Introduction to R Software

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

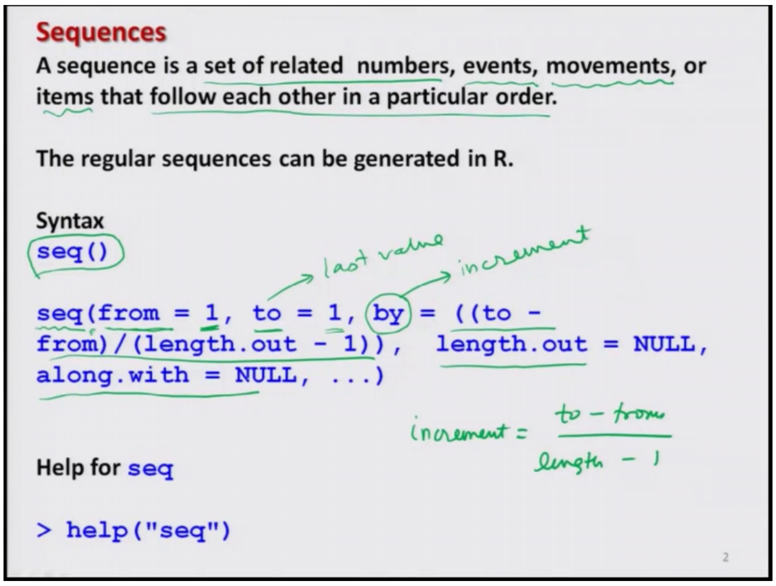
Lecture – 14
Sequences

Welcome to the next lecture on the course introduction to R software. Up to now we have been doing different types of commands and controls in R programming, now from this lecture we are going to learn different aspect of data management for example, how to get the data, how to generate the data, how to arrange the data, how to get input how to get output from the data and so on. So, in this lecture we are going to concentrate on the aspects of sequence. So, the first question comes what is the sequence for example, we always say 1 2 3 4 and so on this is a sequence, 5 10 15 20 that is the sequence if you try to see all the values are arranged in a particular order a sequence can be say 10 9 8 7 and so on.

So, in this case the values are arranged in the descending order and the separation is of one unit. Similarly if I say 10 8 6 4 and so on that means, the values are arranged in the descending order and the separation is 2 units so, that is the sequence. So, in programming at many many places we have to generate a data which has to be in the form of a sequence. We have seen such example in the earlier lectures and every time I had ask you that you will be doing all these things later on. So, this is the time when we are going to learn about sequence right ok.

So, the first question comes what is the sequence.

(Refer Slide Time: 02:04)



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, ...)
```

last value
increment

$$\text{increment} = \frac{\text{to} - \text{from}}{\text{length} - 1}$$

Help for seq

