

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

Basics of Calculations
Lecture - 12
Matrices

Hello friends. Welcome to the course Foundations of R Software and you may recall that in the last couple of lectures so we have learnt the different aspects of calculations and computation in the R software. We have learnt about how to do the calculations, with the scalars, with data vectors and with built in functions.

Now, continuing on the same line, another very important part of calculation is matrix. You know what is the matrix and the rules for matrix operations, they are different than the usual mathematical operations. So, the next question before us is how to handle the matrix in the R software and then, how to do various types of calculations and computation related to the matrix.

Well, this matrix is very important because when you are going to use the R software for various applications in statistics and other areas, many times, the input data has to be there in the format of matrix. The output data is also many times in the format of matrix. So, how to understand it, how to read it and how you can extract a particular part of the matrix, it can be an element, it can be a row vector, it can be a column vector or it can be a sub matrix.

Then after that there are many operations like as transpose, inverse, for finding out eigen values etc., which are very popular when you are trying to use the statistics and other sciences where you want to do the calculations. So now, in this lecture and in the next couple of lectures my target is that I would like to discuss about the matrices.

So, in this lecture, I will try to give you basic idea that how you can define the matrix inside the R software. And then in the forthcoming lectures, I will try to take some common popular operations, which are related to matrices. So, let us begin our lecture and try to understand first that how we can handle the matrix in the R software, ok.

(Refer Slide Time: 02:31)

Matrix

Matrices are important objects in any calculation.

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"), $i = 1, 2, \dots, n, j = 1, 2, \dots, p$.

An element of a matrix can also be an object, for example a string.

However, in mathematics, we are mostly interested in numerical matrices, whose elements are generally real numbers.

So, the matrix theory, you know that matrix theory plays a very important role in doing modeling and other type of things and in R also matrices are important objects in any calculation. The first question comes here what is the matrix? I believe that you all have a reasonable background in the matrix theory, but just for the sake of a quick revision, means I can tell you a matrix is a rectangular array in which the values are arranged in rows and columns.

For example, if I say that there is a matrix which has got p rows and n columns, right. So, this looks like this, I mean there are rows here 1, 2 up to here 3 rows and then there are here columns, 1, 2 etc. and columns and the data is like arranged like 1, 2, 3, 4, 5, 6 and so on, that you know. Now, the values inside this matrix they are the elements of this matrix and an element in the i th row and the j th column is denoted by say X_{ij} like this, X and the subscript ij .

Well, that is the notation, when we try to do this matrix algebra on the blackboard in the book. But when we try to write it down in the say program version means how to write it in the software, then a popular notation is like this one; X and inside the square brackets, you try to write here i and j or i and j they are going to be some values. For example, i can be any values the value between 1 to n and j can be any value between 1 to p .

Now, you can see here, that for the first time, I am now showing you what is the application of this square bracket and you can recall that at some point of time, I had shown you that when we are trying to do the mathematical calculations, then we have this type of simple brackets, then curly brackets and then say square brackets and these brackets have got some important role in the R software.

So, now this is first place where I can show you that this square bracket has a different role, than the mathematical operations. So, now, in case if you try to understand the matrix in a broader way, then the element of a matrix can be an object, for example, a string also. But in mathematics, we are mostly interested in the matrices which have got some numerical values and whose values are some real numbers, right.

(Refer Slide Time: 05:15)

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

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

no. of rows, no. of columns, data vector

```
> x
```

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

4*2=8, no. of rows, no. of columns, data ??

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

So, the first question comes here; how can you create a matrix in the R software? So, we know that whenever we want to create a matrix, there are three ingredients which are needed. We want to know the total number of rows, then total number of columns in the matrix and then we want to know what is the data that has to be arranged in these number of rows and columns. And after that many question comes whether you want to arrange the data row wise, column wise etc.

