

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

Lecture - 29
Sorting, Ordering and Mode

Hello friend. Welcome to the course Foundations of R Software and you can recall that in the last couple of lectures we had discussed various aspects of the data handling. So, continuing on the same line, in this lecture also we are going to take up one more topic of the data handling. This is about Sorting, Ordering and Mode.

So, you know that whenever you are trying to handle the data that is related to calculations, simulations or database management etcetera; many times you need to arrange the data in a particular order either it an increasing order of values, decreasing order of values or you want to order them etcetera.

And, these are the very common operations which you can do any software. So, the question here is how the same operation you can do in the R software and how R works? What is the difference between sorting and ordering? If I ask you, then these are the type of questions which we are going to answer in this lecture.

(Refer Slide Time: 01:46)

Sorting

`sort` function sorts the values of a vector in ascending order (by default) or descending order.

Syntax
`sort(x, decreasing = FALSE, ...)`
`sort(x, decreasing = FALSE, na.last = NA, ...)`

<code>x</code>	Vector of values to be sorted
<code>decreasing</code>	Should the sort be increasing or decreasing
<code>na.last</code>	for controlling the treatment of <code>NA</code> s. If <code>TRUE</code> , missing values in the data are put last; if <code>FALSE</code> , they are put first; if <code>NA</code> , they are removed.

5 1 3
→ 1, 3, 5 (increasing)
→ 5, 3, 1

So, we are going to take couple of examples and then I will try to explain you how do we manage the sorting, ordering and what is the difference between sort and ordering of the data. And, after that I will give you a quick review of the modes right. So, let us begin our lecture.

So, first we talk about sorting. So, I am 100 percent confident that you all know what is the meaning of sorting. Sorting means you are trying to arrange the given data increasing or a decreasing order, right. For example, if I say here I have here value say 5 1 3. So, you would like to arrange an increasing order or an ascending order like 1, 3, 5.

And, in case if you want to increase the data in the descending order that is in the decreasing order, that will be the largest value will come at the first position, then the second largest and then the minimum value, right. So, the question here is that how you can do such functions in the R software. So, we have a function here `sort`, `s o r t`. This is a function which sorts the values of vector in ascending or descending order.

So, if you try to use only here `sort`, the default choice is going to be ascending order, ok. So, now doing this sorting in R software is extremely simple. You simply have to write here the command `sort`, all in lower case alphabets and within the parentheses, you have to write down the data vector in which you have arranged the values and you want to sort them out. After this there is an option here decreasing, `d e c r e a s i n g` and all in lower case alphabets.

And, this is a logical variable and you have to give it its value in terms of say TRUE or FALSE. So, when you try to give it here FALSE, right. So, decreasing is equal to FALSE. If you try to understand the meaning of this statement; that means, decreasing is FALSE, then what is TRUE? Increasing. So, decreasing equal to FALSE means, increasing so that means, the data in the vector `x` is going to be arranged in an increasing order, right.

So, this is how we try to take a call that we have to arrange the data in an increasing or a decreasing order. And, then we have one more option here `na dot last`. This handles the missing values. In case if you try to give here the value TRUE, then the missing values in the data they are kept in the last.

And, if you try to give it value here the FALSE, then they are put at the first place. And, if you try to use here the option here NA, what I have written here, then the missing values are removed. And, after that there are many more options in the sort and I would request you that you try to look into the help, ok.

(Refer Slide Time: 04:25)

Sorting

Example

```
> y = c(8, 5, 7, 6)
> y
[1] 8 5 7 6
> sort(y)
[1] 5 6 7 8
> sort(y, decreasing = TRUE)
[1] 8 7 6 5
```

Handwritten notes:

- $\min(8, 5, 7, 6) = 5$
- $\min(8, 7, 6) = 6$
- $\min(8, 7) = 7$
- $\max(8, 5, 7, 6) = 8$
- $\max(5, 7, 6) = 7$
- $\max(5, 6) = 6$

Now, let me try to take here some examples and try to show you what is really happening and how the things can be done in the R software. So, if I try to take here a data vector say 8, 5, 7, 6. It has 4 values and I try to store it in a variable y. So, now, this is here the data vector here y and now if you try to see the values in the y are not really arranged in an increasing or a decreasing order ok.