```
> help("seq")
```

So, sequence is only a set of related numbers, in case if you try to extend this definition more than this can be a sequence of events, movements or even some items and the important part is that they always follow each other in a particular order; that means, the value which are following it and the values which are followed by they are arranged in a particular order. So, in R programming also a regular sequence can be generated and the general syntax for generating a sequences seq all in small letters and then brackets, inside the brackets we specify different types of commands to generate a sequence.

For example what is that the starting value, what is the final value, what is the increment, and what is the number what is the length and so on. So, for example, the complete command will be that let us try to write down here seq sequence and then bracket is closed the first value is from. So, from is written with equal to sign say for example, if I want to generate say one. So, because is here one and what is the last value that is given by here 2 is from here to this point and yeah means because for the sake of illustration, I am trying to take here value here one and it is here by. By means what should be the increment of the sequence for example, when I say my sequence is 1 2 3 4 and so on that means, the increment value is one unit.

When I say my sequence is 1 3 5 7 and so on then my increment is 2 units, similarly if I try to take a sequence like 10 8 6 and so on then the sequence can be seen as that I want to have a sequence in which the decrement is 2 units or the increment is minus 2 units.

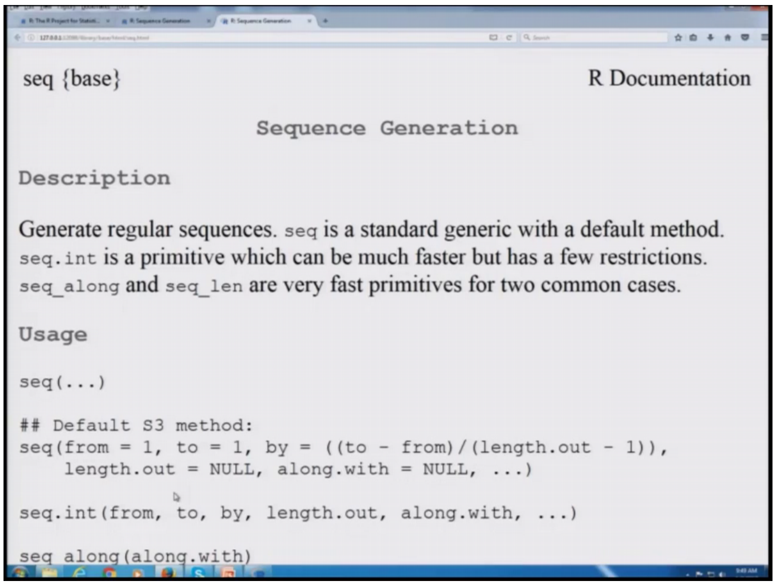
So, in order to denote the increment there are 2 options, that it can be denoted as increment or this can be denoted as decrement and decrement is nothing, but minus times increment. So, that is the rule that we usually follow and this increment is somehow this is calculated simply by taking the difference of the first value and the last value and divided by the length of the sequence. Length of the sequence means how many numbers do we want in that sequence.

For example if I want to have a sequence from 1 2 3; that means, there are 3 units. So, the. So, I am starting from one ending at 3 and the length of the sequence is 3 units. So, that will be 3 minus 1 divided by here by the length of the vector right. This length of the increment that is computed by last value 2 minus here from divided by the length of the vector minus here 1 and this is how the R is computing here right and then if you try to give here different aspects like a length out and then along with and so on.

But definitely now if you try to understand you have already reached to a status where you understand many many things, and then instead of I going into all the details of the syntax seq why do not you try yourself and try to obtain all this data and as I said this can be obtained very easily, you simply have to type here help and then seq inside the bracket followed by in double quotes, and you can get it for example, if I try to find out here this and then if I try to press it over here see here, you can see here now this is the help provided by the R software on its website, and you can see here this is giving you the complete detail.

The only thing what you have to do is that you need to read it carefully.

(Refer Slide Time: 06:30)


seq {base}　　R Documentation

Sequence Generation

Description

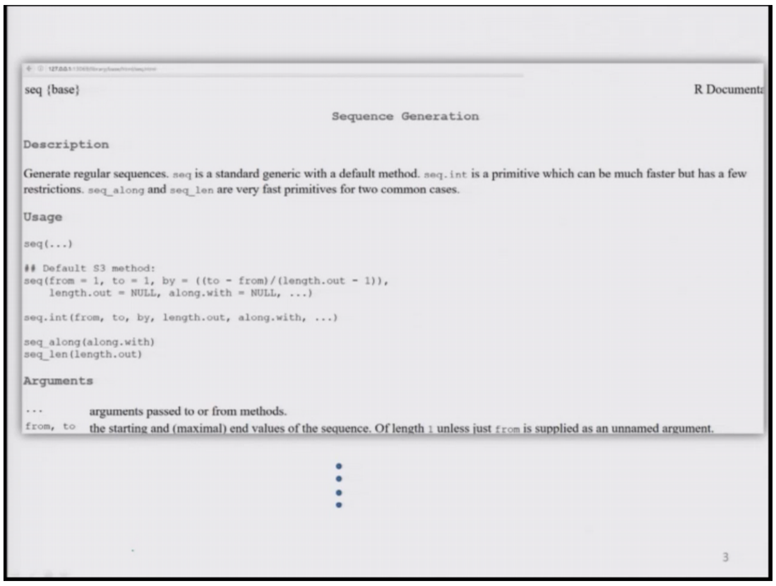
Generate regular sequences. `seq` is a standard generic with a default method. `seq.int` is a primitive which can be much faster but has a few restrictions. `seq_along` and `seq_len` are very fast primitives for two common cases.

Usage

```
seq(...)

## Default S3 method:
seq(from = 1, to = 1, by = ((to - from)/(length.out - 1)),
    length.out = NULL, along.with = NULL, ...)

seq.int(from, to, by, length.out, along.with, ...)

seq_along(along.with)
```

But that is always advisable that whenever you are trying to use any command for the first time, you should read it carefully because that is the most authentic source which is giving you all the help from the developers right. So, I would request you that you try to go with these things and then try to read it, and you can see here they have given different types of a examples also right. I am just scrolling down quickly, but I would request all of you to just have a quick look over this right.

(Refer Slide Time: 07:13)


seq {base}　　R Documenta

Sequence Generation

Description

Generate regular sequences. `seq` is a standard generic with a default method. `seq.int` is a primitive which can be much faster but has a few restrictions. `seq_along` and `seq_len` are very fast primitives for two common cases.

Usage

```
seq(...)