So, you will try to seek answers for all those questions. So, the first thing I would like to discuss here that how to create the matrix. So, in the R software, matrix is the command

to create a matrix which is `m a t r i x` and all are given in the lower-case alphabets. And after that within the parenthesis, you have to indicate the basic information that how the matrix has to be created you want to inform the total number of required rows columns and the data.

So, in order to inform the matrix that how many rows are going to be there, we have a command here `n r o w`; `n r o w` and in case if you write `n r o w` equal to for example, 4. So, this will indicate that there are 4 rows in the matrix. Similarly, in order to know the number of columns in the matrix, we have the command here `n c o l` `n c o l` which means number of columns.

So, if you write here `n c o l` is equal to 2 this will indicate that there are going to be two columns in the matrix. Now when you have 4 rows and 2 columns, then you need to have 4 into 2 is equal to 8 data values, which are to be arranged inside the matrix in rows and columns. And you know any particular element in the matrix has an address and that address is indicated by the row and column. For example, I can say the element in the second row and third column.

So, in order to give the input data inside the matrix, we have a R command here `d a t a`, data and now after this you have to write `d a t a` is equal to, but after that you have couple of options, that you can give the data in the form of a sequence or individual values or in continuation etc. But anyway at this moment, we have not done many things in the R software. So, I would try to keep it here very simple and I try to take here eight values 1, 2, 3, 4, 5, 6, 7, 8 and I use the format of data vector.

So, these values are going to be arranged in 4 rows and 2 columns. So, in case if I try to assign this command in a variable here `x`, and if you try to execute on the R console then `x` will look like this. So, you can see here, this is indicating the first row let us say `r 1` then second row I say `r2` third row say `r3` and fourth row say `r4`. And these are the columns.

So, this is suppose here column 1 and this is here column 2. So, now, you can see here there are 8 positions here where the data has to be placed. So, now, all these values which are given in the data vector they will be placed inside this matrix in these eight

values. So, you can see here how these values are going to be placed. The first value if you try to see the data is in this order 1 to 8.

So, the first value comes here, then the second value, then the third value, then the fourth value and after this the control goes to the first row in the second column. So, first four values are inserted in the first column in four rows. Now, after this the 5th value is going to be entered, then 6th value is going to be entered, then 7th value is going to be entered and then 8th value is going to be entered.

So, now you can see here that this data is entered like this. So, the data is entered here. How- Column wise that first the data is filled up in the column and then in the rows. Yes, that can be interchanged also. So, how to get it done? These are the questions that we are going to understand in the forthcoming slides. So, this you can see here this is the screenshot, if I just write here like this and then you get here this type of matrix, this type of output, ok.

(Refer Slide Time: 10:38)

Matrix parameters

We see:

- The parameter **nrow** defines the row number of a matrix.
- The parameter **ncol** defines the column number of a matrix.
- The parameter **data** assigns specified values to the matrix elements.

The data values from the parameters are written **column-wise** in matrix.

4

So, now if you try to see in this matrix command, I have explained you what is the role of nrow, what is the role of ncol and what is the role of data and these data values are written column-wise. So, that is what you have to understand.

(Refer Slide Time: 10:55)

Accessing any element of a matrix

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

$x_{1,2}$ → value in the 1st row and 2nd column
 $x_{1,2} = 5$

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

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

R Console

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

$x = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}, \begin{bmatrix} 5 \\ 6 \\ 7 \\ 8 \end{bmatrix}$
row column
 $x[3, 2]$

