

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

**Lecture – 15**  
**Sequences**

Welcome to the next lecture on the course Introduction to R Software. In the earlier lecture, we started discussion on the syntax sequence and we learnt through various types of example, that how we can generate the sequences using different types of from value, different types of to value, and different types of increment or say decrement right. One thing what you have to keep in mind, that the value by that is giving an increment.

(Refer Slide Time: 00:49)

**Sequences**

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, ...)
```

Handwritten annotations in green:

- Arrows pointing to 'from = 1' and 'to = 1'.
- An arrow pointing to 'by = ((to - from)/(length.out - 1))' with the label 'Increment'.
- An arrow pointing to 'by = ((to - from)/(length.out - 1))' with the label 'Decrement'.
- An arrow pointing to 'by = ((to - from)/(length.out - 1))' with the label 'Increment'.

2

And, if you remember the syntax what we used was something here like this, where I had used here 3 values from, to and see here by.

So, always remember one thing, that by is trying to give you the increment. And in case if you want here are decrement, then decrement is always taken as say minus time say increment. This is the basic fundamental in generating a sequence, and then this value here from and to they can be positive integers, negative integers, as well as the they can be some fractional values also and similarly for the by also, they can be a positive values, negative values, integer values, fractional values, and all sorts of possibilities are there, and in case if you practice more example taking different types of combinations you will

learn more. And on the same lines you let us try to take here some more examples, and we try to see how the sequence can work.

(Refer Slide Time: 01:54)

The image shows a screenshot of an R console window with the title "Sequences". It contains two examples of the `seq()` function. The first example is `> seq(10)`, which outputs `[1] 1 2 3 4 5 6 7 8 9 10`. Handwritten green annotations include an arrow pointing from `seq(10)` to the output, and wavy lines under each number from 1 to 10. The second example is `> seq(1:10)`, which also outputs `[1] 1 2 3 4 5 6 7 8 9 10`. Handwritten green annotations include an arrow pointing from `seq(1:10)` to the output, and the text `seq(from:to)` written next to it. Below these examples is a separate screenshot of the R console showing the same two commands and their outputs in red text.

So, first example I am trying to take here, that suppose I want to generate 10 values starting from 1 to 10 right. One of the shortcut methods or the default, syntax of the sequence is that if I try to write down here sequence `seq` and inside the bracket I try to write only here 10 then this will give me 10 values by the default increment 1. And if you try to do it here you get here 1, 2, 3, 4, up to here 10 and these are the 10 values. And same command can also be done by, using this command that sequence and say here from separated by colon and say here to.

So, I am trying to say here, I want here a sequence is starting from 1, and which is going up to here 10 and as soon as you do it, you get here a sequence 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, and both these sequences they are the same. So, these are two simple ways to generate sequence of predefined length say 10 starting from 1 to 10. So, you can use any of the options over here, and here I have given the screen shot and, we also try to do it over the screen also.

So, let us try to copy this thing here, say here this and then. So, you can see here this is giving me a sequence of from 1 to 10 and when I try to do it here, sequence 1 colon 10 this is also given as the same sequence. And yeah means here I also have given the screenshot.

(Refer Slide Time: 03:50)

**Sequences**

Assignment of an index-vector

```
> x <- c(9,8,7,6)
> ind <- seq(along=x)
> ind
```

[1] 1 2 3 4

*x = c(9, 8, 7, 6)*

*index vector*

R Console

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

Accessing a value in the vector through index vector

Accessing an element of an index-vector