So, now I try to sort it. So, I use the function here sort and in the parenthesis I try to write here y. So, now, you can see here that the minimum value out of 8 5 7 6 which is here 5. So, this is arranged first. Now, first is out and then the minimum value among 8, 7 and 6 is found this is here 6 and then out of the remaining values 8 and 7 whichever is the minimum value that is found and finally, the last value is here 8. So, this is the order in which the sorting is done and you get here an outcome like here 5 6 7 8.

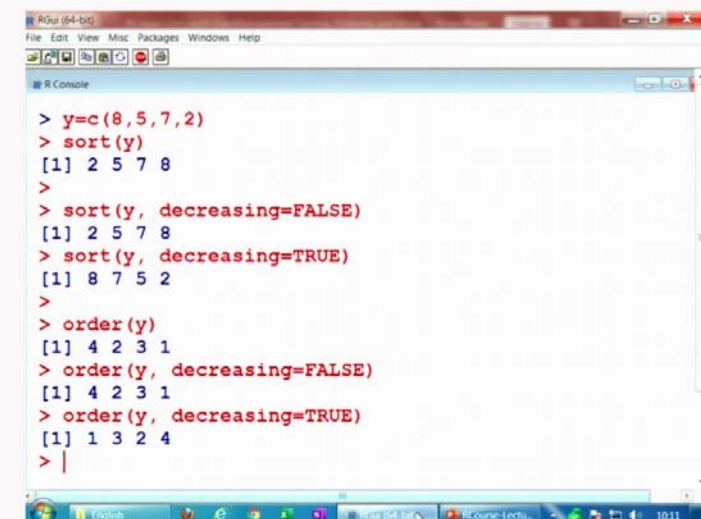
Now, in case if you try to increase this value in a decreasing order, then you have to use here the option decreasing is equal to TRUE right. So, decreasing equal to TRUE what will happen? That whatever the operation, which has been done in the earlier command,

where we got the outcome 5 6 7 8 that is just reversed, right. And, then what will happen?

That, first the maximum value out of this 8, 5, 7, 6 which is here 8 that is obtained and then maximum value among the remaining value 5, 7, 6 that is obtained which is here 7. Then, after this the maximum value among the remaining value 5 and 6 that is obtained 6.

And, then after this whatever is the value remaining 5 that is obtained and this outcome is given here like this. So, you can see here that either of the approaches you can follow to understand what is really happening when we are trying to give the command here, sort y with an option decreasing equal to TRUE, ok.

(Refer Slide Time: 06:59)



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

R Console

> y=c(8,5,7,2)
> sort(y)
[1] 2 5 7 8
>
> sort(y, decreasing=FALSE)
[1] 2 5 7 8
> sort(y, decreasing=TRUE)
[1] 8 7 5 2
>
> order(y)
[1] 4 2 3 1
> order(y, decreasing=FALSE)
[1] 4 2 3 1
> order(y, decreasing=TRUE)
[1] 1 3 2 4
> |
```

So, now let me try to show you these operations first in the R console and then I will try to come to our next topic on order. So, if you see here if I try to take here say c 8, 5, 7 say 2 and if I try to take here sort y. You can see here this is 2 5 7 8 and in case if you try to use here the option decreasing is equal to FALSE, then let us try to see what do you get here. So, this is here the same.

So, you can see here that decreasing equal to FALSE is the default option in the sort function here. And, in case if you try to give this decreasing equal to TRUE, then you

can see here that it is arranged in such a way such that the maximum value occurs first and the minimum value occurs at the end, right, ok.

(Refer Slide Time: 07:53)

Ordering

order function sorts a variable according to the order of variable.

Syntax

