11   12
[2,]   13   14
[3,]   15   16
> rbind(x, y)
      [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
[4,]   11   12
[5,]   13   14
[6,]   15   16
```

(Refer Slide Time: 19:20)



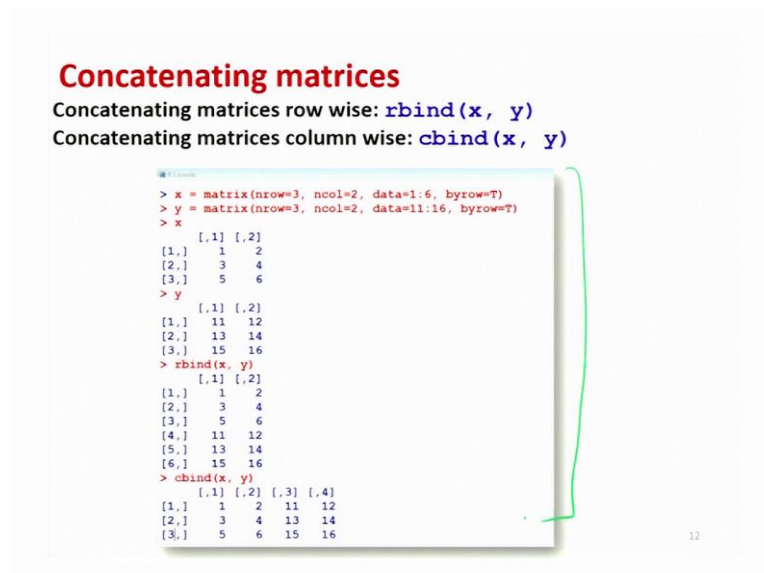
```
> x
  [,1] [,2]
[1,]  1  2
[2,]  3  4
[3,]  5  6
> y
  [,1] [,2]
[1,] 11 12
[2,] 13 14
[3,] 15 16
> cbind(x,y)
  [,1] [,2] [,3] [,4]
[1,]  1  2 11 12
[2,]  3  4 13 14
[3,]  5  6 15 16
> |
```

And similarly in case if you try to take here cbind. So, I am just writing here both the matrices so that you can observe them very clearly it is here like this. So, once again you can see that the matrix operations in these two cases are not difficult to understand and they are very simple to operate also.

(Refer Slide Time: 19:35)

Concatenating matrices

Concatenating matrices row wise: `rbind(x, y)`
Concatenating matrices column wise: `cbind(x, y)`



```
> x = matrix(nrow=3, ncol=2, data=1:6, byrow=T)
> y = matrix(nrow=3, ncol=2, data=11:16, byrow=T)
> x
  [,1] [,2]
[1,]  1  2
[2,]  3  4
[3,]  5  6
> y
  [,1] [,2]
[1,] 11 12
[2,] 13 14
[3,] 15 16
> rbind(x, y)
  [,1] [,2]
[1,]  1  2
[2,]  3  4
[3,]  5  6
[4,] 11 12
[5,] 13 14
[6,] 15 16
> cbind(x, y)
  [,1] [,2] [,3] [,4]
[1,]  1  2 11 12
[2,]  3  4 13 14
[3,]  5  6 15 16
```

Now, you can see here this is the screenshot of the same operation which I just shown you on the R console. So, you can be confident that these things are working, ok.

(Refer Slide Time: 19:45)

Inverse of matrix $A A^{-1}$ Unique inverse
Generalized inverse

`solve()` finds the inverse of a positive definite matrix

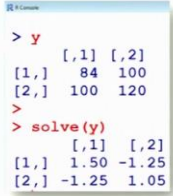
```
> y = matrix( nrow = 2, ncol = 2, byrow = T,
data = c(84,100,100,120))
```

```
> y
      [,1] [,2]
[1,]   84  100
[2,]  100  120
```

`> solve(y)`

```
      [,1] [,2]
[1,]  1.50 -1.25
[2,] -1.25  1.05
```

$y \%*\% \text{solve}(y) = \text{Identity matrix}$



13

Now, I consider the next operation which is finding out the inverse of the matrix. So, you know that whenever you are trying to find out the inverse of the matrix, say there is some matrix here say here A , then you try to denote it by here A inverse. Actually we are trying to find out here the unique inverse. I am writing here this word unique inverse because there is one more concept of inverse of the matrix which is called as generalized inverse, but we are not going to consider this thing here, ok

So, you know that whenever you are trying to find out the inverse of the matrix there are certain conditions which have to be satisfied; for example, the matrix has to be positive definite matrix. So, the R command to find out the inverse of a matrix is `solve` and inside the parenthesis you have to give the matrix. So, now let me give you a very simple example.

For example, if you create here a matrix y which is of order 2 by 2, and the values are 84, 100, 100, 120 which are arranged row wise. So, now, this is your here matrix y and if you want to find out the inverse of this matrix y , you have to simply write down here `solve y` and it will give you this outcome. And if you want to verify if you try to multiply y and this here `solve of y` the outcome of this, then it will come out to be an identity matrix; this is the rule of any matrix.

The only thing what you have to be careful that these things are based on the algorithms. So, in case if you try to increase the order of the matrices that will create more

complexity and then, but this condition will always be holding true. And that is also true that if you try to use different software; that means, that different software are using different types of algorithms to find out the inverse of the matrix.

So, it is possible that if you try to find out the inverse of a matrix of some higher order in different software, then their values may vary very little, means there will be some difference, but the difference will not be so high, but whatsoever be the case this condition will always be holding true. So, in case, this is happening you should not doubt on the integrity of this R software.

(Refer Slide Time: 22:23)

Eigen values and eigen vectors of matrix

Characteristic roots and
Characteristic vectors

`eigen()` finds the eigen values and eigen vectors of a positive definite matrix

