

So, this is another way of giving the data like as 1, 2, 3, 4 up to here 12 and you can see here there are 4 rows and 3 columns. So, we need here 4, 3 are 12 number of values which are given here from 1 to 12. So now, this and this gives us this following matrix. So, you can see here this is my 1st row, this is 2nd row, this is 3rd row and this is here 4th row and similarly here this here is 1st column, this here is 2nd column and this is here 3rd column, you will just try to follow my pen right where I am writing.

So, now you can see here that this rows and column they have got some name. For example, in case if you try to look at this row number 1, so it is indicated by square brackets like this one then 1 and comma and similarly in case if you try to see here this row number 2 this is indicated by square brackets 2 comma. And similarly this 3rd and 4th row are also indicated by square brackets 3 comma and 4 comma. And similarly if you try to look into the column, columns also have been given a name which is square bracket then comma and then 1 this is column number 1.

Similarly, column number 2 here is a square bracket then comma and then 2 and 3rd column is like here square brackets comma and here 3. So, these are the names which are automatically given by the R software to these rows and these columns. And suppose we want to rename these rows and columns that we want to give them another name of our choice. So, in order to do this, we have a command here `rownames` `rownames` and inside the parenthesis you write `x` and then `colnames` that is `colnames` this will rename the names of the columns.

(Refer Slide Time: 03:23)

```
Matrix Operations  
Renaming the row and column names  
> rownames(x) = c("r1", "r2", "r3", "r4")  
> x  
  [,1] [,2] [,3]  
r1    1    5    9  
r2    2    6   10  
r3    3    7   11  
r4    4    8   12  
  
> colnames(x) = c("c1", "c2", "c3")  
> x  
  c1 c2 c3  
r1  1  5  9  
r2  2  6 10  
r3  3  7 11  
r4  4  8 12
```

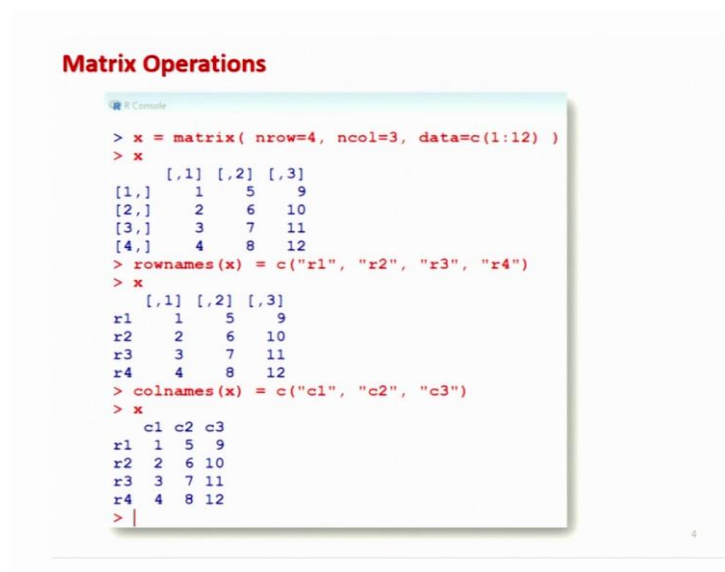
So, if you try to look here in these two operations, suppose I want to change the names of the rows and suppose I want to give them a name like say r 1, r 2, r 3, r 4 like this. So, for that I write here row names, all in lower caps and inside the parenthesis the name of the matrix and then I use the c command to create a data vector and then I try to give these names within this double quotes can see here.

So, these are the characters. So, this will work like that I want to change the names of the rows in the matrix x with these names and if you try to operate, it will change it. For example, now if you try to see at this outcome you can see here this is now changed, ok. And now similarly if you want to change the names of the columns here which are here given here like this.

So, I use here the command colnames and inside the parenthesis I write the name of the matrix x and then I try to give the names which we want to replace. So, suppose I decide that the names of the columns are going to be c1, c2, c3, so I then so I try to give them in the format of data vector using the command c within the parenthesis and you will and all the names inside the double quotes.

So, that they are the characters and if you try to execute it then this function colnames will change the names of the columns of this matrix x and you get here c1 c2 c3. So, this is how you can rename the names of the rows and columns in any given matrix.