## Default S3 method:
seq(from = 1, to = 1, by = ((to - from)/(length.out - 1)),
    length.out = NULL, along.with = NULL, ...)

seq.int(from, to, by, length.out, along.with, ...)

seq_along(along.with)
seq_len(length.out)
```

Arguments

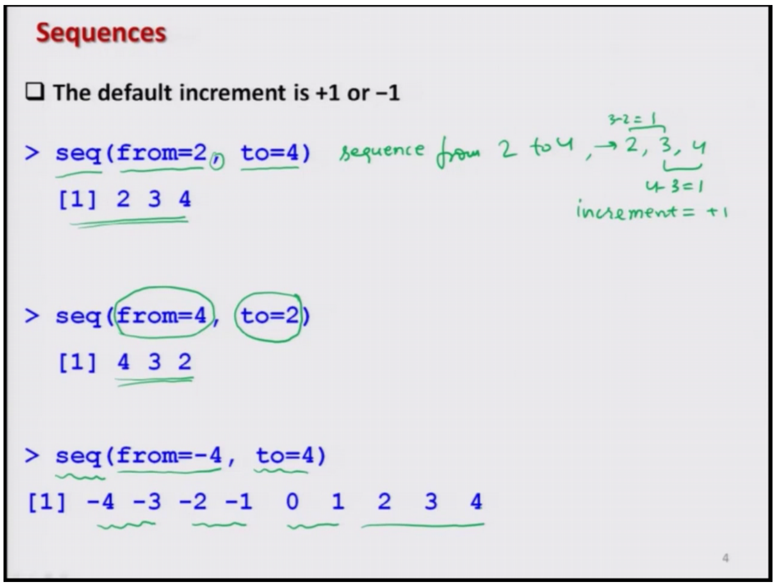
`...`	arguments passed to or from methods.
`from, to`	the starting and (maximal) end values of the sequence. Of length `1` unless just `from` is supplied as an unnamed argument.

⋮

3

Here I am trying to give here a screenshot of the same website that we have seen. So, that is just for the sake of your information although the contents in the and this snapshot app are quite dull, but I request you to go through the site and because is simply a screenshot, just to give you a confidence that what you are getting when you try to seek the help for the function seq right. Now let us try to take several example of this sequence and try to understand how the things are happening right.

(Refer Slide Time: 07:47)



Sequences

- The default increment is +1 or −1

```
> seq(from=2, to=4)
  [1] 2 3 4
```

sequence from 2 to 4, → 2, 3, 4

3-2 = 1

4-3 = 1

increment = +1

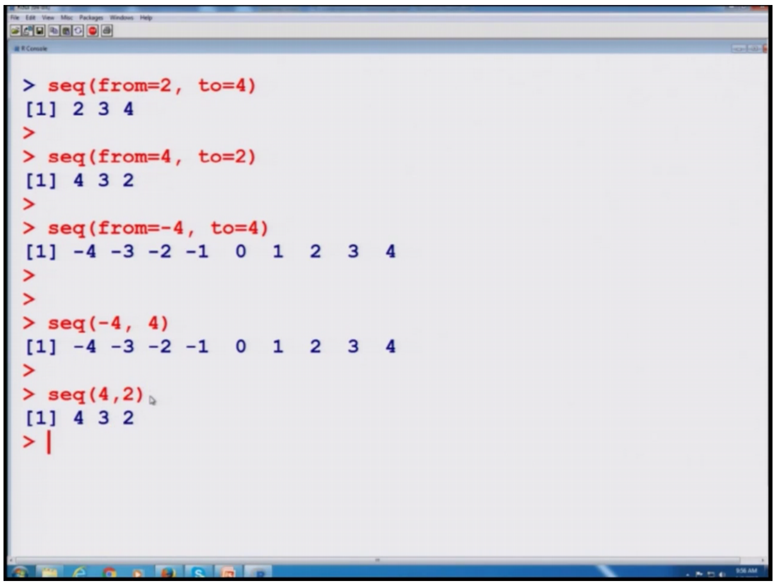
```
> seq(from=4, to=2)
  [1] 4 3 2

> seq(from=-4, to=4)
[1] -4 -3 -2 -1  0  1  2  3  4
```

One thing what you have to keep in mind that the default increment or decrement in the syntax seq sequence is one. So, increment can be of one unit or the increment can be of minus 1 unit that is your decrement. So, for example, let us try to take the first example here I want to generate a sequence from 2 to 4 and I want that all the number should be arranged in an increment of one. So, the sequence will look like 2 3 4, and then you can see here the difference between 3 minus 2 here is 1, the difference between 4 minus 3 here is 1. So, the increment here in each of the value here is plus 1 units.

So, as soon as you try to write down here is seq from equal to 2 and separated by comma and then 2 equal to 4 and as soon as you enter you get here this outcome 2 3 and 4 and similarly you can also go in the reverse direction. For example, if I try to obtain a sequence from 4 to 2, the just the opposite of the first example from 4 to 2 the default decrement is going to be 1 units, and you get here a sequence 4 3 and 2 and another thing which I would like to show you that the values which are taken in from and 2 they need not to be only the positive integers, they can be negative integers also for example, here I am trying to generate a sequence from minus 4 to plus 4.

The default increment is going to be 1 unit. So, I get here a sequence here minus 4, minus 3, minus 2, minus 1 and up to 0 1 2 3 here 4 let us try to see this example on the r.

(Refer Slide Time: 10:02)