```
y = matrix( nrow = 2, ncol = 2, byrow = T, data = c(84,100,100,120) )
```

```
> y
```

	[,1]	[,2]
[1,]	84	100
[2,]	100	120

14

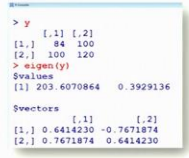
And similarly there is one more operation in the matrix theory which is finding out the eigenvalues and eigenvectors of a matrix or this is also called as characteristic roots and characteristic vectors of a matrix. So, these are very special thing, but this finding out the eigenvalues or the characteristic values that that is taught in the undergraduate classes. So, I thought that let me add it here.

But after that I will surely stop with the matrix operation because there is a long list of such matrix operation that you can do and they can be done in the R software also. So, I try to take here the same matrix here y that I had just used in the case of finding out the inverse of the matrix. So, this is here the matrix here y and then I try to find out the eigenvalues and eigenvectors.

(Refer Slide Time: 23:15)

```
Eigen values and eigen vectors of matrix
> eigen(y)
$values
[1] 203.6070864 0.3929136

$vectors
      [,1] [,2]
[1,] 0.6414230 -0.7671874
[2,] 0.7671874 0.6414230
```



```
> y
      [,1] [,2]
[1,] 84 100
[2,] 100 120
> eigen(y)
$values
[1] 203.6070864 0.3929136
$vectors
      [,1] [,2]
[1,] 0.6414230 -0.7671874
[2,] 0.7671874 0.6414230
```

So, the R command to find out the eigenvalues and eigenvector here is `eigen` all in lower case alphabets, and inside the parenthesis you have to write down the matrix. And, if you try to execute it like as here these two values are the eigenvalues; and these two values here like this one and this one these are the coefficient of the characteristic vector or the eigenvectors corresponding to these eigenvalues.

Well, those who are familiar with the eigenvalues and eigenvectors, they will understand it very easily. And yeah, when you are trying to find out for some higher order values right similar type of differences may come if you try to find it find them from different software, right. So, let me try to show you these things in the R software. So, first let me try to show you the inverse operation. So, let me try to create this matrix and then I try to find out the eigen values of this matrix and the inverse of this matrix.

(Refer Slide Time: 24:28)

```
RGui (64-bit)
File Edit View Misc Packages Windows Help
# R Console
> y = matrix( nrow = 2, ncol = 2, byrow = T, data = c$
> y
      [,1] [,2]
[1,] 84 100
[2,] 100 120
> solve(y)
      [,1] [,2]
[1,] 1.50 -1.25
[2,] -1.25 1.05
> z=solve(y)
> y%*%z
      [,1] [,2]
[1,] 1 0
[2,] 0 1
> z%*%y
      [,1] [,2]
[1,] 1.000000e+00 2.842171e-14
```

So, this is your here matrix y. So, you can see here this is your here matrix y. Now, if you want to find out the inverse of this matrix y this is here like this right. And, in case if you try to save this value in the in some variables say here z. So, then if you try to say here y percentage star percentage z, you can see here this the multiplication of y and solve y is an identity matrix right. And the same thing will also happen if you try to say here z percentage star percentage y.

(Refer Slide Time: 25:07)

```

RGui (64-bit)
File Edit View Misc Packages Windows Help
[1,] 84 100
[2,] 100 120
> solve(y)
      [,1] [,2]
[1,] 1.50 -1.25
[2,] -1.25 1.05
> z=solve(y)
> y%*%z
      [,1] [,2]
[1,] 1 0
[2,] 0 1
> z%*%y
      [,1] [,2]
[1,] 1.000000e+00 2.842171e-14
[2,] -1.421085e-14 1.000000e+00
> |

```

This will also give you close to 0 and nearly 0. So, that is what I wanted to show you, because this inverses are found using some algorithms, ok.

(Refer Slide Time: 25:18)

```

R Console
> y
      [,1] [,2]
[1,] 84 100
[2,] 100 120
> eigen(y)
eigen() decomposition
$values
[1] 203.6070864 0.3929136

$vectors
      [,1] [,2]
[1,] 0.6414230 -0.7671874
[2,] 0.7671874 0.6414230
> |

```

So, now, let me try to show you that for this matrix y , I try to find out here the eigenvalues and eigenvectors. So, the command here is `eigen` inside the parenthesis y and if you try to do here see here this will like this these are the values and these are the eigenvectors.

So, now let me come to an end to this lecture. And, with this lecture I will stop with the matrix operations also.

As I said there is a long list of matrix operations, and if I try to do all of them, possibly the entire course will be spent on only matrix operations. But my objective in this matrix operation part was that I wanted to show you that R can handle the matrix operations also and whatever matrix operations you learnt in your classes, all of them can be done in the R software also.

The only thing is this you have to find what is the correct function or correct command for finding out or doing that operation. And once again, I will just advise you that whenever you are trying to do any matrix operation, try to first see that, what are the conditions which are to be satisfied and those conditions will come only from the theory from those topics which you have studied in your class.

And after this, different people, different student, different candidates, might be requiring different types of matrix operation. So, for that I will say you try to look into the help and try to find out the appropriate operation of your requirement. And, after this I will request you that you try to take some examples, try to create some matrices and try to play in the R software with different type of matrix operations whatever we have learnt so far and this will make you more confident.

And in case if you want to go for the statistics or data science etc., believe me without matrix theory matrix operation you cannot survive even. So, with this objective that you will now try to learn the remaining matrix operations yourself. I will see you in the next lecture with more operation till then, goodbye.