

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

Lecture - 26
More Sequence and Other Operations

Hello friends. Welcome to the course Foundations of R Software and you can recall that in the last lecture we initiated a discussion on the sequence. And in this lecture, we are going to continue on the same topic and we will try to learn some more types of operations which can be done for the sequences. Do you remember that many times in the earlier lectures whenever I wanted to write down the values like 1, 2, 3, 4, 5, then I would try to write 1 colon 5.

And at all those places I was always asking you ok, we will try to do it in the future lecture. So, this is the lecture where we are going to learn about such operations. So, as I get in the last lecture that I took a couple of examples and through those examples I tried my best to explain you that how R will react to those commands, how R is going to give an output based on how we are trying to write the command.

So, in this lecture also we will try to take a couple of examples and we will try to understand that how R works for these sequences. So, let us begin our lecture.

(Refer Slide Time: 01:40)

Sequences

A sequence is a set of related numbers, events, movements, or items that follow each other in a particular order.

The regular sequences can be generated in R.

Syntax

```
seq()  
seq(from = 1, to = 1,  
by = ((to - from)/(length.out - 1)),  
length.out = NULL, along.with = NULL, ...)
```

Help for `seq`

So, as we had understood in the last lecture that a sequence is a set of related number events, movements or items that follow each other in a particular order and we had explored the application of this command `seq`, right, and then I had asked you that if you need to have more information you can look into the help, right.

(Refer Slide Time: 01:59)

```

Sequences
[ ] Continuous sequences with constant unit increment and decrement

> 1:10
[1] 1 2 3 4 5 6 7 8 9 10

> 10:1
[1] 10 9 8 7 6 5 4 3 2 1

> 5:15
[1] 5 6 7 8 9 10 11 12 13 14 15

> 15:5
[1] 15 14 13 12 11 10 9 8 7 6 5
  
```

Now, today I am going to take here another aspect, if you try to write down the earlier command and suppose if I want to write down here sequence from equal to 1 up to here to equal to 2; then what will happen? This will give you a value like 1 2 3 4 up to 10, right and similarly if you try to write down here instead of earlier form 1 to 10 if you try to write down here from 10 to 1, it will give you a sequence like 10 9 8 7 etc. up to 1.

So, now I give you here an alternative to this `seq` command, right. So, I can write the command here to generate the numbers from 1 to 10, like 1 colon 10. So, this 1 the first value is going to give you the option to write the starting point just like from and the next value after colon sign, that is going to give you the option to write the option for the 2, right.

So, if you try to write in this particular way, then it will generate a continuous sequence with constant unit increment and yeah if you try to give the decreasing value then it will give you a decrement, right. So, if you try to see here this will give you here the value like 1 2 3 4 5 6 7 8 9 10, right. And similarly if you want to generate a decreasing

sequence starting from 10 and ending at 1, then you can give here a command like this here 10 colon 1. So, this will start at 10 and then it will go up to 9 8 up to here 1.

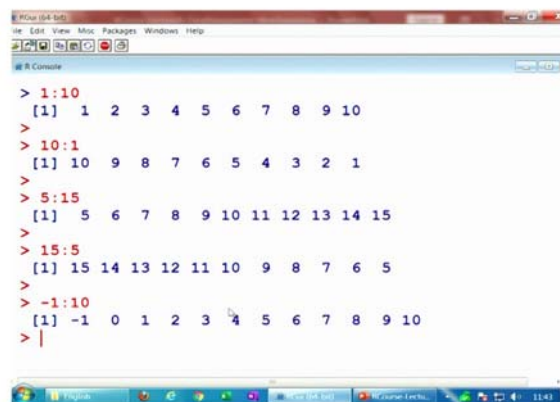
So, this is the outcome that you are going to get, right. So, one question comes here, what is the difference between this type of command and the earlier command based on seq. So, you have seen that in s e q you had many more options, right, but in this type of example you can simply generate the continuous sequence with constant unit increment or decrement only. So, that is the thing.

So, obviously, sequence is a more general command and you have more control over the values than in this present command, but anyway our objective here is to learn both the options and then try to use accordingly then whatever you want, right, ok. So, similarly if I try to take here one more example to show you that these numbers, this from and to they cannot only begin with like here 1 or some other value or they are ending only at 1. You can choose any value actually.

Suppose I want to create here a sequence which is beginning at 5 and it is ending at 15. So, the sequence would be like 5 6 7 up to here 15. So, I have to write down here 5 colon and if you want to start the sequence at 15 and then you want to end at 5. So, this will be 15 14 13 12 etc. So, that is going to be sort of decrement I mean.

So, you are not giving here then option like y is equal to -1 or -2. But once you are trying to write down the value at from which is higher than the value at to then it is going to be a automatic decrement. So, this I try to write down here as a 15 colon 5.