```
> x[ ind[2] ]
```

[1] 8

R Console

```
> x[ ind[2] ]
[1] 8
```

*values*  
c(9, 7, 5)  
↑ ↑ ↑  
1st 2nd 3rd  
↓ ↓ ↓  
1 2 3 → index

Now in the next assignment, I would like to introduce you that how to assign a sequence within index vector. First, let us try to understand what is an index in our vector? Suppose, if I take here any vector here, which is denoted by here c say here 9, 7, 5. So, the value at the first position here is 9. The value at the second position is 7 and the value at the third position is here 5. So, these are actually the values which are taken at first, second and third position. So, I will repeat the statement again. This is the first value, this is the second value and fifth is the third value. So, this I can write down here as a 1, 2, 3, which are nothing, but the index.

Now, suppose I want to assign an index vector. So, what I can do here? I try to explain through an example. Suppose I try to take here a vector x, which is containing 4 values, combined by the operator c 9, 8, 7, 6. And I try to generate, say another sequence, sequence and inside the bracket I write here a l o n g along and I say along is equal to x. And this will give us an sort of index vector, and if I try to execute it and print the value of here ind outcome it gives me 1, 2, 3, 4; that means, 1 has been assigned to the value 9, 2 has been assigned to the value 8, 3 has been assigned to the value 7 and this 4 corresponds to the value here 6, and the same values were given by our vector here x. And these values are the same values which are given over here.

So, then I can say that this is my index vector and that is indicating the position of the values inside the vector x. These types of sequences are really useful when whenever you

are trying to play with some data and I want to call particular type of data. For example, if I have different values and I want to entertain only a particular value from my data, then this index helps us and I can recall a particular value using the index. I try to show this here with a simple example here, suppose I want to access a particular element supposing this in supposing this index vector.

Suppose I want to access the second value in `x`. Now I am talking of the position of the values in the vector. So, now, I want the second value from a vector. Further sake of illustration, I try to take here the same vector here `x`. And suppose I want to recall its second value; that means the value at the second position. And now if you try to see here this value here `8` in the vector here `x`, this is at the second position.

So, now, in order to call this particular value here `8`, I have to write here `x` and one thing what you have to now notice very important, that I am not using here any other bracket except the square bracket. So, you have to write here `x`, the vector from which I want to recall the value, and you have to enclose it by a square bracket and inside the bracket you have to write down the value of this index vector, which you had obtained in the earlier example. For example, this vector here `ind`, this has been obtained here that you can see I am trying to bring you here right.

So, I need the second value, in the `x` vector which is corresponding to the second position in `ind` vector and as soon as I do it here I get here `8`. And let us try to see this thing on the R console also. So, suppose I try to take this sequence here and I try to create my here independent vector here. So, you can see here I am getting these values here as say `ind`.

(Refer Slide Time: 09:30)

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

And now I want to recall here the value of x at the second position, I try to write down here 2, and you can see here that this 8 is the same as this 8. And similarly, suppose if you want to find out here is the value at the fourth position, you can see here this is giving me here 6, here you can see here this 6 and this 6 they are the same.

So, now let us try to come back to our slides and.

(Refer Slide Time: 10:03)

**Sequences**  
Generating sequences of dates

Usage  
`seq(from, to, by, length.out = NULL, along.with = NULL, ...)`

Arguments

- `from` starting date (Required)
- `to` end date (Optional)
- `by` increment of the sequence. "day", "week", "month", "quarter" or "year".
- `length.out` integer, optional. Desired length of the sequence.
- `along.with` take the length from the length of this argument.

Now, it is also possible, to generate the sequence of dates. For example, if you are dealing with a data, that is arranged with respect to dates. For example, the value of x on

first January, second January, third January, or say the value of  $x_n$  and say this month of January, month of February, month of March and so on or the value of  $x$  in say 2012, 2013, 2013 and so on. So, in this case we would like to generate a data that contains a vector of say here dates.

So, now we are going to learn, how to generate such sequence of dates under different types of conditions. But before that I would try to introduce you that there are two commands by which I can always find out the that time and date of the system. The system is your computer system. So, in case if I want to find out the time, I simply have to use here the command `sys dot t i m e` and then the bracket. Note one thing that this the first `S` is going to be say capital letters and say another `s` is going to be small letters whether wise this will not work.

For example, here if I try to execute here, I get here this data, that is here the this is giving me here the date and this is here giving me here the time, and this is here giving me the time zone right. And remember one thing this time zone dates time they are under your control that whenever you want to generate it you can generate them and this time zone can be changed depending on the information that is feuded in the system. And similarly, if you want to find out the date feeded in the system, so that can be obtained by capital `S` small `y` small `s` dot `date` and then in the date also you have to keep that this `d` is going to be in the capital letters and after that all small letters ate and this bracket sign.

So, just note that this capital `S` and capital `D`, they are here in the capital letters. So, if you try to do it here, you get here the system date right. For example, this is first January 2017, right; and now here is the screenshot of the same thing right. Now let us try to understand, what is the syntax for generating a sequence of dates. It is more or less like the earlier one. That I have to write down the syntax `seq`, first value is going to be from, second value is going to be here two and then third value is going to be the `y` that is giving you the increment value and then the same thing length out and along with whatever we have done earlier.

The main difference in generating the dates and generating a numerical value is as follows: that when I am trying to write down here this here from, this from is going to be a starting date and when I am trying to give you here a value of here to this to is going to be the end date. Out of the starting and end date, this starting dates is required where is

the end date is optional. And when I am trying to give it to here the values for here by, this by depends on you. And this can be a day, this can be a week, this can be a month, this can be a quarter or this can be an year.

So, that depends on the requirement whether we want to generate a sequence of date with respect to day, week, month, quarter or year that we have to specify. And then length out this is going to an integer value, this is optional this is not compulsory, but this will give you the desired length of the sequence that how many values you want in a sequence. And similarly here this here along with this has the same rule as we understood in the earlier case, that it takes the length from the length of the argument, and they try to generate the sequence along with that vector right.

(Refer Slide Time: 15:03)

**Sequences**  
Generating sequences of dates *from*  
*as.Date("year-month-day")*  
Sequence of first day of years *1 Jan 2010 to 1 Jan 2017 Years*