```
> seq(from=2, to=4)
[1] 2 3 4
>
> seq(from=4, to=2)
[1] 4 3 2
>
> seq(from=-4, to=4)
[1] -4 -3 -2 -1  0  1  2  3  4
>
>
> seq(-4, 4)
[1] -4 -3 -2 -1  0  1  2  3  4
>
> seq(4,2)
[1] 4 3 2
>
```

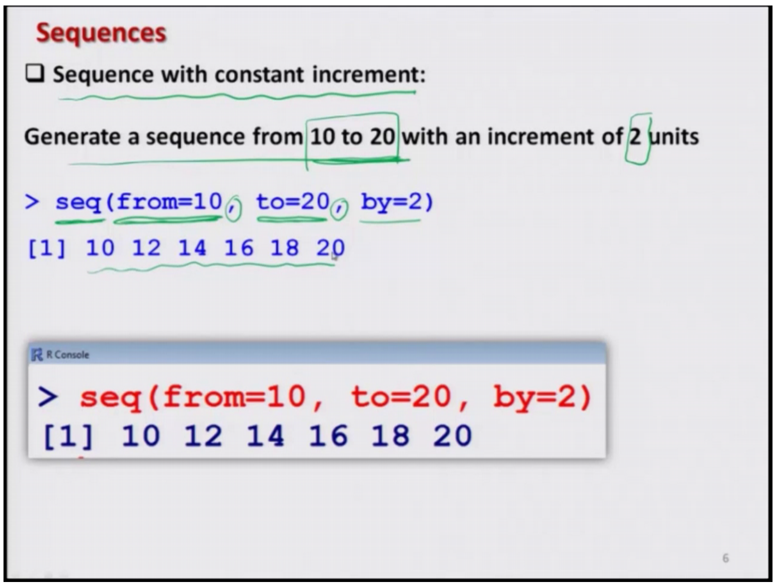
Now let us try to have the first example from in which we are trying to generate a sequence from 2 to 4. So, as soon as you type this command and say here enter you get a sequence of a numbers 2 3 and 4, and similarly if I try to have here another command in which I would like to generate a sequence from 4 to 2, then we see here that we get an outcome here 4 3 and 2.

And similarly if I try to have here another sequence from minus 4 to 4, then you can see here I get the same thing. So, you can see here that in any situation wherever you want to generate a data in the form of a sequence you do not have to write down the entire vector; means if you remember earlier we had written a vector sometimes say 1 2 3 4 5 6 7 8 and these values were combined by the operator c. So, you need not to write the entire vector, but you can simply write down here same sequence from 1 to 8 and here I would also like to emphasize that it is not important to write the from and to also, you can directly write these values and it will take it automatically as a default.

For example here you can see here that in the same command say for example, I do not write here from and to, and I delete it and I simply write sequence inside the bracket minus 4 to 4 and I get here the same value. So, the first value in the bracket that is always taken as a say from and the second value in this bracket here is taken as here to. So, that is also a shorter form for example, in this case also if you want to have a sequence from 4 to 2, then I can delete this to and from and the first value here this is giving me here default value as say from and this is giving here as a default value as here 2, and as soon as you enter you get this thing the sequence 4 3 2.

So, you can see here that is pretty convenient and in the next slide I am simply trying to give you a screenshot of the same output which we have just done right.

(Refer Slide Time: 12:25)



Sequences

❑ Sequence with constant increment:

Generate a sequence from 10 to 20 with an increment of 2 units