```
order(x, decreasing = FALSE, ...)
```

increasing

```
order(x, decreasing = FALSE, na.last = TRUE, ...)
```

x Vector of values to be sorted

decreasing Should the sort be increasing or decreasing

na.last for controlling the treatment of **NA**s.
If **TRUE**, missing values in the data are put last;
if **FALSE**, they are put first;
if **NA**, they are removed.

After this our next command is order. So, the question here is first of all what is an order? What is the difference between sorting and ordering and how to get it done in the R software? So, first I explain you what is the command in the R software to order the value and then I will try to explain you what is the difference between this sorting and ordering, right.

So, the function in the R software for ordering the value is `order`, order and then whatever values in the data vector you want to order you have to store in a variable like here `x`. Just like what we did in the case of `sort` command. So, you write down here `order`, then within parenthesis you try to write down here and then after that we have a option here `decreasing` is equal to `TRUE` or `FALSE`.

So, when you are trying to use here the option `decreasing`, all in lower case alphabet, `decreasing` is equal to `FALSE`. Then, as the meaning suggests that you do not want decreasing. So, what do you want? You want here increasing. So, if you want to order the value in an increasing order of the variable, then you have to use here say `decreasing` is equal to `FALSE` or that is the default option. So, you do not even need to write it.

And, in case if you want to arrange them in the decreasing order of the variable then you have to use here decreasing is equal to TRUE ok. And, similarly here if you try to add here one more option here na.last is equal to TRUE, then this is a variable. If you try to give it here TRUE, then the missing values in the data are put at the last that is towards the end.

If you write it as FALSE, then they are put at the first place and if you try to write down here NA, then they are removed and the order is obtained for the remaining values; the values, which are available in the data vector x. So, this operation is just like what we have learnt in the case of sort.

(Refer Slide Time: 09:53)

Ordering

Example

```
> y = c(9, 8, 5, 7, 6)
> y
[1] 9 8 5 7 6
```

Value	9	8	5	7	6
Position	1	2	3	4	5

Handwritten notes: *Table 1*, $\min(9, 8, 5, 7, 6) \rightarrow 5$, $\min(9, 8, 7, 6) \rightarrow 6$, $\min(9, 8, 7) \rightarrow 7$, $\min(9, 8) \rightarrow 8$

Ordered value	5	6	7	8	9
Position	3	5	4	2	1

Handwritten notes: $(3, 5, 4, 2, 1)$

But, now first we try to understand, what is the difference between sorting and ordering, right. So, if you try to see here, I try to take here a data vector which has five

values; 9, 8, 5, 7, 6 right. Now, if you try to see whenever we are writing a value in a data vector, then there are two components. One is the value and second is the position. For example, if you try to see here this value 9, this is at the 1st position, this value 8 this is at the 2nd position, 5 is at the 3rd position, 7 is at the 4th position and 6 is at the 5th position.

So, in order to explain you I try to just write down these values here. So, here I am trying to write down the all values 9 8 5 7 6 and then here location or positions, their address in the data vector which is 1 2 3 4 5 for all these values. Now, what you try to do that first you try to order the values. So, if you try to see out of this 9 8 5 7 6; what I try to do here

whatever is the minimum value of means among 9, 8, 5, 7, 6 I try to write down here, this is here 5.

Then, the minimum value out of the remaining values 9, 8, 7 and 6, this is here 6. Then, the minimum value out of the remaining values 9, 8, 7 this is here 7. Then, the minimum value between the remaining values 9 and 8, this is here 8. And, then after that we have the value here 9. So, these are my ordered values. So, I try to write down here them here like this say this value 5 6 7 8 9. I am writing here 5 6 7 8 and here 9.

Now, if you try to compare these values from the values which are given here, let me call it here let us say table 1. In the table 1 what I try to do here, I try to write down these values from the table, but I try to move them with their position values. So, I will try to see here, what is my first value here? 5. So, I will go here and I will try to pick up here 5, I will try to bring it here to the 1st value and I try to write down here, 5 and here 3.

So, that is what I am writing here. After this whatever is my next value which is here 6th, I go to the table number 1 and I try to pick up and I try to bring it here, the value along with the position. After this I try to pick up the 3rd value which is here 7. So, I try to pick up here the value here 7 and its position and I try to pick up and bring it here. And, after that I try to do the same operation with here 8.

So, I try to bring this 8 and its position to here, this value. And after finally, can I try to choose the last value 9 which is at position number 1 and I bring it here. So, if you try to see here now, what is the value of these positions here? This is here 3, 5, 4, 2, 1 ok. Just try to keep in mind.

(Refer Slide Time: 13:11)

Ordering

Example

```
> y  
[1] 9 8 5 7 6
```

Value	9	8	5	7	6
Position	1	2	3	4	5

Ordered value	5	6	7	8	9
Position	3	5	4	2	1

```
> order(y)  
[1] 3 5 4 2 1
```

Sort → arrange the values
Order → arrange the location of the values