(Refer Slide Time: 05:12)

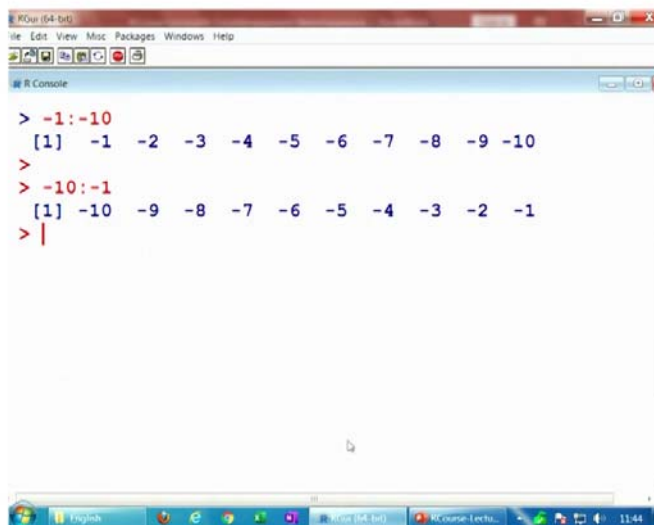


```
R Console
> 1:10
[1] 1 2 3 4 5 6 7 8 9 10
> 10:1
[1] 10 9 8 7 6 5 4 3 2 1
> 5:15
[1] 5 6 7 8 9 10 11 12 13 14 15
> 15:5
[1] 15 14 13 12 11 10 9 8 7 6 5
> -1:10
[1] -1 0 1 2 3 4 5 6 7 8 9 10
> |
```

So, before I move forward let me try to show you these options on the R console so that you get more confident and then I can move further. Suppose if I want to generate here a sequence from 1 to 10, you can see here this is like this and if I want to generate here a sequence from 10 to 1, you can see here this can be done here like this.

And if you want to generate here another sequence from say 5 to 15, you can see here this is the way I can write it down, right. And after that if you want to generate a sequence which is beginning from 15 and going up to 5, you can write down here like this, right. And if you try to see here what happens if you try to take care the negative numbers say from -1 to say here 10 what happens let me try to see yes it also considers the minus number.

(Refer Slide Time: 05:54)



```
R Console
> -1:-10
[1] -1 -2 -3 -4 -5 -6 -7 -8 -9 -10
>
> -10:-1
[1] -10 -9 -8 -7 -6 -5 -4 -3 -2 -1
> |
```

In case if you try to now give here say -1 to here -10, what do we expect what will happen? This will go from -1 to -10 and if you try to give here from -10 to here -1, you can see here this is also now working, right.

(Refer Slide Time: 06:12)

```
Sequences  
□ Continuous sequences with constant unit increment and decrement  
  
> -1:-10  
[1] -1 -2 -3 -4 -5 -6 -7 -8 -9 -10  
  
> -10:-1  
[1] -10 -9 -8 -7 -6 -5 -4 -3 -2 -1  
  
> -5:-15  
[1] -5 -6 -7 -8 -9 -10 -11 -12 -13 -14 -15  
  
> -15:-5  
[1] -15 -14 -13 -12 -11 -10 -9 -8 -7 -6 -5
```

So, now after this you will have some confidence and I will try to explain you these things. So, you can see here that is the same thing I just shown you on the R console itself, that if you try to take here from and to as negative values then even then it gives you a sequence -1, -2 up to here -10.

And if you try to take here the -10 to -1; that means, -10 colon -1 then the value begins at -10 and then it will go up to -1 and you get here this sequence. Now, then if you try to take any intermediate sequence also for example, you are not starting or ending at -1, but you are trying to start at -5 and you want to end it at -15.

So, this will start with -5 -6 -7 etc. and then it will end at -15, which is here like this. And then similarly, if you want to reverse that you want to start at -15 and then you want to end it at -5 that also you can give here, -15 colon -5 and it will give you here a sequence which is beginning at -15 and it is ending at -5, right.

Now, I try to give you here an example that you can begin with any fraction number also and then the main thing is that you have to observe that in such a case what R is trying to do. The point at which it is going to finish the sequence what happens to that. So, from this objective I try to take here one more example here, in which this sequence is beginning at 1.23 and it is ending at 10.

So, now you can see here what will happen it will begin at 1.23 and then the default increment is plus 1 this will go to 2.23 3.33, 23, 4.23 etc. then finally, it will come to 7.23, 8.23 from here 9.23 and after this the number will be here 10.23, but the sequence has to go only up to here 10. So, this 10.23 will not be considered and the sequence will stop at 9.23.

So, the sequence is going to stop at a value which is less than or equal to the value which is given after the colon sign for example, it is here 10. So, that is why the sequence is stopped here at a value which is less than or equal to 10. So, that is what you have to keep in mind. Now, similarly as I have taken here an example where the second number is an integer.

So, if I want to take it any other number for example, if I want to create here a sequence like 1.23 and which is ending at 10.54, the logic is going to be the same as I explained you, right. So, you can see here the sequence will start here with a constant unit increment 1.23, 2.23, 3.23, 4.23 up to here, then 8.23, 9.23 and from 9.23, it will come to 10.23 and after that the value will be 11.23.