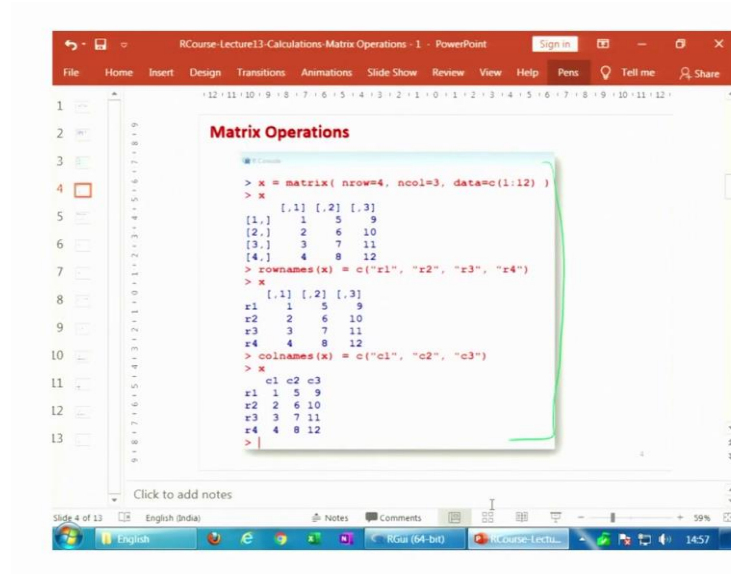
(Refer Slide Time: 04:57)



```
Matrix Operations
# Console
> x = matrix( nrow=4, ncol=3, data=c(1:12) )
> x
      [,1] [,2] [,3]
[1,]  1   5   9
[2,]  2   6  10
[3,]  3   7  11
[4,]  4   8  12
> rownames(x) = c("r1", "r2", "r3", "r4")
> x
      [,1] [,2] [,3]
r1     1   5   9
r2     2   6  10
r3     3   7  11
r4     4   8  12
> colnames(x) = c("c1", "c2", "c3")
> x
      c1 c2 c3
r1     1  5  9
r2     2  6 10
r3     3  7 11
r4     4  8 12
> |
```

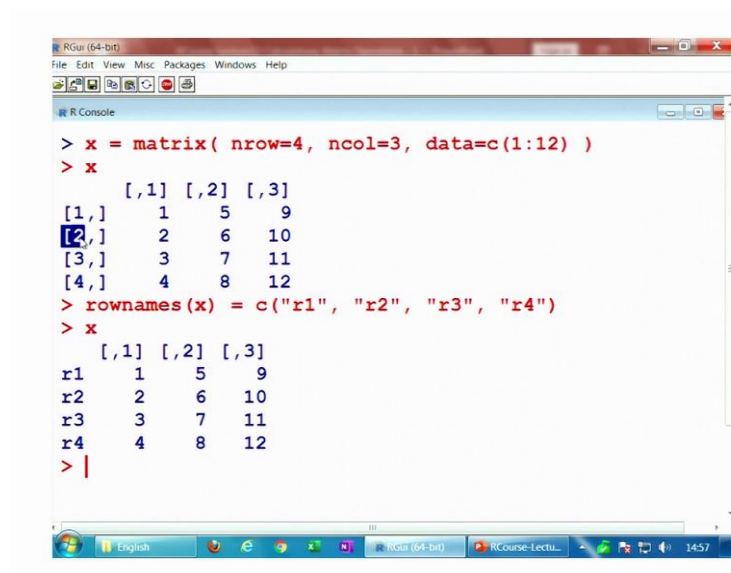
For example if you try to see this is here the screenshot of the same operation when you try to do it on the R console.

(Refer Slide Time: 05:04)



So, let us try to do this operation on the R console and try to see that what do you get here?

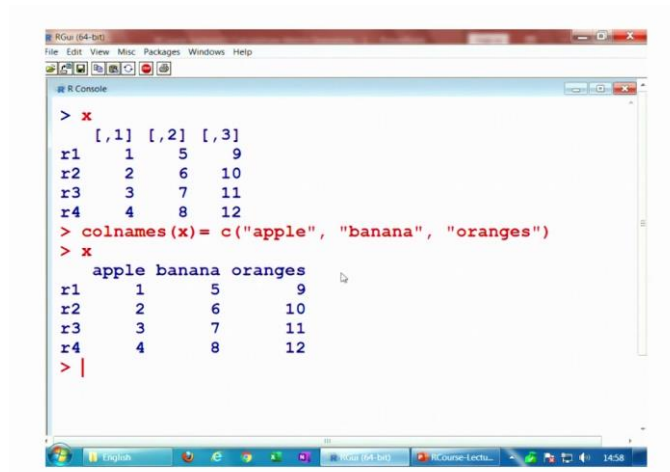
(Refer Slide Time: 05:12)



So, first we try to define here this matrix x. So, you can see here this is here like this and now then I try to operate here the command rownames on this matrix x. So now, if you

try to see here this x this will come out to be here like this, where the earlier names were like this 1 2 3 4, but now the new names are here r1, r2, r3 and r 4. And similarly in case if you try to change the name of the column.