```
> seq(as.Date("2010-01-01"), Start date as.Date("2017-01-01"), End date by = "years")
```

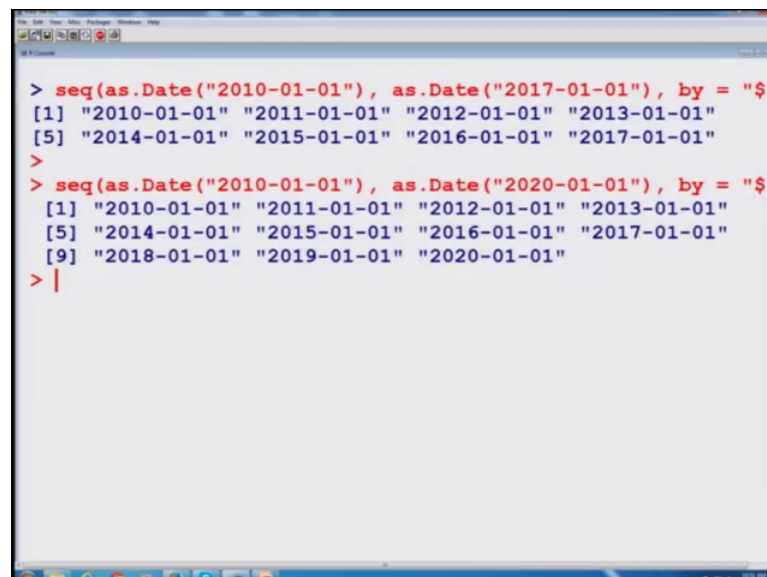
```
[1] "2010-01-01" "2011-01-01" "2012-01-01" "2013-01-01"  
[5] "2014-01-01" "2015-01-01" "2016-01-01" "2017-01-01"
```

```
> seq(as.Date("2010-01-01"),  
+ as.Date("2017-01-01"), by = "years")  
[1] "2010-01-01" "2011-01-01" "2012-01-01" "2013-01-01"  
[5] "2014-01-01" "2015-01-01" "2016-01-01" "2017-01-01"
```

So, now let us try to take here is some examples, and we try to see what is really happening. A one thing, what you have to keep in mind. That when we are trying to give here the from values and two values in the format of date, there is a special format in which you need to enter the data. The special format is this, you have to write as small letters, then date in such a way so, that capital D and all let small letters ate Date, then inside the bracket, you have to write down the Date, but those days have to be enclosed inside the double quotes, and this date has to be given by here say year, 4 digits and then by here say here month and by here see here day.

So, whenever you want to give the from or two value in the format of a date, that has to be given in this format. So, suppose I want to generate a data from 1st January 2010 to 1st January say 2017, and I want this data to be in the annual modes. So, this data has to be generated with respect to years. So, I would like to take the sequence here as a sequence and this is my here from date. So, you can see here, I am writing here inside the bracket, this as dot capital Date and then inside here this is the date from where I want to start my sequence. Then separated by this comma date up to which I want my sequence that is the end date. So, this is the first date is start date and the second date is end date. And these values are to be arranged with respect to years right. So, as soon as you try to do it, I get here this type of outcome and the screen shot of this outcome is also given here and let us try to do so over the R console.

(Refer Slide Time: 17:32)



```
> seq(as.Date("2010-01-01"), as.Date("2017-01-01"), by = "$
[1] "2010-01-01" "2011-01-01" "2012-01-01" "2013-01-01"
[5] "2014-01-01" "2015-01-01" "2016-01-01" "2017-01-01"
>
> seq(as.Date("2010-01-01"), as.Date("2020-01-01"), by = "$
[1] "2010-01-01" "2011-01-01" "2012-01-01" "2013-01-01"
[5] "2014-01-01" "2015-01-01" "2016-01-01" "2017-01-01"
[9] "2018-01-01" "2019-01-01" "2020-01-01"
> |
```

So, you can see here, that you are getting this type of data. And suppose if you want this data form say 2010 to say 2020, you can simply have to change the end date and you get here this type of outcome and so on.



(Refer Slide Time: 17:55)

```
Sequences  
Generating sequences of dates from  
as.Date("year-month-day")  
Sequence of first day of years 1 Jan 2010 to 1 Jan 2017 Years  
> Start date seq(as.Date("2010-01-01"), End date as.Date("2017-01-01"), by = "years")  
[1] "2010-01-01" "2011-01-01" "2012-01-01" "2013-01-01"  
[5] "2014-01-01" "2015-01-01" "2016-01-01" "2017-01-01"  
  
R Console  
> seq(as.Date("2010-01-01"),  
+ as.Date("2017-01-01"), by = "years")  
[1] "2010-01-01" "2011-01-01" "2012-01-01" "2013-01-01"  
[5] "2014-01-01" "2015-01-01" "2016-01-01" "2017-01-01"
```

And, similarly if you want to change the data for say this month or year, that also you have to simply change the option and instead of your here years you have to give the days, month, quarter or say anything else.

(Refer Slide Time: 18:08)

```
Sequences  
Generating sequences of dates  
Sequence of days  
> from seq(as.Date("2017-01-01"), " " by = "days", " "  
length = 6)  
[1] "2017-01-01" "2017-01-02" "2017-01-03" "2017-01-04"  
[5] "2017-01-05" "2017-01-06"  
  