```
> order(y, decreasing = TRUE)  
[1] 1 2 4 5 3
```

Now, I do one thing here. First you try to look here. I try to execute this command order y on the R console and I get here the outcome 3 5 4 2 1. So, now the question is what is this 3 5 4 2 1? So, whatever I have done here, if you try to look what I have written in my in my hands 3, 5, 4, 2, 1; well I have this copied and pasted this 2 tables here also. So, you can see here this is your here 3 5 4 2 1. So, this is the outcome here.

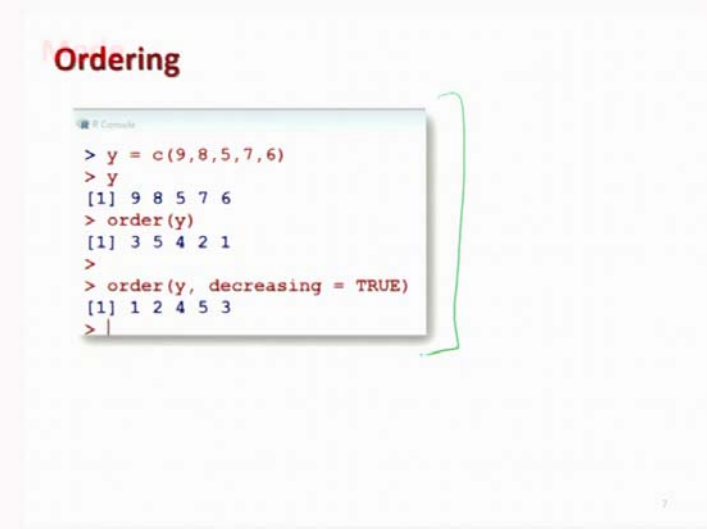
So, if you try to see many times people get confused that the sorting and ordering in the R software they are the same, right. But, here you have to be very careful and that is what I always emphasize that you have to understand what R is trying to do and accordingly you have to manage your actions.

So, if you try to see here when we are trying to use the command here sort, then it is trying to arrange the values. But, when I am trying to use the command order, then it is trying to arrange the location of the values right. So, it is here like this. So, this order command, this is giving us here an outcome 3 5 4 2 1 which is the location of the values when they are ordered.

Or, I will say in very simple words that when you are trying to sort the values, you are not only sorting the values, but you are also trying to sort their corresponding positions. In the sense, that whenever the value is sorted along with the value its position is also carried over.

So, after the sorting, the sort command will try to give you the values and the order command will try to represent the values of their locations in the ordered data vector. So, that is what is happening. So, if you try to use here the command here order with the decreasing is equal to TRUE, then you will you can see here this is like 1 2 4 5 3. What is happening here? This is just the reverse order of 3 5 4 2 1, 3 5 4 2 1, right. So, this is the difference between sorting and ordering, right.

(Refer Slide Time: 15:40)