(Refer Slide Time: 05:39)



```
RGui (64-bit)
File Edit View Misc Packages Windows Help
R Console
> x
  [,1] [,2] [,3]
r1   1   5   9
r2   2   6  10
r3   3   7  11
r4   4   8  12
> colnames(x) = c("apple", "banana", "oranges")
> x
  apple banana oranges
r1     1     5     9
r2     2     6    10
r3     3     7    11
r4     4     8    12
> |
```

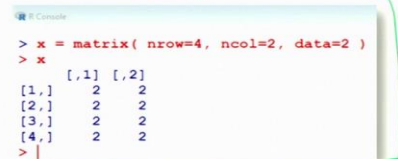
So, let me try to give some other name suppose if I give here the command here column names of this matrix x, suppose if I try to give it here say c say apple, banana and say oranges. So, even you can do these things you are simply giving the name. So, now if you try to see here, now the names are here apple, banana and oranges. So, this is how you can change the names of the rows and columns in any given matrix. So now, we come back to our slides and we try to learn some more operations.

(Refer Slide Time: 06:10)

Matrix Operations

Assigning a specified number to all matrix elements:

```
> x = matrix( nrow=4, ncol=2, data=2 )
> x
  [,1] [,2]
[1,]  2   2
[2,]  2   2
[3,]  2   2
[4,]  2   2
```



```
R Console
> x = matrix( nrow=4, ncol=2, data=2 )
> x
  [,1] [,2]
[1,]  2   2
[2,]  2   2
[3,]  2   2
[4,]  2   2
> |
```

The slide includes a diagram of a 4x2 matrix with all elements equal to 2. Green arrows point from the row and column indices to the corresponding elements in the matrix. A green box highlights the R console output below the diagram.

Now, suppose you need a matrix in which all the values should be the same, well these are the different type of operations which we need when we are trying to handle the real data set and it is useful. So, my objective here is that I want to create here a matrix in which all the values are the same.

So, suppose I try to create a matrix of order 4 by 2; that means, there are four rows and two columns and I want that in this matrix all the values should be 2. So, in order to do this thing I simply write down here data is equal to 2 and if I use all these parameters in the matrix command then and assign the value in a variable x, then x will come out to be like this. You can see here this is the 1st row 2nd row 3rd row 4th row, 1st column 2nd column and all the values are here only the 2.

And similarly if you want you can choose any other value and this is here the screenshot of the same operation when you try to do it in the R console now.

(Refer Slide Time: 07:13)

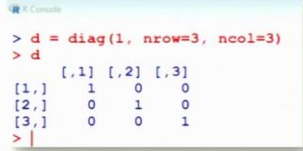
Matrix Operations: Diagonal matrix

Construction of a diagonal matrix, here the identity matrix of a dimension 3:

```
> d = diag(1, nrow=3, ncol=3)
> d
```

	[,1]	[,2]	[,3]
[1,]	1	0	0
[2,]	0	1	0
[3,]	0	0	1

Handwritten notes: The matrix is annotated with green circles around the diagonal elements (1, 1, 1) and the word "data" written next to the matrix. To the right, a diagram shows a square matrix with a diagonal line and the word "diagonal" written above and below it.



```
> d = diag(1, nrow=3, ncol=3)
> d
```

	[,1]	[,2]	[,3]
[1,]	1	0	0
[2,]	0	1	0
[3,]	0	0	1

Similarly, when you are trying to do matrix operation, many times we need a diagonal matrix, the diagonal matrix is a matrix in which the off diagonal elements are 0 and there are only some nonzero values in the diagonal you know that if you have a square matrix suppose then all these values on this diagonals, they are called the diagonal element and all these values on this so called triangles here and here they are called as off diagonal elements, these are off diagonal, right.

So, in case if you want to see here then we do here then you have to simply use here the command `diag` and you have to define here the values which you want to put on the diagonal. Suppose I want to create an identity matrix, identity matrix is a matrix in which the diagonal elements are 1 and the off diagonal elements are 0. Suppose I want to create a 3 by 3 identity matrix and in that case I will write down here `nrow` equal to 3 and `ncol` equal to 3.