But you can see here that this value is greater than the last value 10.54, which is given here. So, that is why the program will stop here and it will not go to 11.23, but instead of going up to 10.54 it is stopping at 10.23. So, this is how this R works. Similarly, if you try to take here one more example, where the value which is the starting point that is greater than the value which is ending point. So, I try to create here a sequence like 10.54 to 2.23 like this, 10.54 colon 2.23.

So, now, what will happen? The sequence will begin at 10.54; 10.54 to 9.54, 9.54 to 8.54 and so on. It will continue it will come to here 3.54 to 2.54 and after this the value is going to be 1.54. But this value what will happen here? You have to compare this 1.54

with this 2.23. So, 2.23 is coming earlier and after that this 2.54 is coming. So, it will stop after this, right.

(Refer Slide Time: 10:40)

Sequences
□ Continuous sequences with constant unit increment and decrement

> -1.23:-10
[1] -1.23 -2.23 -3.23 -4.23 -5.23 -6.23 -7.23 -8.23 -9.23 -10.23 X

> -5.23:6
[1] -5.23 -4.23 -3.23 -2.23 -1.23 -0.23 0.77 1.77 2.77 3.77 4.77 5.77 6.77 >6 stop

So, similarly if you try to take here one more example, where I try to take the negative values in the similar example what I just shown you. Suppose I try to take here a sequence which is beginning from -1.23 and it is ending at -10. So, what will happen here? It will begin at -1.23 -2.23, then -3.33 up to here it will come to -9.23 and after that the next value is -10.23, but now this value up to where the sequence has to end is -10.

So, this value will not be executed and the sequence will start at a -1.23, but it will finish at -9.23, that is smaller than the value of -10, right ok. And similarly if you try to take here a combination where the point which is starting that is a negative value and the place where it is ending that is a positive value.

So, what will happen here? This will come begin as -5.23, then -4.23 -3.23 and then somewhere it will become here positive also to 0.77 and then it will continue. And finally, it will come to 5.77 and after this the next value is going to be 6.77, but this value is greater than the value of 6. So, the program will stop here and this will not be executed. So, you can see here this is how this program is going to work, right.

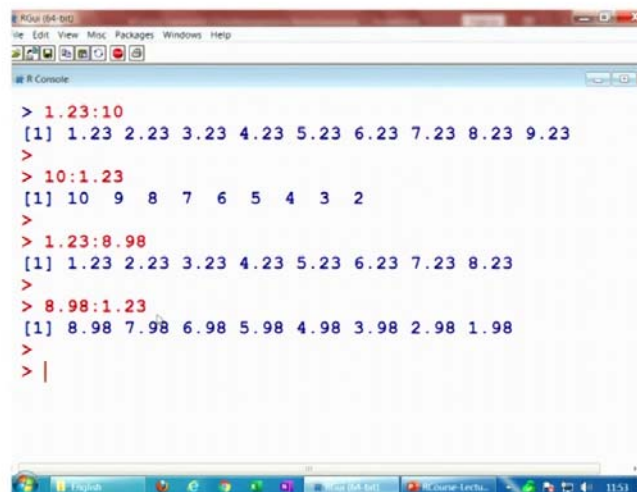
(Refer Slide Time: 12:02)

Sequences

```
> 1.23:10
[1] 1.23 2.23 3.23 4.23 5.23 6.23 7.23 8.23 9.23
> 1.23:10.54
[1] 1.23 2.23 3.23 4.23 5.23 6.23 7.23 8.23 9.23 10.23
>
> 10.54:2.23
[1] 10.54 9.54 8.54 7.54 6.54 5.54 4.54 3.54 2.54
>
> -1.23:-10
[1] -1.23 -2.23 -3.23 -4.23 -5.23 -6.23 -7.23 -8.23 -9.23
>
> -5.23:6
[1] -5.23 -4.23 -3.23 -2.23 -1.23 -0.23 0.77 1.77 2.77 3.77 4.77 5.77
>
> |
```

And this is here the screenshot of all these operations. So, look let me try to show you these things on the R console here, we before we try to move further, right. So, if you try to see here I try to take here the same example here.

(Refer Slide Time: 12:15)

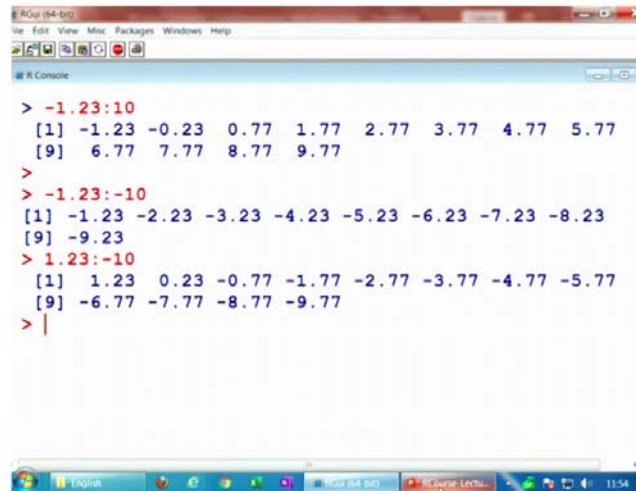


```
> 1.23:10
[1] 1.23 2.23 3.23 4.23 5.23 6.23 7.23 8.23 9.23
>
> 10:1.23
[1] 10 9 8 7 6 5 4 3 2
>
> 1.23:8.98
[1] 1.23 2.23 3.23 4.23 5.23 6.23 7.23 8.23
>
> 8.98:1.23
[1] 8.98 7.98 6.98 5.98 4.98 3.98 2.98 1.98
>
> |
```

Suppose I want to create here 1.23, 2 here say here 10 this comes here like this the sequence is going to stop at 9.23. And similarly, if I try to take care starting at 10 to 1.23, then it will be like here this 10 9 8 7 up to here 2, right. After this the value is going to be here 1, which is lower than the value of 1.23. So, it will stop here, right.

