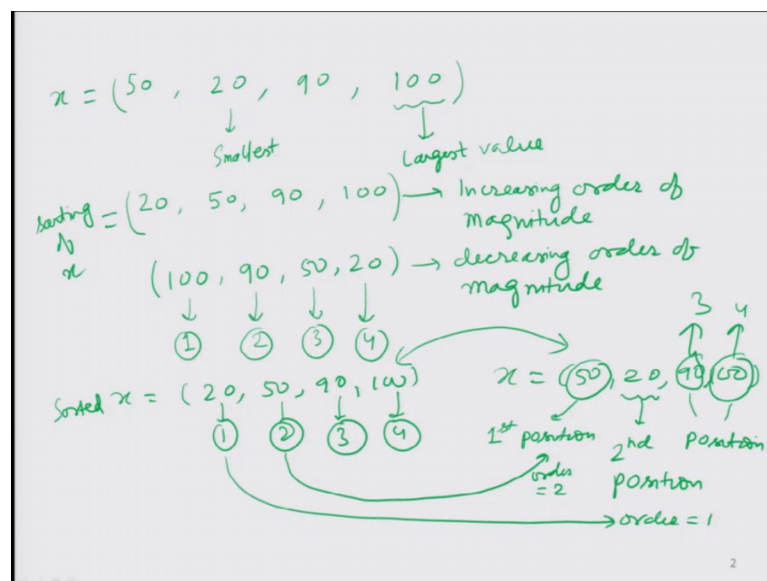


**Introduction to R Software**  
**Prof. Shalabh**  
**Department of Mathematics and Statistics**  
**Indian Institute of Technology, Kanpur**

**Lecture - 17**  
**Sorting and Ordering**

Welcome to the next lecture on the course Introduction to R Software. In this lecture we are going to talk about two options that are available for the data management and data generation this is sorting and ordering. That means how to sort a data and how to order the data or how to know what is the order of the data. These are the questions which we are going to answer in this lecture, but the first question comes what is sorting and what is ordering

(Refer Slide Time: 00:50)



So, suppose I take suppose several value say here 50, say here 20, say here 90, and see here 100. You can see here that these are some values which are arranged in some vector here say called is here  $x$ .

Now, when I say sort, sort means rearranging all those values in an increasing order of magnitude or decreasing order of magnitude. For example, if you try to see here what is the smallest value among this thing this is here the 20 this is the smallest value, and what is the maximum value here? This is here 100. So, this is the largest value

Now there are 2 more values which are contained between 20 and 100, and in case if you try to rearrange these 4 values in the increasing order of magnitude, then this can be written as say here 20, then 50 and then 90 and then 100. So, this is increasing order of magnitude.

So, I would say that we have sorted the data in x and the sorting of data in x results in this vector 20, 50, 90 and 100 similarly this sorting can be in the reverse order also; that means, the largest value at the first position say 100, then the second is smallest value that is 90 then the third is smallest value 50 and the smallest value that is 20.

In this case what we are trying to arrange the data in the decreasing order of magnitude. So, this is called sorting. Now if you try to see when we are trying to do the sorting we are trying to put the smallest or the maximum value at certain position for example, you can see here that this is the position number one, when we are trying to arrange the data in the decreasing order, and this is the position number 2 and for 50, this is position number 3 and for 20 this is position number 4. And similarly when we are trying to sort the data in say increasing order of magnitude, then we have seen that this data is coming here and in this case this is as the first place, this is at the second place, this is at the third place and this is at the fourth place.

So, when we say that we want to order the data; that means, we need to know; what is the order value of a particular value inside the vector. For example, in this case if you try to see here this is the sorted data and this is here the original data 50, 20, 90 and here 100.

Now, in case if you try to compare this and this what happens here? Here the 20 is coming at the second position, but remember this is the second position, but the order of 20 is coming from here order equal to here 1 and similarly if you try to take here another value here 50 then 50 is coming in this vector at the first position.

But the order of this 50 is coming over here this is here 2. So, this is here order equal to 2 and similarly for this 90 value here the order here is 3 and for. So, this here 100 value the order here is 4. So, in this case the positions of 90 and 100 are coinciding with the original order in the x vector. So, this is what we mean by sorting and ordering now how to get this thing done in this R.