```
Ordering
R Console
> y = c(9,8,5,7,6)
> y
[1] 9 8 5 7 6
> order(y)
[1] 3 5 4 2 1
>
> order(y, decreasing = TRUE)
[1] 1 2 4 5 3
> |
```

And, you can see here this is here, the screenshot of the same outcome. And, now let me try to first show you these things on the R console and then I will try to move forward. So, let me try to show you this order here also so, that you can also see what is the difference right. So, I try to write down here, order of this here y. So, y here is 8, 5, 7, 2 like this, right.

So, if you try to see here this is now here 4 2 3 1. So, this is here 4 2 3 1, but when you try to sort it the values of why this was 2 5 7 8. And, similarly if you try to use here the command is equal to here decreasing is equal to here FALSE. Then, you can see here this is the same outcome, right.

But, in case if you try to make this option decreasing is equal to TRUE, then it will be just reverse. It is 4 2 3 1. Now, this will become here 1 3 2 4 or 4 then 2 then 3 and then 1 in the reverse order. So, this is how we try to sort and order the values.

(Refer Slide Time: 16:56)

Mode ✓

Every object has a mode. ✓

The mode indicates how the object is stored in memory: as a

- ✓ number, ✓
- ✓ character string, ✓
- ✓ list of pointers to other objects, ✓
- ✓ function etc. ✓

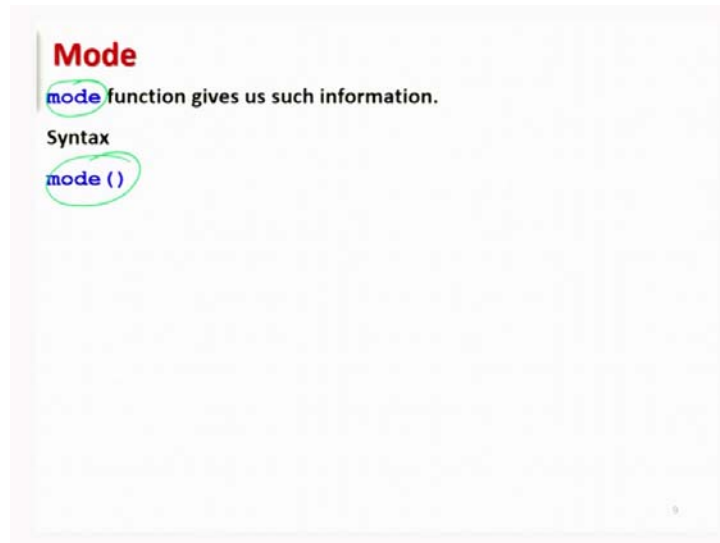
*mean
median
mode* /

So, once again the more important part here is that you have to understand number 1 what is the difference between sorting and ordering in R software and, how these functions are going to be executed in the R software. And, one request please do not get confused that sorting and ordering they are the same thing in the R software. And, they are going to produce the same outcome which is not happening here, I have shown you.

Now, after this I come to another option here, another topic here which is about mode. About mode, we have briefly learnt these things. If you recall that in the beginning of the lectures, in the first few lectures we had talked about the mode. And, I had told you very clearly at that time itself that with the name mode, do not get confused with the mode that is used in statistics. Like as mean, median or mode which are used to measure the central tendency of the data, right.

So, this is not that mode. Remember, this is like modes of transportation, modes of transportation are say train, air, road etc. So, that is this mode. So, I would just like to give you here a brief overview of this mode here, because now we have learnt some more topics in this. So, every object has got a mode and the mode indicates how the object is stored in the memory. For example, stored as a number, characteristic string, list of pointers to the object or function or there are couple of other things also, right.

(Refer Slide Time: 18:33)



So, in order to know the mode of the object you know the command here is mode; mode and you have to write down here mode. And, and inside the parentheses you have to write down the data vector or the object of which you want to find out the mode.

(Refer Slide Time: 18:48)

Object	Example	Mode
Number	1.234	numeric
Vector of numbers	c(6, 7, 8, 9)	numeric
Character string	"India"	character
Vector of character strings	c("India", "CANADA")	character
Factor	factor(c("MP", "UP"))	numeric
List	list("India", "CANADA")	list
Data frame	data.frame{x=1:2, y=c("India", "CANADA")}	list
Function	print	function

So, for example, if you take a number like as here 1.234, then its mode is numeric. And, when you are trying to take a data vector in which all the values are some number. This is essentially a vector of number. So, this is also going to have the mode as numeric. In case, if you try to take a word like India and you try to write down within the double quotes, then this is actually a character string and the mode of this is character.

And, similarly if you try to take a data vector consisting of this character string like as India and say CANADA, you can see here; then yeah this is again a vector of character string. So, its mode is again character. Now, similarly if you try to take here say here factor. So, factor is such a thing we have not done up to now. But, factor are the thing which tries to convert or assign a number to some character, that is one of the very simple definition I can explain you at this moment. And, in the next couple of lectures we are going to talk about the factor.

So, if I try to write down here character string like as here MP and UP, MP means Madhya Pradesh, UP means Uttar Pradesh and they are the character string. They are the data vector of characters. So, they are going to be character as you have seen here. But, in case if you operate the factor command over this, then this is this factor has a mode which is numeric. So, that is something new which you are doing it and that was my reason that, why I took this topic of mode once again.

After this you are going to learn about one more topic which is about list. So, that we are going to learn very soon. So, if you try to write down some character strings inside the double quotes within the bracket and you try to use here list. Then, this is actually called as also a list and this mode is also list, list. And, similarly we are going to learn about one more aspect in the forthcoming lecture, this is about data frame, right.

So, data frame is something like you can think as a spreadsheet; spreadsheet in which you try to arrange the data in rows and columns, right. So, you know that in spreadsheets you can arrange the numerical data as well as character data. One software which can provide you the spreadsheet is Microsoft Excel software. So, and in usual words we call it as an Excel file.

So, in Excel file you can give the names as well as number, you can also do the mathematical operations also. So, that is why this data frame is the framework in which you can understand in a very simple language that we are going to handle the spreadsheet. So, this is called here as a data frame and its mode is also here as a list.

And, after this we have handled many such functions for example, print, sum etc. So, these things they are the function and their mode is function, right. So, that is what you have to keep in mind so, that in the forthcoming lectures whenever we are trying to do it, you can understand them.

(Refer Slide Time: 22:04)