R Console  
> seq(as.Date("2017-01-01"), by = "days", length = 6)  
[1] "2017-01-01" "2017-01-02" "2017-01-03" "2017-01-04"  
[5] "2017-01-05" "2017-01-06"
```

Now, we take this another example; in which I try to generate the data, with respect to days. The format here is the same, you have to write down the sequence and the dates has to be in the special format like capital D and then date and inside the brackets you have to write down the dates inside the double quotes. So, I try to write down here the

starting date, because this is compulsory then that I have to give from. And then I am saying that I want a sequence by days not by years as in the earlier example, and I try to write down these days inside the double quotes, and I am now giving here length 6, I am not giving here the end date. So, I need here the 6 values of the dates, starting from 1st of January 2017 and these 6 values as to be incremented with respect to days. And as soon as I give it on the R console, I get here this type of outcome. So, you can see here this is 1st January, this is 2nd January, this is 3rd January, this is 4th January, and so on 5th January and 6th January 2017.

So, this is how I can generate the dates and here is the screen shot, that we can try yourself right.

(Refer Slide Time: 19:38)

```
Sequences
Generating sequences of dates

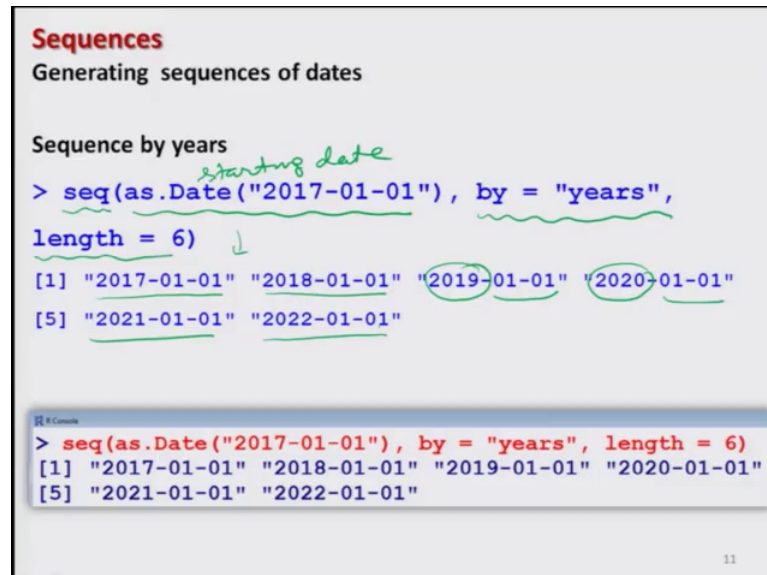
Sequence of months
> seq(as.Date("2017-01-01"), by = "months",
length = 6)
[1] "2017-01-01" "2017-02-01" "2017-03-01" "2017-04-01"
[5] "2017-05-01" "2017-06-01"

> seq(as.Date("2017-01-01"), by = "months", length = 6)
[1] "2017-01-01" "2017-02-01" "2017-03-01" "2017-04-01"
[5] "2017-05-01" "2017-06-01"
```

Similarly, I am trying to take the same example, where I am trying to generate the data with respect to months. So, here I am trying to take the same a starting days in the format of sequence, say the data starting from 1st of January, written as dot capital D ate and inside the bracket the starting date starting date and now I am saying that it has to be by months, not by years or days as in the earlier example. And I want a sequence of length 6 so; that means, the data has to be start from 1st of January 2017 and then after that I need a regular a dictator at an interval of 1 month, and I need 6 values. So, as soon as I enter this command in the R console, I get this output. So, you can see here this output is containing 1st January, 1st February, 1st March, 1st April, 1st May, and say here first of

June. Remember this year this is here, month and this is here day and this is here the screen shot of the same data that you can try yourself.

(Refer Slide Time: 20:57)



The screenshot shows a slide titled "Sequences" with the subtitle "Generating sequences of dates". Under the heading "Sequence by years", there is a code snippet in R: `> seq(as.Date("2017-01-01"), by = "years", length = 6)`. The starting date "2017-01-01" is annotated with a green handwritten note "starting date" and a green underline. The output of the command is displayed in two lines: `[1] "2017-01-01" "2018-01-01" "2019-01-01" "2020-01-01"` and `[5] "2021-01-01" "2022-01-01"`. The dates "2019-01-01" and "2020-01-01" are circled in green. Below the code, there is a small window titled "R Console" showing the same command and output. The slide number "11" is visible in the bottom right corner.

Now, I try to get the same data, with respect to years. So, I need here a sequence, and this is my starting date. Starting date is again the same, 1st of January 2017. And now I want this data to be arranged in the years right and I need 6 values so; that means, the values will start from 1st of January 2017, after that it will go for 6 years at an interval of 1 year. So, you can see here that as soon as I entered this command over the R console, the outcome is 1st January 2017, 1st January 2018, 1st January 2019, 1st January 2020, 1st January 2021, and 1st January 2022.

So, you will get all this data and the data can be arranged in the sequence that can be used earlier right. And, similarly in case if you want to write a more general program where you do not want to fix the start date and end date inside the program.

(Refer Slide Time: 22:01)