```
> seq(from=10, to=20, by=2)
[1] 10 12 14 16 18 20
```

R Console

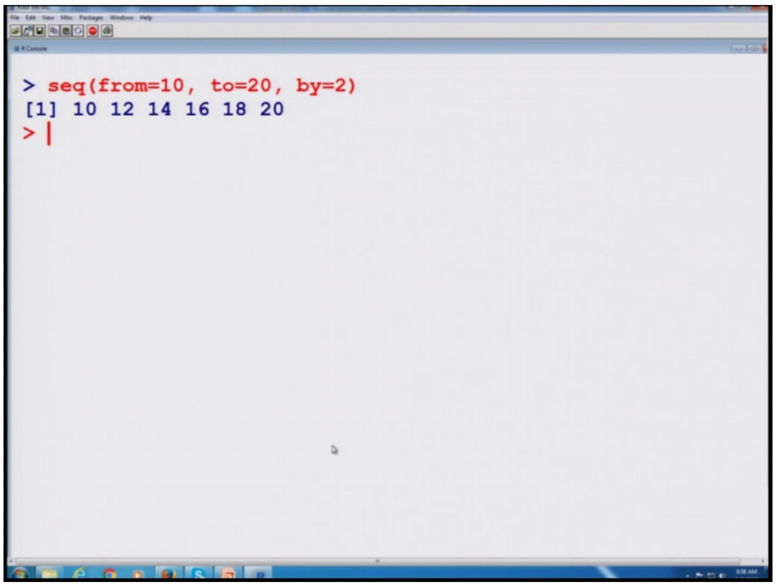
```
> seq(from=10, to=20, by=2)
[1] 10 12 14 16 18 20
```

6

Now, I try to takes another situation where we want to generate a sequence with some constant increment for example, the default increment in a sequence operator is 1; 1 unit plus 1 or say minus 1, but suppose I want an increment of say 2 units or 2.5 units or say minus 3 units or say minus 3.6 units and so on and this case we can also generate a sequence which has got a constant increment which is not equal to the default increment of 1 unit ok.

So, now I try to take here an example here where I try to generate a sequence from 10 to 20, and I want the increment to be of 2 units. So, now, I write down the syntax here as say seq which is for the sequence, first value is going to be here from 10 second value separated by this comma is going to be to is equal to 20 that mean from 10 to 20, and this increment is going to be written over here separated by this comma say by equal to 2 and as soon as you enter here you get here this sequence 10 12, 14, 16, 18, 20. So, why not to do this here on the R console itself so that we get more confidence that this is really happening right.

(Refer Slide Time: 13:56)



```
> seq(from=10, to=20, by=2)
[1] 10 12 14 16 18 20
>
```

So, you can see here we get here the same outcome right and here is the screenshot of the same thing here. So, similarly now onwards I am going to take different type of situations in which I would try to explain you that what type of sequences are possible to generate in r, and now you have a confident that whatever I am writing here they are really happening and I am also giving you the screenshot and I would request all of you that you please try to experiment these commands yourself with your own hand, unless and until you do not do it you will not gain confidence in programming.

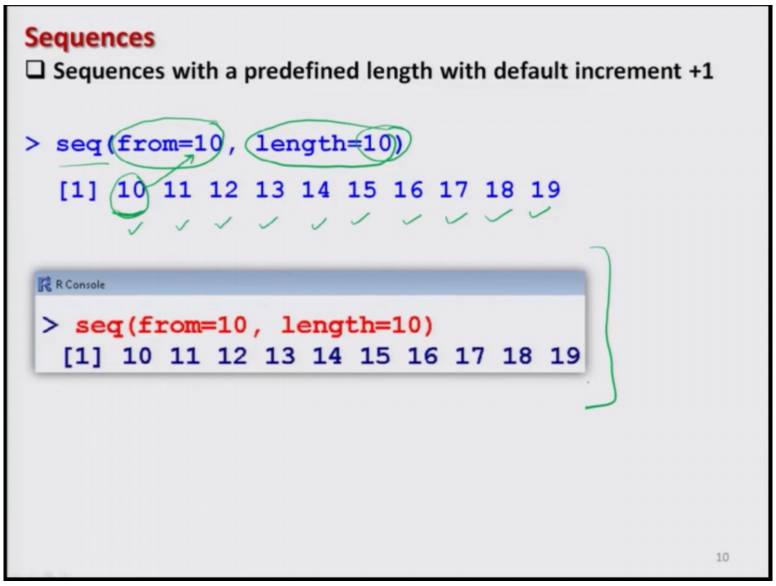
And another thing is this when you are trying to execute this program by your hand you will also remember them. So, in this example I am simply trying to take the same example, but in this case I am trying to generate a sequence from 20 to 10, that is in a decreasing order and I want to give here a decrement of here 2 units. So, this decrement is actually as I told you this is minus times say increment right. So, I try to write down my here syntax as here say seq inside the bracket from 20 separated by comma and then to 10.

And now I have to give here the increment now I have 2 options either I try to give here increment as see here minus 2 or see here decrement as here 2. So, now, if you try to see what is the relationship between increment and decrement, this decrement is nothing the minus times increment. So, I am trying to give here an increment of 2 units and when I am trying to give here that decrement I can give here minus 2 units right and as soon as I do it I get here the value from 20 to 18 something like 20 plus minus times here 2, and this gives me here the value 18, from 18 to 16 this give me here 18 minus 2 that is 16 and so on this continues up to here till where I get the values here 12 minus 2 equal to here 10. And this is the screenshot of with this operation which will give you a confidence that whatever we have done that really works well right.

Now, I try to take see here another example where I try to find out here a downstream sequence; that means, the values are decreasing and there is a constant increment. So, I try to generate a sequence starting from 3 to minus 2 which is going in the downward direction that is a decreasing sequence and there is going to be a decrement of 0.5 units. So, there are 2 things which I would like to show you here through this example number one these values from and to they can be negative integers also and this increment need not always be in an integer form this can be a fraction also.

So, here I try to write down the same sequence; from 3 to minus 2 separated by this comma, and then by here I am trying to give here a decrement. So, that is going to be a minus times increment and which is here minus 0.5. And as soon as you try to do it here you get here the same sequence. So, let us try to do it here on the R console and try to see whether are we getting the same thing here, you can see here that we are getting here the sequence starting from 3 and then it is decreasing by 0.5 units, then again by 0.5 units and so on.

(Refer Slide Time: 18:02)



Sequences

❑ Sequences with a predefined length with default increment +1