And then if I try to operate it, it will give me here this matrix in which you can see here the values of the diagonal element is only 1 and all other off diagonal elements they are 0. Well, in case if you want you can also do the same operation using the matrix command, but in that case you have to be very careful when you try to define the value of the data and the option by row. This data has to be assigned in such a way, so that it is arranged in the way I want. So, in order to avoid those complications this `diag` command directly gives us the diagonal matrix.

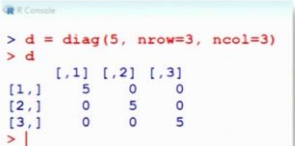
(Refer Slide Time: 08:50)

Matrix Operations: Diagonal matrix

Construction of a diagonal matrix with all numbers as 5 of a dimension 3:

```
> d = diag(5, nrow=3, ncol=3)
> d
```

	[,1]	[,2]	[,3]
[1,]	5	0	0
[2,]	0	5	0
[3,]	0	0	5

$$\begin{pmatrix} 5 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 5 \end{pmatrix}$$


$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix}$$

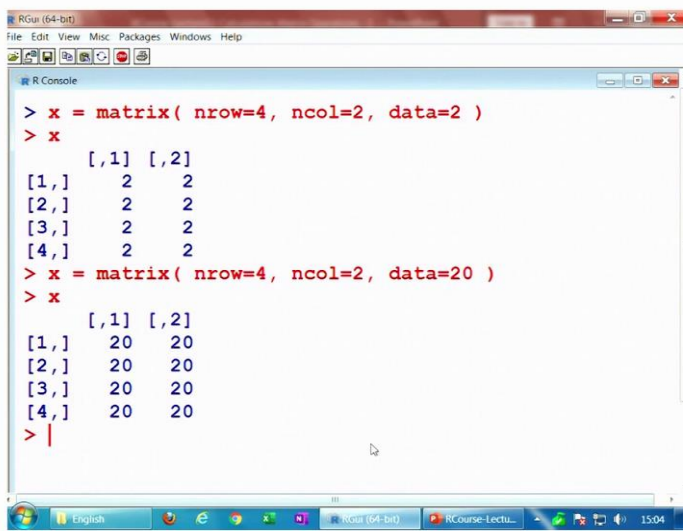
7

And similarly in case if you want to choose here some other value and the diagonal matrix suppose I want to create matrix in which the there are three rows and three columns and all the values on the diagonal element they are 5 and all the other values on the off diagonals they are 0.

So, in that case I have to simply use the same command here `diag` and then `nrow` equal to 3, `ncol` equal to 3. So, that it generates a 3 by 3 matrix and the value on the diagonal elements is 5. So, if you try to see here it will give you this type of operation. And now you think that in case if I want to have a matrix in which the diagonal elements are like 1, 2, 3 and off diagonal elements are 0.

How to do it, think about it? But before that let us try to first understand these operations on the R console, so if you try to see here if you want to create such a matrix in which all the elements are going to be 2.

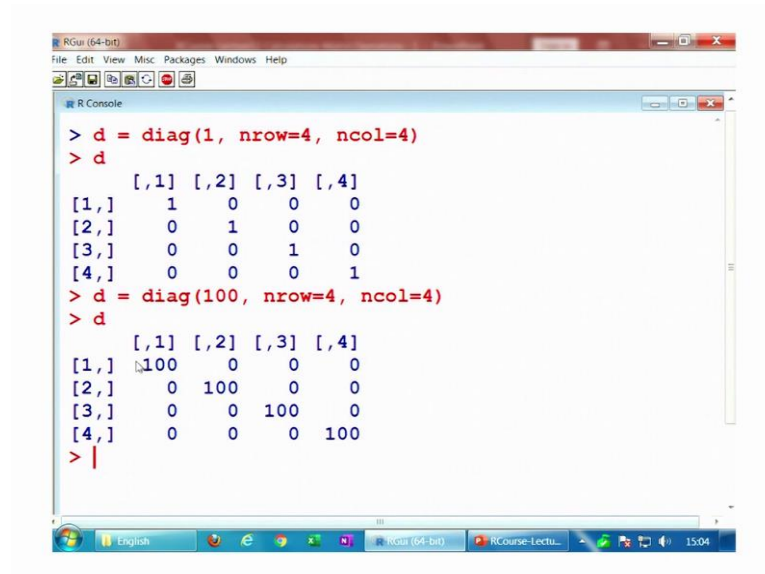
(Refer Slide Time: 09:47)



```
> x = matrix( nrow=4, ncol=2, data=2 )
> x
      [,1] [,2]
[1,]    2    2
[2,]    2    2
[3,]    2    2
[4,]    2    2
> x = matrix( nrow=4, ncol=2, data=20 )
> x
      [,1] [,2]
[1,]   20   20
[2,]   20   20
[3,]   20   20
[4,]   20   20
> |
```

Then let me try to show you here if you use the same command it gives you here 2. And if you want to give here some other value say 20, then if you try to see here this will give you all the values in the 4 by 2 matrix as 20.