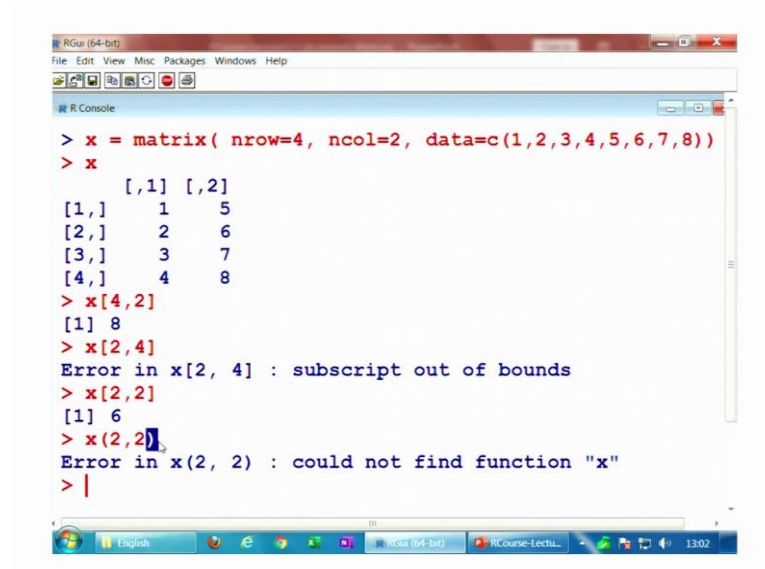
Now, the next question comes here how do you identify or how do you access a particular value in the matrix. So, one thing I can share with you that is coming from the matrix theory actually, that when I try to write down here x say here 1, 2. So, this means this is the value in the first row and second column. So, this 1, 2 is indicating like here the position of row and position of column.

So, for example, in this matrix which is given here, if you try to see in case if I see here x 1, 2. So, 1 means the first row like I say like this and 2 is the second column so like this. So, this value at the intersection of these two lines this is the value of x 1, 2 which is equal to here 5. So now, the same thing I want to do in the R software. So, for that the question is what type of command are we going to use? So, in case if you want to access a single element for, of a matrix say x, because you have to give the matrix a name. So, here I have given the matrix name as x, then inside this square brackets you have to first write the position of the row and then you have to write the position of the column. From there, you want to access that element. For example, in case if I want to access the element in the third row and second column, then I have to write down here x third row; that means, r equal to 3 and column is the second, so c is equal to 2.

So, now if you try to see here, if I try to write down here x 3, 2. So, what is this value? This is here my row number 3 and this is my here column number 2. So, this value at the intersection is 7. So, now, as soon as you write here x 3, 2 and the 3, 2 they are separated

by comma and then they are written within the square brackets, ok. So, this value will come out here 7 and this 7 is actually here, this 7. And if you try to execute it on the R console also there will not be any problem, ok, right.

(Refer Slide Time: 13:22)



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

R Console

> 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
> x[4,2]
[1] 8
> x[2,4]
Error in x[2, 4] : subscript out of bounds
> x[2,2]
[1] 6
> x(2,2)
Error in x(2, 2) : could not find function "x"
> |
```

So, before I move further, let me try to give this example, it on to you on the R console. So, let me try to define my here matrix here like this you can see, x is here matrix nrow equal to 4, ncol equal to 2 etc. And if you try to enter here, then you try to see the value of here x this comes out to be like this. So, you can see here now the values are entered 1, 2, 3, 4 column wise and then 5, 6, 7, 8, right and in case if you want to access any particular element from here, you can write the name of the matrix x. And then suppose if I say here 4 comma 2.

So, 4 comma 2 will come here 8, because this is the fourth row and the second column and this value here is 8. And similarly, if you try to take here say x 2 comma 4, then what will happen do you what you expect this is your here second row and fourth column. But where is the fourth column? I have only had two column, column number 1, column number 2.

So, let us see what R does, it will give you error. That the subscripts out of bounds; that means, you are trying to give the value of the column which is not in the range of the columns which are present in the matrix.

So, rather if you try to write down here something like x 2, 2 what does this mean? This is the second row and second column. So, the value here is 6. So, if you try to enter here, you will get here the value 6. And in case if you try to use here some other bracket like x 2 comma 2, you can see here this will not work because it cannot understand what this parenthesis is trying to indicate. So, you have to use only the square bracket that is going to make the R understand that this is a command for the matrix.

(Refer Slide Time: 15:25)

Entering data column wise in a 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(1,2,3,4,5,6,7,8))
```

OR

```
x = matrix( nrow=4, ncol=2,
            data=c(1,2,3,4,5,6,7,8), byrow = FALSE)
```

> x

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

logical TRUE FALSE
default

So, now we come back to our slides and try to see some more operations, ok. So, now you have seen, that when you are trying to define a matrix in this particular way as you have done, this data is going to enter column wise that you can see here, the data is entered here, column wise, like 1, 2, 3, 4 and then 5, 6, 7, 8 like this.