(Refer Slide Time: 05:43)

**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, ...)
```

**x**                      Vector of values to be sorted

**decreasing**          Should the sort be increasing or decreasing

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

3

So, there is a function what we call as here say sort s o r t. This is a function which sorts the values inside a vector depending on whether we want to arrange the values in the ascending order or the descending order when you use the sort function which is embedded in the base package of r this takes the ascending order as default option. So, unless and until you specify that you want to sort the data in the descending order or the decreasing order of the magnitude, the usual option will always be in the ascending order.

So, this is the syntax first you try to write s o r t and then the values which are contained inside the vector x, and now you have to say here whether decreasing is false or decreasing is true and remember one thing that true and false these are the 2 reserved words which are the logical value and based on that it will try to give you the outcome right.

I know there is another option that in case if some values in the x vector are missing they can also be handled using the command n a dot l a s t that is NA dot last; that means, there is missing values are going to be denoted by here NA and in case if I say that this statement is true; that means, the missing values in the data are put in the last and if I say this value is false then these values are put in the first that mean first the missing value occurs or the missing value are occurring at the end of the x vector and if I say NA; that

means, there is no missing value and we and the control of r will not handle say any missing values and you will simply ignore it.

So, now, let us try to take some example and we try to understand how this r is working.

So, I try to take a vector.

(Refer Slide Time: 08:11)

**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 calculations:

$8 \quad 5 \quad 7 \quad 6$

$5 \equiv \min(8, 5, 7, 6) \equiv 5$

$6 \equiv \min(8, 7, 6) \equiv 6$

$\min(8, 7) \equiv 7$

$8$

$\max(8, 5, 7, 6) \equiv 8$

$\max(5, 7, 6) \equiv 7$

$\max(5, 6) = 6$

$5$

Of 4 values 8 5 7 and 6 which are arranged in a vector it is combined by this c and this vector is over here y. So, you can see here this is my here y vector.

Now, when I try to say here sort; now let us try to see what is this sort command over this vector y does. One thing you have to keep in mind that here the default value is to arrange the values in ascending order. So, now, if you try to see here we have here 4 values 8 5 7 and 6 when I say sort y the control comes to this data set and it find which is the minimum value. So, in one can see here that the minimum value here is 5. So, it keeps it at the first place now from the remaining value 8 7 and 6, it against try to find out the next minimum value and this comes out to be 6. So, 5 was essentially the minimum value between 8 5 7 and 6 and 6 is the minimum value with the remaining value 8 7 and 6. So, the answer comes out to here 5 and here 6.

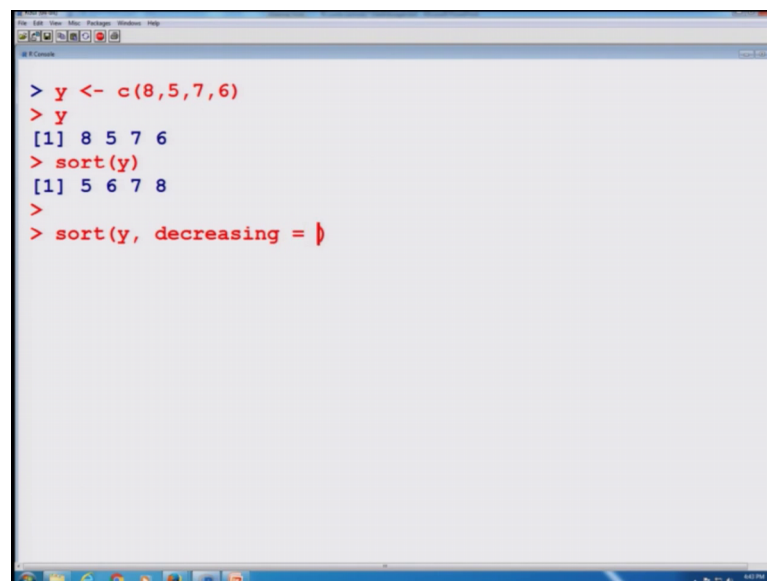
Now, the remaining values I have here these are here 8 and 7. So, so again r tries to find out the minimum value between 8 and 7 and this comes out to be here 7. And now there is only one value which is left which is the maximum value among all the given values

and this here is 8 and this is the same sequence 5 6 7 and 8, which is given here by this sort and suppose I want to have this sorting in the decreasing order. So, what I have to do I have to use the same command sort or y, but I have to write down here decreasing is true; so decreasing all in a small letter equal to true, true in capital letter.

Now, what will happen here that it will try to take up these values 8 5 7 and 6 and first it will try to find out the maximum of this thing. The maximum here is 8 then out of the remaining values 5 7 and 6, it will try to find out the maximum value and this comes out to be here 7. Then from the remaining value here 5 and 6 this will try to find out the maximum value and this comes out to be a 6 and the remaining value is now here 5. So, the sequence in that decreasing order becomes 8 7 6 and 5 and which is denoted here.

So, this is how the sorting happens, and we are kind of side to see how this can be done over the r console. So, let me try to take this data over here.

(Refer Slide Time: 11:27)



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

Why so, you can see here why is this thing and then if I try to say here sort y this gives me the same output and when I try to say sort y we are decreasing equal to true it gives me this outcome. So, now, I just try to come back to a slide and we see here that in the next slide I have given the screenshot.

(Refer Slide Time: 12:02)

**Ordering**

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

Syntax

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

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

<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 NAs. If TRUE, <u>missing values in the data are put last;</u> if FALSE, <u>they are put first;</u> if NA, they are removed.

6

Now, you try to consider the ordering of the data. The ordering of the data can be done using the syntax or a function say `order`. This `order` function actually sorts a variable according to the order of the variables and the syntax here is simply you have to give the order and the values which are contained in the vectors say here `x`, and now you have to see here whether the this arrangement has to be in the decreasing order or in the increasing order that can be controlled by writing here `false` or `true` the default is decreasing.

And similarly as we have done in the case of sorting, there are some other option like is `na.last` it is trying to give us an idea that how the missing value should be entertained means, if I try to say here this is equal to here `true` then `true` means missing values in the data are put in the last; that means, end of the vector and if I say here `false`; that means, all the missing values are put in the first position and if I say `na.last` is equal to `NA` that means, they have been removed and the control will not consider `NA` of the missing value in the computation.

(Refer Slide Time: 13:33)

**Ordering**

**Example**