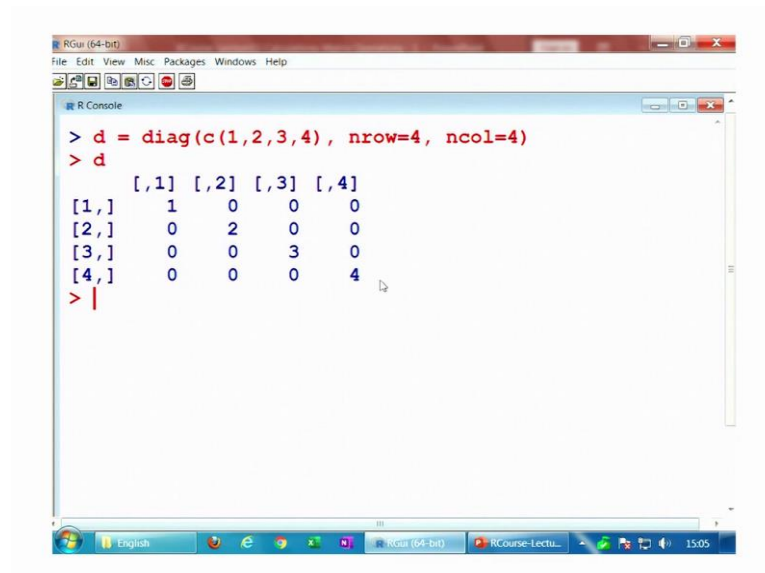
(Refer Slide Time: 10:04)



```
> d = diag(1, nrow=4, ncol=4)
> d
      [,1] [,2] [,3] [,4]
[1,]  1   0   0   0
[2,]  0   1   0   0
[3,]  0   0   1   0
[4,]  0   0   0   1
> d = diag(100, nrow=4, ncol=4)
> d
      [,1] [,2] [,3] [,4]
[1,] 100   0   0   0
[2,]  0 100   0   0
[3,]  0   0 100   0
[4,]  0   0   0 100
> |
```

And similarly in case if you try to define here a diagonal matrix say `diag`, then the data here is 1 and suppose I define here `nrow` is equal to 4 and `ncol` is equal to 4, then this will give me an identity matrix of order 4 like this. And if I try to change the value here on the diagonal, suppose I want to have a 100. So now, you can see here this matrix here is like this where all the diagonal values are here 100.

(Refer Slide Time: 10:33)



```
> d = diag(c(1,2,3,4), nrow=4, ncol=4)
> d
      [,1] [,2] [,3] [,4]
[1,]  1   0   0   0
[2,]  0   2   0   0
[3,]  0   0   3   0
[4,]  0   0   0   4
> |
```

And as I ask you that if you want to suppose give here a values like here four values 1, 2, 3, 4 like this. So, then how to give it means on the diagonal the values are 1, 2, 3, 4 and

off diagonal elements are 0. So, I can use here my logic and I can give here the data in the using the data vector command and let us see what happens you can see here now so this 1, 2, 3, and here 4.

So, you see now you can think that how you can combine different types of operations, which are possible in matrix theory and then try to experiment with them try to play with them and try to see what is really happening, right.

(Refer Slide Time: 11:12)

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

TRUE → T
 FALSE → F

```
> 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

Transpose

1	3
2	4

```
> 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
> |
```

So, now after this I come to some more operations, I am sure that those who are familiar with the matrix theory, they must be knowing about the transpose of a matrix. Transpose of a matrix that means, the rows and columns are interchanged. For example, if I try to take care of very simple example like as here if I take a matrix here 1, 2, 3, and here 4. So now, in case if I try to find out the transpose of this matrix then these rows and columns will be interchanged.

So, this first row this will become here like first column 1, 2 and this 2nd row will become here 2nd column like this. So, if you want to do such an operation in the R Software then how to get it done? So, for that suppose we define here a matrix of order 4 by 2, which has the data values 1, 2, 3, 4 up to 8 and by row is equal to TRUE and you know that instead of using TRUE and FALSE we can also use here capital T and capital F respectively.

So, I try to use here t and then gives me here this type of matrix in which the observations are row wise arranged you can see 1 2 3 4 5 6 7 8.

(Refer Slide Time: 12:22)

Matrix Operations: Transpose

Transpose of a matrix X: X' *t (matrix)*

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

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

Now, I try to find out its transpose, because the command to find out the transpose of a matrix is simply T and inside the parenthesis, you have to write down the matrix. So, if you see here I try to write down here T and inside parenthesis this is x and I try to give it a name say here xT and now you can see here this is changed.

So, you can see here from the screenshot that this column which was 1 3 5 7, now this becomes here a row. First column becomes first row and this 2nd column this is 2, 4, 6, 8 this becomes here a 2nd row. So, this is the transpose operation and you can very easily do it in the R software.

(Refer Slide Time: 13:03)

Matrix Operations: Row and column sums

Finding the row and column sums