Similarly, in case if you try to take here this example here like as here when the 2 numbers are both are fraction. So, if I try to take it here 1.23 2 here say 8 point say here 98, you can see here it will go up to 8.23 and then after that it will stop here. And similarly, if you try to take here 8.98 up to 1.23 it will go exactly in the same way, right. It will begin at 8.98, but it will finish at 1.98 because after this the value will come that will be smaller than the value of 1.23, right.

(Refer Slide Time: 13:14)



```
R Console
> -1.23:10
[1] -1.23 -0.23 0.77 1.77 2.77 3.77 4.77 5.77
[9] 6.77 7.77 8.77 9.77
>
> -1.23:-10
[1] -1.23 -2.23 -3.23 -4.23 -5.23 -6.23 -7.23 -8.23
[9] -9.23
> 1.23:-10
[1] 1.23 0.23 -0.77 -1.77 -2.77 -3.77 -4.77 -5.77
[9] -6.77 -7.77 -8.77 -9.77
> |
```

Now, in case if you try to take here some negative values also. So, -1.23 up to here 10, you can see here this is a like this and then if you try to take here both the values to be here negative even then it is beginning at -1.2 ending at -9.23. And even if you try to take here the first value to be positive and the second value to be -1.23 to -10, you can see here this is working because it automatically detects whether the sequence has to increase or it has to decrease.

So, if you try to see I have taken here a similar type of examples and so now, you are going to be confident.

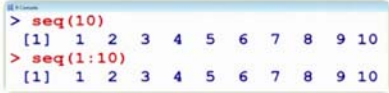
(Refer Slide Time: 13:49)

Sequences

```
> seq(10)
[1] 1 2 3 4 5 6 7 8 9 10
```

is the same as

```
> seq(1:10) seq( from = 1, to = 10 )
[1] 1 2 3 4 5 6 7 8 9 10
```



```
> seq(10)
[1] 1 2 3 4 5 6 7 8 9 10
> seq(1:10)
[1] 1 2 3 4 5 6 7 8 9 10
```

Now, some more basic operations. So, when you try to write down here command like sequence like this `seq` and inside the parenthesis you write the value 10. So, this is going to give you a sequence which is beginning at 1 and it is ending at 10. And this is the same as if you try to write down the sequence here 1 colon 10 or sequence say from equal to 1 to equal to 10 like this.

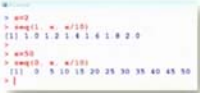
So, they are going to give you the same result. So, I just wanted to show you that what happens so that if you are trying to use it somewhere you know how is the outcome going to be now.

(Refer Slide Time: 14:23)

Sequences

Sequences with a predefined variable and constant increment

```
> x=2 from to by
> seq(1, x, x/10) seq(1, 2, 2/10)
[1] 1.0 1.2 1.4 1.6 1.8 2.0
```



```
> x=50
> seq(0, x, x/10) by = 50/10 = 5
[1] 0 5 10 15 20 25 30 35 40 45 50
```

Second thing whatever is the outcome of a sequence, that can be defined in terms of variables also, that can be controlled by the variable also. For example, when you are trying to give here the value like from and to etc. So, we are trying to give here some fixed values, but these values can also be some variable, right. So, this is really going to help you when you are trying to write down the programs.

So, for example, if you try to see here, I try take I take an example here that I consider x equal to 2, x is some variable and then I try to define here a sequence in terms of this variable. So, sequence from 1 to here x. So, whatever x we are going to give as input that is going to be used here.

And then after this the third value which I am writing here I just want to show you that although I am not writing here from to and by, but in the sequence command if you try to write down the 3 values separated by comma, then the first value is going to be from second value is going to be to and third value is considered as by that is the default, right.

So, if you try to see here essentially you are trying to write down here that sequence beginning from 1 going up to 2 and then this increment is going to be 2 upon 10, which is going to be here 0.2, right. So, if you try to see here this will give you a sequence like 1, 1.2 up to here 2. And similarly, if you try to change the value of the see here x to be here 50 and you want to have a sequence which is beginning from 0 going up to 50 and then the by that is the increment is here 50 upon 10, which is equal to here 5 units.

Then it will become here the sequence will begin from 0 it will come at the increment of 5 10 and it will finish at 50, right. So, you can see here these are the very simple operation which are based on your knowledge which you have learnt up to now, you know how to define a variable and now you are trying to learn how to define a sequence and now you are trying to combine them together.