So, now, my objective is that suppose I want to enter this data row wise. So, how should I control it? So, when you are trying to enter the data column wise, 1 to 8, it goes like this 1, 2, 3, 4 and then it goes to second column 5, 6, 7, 8. But when you want to write down the data in row wise, it means that 1 after this it will be 2 will be assigned in the second column and then it comes to 3 and then it will go to 4, then it comes to 5 in the first column and 6 in the second column, then 7 comes in the first column and 8 comes in the second column.

So, now, how to control it? So, for that we have here an option which is `byrow`, byrow; this by row this is a logical variable. So, this can take two possible values, one is here TRUE and another here is FALSE. So, now, in case if you write byrow is equal to FALSE and try to think for a while, what does this mean? You are trying to say by row is FALSE; that means, the data has not to be arranged by rows. When the data is not to be arranged in rows then what is the next option? The data has to arrange in the columns only.

So, in this case, if you try to see here the outcome comes out comes out to be like this data is arranged in the column, but what is this mean? If you try to compare, this command which is not using byrow and the second command which is using here the byrow. When you are trying to use byrow then or you are not using the byrow, it is giving you the same outcome.

This means when you are not using the option of byrow then matrix command is assuming byrow equal to FALSE as default. That either you will give it or not, it will always assume that byrow is equal to FALSE.

(Refer Slide Time: 17:55)

Entering data row wise in a matrix

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

```
> y = matrix( nrow=4, ncol=2,
              data=c(1,2,3,4,5,6,7,8), byrow = TRUE)
```

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

7

So, now the question comes if you want to arrange your data row wise, then what you have to do that is now very obvious simply try to use byrow is equal to TRUE. And in case if you try to use the same command, but now I am using it in a different variable

name say here y. So, matrix command with nrow equal to 4, ncol equal to 2 and data is the same, but by row is equal to now TRUE. Now, you can see the outcome of y.

So, you can see here 1 is coming here, then it goes to 2 in the second column. Then the next value comes with the first column, it is here 3 and then it goes to the fourth column which is here 4 and then it comes with the first column as 5, then go to the next column as 6. Then it comes to the first column as 7 and then goes to the next column as 8. So, you can see here this operation is going like; this is row wise assignment of the data.

So, now you can see this is in your control that how you want to assign the data in the matrix.

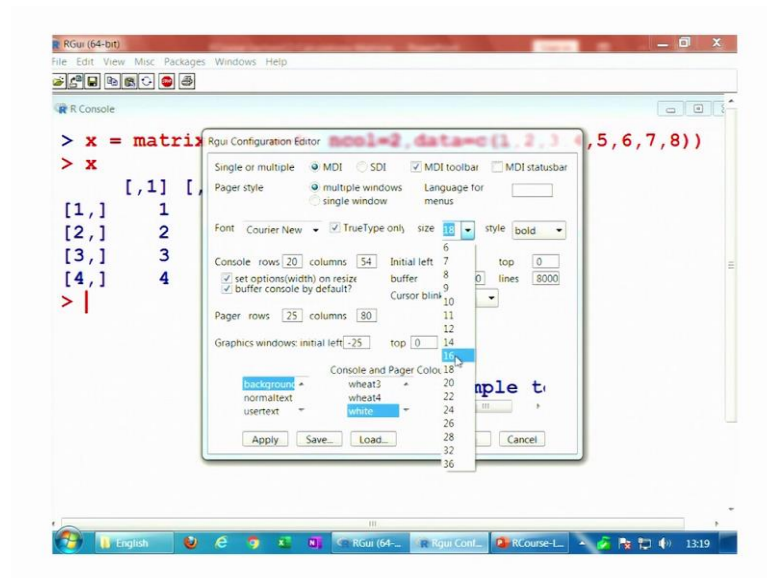
(Refer Slide Time: 18:50)