```
Sequences  
Generating sequences of dates  
  
To find sequence with defining start and end dates  
  
> startdate <- as.Date("2016-1-1") | Jan 2016  
> enddate <- as.Date("2017-1-1") | Jan 2017  
  
> out <- seq(enddate, startdate, by = "-1  
month")  
[1] "2017-01-01" "2016-12-01" "2016-11-01" "2016-10-01"  
[5] "2016-09-01" "2016-08-01" "2016-07-01" "2016-06-01"  
[9] "2016-05-01" "2016-04-01" "2016-03-01" "2016-02-01"  
[13] "2016-01-01"
```

But, you want to control it yourself, that as soon as you specify a start date or say end date the outcome should come in a particular format, that also can be done that this in that case the start date and end date they have to be assigned a value in the form of a variable. So, for example, if I want to control each and everything in this format, then I would say specify here say date of a start and date of end. Well, I am trying to just for the sake of better understanding, I am trying to a give it a variable name start date. So that you can remember, this otherwise you can take any value x, y, z, a, b, c, whatever you want. And simply I try to a two choose here two variable start date and end date and I try to give the same value here in this one.

As date and whatever is my start date and say end date. For example, here I am giving the start date to be here 1st January 2016 and the end date is 1st January 2017. And suppose I want a data from end date, to say start date and this increment is going to be minus 1 that mean it is going to be here decrement of minus 1 month.

As soon as, you try to use this data you get here this type of data. Therefore the sake of illustration I try to do it for you so that you can gain some more confidence, see I am defining here the start date, then I am defining here the end date.

(Refer Slide Time: 23:57)

```
Sequences  
Generating sequences of dates
```

```
> startdate <- as.Date("2016-1-1")  
> enddate <- as.Date("2017-1-1")  
> out <- seq(enddate, startdate, by = "-1 month")  
> out  
[1] "2017-01-01" "2016-12-01" "2016-11-01" "2016-10-01"  
[5] "2016-09-01" "2016-08-01" "2016-07-01" "2016-06-01"  
[9] "2016-05-01" "2016-04-01" "2016-03-01" "2016-02-01"  
[13] "2016-01-01"
```

13

And, then I am trying to write down, because here this output in the variable out. Now as soon as you press here and see the output here comes out to be like this. So, you can see here all this data has been arranged in the decrement, that is starting from 1st of January 2017, this is coming to be 1st December 2016, then 1st of November 2016, then 1st of October 2016 and so on. And this is the screenshot of whatever we have done now, so that you can try yourself.

(Refer Slide Time: 24:45)

```
Sequences  
Generating sequences of letters
```

letters is used to find sequence of lowercase alphabets

```
> letters  
[1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n"  
[15] "o" "p" "q" "r" "s" "t" "u" "v" "w" "x" "y" "z"
```