```
Mode  
Example  
> mode(2.432)  
[1] "numeric"  
  
> mode(c(3,4,5,6,7,8))  
[1] "numeric"  
  
> mode("India")  
[1] "character"  
  
> mode(c("India", "CANADA"))  
[1] "character"
```

So, just for the sake of example, I try to show you here that if you try to find out the mode of 2.432, this will come out to be numeric. In case, if you try to take the mode of a data vector consistent of only the numerical value 3, 4, 5, 6, 7, 8, then it is numeric. And, if you try to take here some character like as India, then its mode is coming out to be character.

And, if you try to take a data vector of this character string and try to find out its mode, say I try to consider here two strings India and CANADA. This comes out to be here as a character right and this is here the screenshot.

(Refer Slide Time: 22:42)

```
Mode:  
Example  
> mode(factor(c("UP", "MP"))) )  
[1] "numeric"  
  
> mode(list("India", "USA"))  
[1] "list"  
  
> mode(data.frame(x=1:2, y=c("India", "USA")))  
[1] "list"  
  
> mode(print)  
[1] "function"
```

And, similarly if you try to take the factor of characters like as UP, MP then its mode comes out to be here numeric. And, if you try to consider here list say India and USA, there are two strings, there are two characters. Then, its mode will also come out to be here as a list. And, if you try to consider here the mode of a data frame, that I will try to show you that what is this thing, then this will also come out to be here as a list. And, if you try to find out the mode of a function like print for example, it will come out to be here as a function right.

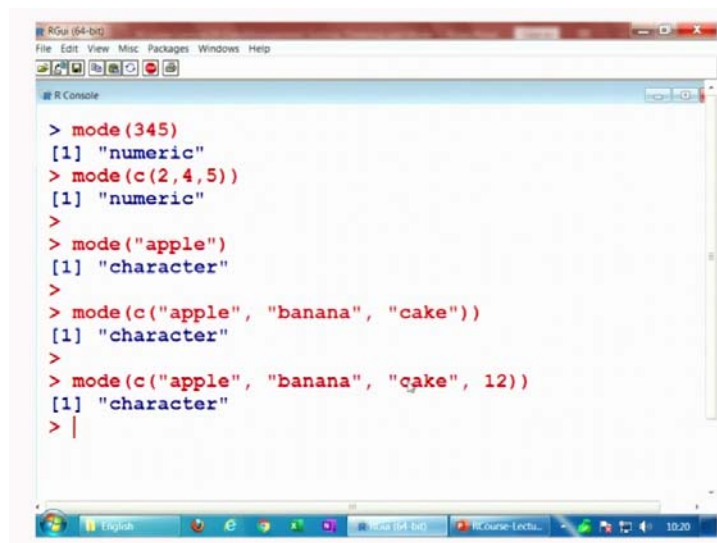
(Refer Slide Time: 23:19)