```
> 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
```

rowSums(x) Finds the sum of numbers in rows

colSums(x) Finds the sum of numbers in columns

Similarly, in case if you want to find out the sum of the values in the rows and columns, then how to get it done? So, I will just take an example to explain you. So, suppose I define here a matrix like this which is a 4 by 2 matrix with the data 1 to 8 and it is here like this. So now, in case if you want to find out the sum of the elements on the rows and columns, what does this mean? If I try to say here the sum of the values in the first row this is here 1 plus 5 which is here 6.

And similarly, in the 3rd row it is 3 plus 7 which is equal to here 10 and similarly in case if you want to find out the values in the first column and try to find out their sum, this will be 1 plus 2 plus 3 plus 4 which is equal to here 10. So, in order to do such operations in the R software which are required many times while doing calculations we have a command here row sums and column sums.

But if you try to see here how these commands have been spelled for rowSums it is r o w S u m s, where this S which is the first S in the sum this is capital this is upper case and similarly in the 2nd command c o l S u m s and in this case also this S is capital. So, this will find the column sums. So, in case if you simply try to give here the rowSums or colSums and inside the parenthesis you give the name of the matrix it will find out all the sums.

(Refer Slide Time: 14:31)

Matrix Operations: Row and column means

```

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

$(1+5) = 6$
 $(2+6) = 8$
 $(3+7) = 10$
 $(4+8) = 12$

$(1+2+3+4) = 10$ $(5+6+7+8) = 26$

```

> rowSums(x)
[1] 6 8 10 12

> colSums(x)
[1] 10 26
  
```

```

> 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
> rowSums(x)
[1] 6 8 10 12
> colSums(x)
[1] 10 26
  
```

For example, you can see here first side first you try to look here, can see here row sums of x it is 6, 8, 10, 12 and column sums of x is 10, 26 like this. So, if you try to see what it is trying to do? It is trying to add here 1 plus 5 which is here like this 2 plus 6 here 8, 3 plus 7 here this is 10 and 4 plus 8 it this is here 12.

And similarly, if you try to find out the column sums here 1 plus 2 plus 3 plus 4 this is here 10 and 5 plus 6 plus 7 plus 8 which is here 26 and these are the values 6, 8, 10, 12 which are given here and these are the values 10 and 26 which are given here. So, you can see here, right.

(Refer Slide Time: 15:16)

Matrix Operations: Row and column means

Finding the row and column means

```
> 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

Handwritten calculations:

- For the first row: $\frac{1+5}{2} = 3$
- For the first column: $\frac{1+2+3+4}{4} = \frac{10}{4} = 2.5$

rowMeans(x) Finds the means of rows

colMeans(x) Finds the means of columns

12

So, similarly instead of finding out the sums if you want to find out the arithmetic mean of the values in the rows and columns, then similarly those values can be found very easily. So, suppose if I try to take here the same matrix here which we have just considered for finding over the row sums and column sums. So, we have this matrix here and now we want to find out the arithmetic mean of the values in the rows and columns.

For example, if you try to see here, in the first row I have the values here 1 plus 5. So, the arithmetic mean will be here 1 plus 5 divided by 2 which is here 3 and similarly if you try to see the sum of the values in the first column 1 plus 2 plus 3 plus 4 divided by 4 which will be equal to here 10 upon 4 is equal to 2.5, right.

So, in order to find out this means we have the command here rowMeans and colMeans, but you have to be very careful that here this M this is in the uppercase alphabet, that is capital M. So, the spelling here is r o w capital M e a n s out of is e a n s is again in the lower case and similarly in the column means c o l in the lower case col then capital M and then in the lower case alphabet e a n s and then inside the parenthesis you have to give the name of the matrix.

(Refer Slide Time: 16:34)

Matrix Operations: Row and column means