```
> letters  
[1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n"  
[15] "o" "p" "q" "r" "s" "t" "u" "v" "w" "x" "y" "z"
```

14

Now, after this time and dates, there is another option to generate the data. We all know that there are 26 letters in the English alphabets, a, b, c, d up to z and these letters can be written in say lowercase or say uppercase, what we call in common language as a small letter alphabet and capital letter alphabet right.

So, in programming many times, we would like to use this alphabets to generate a sequence and then in turn later on we can identify index vector corresponding to the letters, and based on that we can recall a particular letter in the programming language. So in order to do so, let us try to understand how to do it. First of all there are two commands to generate a sequence of letters, in uppercase and lowercase right. So, now, we are trying to take up the issue how to generate a sequence of letters. So, a word letters all written in say lowercase letters is a used to find out the sequence of lowercase alphabets.

So, as soon as you type here letters, you will get here this type of outcome and you will get here all a small letter alphabets a, b, c, d, e, f, up to here x, y, z and this is the screenshot here. They will try to show you over the R console also, but before that let us try to take some example from this sequence. Now you have to understand, one thing that whatever outcome you are getting here inside the letters, that is actually inside a vector and all these values are combined by here c.

(Refer Slide Time: 26:33)

**Sequences**  
Generating sequences of letters

`letters[from_index:to_index]` is used to find sequence of lowercase alphabets from a particular index to a specified index.

`> letters[1:3]`  
[1] "a" "b" "c"

`> letters[3:1]`  
[1] "c" "b" "a"

`> letters[21:23]`  
[1] "u" "v" "w"

`> letters[2]`  
[1] "b"

Handwritten annotations: A table at the top right shows 'a' at index 1, 'b' at index 2, 'c' at index 3, and '...' at index 26. Green arrows point from the text 'index' to the '1' and '3' in the first code example, and from 'index' to the '3' and '1' in the second code example.

So, now, in case if you want to recall a particular letter from this sequence of alphabets. This is done by the index vector corresponding to this vector of alphabets. What is this index vector?

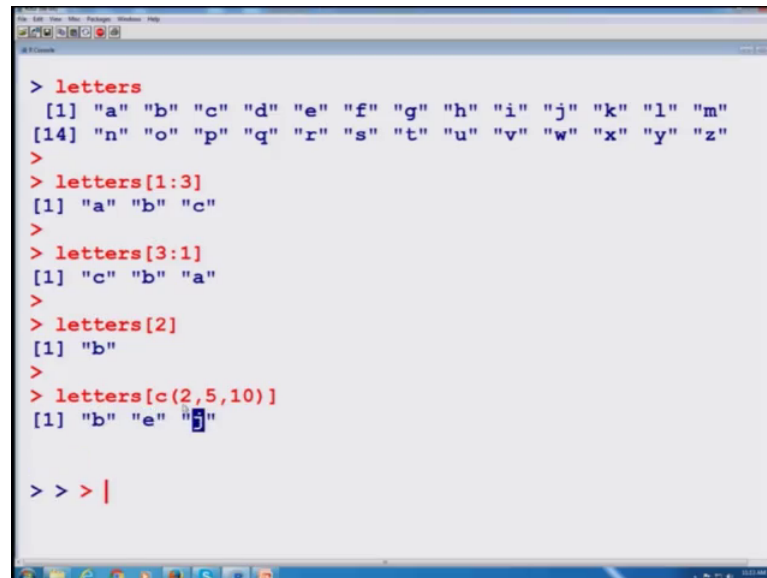
Now, suppose from the sequence of alphabets, I want to say extract some of the alphabets only. So, the command for this thing is this, that I have to write down here the command letters in the form of letters all in small letters lay lowercase, then I have to give here the starting value here from and then this is separated by a colon and then I have to give here 2, and what is this here from and to? You have to remember one thing that, this from and to they are coming from this index vector. What is the index vector? As soon as you write down here the say here three letters here a, b and c, then the position of the alphabet a in the vector is 1, the position of the second alphabet is 2 and the position of the third alphabet is 3 and so on this will go up to here z, and z is at the 26 place. So, this is a sort of index vector which is giving an index automatically to the alphabets.

So, suppose I want to recall here the first 3 letters a, b and c. So, a, b and c they have got an index 1, 2, 3. So, I can write down here say letters inside this square bracket from the index position 1, 2 index position 3. And as soon as you try to do it here, you will get here a, b and c. The same thing can be done in the reverse ordering also, suppose I want a sequence of letters in the format say c, b and a. So, I can give here the index position here a say 3 which is of c, and separated by this colon, and then the index position of a.

And, as soon as you try to enter you will get this outcome which is in the format of c, b and a. And it is not that you have to start or say and only form c here first, you can give it here any number. For example, if I want to generate the alphabets from index position 21 to 23, then I have to write then 21 colon 23 inside the square bracket followed by this syntax letters, and we get here the letters at the 21st position, 22nd position and 23rd positions which are u, v and w respectively. And suppose if I want to recall only one particular element that is also possible. Suppose I want to recall the alphabet at the second position. So, I have to give it here letters and inside the square brackets only here to right.

So, you can see here, I can get here because here b. So, why not to do all these things on the R console here and, try to see how it happens.

(Refer Slide Time: 29:57).



```
> letters
[1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m"
[14] "n" "o" "p" "q" "r" "s" "t" "u" "v" "w" "x" "y" "z"
>
> letters[1:3]
[1] "a" "b" "c"
>
> letters[3:1]
[1] "c" "b" "a"
>
> letters[2]
[1] "b"
>
> letters[c(2,5,10)]
[1] "b" "e" "j"
> > > |
```

For example, first of all I try to type here letters, you can see here I am getting the all 26 letters. Now, suppose I want to have here letters from 1 to 3. I am to getting here 1 to 3. Suppose I want to get here from letter 3 to 1 index positions, I am getting here this 1.

Suppose, I want to get here only a letter at position number 2, this I am getting here like this thing. Now suppose I want to have some letters, which are at say position number say 2, 5 and 10. So, I can write down a vector of indices 2, 5 and 10 and I try to enclose it using the brackets followed by the c command, combined command and as soon as I try to do it here, I will get here three letters b that 2nd position, e is at 5th position and j is at 10th position.

So, you can see here all sorts of operations are possible. So, here is the screenshot whatever I did and you can practice it.

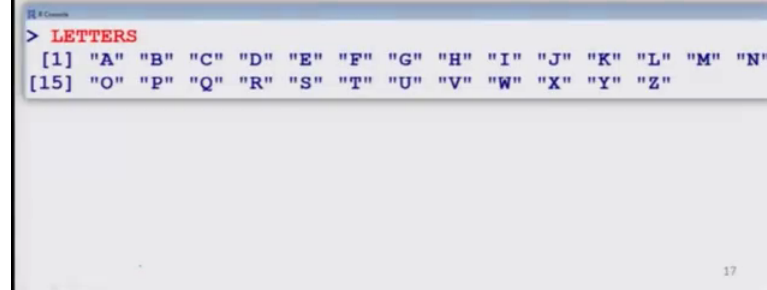


(Refer Slide Time: 31:23)

**Sequences**  
Generating sequences of alphabets

**LETTERS** is used to find sequence of uppercase alphabets