Entering data row wise and column wise in a matrix

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

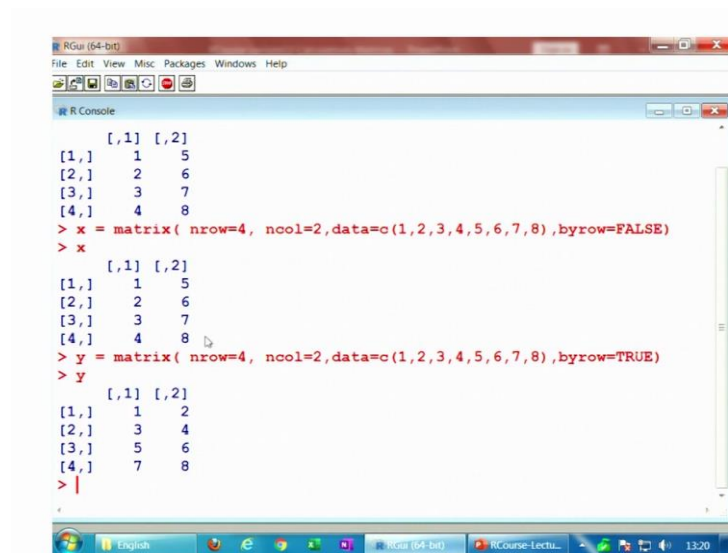
And if you try to see here the screenshot, that when you are trying to see here there is no option, by row is equal to TRUE or FALSE you are not writing anything, then the data is this here like this column wise. And when you are writing here by row is equal to TRUE then the data here is row wise like this.

(Refer Slide Time: 19:15)



So, let me try to show you these operations on the R console also so that you get here more confident. So, you can see here, this is your here in matrix x, where you have not used any command over here and you can see here this is your here x. Now, in the same command, let me try to reduce the font so, that you can see everything on the same scale, suppose I make it here 16.

(Refer Slide Time: 19:36)



So, now you can see here, that I try to add here byrow is equal to FALSE. Now, you can see here the value of here x, this is here like this. Let me reduce the font size to be like this so that you can see clearly. Yes, now it is clear. So, you can see here when you are trying to use here byrow equal to FALSE, then the outcome here and here that is the same. But now in case, if you try to use here this outcome and you try to make it here true.

And you try to assign it in a new matrix, say y then you can see here y. Now, you can see here, when you have by row equal to FALSE then the data is like 1, 2, 3, 4 and when you are trying to use here by row equal to 2, then the data like 1, 2, 3, 4 and so on. So, that is the difference between the two, right, ok. So, now, let me come back to our slides and try to see some more operations, right.

(Refer Slide Time: 20:32)

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

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

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

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

So, now let us try to consider the same matrix and with then we try to learn some more operations on this matrix x, which has four rows and two columns with the data 1 to 8. So, this is your column wise arrangement and we try to learn here something here more.

(Refer Slide Time: 20:48)

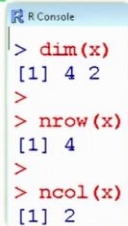
Properties of a Matrix

We can get specific *properties* of a matrix:

```
> dim(x) # tells the dimension of matrix
[1] 4 2
```

> nrow(x) # tells the number of rows
[1] 4

> ncol(x) # tells the number of columns
[1] 2



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

```
> dim(x)
[1] 4 2
> nrow(x)
[1] 4
> ncol(x)
[1] 2
```

Now, whenever you are trying to use a matrix. Suppose some matrix is given to you and you want to know the information about this matrix.