(Refer Slide Time: 16:23)

```
Sequences
Outcome of sequences can be stored.
x = seq(1, 50, 1/2)
x
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0 8.5 9.0
[18] 9.5 10.0 10.5 11.0 11.5 12.0 12.5 13.0 13.5 14.0 14.5 15.0 15.5 16.0 16.5 17.0 17.5
[35] 18.0 18.5 19.0 19.5 20.0 20.5 21.0 21.5 22.0 22.5 23.0 23.5 24.0 24.5 25.0 25.5 26.0
[52] 26.5 27.0 27.5 28.0 28.5 29.0 29.5 30.0 30.5 31.0 31.5 32.0 32.5 33.0 33.5 34.0 34.5
[69] 35.0 35.5 36.0 36.5 37.0 37.5 38.0 38.5 39.0 39.5 40.0 40.5 41.0 41.5 42.0 42.5 43.0
[86] 43.5 44.0 44.5 45.0 45.5 46.0 46.5 47.0 47.5 48.0 48.5 49.0 49.5 50.0
y=2*x
y
[1] 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
[22] 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43
[43] 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64
[64] 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85
[85] 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
```

So, these concepts are going to be really useful when you are trying to do the programming. Similarly, just for the sake of here example I try to show you that the outcome of a sequence can also be stored because, here I am trying to take those examples where I can show you the output on the screen in some bigger fonts, right.

So, suppose I try to take here a sequence like a which is beginning at 1. So, this is here from and this is ending at to equal to 50 and then it is increased by 0.5. So, I give it here 1 by 2. So, you can see here this is the outcome starting from here 1 and then ending here at 50 now I try to multiply all the values in the sequence y 2 and I try to store them in a new variable y.

So, I write down here y is equal to 2 into x and then I try to you can see here that all these values are multiplied by this is the value 1, which is multiplied here by 2, this is 2. And similarly, this is the value here 2.5 which is multiplied by 2 this gives you here value 5. So, you can see here that all these values are there and finally, this 50 value is multiplied and this is your 100.

(Refer Slide Time: 17:31)

Sequences
Outcome of sequences can be stored.

```
> x = seq(1, 50, 1/2)
> x
 [1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0 8.5 9.0
 [18] 9.5 10.0 10.5 11.0 11.5 12.0 12.5 13.0 13.5 14.0 14.5 15.0 15.5 16.0 16.5 17.0 17.5
 [35] 18.0 18.5 19.0 19.5 20.0 20.5 21.0 21.5 22.0 22.5 23.0 23.5 24.0 24.5 25.0 25.5 26.0
 [52] 26.5 27.0 27.5 28.0 28.5 29.0 29.5 30.0 30.5 31.0 31.5 32.0 32.5 33.0 33.5 34.0 34.5
 [69] 35.0 35.5 36.0 36.5 37.0 37.5 38.0 38.5 39.0 39.5 40.0 40.5 41.0 41.5 42.0 42.5 43.0
 [86] 43.5 44.0 44.5 45.0 45.5 46.0 46.5 47.0 47.5 48.0 48.5 49.0 49.5 50.0
>
> y = 2*x
> y
 [1] 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
 [22] 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43
 [43] 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64
 [64] 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85
 [85] 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
> |
```

So, you can see here the outcome of the sequence can also be stored in some variable and this is here the screenshot of the same operation, right. So, before I move further let me try to show you these operations on the R console so that you become here more confident.

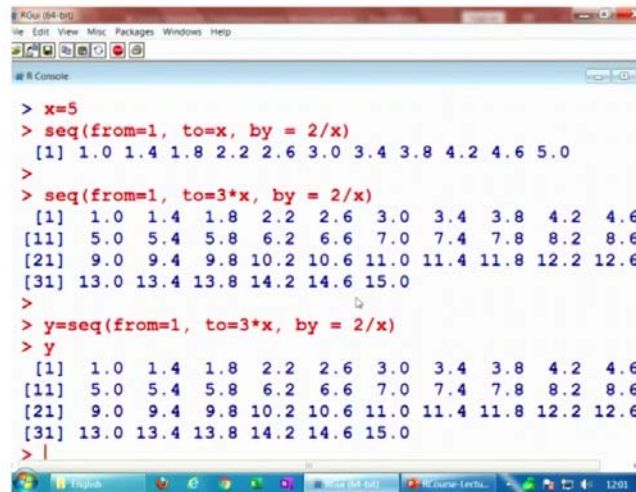
(Refer Slide Time: 17:43)

```
> seq(10)
 [1] 1 2 3 4 5 6 7 8 9 10
> seq(1:10)
 [1] 1 2 3 4 5 6 7 8 9 10
> 1:10
 [1] 1 2 3 4 5 6 7 8 9 10
> |
```

So, now if you try to see here sequence, if I write down here 10. So, this is giving me here the value like this and if I try to write down here sequence 1 colon 10, this is also giving me the same value and if I try to write down here 1 colon 10 this is also give me

the same value. So, yeah you can use whatever you want depending on your need and requirement.