```

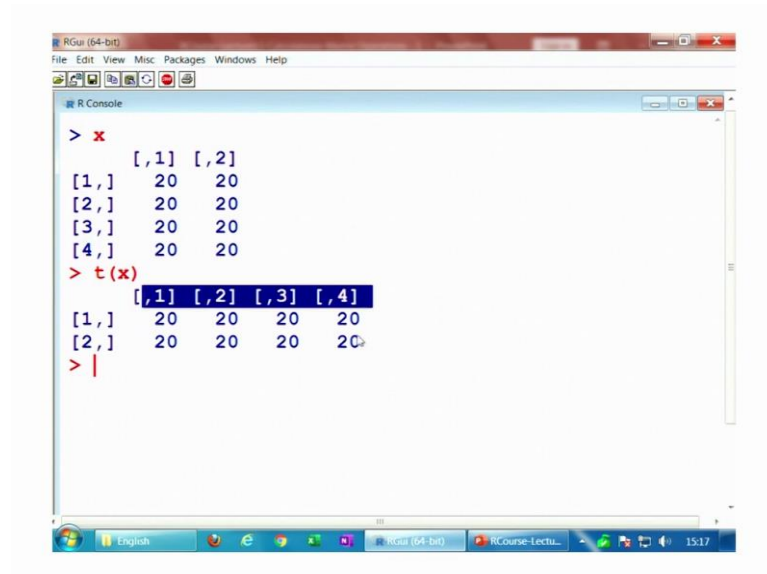
> x
  [ ,1] [ ,2]
[1,]  1  5
[2,]  2  6
[3,]  3  7
[4,]  4  8
(1+5)/2 = 3
(2+6)/2 = 4
(3+7)/2 = 5
(4+8)/2 = 6
(1+2+3+4)/4 = 2.5
(5+6+7+8)/4 = 6.5
> rowMeans(x)
[1] 3 4 5 6
> colMeans(x)
[1] 2.5 6.5
  
```

So, if you try to see this is the very simple operation, that if you try to look here it will try to say here 1 plus 5 which is equal to here 1 plus 5 divided by 2 which is 3, then 2 plus 6 which is here 8 divided by 2 which is here 4 and then 3 plus 7 which is here 10, 10 divided by 2 which is here 5, 4 plus 8 which is here 12, 12 divided by 2 which is here 6.

So, this 3, 4, 5, 6 you can see here this has been obtained here in the command rowMean x and the same output is obtained here 3, 4, 5, 6. Now in case if you want to find out the column means you know that you have to find out the sum of the values in the first column that is 1 plus 2 plus 3 plus 4 and then divided by 4 which will be here 2.5.

And the means of the value in the 2nd column it is 5 plus 6 plus 7 plus 8 divided by 4 which is here 6.5 and you can see here these 2 values are appearing here in the outcome of the command called means, which is which are 2.5 and 6.5. So, let us try to do these operations on the R console and then try to see how you can obtain them so, right, ok.

(Refer Slide Time: 17:50)



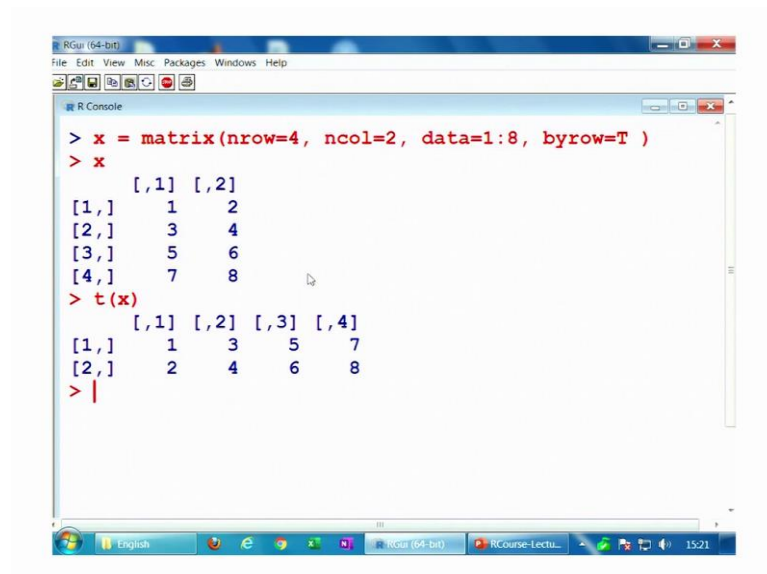
```
RGui (64-bit)
File Edit View Misc Packages Windows Help