```
> seq(from=10, length=10)
 [1] 10 11 12 13 14 15 16 17 18 19
```

```
> seq(from=10, length=10)
 [1] 10 11 12 13 14 15 16 17 18 19
```

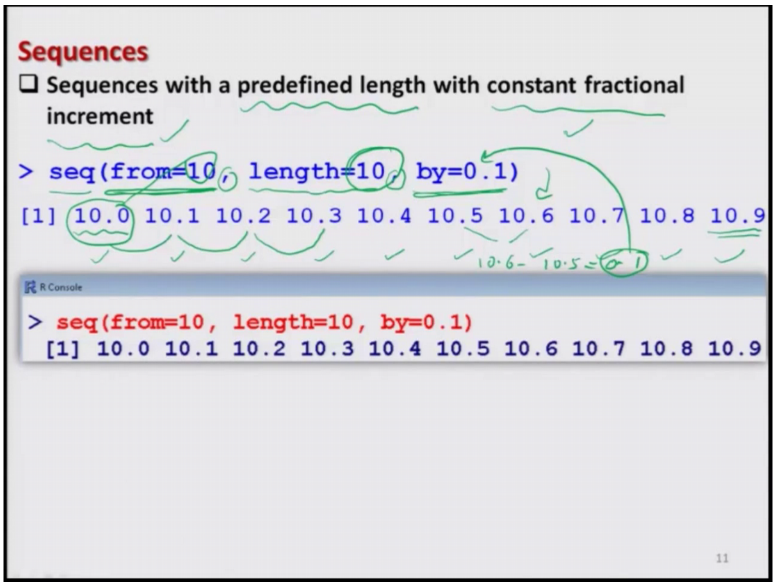
So, now, in the next example I am trying to take another issue that I want to generate a sequence of predefined length. Predefined length means I know that how many elements I want, how many values in a sequence I want to generate right and the default increment I am going to use here as say plus 1, in that case you simply have to write here 2 this is the last value and before that you want a sequence of length 10; that means, there are going to be 10 values and the sequence is going to end by the value 10. So, as soon as you try to do so, you get here a sequence one to 10 and that is correct also the last value here this 10 is this thing.

And if you try to count on this number 1 2 3 4 5 6 7 8 9 10 this is here this value that is the length of the sequence here is 10. So, even if you do not know the starting value or the end value, which are little bit confusing whenever you are trying to use a fractional increment you need not to worry, you simply have to give the 2 value and then you have to specify the length of the sequence and a same thing we can also do at the from also, but here you can also have a look at this screenshot and do not you try it yourself.

Now, the same example I am trying to take here, but instead of here 2, I am trying to use here from; that means, I want to start my sequence with 10 and I want a sequence of length 10. So, you can see here as soon as you write down in the a format of our sequence, the starting value comes out here in this output here as say 10 and then after that this 10 is corresponding to this 10 and then you can see here how many values are there? 1 2 3 4 5 6 7 8 9 10 and this value here is 10 the length of the sequence is 10; that means, there are 10 values in the sequence and starting from 10.

And this is the screenshot of the outcome. So, you please try it yourself.

(Refer Slide Time: 20:43)



Sequences

- Sequences with a predefined length with constant fractional increment