(Refer Slide Time: 18:14)



```
R Console
> x=5
> seq(from=1, to=x, by = 2/x)
[1] 1.0 1.4 1.8 2.2 2.6 3.0 3.4 3.8 4.2 4.6 5.0
>
> seq(from=1, to=3*x, by = 2/x)
[1] 1.0 1.4 1.8 2.2 2.6 3.0 3.4 3.8 4.2 4.6
[11] 5.0 5.4 5.8 6.2 6.6 7.0 7.4 7.8 8.2 8.6
[21] 9.0 9.4 9.8 10.2 10.6 11.0 11.4 11.8 12.2 12.6
[31] 13.0 13.4 13.8 14.2 14.6 15.0
>
> y=seq(from=1, to=3*x, by = 2/x)
> y
[1] 1.0 1.4 1.8 2.2 2.6 3.0 3.4 3.8 4.2 4.6
[11] 5.0 5.4 5.8 6.2 6.6 7.0 7.4 7.8 8.2 8.6
[21] 9.0 9.4 9.8 10.2 10.6 11.0 11.4 11.8 12.2 12.6
[31] 13.0 13.4 13.8 14.2 14.6 15.0
> |
```

Now, after this I try to take here I show you this example that I try to take this command here like this, that I am trying to consider here a variable. So, if you try to see here, I try to choose here a variable suppose here 5 and then I try to give here a command like here like this sequence, which is beginning from. So, here from equal to 1 and then to up to here see here see here x and then y is going to be here say 2 upon say x, you can see here this is the value.

So, x is coming here 5 and then this by is here 2 upon 5, right. So, this is point 4 and even if you want to give here say 2 also you can give it here as a new variable say 3 into x, you can see here these values are here, right. And in case if you want to store these values, suppose I can store this is the value of this command here as say y.

So, y is equal to this we can recall here this value by y you can see here this is the value y here and then after that I try to define here a new sequence which is z is equal to 2 into y. You can see here every value will be multiplied by here 2 and you can see here this is your here is z.

(Refer Slide Time: 19:03)

```
RGui (64-bit)
File Edit View Misc Packages Windows Help
[11] 5.0 5.4 5.8 6.2 6.6 7.0 7.4 7.8 8.2 8.6
[21] 9.0 9.4 9.8 10.2 10.6 11.0 11.4 11.8 12.2 12.6
[31] 13.0 13.4 13.8 14.2 14.6 15.0
>
> y=seq(from=1, to=3*x, by = 2/x)
> y
[1] 1.0 1.4 1.8 2.2 2.6 3.0 3.4 3.8 4.2 4.6
[11] 5.0 5.4 5.8 6.2 6.6 7.0 7.4 7.8 8.2 8.6
[21] 9.0 9.4 9.8 10.2 10.6 11.0 11.4 11.8 12.2 12.6
[31] 13.0 13.4 13.8 14.2 14.6 15.0
> z=2*y
> z
[1] 2.0 2.8 3.6 4.4 5.2 6.0 6.8 7.6 8.4 9.2
[11] 10.0 10.8 11.6 12.4 13.2 14.0 14.8 15.6 16.4 17.2
[21] 18.0 18.8 19.6 20.4 21.2 22.0 22.8 23.6 24.4 25.2
[31] 26.0 26.8 27.6 28.4 29.2 30.0
> |
```

So, 1 is multiplied by 2 it becomes here 2 1.4 multiplied by 2 this becomes a 2.8 and so on and all these values are stored in the vector z in a new variable z. You can see here that all these operations are possible here and how I try to give you here one more aspect of this sequence, right.

(Refer Slide Time: 19:29)

Sequences: Index vector

Assignment of an index-vector

`x = c(9, 8, 7, 6)` values → 9, 8, 7, 6

`ind = seq(along=x)` Positions 1 2 3 4 → index

`[1] 1 2 3 4` → index no = 2

Accessing a value in the vector through index vector

Accessing an element of an index-vector

`> x[ind[2]]` → value

`[1] 8`

```
> x = c(9, 8, 7, 6)
> ind = seq(along=x)
> ind
[1] 1 2 3 4
> x[ ind[2] ]
[1] 8
> |
```

And this will give you one more operation by which you can do many data manipulations, there is a command here along. So, I want to show you here that first of all that what is the application of along a l o n g and then I want to show you that how

you can assignment, how you can assign the values and an index vector. Do you know what is index vector? Do you see that whenever you try to read a book, there is an index, where the values are trying to indicate the locations, right.

There is an index that ok chapter 1 page number 20, chapter 2 page number 40 and so on. So, similarly when you are trying to consider the data vector, every value in the data vector has got a location and that location has to be indexed. So, now what will be the advantage? Suppose you try to take the example of your book, suppose you want to read the chapter 2.

So, you will see from the index that the chapter 2 is on the page number 40. So, will you will simply jump to the page number 40 and in case if you want to know that what is the location of the chapter number 2, then also you need to know that this is the location of the index at 2 and what is the value which is at the index number 2, that is page number 40.

So, let us try to consider a very simple example to illustrate all this. So, let me try to consider here an example as data vector which has 4 values 9 8 7 6, right. So, now, if you try to see here these are the values here which I have written here, 9 8 7 6 and these are the positions.