```
> y <- c(8,5,7,6)
> y
[1] 8 5 7 6
```

**Increasing Order**

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

**Decreasing Order**

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

8	5	7	6
place 1	place 2	place 3	place 4

increasing

Addresses: 2, 4, 3, 1

decreasing order

8, 7, 6, 5

place 1, place 3, place 4, place 2

So, now let us try to take some examples and try to understand. Now I try to take here the same data set say  $y$  which was containing 4 values 8 5 7 and 6, that we can see here this is the values of here  $y$  and when I say order of  $y$  what do you think should happen means we have seen that here I have here 4 values 8 5 7 and 6, there are 4 values. So, we have to find out their position right.

Now, there are 2 types of position one is the place value. The place value here is this is my here place number 1 this is here place number 2 this is place number 3 and this is here place number of 4 it is something like if you try to visualize this thing suppose I have a sill which is something like this, and it is saying that this is the address of the first value, this is the address of the second value, this is the address of the third value and this is the address of the fourth value.

But now when I try to arrange it in the increasing or decreasing order we have seen in the last example that when we want to arrange these values in the increasing order then these values will come out to be 5 6 7 and 8.

Now, if you try to see; what is the address of value 5 here. We try to write down the address the address is coming from here this is saying that 5 number value is at the second order or at place number 2 what about here 6? 6 is at the place number here 4, what about here is 7? 7 here is at place number here 3 and what about here 8? This 8 here is here somewhere here and the address of the place of 8 is 1.

So, now I should get here the vector here 2 4 3 and 1 and which is trying to denote the addresses of say here 8 5 7 and here 6 and this is what we are getting over here and similarly when I am saying that I want to have this order in the decreasing sequence and if I write here instead of false I right here decreasing way equal to true here, then just the opposite of this happens.

These values 8 5 7 and 6 when they are arranged and say here decreasing order then these values will it come out to be say here first the maximum value 8, then the second largest value 7, then the third largest value 6 and lastly the smallest value which is here 5.

Now, you simply have to see; what are their addresses in the place. So, let us try to copy their addresses from this place which I am trying to denote here by star and encircle, if you try to see here the address of value 8 here is place 1 and the address of value 7 is here place say here 3, the address of value 6 is place 4, and the address of value 5 is here place 2 using the addresses which I have denoted above by star.

So, now I should get here the outcome here I say the first value should be here 1 then 3 then 4 and then here 2 and this is precisely what we are getting over here. So, you can see here that whenever we are trying to use the function sort, sort is giving us the value which are arranged in the increasing or decreasing order of the magnitude of the values whereas, when we are trying to say the order then order is trying to give us the addresses of these values in the original vector.

Let us try to do this thing on the r console and see what really happens. So, here you can see here we already have a vector that we defined here as say here y, and when I try to find out the order of here y this is your 2 4 3 1, this is the same outcome that we have obtained and in case if I try to do the reverse of this thing, then we get it here 1 3 4 2 and you can see here that in this operation order of y, when the values are arranged in ascending order and when the values are arranged in the descending order they are just the opposite thing.

So, whatever order I am giving here as a 2 4 3 1, that is simply reversed in the second situation as a 1 3 4 2. And here I have placed the screenshot of the same operation that we have obtained here.



So, now we stop here and you take some time to practice the sequence orders and sorting. Try to take some example from the book, from the assignment and some other issues. We will be discussed in the next lecture, till then goodbye.