```
> seq(from=10, length=10, by=0.1)
[1] 10.0 10.1 10.2 10.3 10.4 10.5 10.6 10.7 10.8 10.9
```

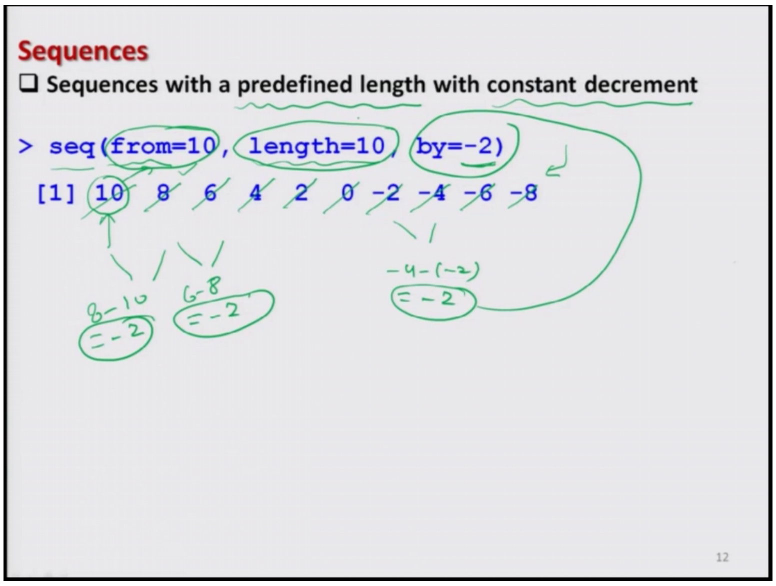
```
> seq(from=10, length=10, by=0.1)
 [1] 10.0 10.1 10.2 10.3 10.4 10.5 10.6 10.7 10.8 10.9
```

Now I try to take another example on the similar lines, but here I do not use the default increment, but I try to use here the constant fractional increment. So, I have here 2 types of information, one is that what is the predefined length and what is the constant fractional increment. Now I have to specify the 2 values, in the first case I will try to use the frown value and in the second case I will try to use the 2 values as I did in the last 2 example right. So, here I try to write down here sequence inside the bracket, the from value that I want my sequence to start from 10 and the length of the sequence is going to be 10 separated by this comma.

And the increment is going to be here 0.1 which is a written after a separator comma so; that means, I want to start my values from see here 10 and after that I need here 10 values and which are differing by 0.1. So, you can see here the outcome as soon as you enter here, the first value comes out to be here 10.0 this 10.0 is the same as here this 10. Now after this you can see here that the values are differing by 0.1 10.0 to 10.1 10.1 to 10.2 10.2 to 10.3 and so, on this goes up to here 10.9.

And how many values are this? 1 2 3 4 5 6 7 8 9 and here 10 and this 10 is this thing and the difference between this and this which is here 10.6 minus 10.5 which is equal to 0.1 this is here this value. So, this is how I can generate a sequence of predefined length with a constant fractional increment.

(Refer Slide Time: 22:45)



Sequences

- Sequences with a predefined length with constant decrement