So, one information in which you are always interested in the dimension of the matrix, that how many number of rows are there, how many number of columns are there and so on. So, in order to know the dimension of the matrix we have a command here `dim`, `dim`. And within the parenthesis if you write down the name of the matrix it will give you the information like is here, like this 4 comma 2, 4 is the number of rows and 2 is the number of columns.


And in case if you want to know only the number of rows, then the command here is `nrow` and parenthesis inside the parenthesis you write the name of the matrix. So, this will be your here `nrow` and say inside parenthesis `x` and the answer will come out to be here 4. Why? Because you can see here in this matrix `x`, you have here 1, 2, 3, 4 rows and how many columns are there 1 and here this 2.

So, now in order to know the columns of a matrix, you simply have to use the command here `ncol`; `ncol` and inside the parenthesis you have to write the variable in which the matrix is saved. And it will explain you the number of columns present in the matrix. So, it is here 2. So, you can see here these are not very big operations.

(Refer Slide Time: 22:08)

Properties of a Matrix

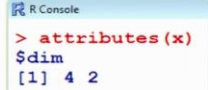
```
> mode(x) # Informs the type or storage mode of an object, e.g., numerical, logical etc.
[1] "numeric"
```



```
> mode(x)
[1] "numeric"
```

attributes provides all the attributes of an object

```
> attributes(x) #Informs the dimension of matrix
$dim [1] 4 2
```



```
> attributes(x)
$dim
[1] 4 2
```

11

And similarly, if you want to know the mode of the matrix, do you remember mode we had done in the earlier lecture, where we had numeric character its etc. and remember this mode was not the mean, median, mode, this is statistical mode.

So, this is going to inform us that how the values in this matrix x are stored. So, this is you can see here numeric and similarly we have here one more command here attributes. So, this also provides all the attributes of an object actually.

So, in this case because this is a matrix so, if you try to write down here attributes x inside the parenthesis, then it will give you this type of information which is trying to tell you the dimension of this x which is here 4 and 2, which is 4 rows and 2 columns. And the spelling of this attributes is a double t r i b u t e s in all in lower alphabets lowercase alphabets, small alphabets.

(Refer Slide Time: 23:08)

Help on the Object "Matrix"

To know more about these important objects, we use R-help on "matrix".

```
> help("matrix")
matrix package:base R Documentation
Matrices
Description:
' matrix' creates a matrix from the given set of values.
' as.matrix' attempts to turn its argument into a matrix.
' is.matrix' tests if its argument is a (strict) matrix. It is generic: you can write methods to handle specific classes of objects, see Internal Methods.
```

12

So, this is how you can obtain this type of information. Now, in this matrix there are many other options which are available that how you can handle this matrix. So, I will not go into all the details, but I will request you is that you try to see into the help matrix.

(Refer Slide Time: 23:30)

```
Then we get an overview on how a matrix can be created and what
parameters are available:
Usage:
  matrix(data [= NA,nrow = 1,ncol = 1,byrow = FALSE,
            dimnames = NULL)
  as.matrix(x)
  is.matrix(x)
Arguments:
  data: an optional data vector.
  nrow: the desired number of rows
  ncol: the desired number of columns
  byrow: logical. If 'FALSE'(the default) the matrix is filled by
        columns, otherwise the matrix is filled by rows.
  dimnames: A 'dimnames' attribute for the matrix: a 'list' of
            length 2.
  x: an R object.
```

So, try to type help and within double quotes, type here matrix it will give you all the description matrix as matrix, is matrix etc.