```
Mode  
> mode(factor(c("UP", "MP")))  
[1] "numeric"  
> mode(list("India", "USA"))  
[1] "list"  
> mode(data.frame(x=1:2, y=c("India", "USA")))  
[1] "list"  
> mode(print)  
[1] "function"
```

So, this is here the screenshot of the same operation. And, now I try to show you these operations on the R console so, that you can understand them very easily, right.

(Refer Slide Time: 23:29)



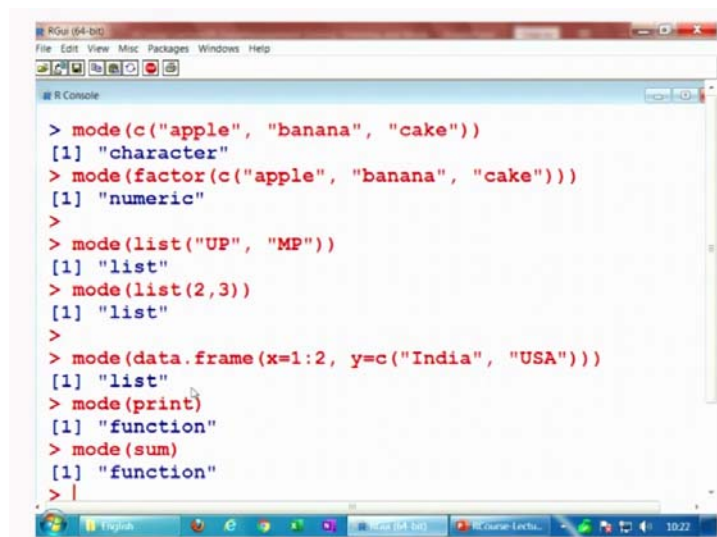
```
RGui (64-bit)  
File Edit View Misc Packages Windows Help  
R Console  
> mode(345)  
[1] "numeric"  
> mode(c(2,4,5))  
[1] "numeric"  
>  
> mode("apple")  
[1] "character"  
>  
> mode(c("apple", "banana", "cake"))  
[1] "character"  
>  
> mode(c("apple", "banana", "cake", 12))  
[1] "character"  
> |
```

So, if I try to find out here the mode of some number here 3 4 5, you can see here this is numeric. And, if I try to find out the mode of a data vector say 2, 4, 5 etc. This will also come out to be here numeric. Now, in case if you try to find out here the mode of a character like here apple, right. So, you can see here this is a character.

And, in case if you try to take the data vector of the character. Suppose, if I take it take here see here apple and say here banana and say here cake right. You have to put all these things inside the double quotes so, that R can understand that these are your strings. So, this is also a character.

Now, if I add here one more value here like here as a 12. So, 12 is a number. So, what do we expect what will happen, what will be the mode of this data vector which has apple, banana, cake and then number 12? So, this will again be a character. Why? Because, once in a numerical vector if you try to add any character string, the mode will become correct because you cannot do the mathematical operation on it right.

(Refer Slide Time: 25:01)



```
> mode(c("apple", "banana", "cake"))
[1] "character"
> mode(factor(c("apple", "banana", "cake")))
[1] "numeric"
>
> mode(list("UP", "MP"))
[1] "list"
> mode(list(2,3))
[1] "list"
>
> mode(data.frame(x=1:2, y=c("India", "USA")))
[1] "list"
> mode(print)
[1] "function"
> mode(sum)
[1] "function"
> |
```

So, similarly if you try to take here the these examples here like as here factor. So, if you try to take care of the same, let me try to clear the screen. So, you can see here that mode of this data vector is a character. But, when you try to find out the mode of the factor of the same variable here, you can see here what happens. This comes out to be numeric, right.

So, we will try to understand what is the operation of a factor so, but here at this moment you can see that factor is trying to convert the nature of the mode which is from character to a numeric. And, then I will try to go for here say here list command, say here mode of say here list, say here UP and say here MP. And, you can see here this is here a list, right. And, even in the list if you try to remove the character and if you try to put here some values also, you can see here it will remain as a list. It is not going to change as a numeric or a character.

Similarly, if you try to take here the mode of our data frame so, you can see here this is also here a list. And, if you try to find out here the mode of a function like as here print. So, you can see here this is function and if you try to take care any other say this function like as here sum, you can see here this is again a function.

So, now we come to an end to this lecture and you can see here that this was a pretty simple lecture. But, the main objective is that it was very important to understand what is the difference between sorting and ordering, right. People try to take it exactly in the same way and yeah they have the similar meaning also right. But, the way R is trying to consider it that is more important for you to understand, because you have to do all the operations in the R software.

And, you have to give the instruction to the R software, but what is to be done only you know. So, that is why these operations are very important to understand that whether you want to find out the values or whether you want to find out the values of their locations. And, based on that you have to choose the sorting and ordering commands.

And, similarly for the mode although we already had done it, but now you can see here, I have introduced here some more types of modes which we are going to handle in the next couple of lectures. So, I thought that ok, let me give you here this brief idea so, that you are comfortable at this moment.

Yeah, at the moment you may not be able to understand what is the difference between list, factor or say data frame. But, that is my promise that I will try to clarify all these concepts very soon. So, now my request is that why do not you try to take some numerical values and try to operate with this sorting, ordering and mode operations and try to see how do you get an outcome.

But, the main thing is for example, if I take the operation of mode. So, try to take some value, try to first think what do you expect the mode is going to be like this and then you try to execute it. So, if your thought process and the outcome of the software matches; that means, you are done. You have understood that how R is going to think. So, you try to think, try to practice and I will see you in the next lecture.

Till then goodbye.