In this data vector also if you try to see here I can assign every number of seat. So, this is here seat number 1, seat number 2, seat number 3, and seat number 4. So, these are the positions of these values 9 8 7 6, right. So, which I am written here like this. So, this is actually here the index and I try to combine all these values in a vector the this will be called as index vector.

So, now, the question is how you can create such an index vector. So, for that you have to use the command sequence `seq` and inside the parenthesis you have to write along `x` a long all in lower case alphabet equal to `x`. So, what will happen? This along will try to read the data vector in the `x` and then it will assign the values 1 2 3 4 etc. depending on the number of values in the data vector. So, if you try to see the outcome here, the outcome will come out to be here 1 2 3 4.

So, this outcome is indicating that is stored in the variable here `ind`, that it is indicating that in the index number 1 that is corresponding to the first value in the data vector `x`.

Index number 2 that is corresponding to the second value in the data vector x, the value at index number 3 this is corresponding to the third value in the data vector x and the fourth value in the index number which is 4 this is corresponding to the fourth value in the data vector x, right. So, now what is the advantage?

Now, in case if you want to know that which value is located at this index number here 2. So, you know that at index number 2 the value here is 8 like this one, but this is I am showing you manually, I want to do it using the R software. So, in order to do the thing that if you want to access a particular value in a vector using the index vector, what you have to do? First you try to create the index vector and then you try to write down the position in the index vector that you want to access. And this has to be written here like.

This i n d and then inside the square brackets we have to write down here the location or the value which is there in the index number and then you have to write down here a data vector x in which you want to find out that what is the value at the second place in the data vector x and you have to write it inside the square bracket. So, this is like data vector, the square brackets and then inside that there is an index vector and then inside the square brackets you have to give the value, value that is in the index vector.

And now it will give you here the value here 8. So, if you try to see what is happening first you try to write down the value in the index vector, enclose it by the square bracket and then write down the value or the name of the index vector in which you have a stored the values and then you try to write down the original data vector in which you want to know where is this value and then close them by this square bracket.

So, this is how you can access that. For example, in this case what is the value of the data in the data vector at the index number 2 or at the second position in the index vector, right.

(Refer Slide Time: 24:19)

Sequences
Generating sequences of dates
Generating current time and date
Sys.time() command provides the current time and date from the computer system.

```
> Sys.time()
[1] "2021-11-29 21:23:57 IST"
```

Sys.Date() command provides the current date from the computer system.

```
> Sys.Date()
[1] "2021-11-29"
```

Handwritten annotations on the slide: The output of Sys.time() is annotated with 'Year', 'Month', 'Date', 'hour', 'minute', and 'seconds' pointing to the corresponding parts of the string. The output of Sys.Date() is annotated with '29 November 21'. A small inset window shows the full R console output for both commands.

So, if you try to see here this is a very simple command, I will try to show you on the R console, but it is more important for you to understand what is really happening that is more important, right. Now, ok after this I would like to show you some more types of sequence like a sequences of dates, etc. for example, you can create the sequences with respect to days, months, years, etc.

But before that I want to show you that in R software it is possible to get the time and date and then once you have this idea then in the next lecture I will try to show you that how you can generate the sequence related to the dates in terms of year, month, etc. So, and this type of command is very useful when you are trying to generate a report. So, first let me try to show you what I am trying to do and then I will explain you the utility.

So, in case if you want to generate the current time and date, then you have the command here sys dot time. So, this is like a time of the system, system means the computer system on which you are working. So, if you try to see here the syntax is like this, capital S and then y and s further they are in the lower case and that dot time t i m e all in lower case alphabets and then you have to write down the parenthesis. So, this is going to give you the current time and date from the computer system.

So, if you try to see here when I try to execute this command here Sys dot time, then it is giving me this type of outcome. So, this is giving me here this is year 2021, this is here month which is here month number here is 11, that is November and this is here the date

that is 29th. So, this is giving me a date here 29th-November-21 and that time here is this is here hours, this is here minutes and this is here seconds and this is IST, right.

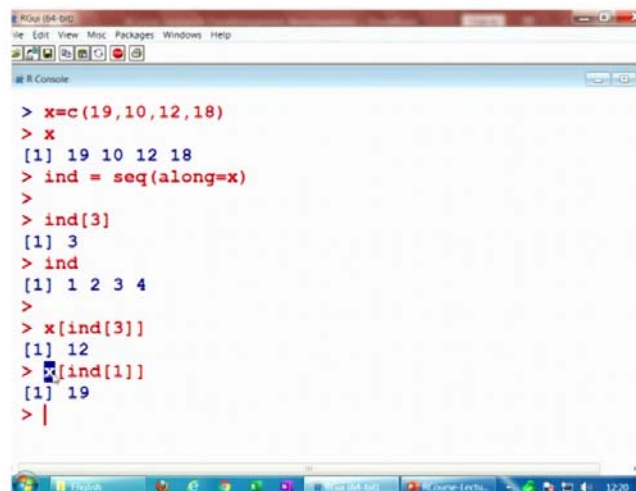
So, if you try to see here I get a this outcome, but certainly when I try to show you it on the R console you will get a different value because this is giving me the time when I had appeared the slides and I had use this command. So, when you are trying to use it on your computer do not expect that this is going to be the same, but that is going to give you the time and date when you are trying to use this command, right.