(Refer Slide Time: 23:37)

```
Then, the meaning of each parameter is explained:
Details:
If either of 'nrow' or 'ncol' is not given, an attempt is made
to infer it from the length of 'data' and the other parameter.

If there are too few elements in 'data' to fill the array, then the
elements in 'data' are recycled. If 'data' has length zero, 'NA'
of an appropriate type is used for atomic vectors and 'NULL' for
lists.

'is.matrix' returns 'TRUE' if 'x' is a matrix (i.e., it is _not_
a 'data.frame' and has a 'dim' attribute of length 2) and
'FALSE' otherwise.

'as.matrix' is a generic function. The method for data frames
will convert any non-numeric/complex column into a character
vector using 'format' and so return a character matrix, except
that all-logical data frames will be coerced to a logical matrix.
```

And after that it will give you the usage then the details about different arguments, dim names and then after that it will give you these type of detail.

I have simply taken it from the help which is available in the R, right. So, I will not take detail it here, because now you know all the things I already have explained you the concept of say is and as for example, if you use here this command as dot matrix. So, this will simply show you whether the variable here is in the format of matrix or not and in case if you want to convert any number into a matrix, then you have to use as dot matrix right.

(Refer Slide Time: 24:06)

```
Finally, references and cross-references are displayed...
References:
Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) _The New
S Language_. Wadsworth & Brooks/Cole.

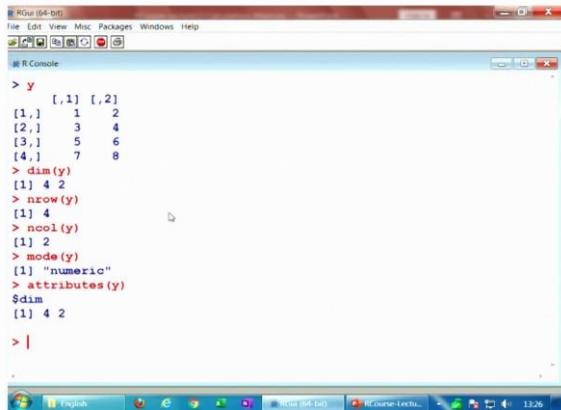
See Also:
'data.matrix', which attempts to convert to a numeric
matrix.

.. as well as an example:

Examples:
is.matrix(as.matrix(1:10))
data(warpbreaks)
!is.matrix(warpbreaks)# data.frame, NOT matrix!
warpbreaks[1:10,]
as.matrix(warpbreaks[1:10,]) #using
as.matrix.data.frame(.) method
```

So, all these things are there, I would request you that you please try to look into this help menu and try to read it here. And I will try to show you these things on the R console about these two operation dimension nrow, ncol and more attributes. So, let me try to show you these things on the R console.

(Refer Slide Time: 24:30)



```
R Console
> y
  [,1] [,2]
[1,]  1   2
[2,]  3   4
[3,]  5   6
[4,]  7   8
> dim(y)
[1] 4 2
> nrow(y)
[1] 4
> ncol(y)
[1] 2
> mode(y)
[1] "numeric"
> attributes(y)
$dim
[1] 4 2
> |
```


So, you can see here now, let me try to see here y is our this matrix and if I want to find out the dimension of this here y, this is here 4, 2. In case if I want to find out the nrow in the matrix y, you can see here this is here 4. And in case if I want to find out their ncol of this y this is here, like this 2.

And if I try to find out here the mode of y, this will be here numeric and if I try to find out the attributes of here y, this will be like 4, 2. So, you can see here getting the information about the matrix is not difficult at all ok. So, now we come to an end to this lecture this was a very simple short lecture. Short in terms of the content of the lecture.

My idea was that ok this is the first lecture where you are trying to understand the concept of matrix for the first time. So, I should stop here. So, that you get some time to do all these operations yourself in the R console. It is not difficult, the only thing is this you have to settle down these concepts in your mind that how this nrow, ncol, data, byrow, etc. they are going to work.

Once you are familiar with these objects, after that I promise you handling with the matrices is very simple and it will be very easy for you to do such operation. So, you try to have a quick revision, try to practice it and I will see you in the next lecture with more details on matrix operation till then good bye.