```
> LETTERS
[1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N"
[15] "O" "P" "Q" "R" "S" "T" "U" "V" "W" "X" "Y" "Z"
```



17

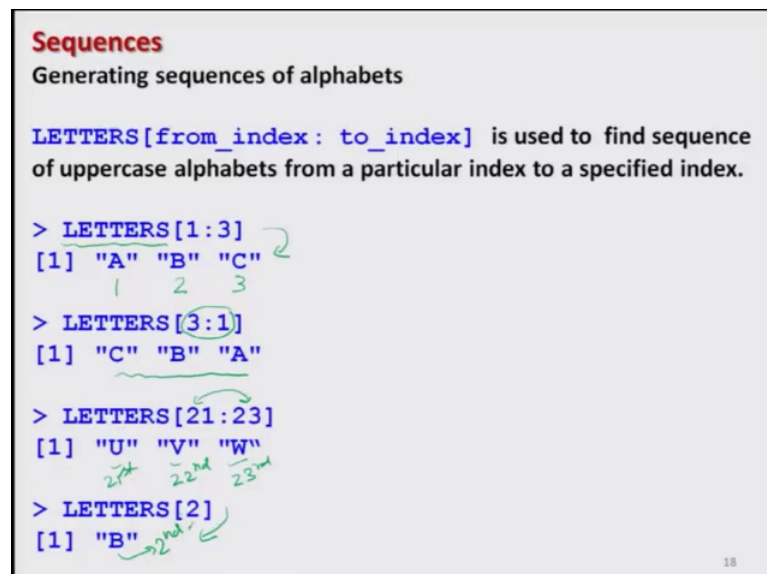
Now, similar to is small a letter or say lowercase I can also do the here the same thing with the capital letters or the uppercase alphabets in English. So, similarly if you want to generate a sequence of say uppercase or say capital letters. You simply have to use the command say letters all written in the uppercase or the capital letters. And as soon as you try to do it here, you get here the same sequence of alphabets and exactly on the same line as you did earlier.

(Refer Slide Time: 31:56)

**Sequences**  
Generating sequences of alphabets

**LETTERS[from\_index: to\_index]** is used to find sequence of uppercase alphabets from a particular index to a specified index.

```
> LETTERS[1:3]
[1] "A" "B" "C"
     1  2  3
> LETTERS[3:1]
[1] "C" "B" "A"
> LETTERS[21:23]
[1] "U" "V" "W"
     21st 22nd 23rd
> LETTERS[2]
[1] "B"
     2nd
```



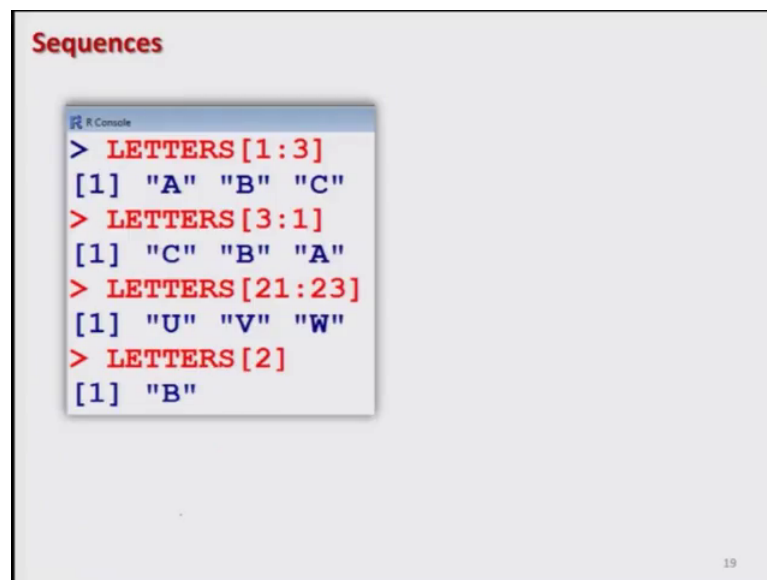
18

If you want to take out any particular letter or say any sequence of particulars the letter you have to use the same command that we use with say small letters or say lowercase alphabets with the uppercase alphabets say letters.

For example, I try to repeat the same example what I did earlier, that instead of a small letters I am trying to take here the capital letters. And I want to have a sequence of here 1, 2, 3. So, you can see here that I am getting here the letter capital letter at position number 1, capital letter at index 2 and the third letter at index number 3 that is capital C. And the reverse can be obtained by the same thing with letters inside the square bracket, 3 colon 1 and this will gave me the A, B, C in the reverse order, that is c, b, a and similarly if I want to have here the positions from 21 to 23rd in the indices vector, then I get here U, V and W. The U is at 21st position, V is at 22nd position, and W is at 23rd position. And if I want we have a particular letter form a particular position, I can give it here here say letters and then 2 and then b is at the 2nd position.

So, this is how I can obtain different types of thing and you can see here.

(Refer Slide Time: 33:22)



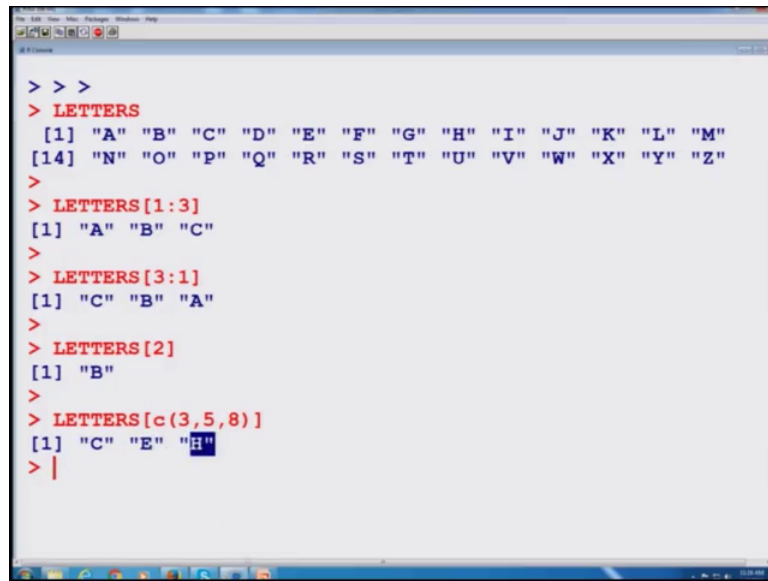
The screenshot shows an R console window titled "R Console" with the following text:

```
Sequences  
> LETTERS[1:3]  
[1] "A" "B" "C"  
> LETTERS[3:1]  
[1] "C" "B" "A"  
> LETTERS[21:23]  
[1] "U" "V" "W"  
> LETTERS[2]  
[1] "B"
```

The word "Sequences" is written in red at the top left of the console window. The R prompt ">" is in red, and the code and output are in blue. A small number "19" is visible in the bottom right corner of the console window.

This is the screenshot and because for the for the sake of your understanding, I will try to do it here on the R console also.

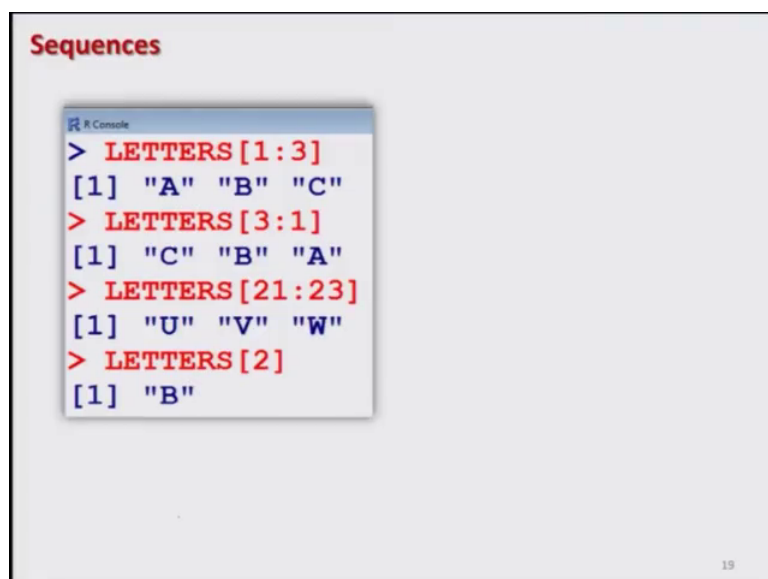
(Refer Slide Time: 33:36)



```
> >
> LETTERS
[1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M"
[14] "N" "O" "P" "Q" "R" "S" "T" "U" "V" "W" "X" "Y" "Z"
>
> LETTERS[1:3]
[1] "A" "B" "C"
>
> LETTERS[3:1]
[1] "C" "B" "A"
>
> LETTERS[2]
[1] "B"
>
> LETTERS[c(3,5,8)]
[1] "C" "E" "H"
> |
```

You can see here, you are getting here the same thing and if you want to recall here, any particular type of sequence you can say here letters A, B, C and if you want to have it in the reverse order you can have a it here, letter 3 to 1. If you want to have a here only a particular position say letter at 2, you get here 2. And if you want to have here (Refer Time: 33:58) let us say more than one position at a time say in letter number at 3rd index, 5th index or see it 8th index. Let us I can give in the form of a vector c 3, 5, 8 and as you enter you get here three letter C is at position number 3, E is a position number 5 and 8 is at position number 8 right. So, after this we stop here with this lecture.

(Refer Slide Time: 34:25)



```
Sequences
R Console
> LETTERS[1:3]
[1] "A" "B" "C"
> LETTERS[3:1]
[1] "C" "B" "A"
> LETTERS[21:23]
[1] "U" "V" "W"
> LETTERS[2]
[1] "B"
```

And we have done with the aspects of sequence although that is difficult to take each and everything, but I have taken sufficient number of examples, through which you can have a fair idea that, how do we generate the data using a sequence command. So, my request to all of you is that you try to practice it, and in the next turn we will come up with some more commands till then goodbye.