```
> seq(from=10, length=10, by=-2)
 [1] 10  8  6  4  2  0 -2 -4 -6 -8
```

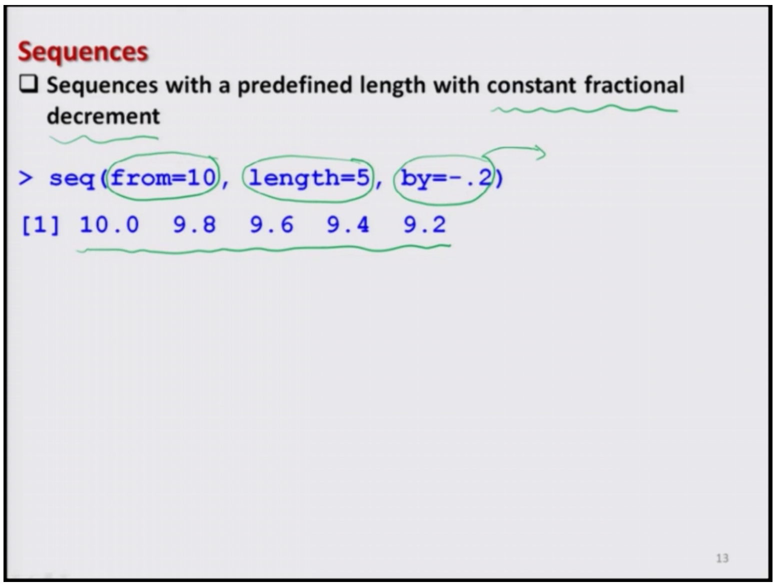
So, now, let us try to take another example here in which I am trying to have the information on that I want a sequence of predefined length, and I know the value of here from which is given over here, and I need a constant decrement.

So, just for the sake of example I try to take here a sequence which is here and then I try to give it here from 10 separated by length 10 and then the increment is or the decrement is going to be here denoted by minus 2, that is an increment of minus 2 that is actually decrement once you enter then you get here this type of sequence. So, you can see here this is the value which is the starting value and that is given over here, and now if you try to see the difference between 10.8 is 8 minus 10 which is equal to here minus 2 the difference between 6 minus 8 here is say here minus 2.

And similarly here if you try to take care anything over here then the difference here is minus 4 minus 2 this is equal to here minus 2 and this minus 2 this minus 2 this minus 2 that is given here that is the constant increment, and the values which I am getting here they are 1 2 3 4 5 6 7 8 9 and 10 and this is the length of the sequence where I am trying to get here 10 values starting from 10 at a decrement of minus 2 right ok.

Now, let us try to take another example, but before that let us try to show you here whether this is really happening in the R console or not so you can see here that this is really happening.

(Refer Slide Time: 24:41)



Sequences

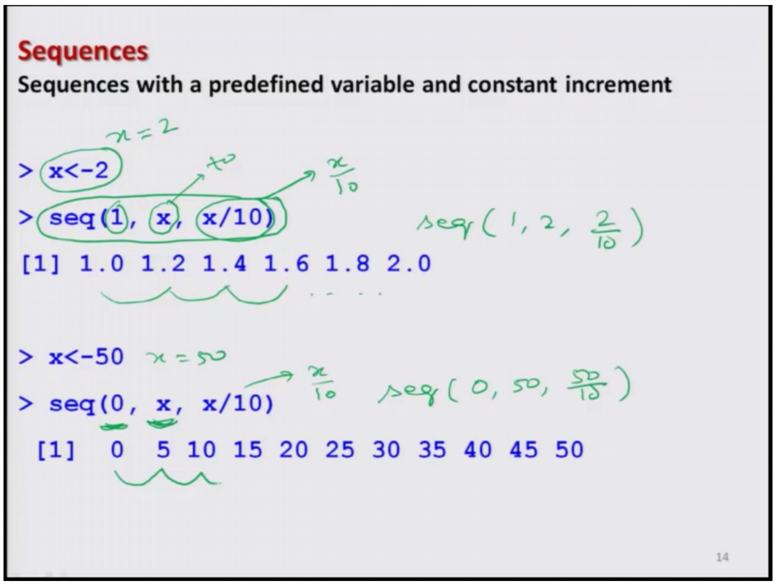
- Sequences with a predefined length with constant fractional decrement

```
> seq(from=10, length=5, by=-.2)
[1] 10.0  9.8  9.6  9.4  9.2
```

13

So, now, let us try to take here another example and here I am trying to take the same example which I did just now, but I am making a little difference by adding here the constant fractional decrement that I want to get here a sequence of length 5 that is starting from the value 10, and the decrement is going to be 0.2 that is increment is going to be minus 0.2 right.

And as soon as you try to do it here you get here the sequence of this type. So, the idea which I want to convey by this example is that the increment or the decrement can be integer that can be fractional, that can be positive as well as that can be negative, all sorts of possibilities are there right and let us try to do this example on the R console also. (Refer Slide Time: 25:53)



Sequences

Sequences with a predefined variable and constant increment

```
> x<-2
> seq(1, x, x/10)
[1] 1.0 1.2 1.4 1.6 1.8 2.0

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

So, that you can check the you can see here you are getting the same outcome which I have given here. Now let us try to take say here another example that is the more general format.

Suppose I want to write a program in which I need to generate some data depending on different types of values. So, here up to now in all this example I am trying to give a specific value or from to by and so on, but that can be controlled by a function also for example, if I say there is some sequence which I want to generate there should start from here see here 1 and it should go to x and the increment is going to be here x by 10.

So, that is a very general format and now if I try to specify the value of here x suppose for example, I try to specify here x as here 2 so; that means, here I am trying to generate here a sequence a 1 to 2 with an increment of 2 upon 10 that is 0.2, and as soon as you try to run this program you get here a sequence of 1 to 1.2 to 1.4 to 1.6 and so on.

And similarly if I want to take another value you see here suppose I take here x equal to 50 and I consider a different sequence which is starting from here 0, and that is going up to x and the increment is going to be here x upon 10 then that means, I am interested in getting a sequence which is 0, which is starting from 0 and it is going up to 50 and the increment is going to 50 upon 10 which is here 5 units. So, you can see here I am getting here a sequence 0 5 10 15 20 25 and so on right and here is screenshot of the same operation you can see over here.

Now, we stop here and in this lecture I have taken different types of a sequences in which I have used that sequence from to and by, and I have used the option of length also and I have taken various example to show you that what type of data can be generated just by using a single command sequence.

So, I would request you that you try to practice this thing and in the next lecture I would try to take some more example on the sequence, to demonstrate how the data can be generated automatically using the syntax seq till then goodbye.