R Console

> x
  [,1] [,2]
[1,]  20  20
[2,]  20  20
[3,]  20  20
[4,]  20  20
> t(x)
  [,1] [,2] [,3] [,4]
[1,]  20  20  20  20
[2,]  20  20  20  20
> |
```

So, let us try to use our x which we had defined earlier, but this is all 20. So, if you try to see here the transpose of this matrix here x, you can see here all the values are the same, because it is a constant matrix and but the number of rows and number of columns are changed. Here there are 4 rows and 2 columns, but now here there are 2 rows and 4 columns. So, why not to take the same example that we have considered here in our slides?

(Refer Slide Time: 18:22)



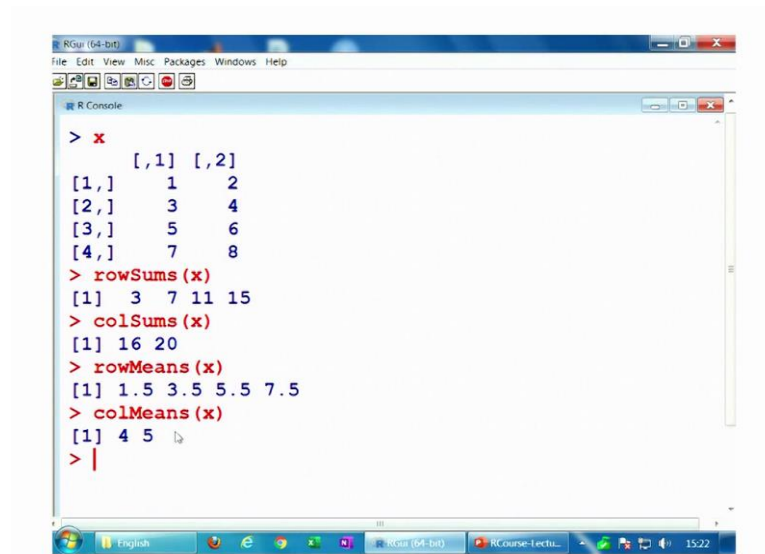
```
RGui (64-bit)
File Edit View Misc Packages Windows Help

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

So, let us try to consider here the same matrix here this is here like this. So, this is the matrix here. So, this matrix has 4 rows and 2 columns, ok and the data is arranged like 1, 2, 3, 4, 5, 6, 7, 8 which are row wise. So now, if you try to find out its transpose then you can see here this is changed. The first row becomes here the 1st column 2nd row 3 and 4 becomes here the 2nd column and so on. So, this is how the transpose operation is done.

(Refer Slide Time: 18:55)



```
R Console
> x
      [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
[4,]    7    8
> rowSums(x)
[1]  3  7 11 15
> colSums(x)
[1] 16 20
> rowMeans(x)
[1] 1.5 3.5 5.5 7.5
> colMeans(x)
[1] 4 5
> |
```

Now, we try to find out the rowSums colSums rowMeans and colMeans. So, let us try to use the same matrix. So, if you try to see here if I try to find out here rowSums of here x, this will come out of here like this 1 plus 2 this is 3 which is here like this 3 plus 4 is the 7 which is here like this and so on.

And similarly if you try to find out here colSums of this matrix here x this is here like 16 and 20, right, this is 1 plus 3 plus 5 plus 7 is 16 and so is the sum of the 2 plus 4 plus 6 plus 8 which is here 20. Now similarly if you want to find out the rowMeans here you have to be very careful when you are trying to write down the spelling. So, rowMeans will become here like this. So, you can see here every row has got the two elements. So, if you simply try to divide 3, 7, 11, 15, by 2 you get here the rowMeans here like this 1.5, 3.5, 5.5 and 7.5.

And similarly if you want to find out here the column means, the column means of x is obtained here like 4 and 5. So, there are four elements in the each of the column. So, if

you try to divide the column sums by 4 you get the answer which is here 16 divided by 4 is 4 and 20 divided by 4 is 5. So now, you can see here that these are very simple operations, these are very elementary operations also, but when you are trying to do the bigger programming you write the big programs, then possibly these concepts are going to be used at many many places and that was my objective that why I have chosen this operation.

But now I have taken very small number of operations, so that you can settle this concept inside your mind and you have time to practice it. So, you try to practice this command try to take some examples yourself and try to see that whatever calculation you are trying to do manually, is it matching with the outcome on the R Software. So, you try to practice it and I will see you in the next lecture till then goodbye.