So, similarly if you want to have only the date, then the command here is sys dot date, but you have to be careful when you try to write down this command. This is here say capital S lower case y lower case s and then dot and then capital D is also capital upper case then all a t e that is a lower case and then you have to write down the parenthesis.

So, if you try to execute this command on the R software, it will give you the value like s y s dot d a t e where this S and this D they are going to be in the upper case, it will give you here the value like this. So, this is going to give me the same date which I shown you here, which is 29th-November-21, right and this is here the screenshot just to show you that this was the time when I appeared the slide ok.

So, now let me try to show you these commands on the R console also so that you get here more confident.

(Refer Slide Time: 27:20)



```
> x=c(19,10,12,18)
> x
[1] 19 10 12 18
> ind = seq(along=x)
> ind[3]
[1] 3
> ind
[1] 1 2 3 4
> x[ind[3]]
[1] 12
> x[ind[1]]
[1] 19
> |
```

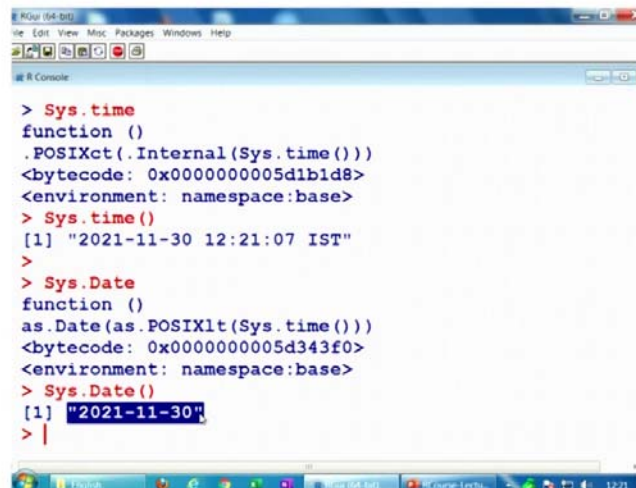
So, I try to create here a data vector say here c say 19, 10 say 12, right this is my here x data vector and now I try to create here the index vector like this. And now in case if you want to know, number 1 what is there at the index number 3, let us see what happens is 3 why because your index vector is here like this 1 2 3.

So, when you are trying to write down only here ind, the name of the index vector and inside the square brackets the 3, this 3 is corresponding to the value which is here and the third position of the index vector. Now, you want to know what is there on the third position in the data vector x. So, I have to write down here x and then the index position inside the square bracket and it will give you here the 12.

So, you can see here this is the third position here in the data vector x, whose value here is 12. And similarly, if you try to say here what is the value at here 1, you can see here the value at first position in the data vector is 19. So, when you are writing like this. So, it is going to the position in the index vector at first and then it will try to find out what is the value of x at the first location.

So, you can see here this is not a very difficult command and then I try to give you here these two commands sys dot time and sys dot date.

(Refer Slide Time: 28:35)



```
> Sys.time
function ()
.POSIXct(.Internal(Sys.time()))
<bytecode: 0x000000005d1b1d8>
<environment: namespace:base>
> Sys.time()
[1] "2021-11-30 12:21:07 IST"
>
> Sys.Date
function ()
as.Date(as.POSIXlt(Sys.time()))
<bytecode: 0x000000005d343f0>
<environment: namespace:base>
> Sys.Date()
[1] "2021-11-30"
> |
```

So, now you can see here that when I am trying to do this is going to give me a time and date when I am trying to record this video, but if you try to give it without this

parenthesis, it will not work, right. So, you have to give it here like this only, ok and then it will give you this value here, right.

So, this is here 30th-November-21 and I am recording this video at 12:21 means 12 hours 21 minutes 07 second IST, right. Similarly, if you try to write down here sys dot date and if you do not use here parenthesis this will give you error, but if you try to use here parenthesis this will give you the date this 30th-November-2021, right. So, you can see here that it is not a very difficult thing to learn, right.

So, what we have done today? We have considered almost the same command that we have learned in the last lecture, but we have understood a different way. Now, the question is whether you want to use this way of writing numbers with the help of colon sign or with the s e q command, that decision lies with you have to see what you really want to do what type of a sequence you want to generate as long as you are trying to generate simple sequence, possibly this will work.

But if you want more option then the sequence command will be there. And similarly, if you try to see I have told you here the commands for finding out the time and date. These commands are going to be very useful when you are trying to write a program to generate a report for example, you have given some assignment to your office people and you want to know that when they completed and when they had generated the report.

So, you will write many many commands and among those commands, if you write this sys dot time and sys dot date. So, they will also generate the date and time automatically. So, that will give you a foolproof system where you can identify that when this report was generated. So, you try to take some example, try to practice it and I will see you in the next lecture with more commands on the sequence, till then goodbye.