

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

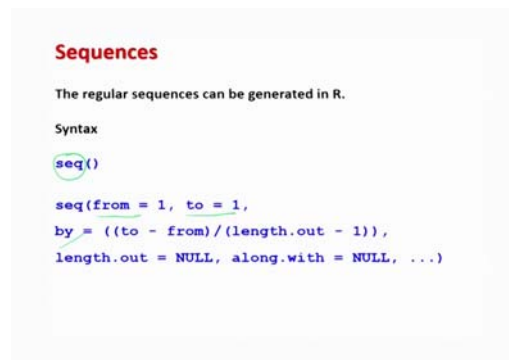
Lecture - 27
Sequences of Dates and Alphabets

Hello friend, welcome to the course Foundations of R Software and you can recall that in the last two lectures we were talking about the sequences and we had considered different types of command, their options to generate, various types of sequences in the R software. Now, in this lecture we are going to continue with the same topic, but we are going to learn here how to generate the sequence of a different type that is the sequence of the dates and sequence of the alphabets.

Up to now what you have done you have considered the sequence which are numerical. So, the first question comes here, why should I learn it? So, you know that whether whenever you are trying to handle different databases and you want to generate some report or you want to manipulate some data file, it is possible that sometime you will need some particular dates which are in a sequence like a January 21, February 21 etc.

So, how to do it in the R software and similarly you have alphabets like as a b c d they are in lower case and upper case. So, somehow so, sometime we want to generate the sequences of these alphabets so, the question is how to get it done? So, why not to begin this lecture and try to learn? So, I will try to take care some example and through those examples I will try to explain you how you can generate the sequences of dates and alphabets.

(Refer Slide Time: 01:59)



Sequences

The regular sequences can be generated in R.

Syntax

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

So, let us begin our lecture. So, you can recall that in the last two lectures we had learned the sequence command in which the command was `seq` in which you need couple of ingredients like from where you want to begin the sequence, where you want to stop the sequence, what is the increment, how you can control it. So, that is going to be the by and some other sequences and after that we also have learnt for example, how to use the colon command to generate a sequence.

(Refer Slide Time: 02:24)

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.

30 Nov 21
30 Nov 22
30 Nov 23
30 Nov 24
30 Nov 25

So, now in this lecture today first we want to learn that how to generate the sequences of dates. right. So, sequences of dates for example, could be suppose 30th November to 21 and suppose you want to generate a sequence of 5 dates and the increment is going to be either by day, by week, by quarter or by year. So, suppose if you want to generate say 5 dates which are incremented at a annual level. So, this will be 30th November 21, 30th November 22, 30th November 23, 30th November 24 and 30th November 2025 and so on.

So, this is how the sequence of dates will look like. So, now, the basic command to generate the sequence is the same as we had learnt earlier that is `seq`, but the way you are trying to give different types of option like from, to, by, etc. they are going to be little bit different, right. So, for example, if you try to see here the way you have used the sequence command, first you have to give the 'from'.

So, now this from is going to be the starting date and it is a compulsory argument, that you have to give this value. Then you have to give the next value here I say to that is to is the and date that when you want to finish your sequence at what date. So, that is an optional and then we have here the third value which is here by.

So, by can be here day, by can be here week, month, quarter or year. So, what is the use? That if you want to get a sequence in which the dates are changing by the day then you have to write by is equal to this "day" inside the double quotes. Similarly, if you want to get the dates on a weekly basis then you have to write down here week and then you have to write "week" inside the double course and so on.

And similarly you have here length out and along with so, this length out is an integer and it helps in getting the desired length of the sequence and along with is the command which take the length from the length of this argument, right. So, we will try to take here some examples and try to understand how are you going to give this input, so that you get the output of the required type.

(Refer Slide Time: 04:30)

The image shows a slide titled "Sequences" with the subtitle "Generating sequences of dates". It contains R code and its output, annotated with handwritten notes. The code is: `> seq(as.Date("2010-01-01"), as.Date("2017-01-01"), by = "years")`. The output is: `[1] "2010-01-01" "2011-01-01" "2012-01-01" "2013-01-01"` and `[5] "2014-01-01" "2015-01-01" "2016-01-01" "2017-01-01"`. Handwritten notes include: *seq(from, to, by)*, *1 Jan 2010 → yearly*, *1 Jan 2017*, *to =*, and *year month day*. A second code block shows the same command with a plus sign: `> seq(as.Date("2010-01-01"), + as.Date("2017-01-01"), by = "years")` with the same output.

So, suppose first I try to consider; how to generate the sequences of years, suppose, I want to generate the sequence of first day of years, right. So, ok, the way I am going to write down these commands here is like the first value is going to be here from, inside the arguments inside the parenthesis.

Second value is going to be to and third value is going to be here by, because here the commands are quite long. So, I am just trying to express them in the shorter way, so that they can be accommodated on my screen. That is the only reason otherwise you can always write say from equal to this to equal to this and by equal to this that is the correct approach actually.

Now, suppose I want to generate the dates starting from 1st of January 2010 to 1st of January 2017 and these dates are going to be changed yearly. So, now, you have to see how are we going to give this input. Well, it is not difficult for you to understand, you see first I am writing here from and from I am writing here like this suppose I want to give you the date 1st January 2010. So, this is going to be like “2010 hyphen 01 hyphen 01”.

So, this is here year, this is here month and this is here day and this is written inside the double quotes. And then we want to read this value as date, right because that can be character also or in order to inform the R to read this number as a date we try to give here a command as dot Date, but you have to be very careful that a s that is in lower case, alphabet capital D is here in the date and then after that all other alphabets ate they are in the lower case and then so, this become as dot Date.

So, now, you already have done couple of such commands where you used to have something like as like as dot numeric, as dot character etc. So, you now you know that what is the meaning of this as our date. So, similarly as you have learnt about this as dot numeric, as dot character so, similarly we have here as dot Date and this is the way I would like you to learn this course content in this lecture because if I try to cover each and everything it will be a very long lecture.

So, I am trying my best to give you the basic fundamentals. So, once I have told you that how you can use such command as dot numeric, as dot character, then it should be obvious that when I am trying to use as dot Dates that is simply going to convert the given number into a date, right. So, I give here from and then I try to give it here to as date and then I try to write down here in the same format “2017 minus or say hyphen 01 hyphen 01”.

Now, after this I have to write down here by. So, by because I need these dates to be incremented by years so, I try to write down here inside the double quote “years” and if you try to see here this outcome will be there the first value is going to be “1st January 2010” then, “1st January 2011”, then “1st January 2012”, 2013, 14, 15, 16. And finally, 1st January 2017 and this is here the screenshot. So, you can see here this is not a very difficult thing to understand.

(Refer Slide Time: 07:43)

```
Sequences
Generating sequences of dates

Sequence of days

> 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"

> 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"
```

And let me try to give you some more example and then I will try to show you on the R console. So, similarly if you want to generate this sequence of days, right and now well, if you want to take the same example here and you want to change this years to date, months, etc. you can do it without any problem, right.

But now I want to generate here a sequence of days where I note the starting date, but I do not know the last date, but my requirement is that I need 6 dates, right. So, suppose for example, I want to generate the 6 dates starting with 1st January 2017. So, now, what will be those dates 1st January, 2nd January, 3rd January, etc.

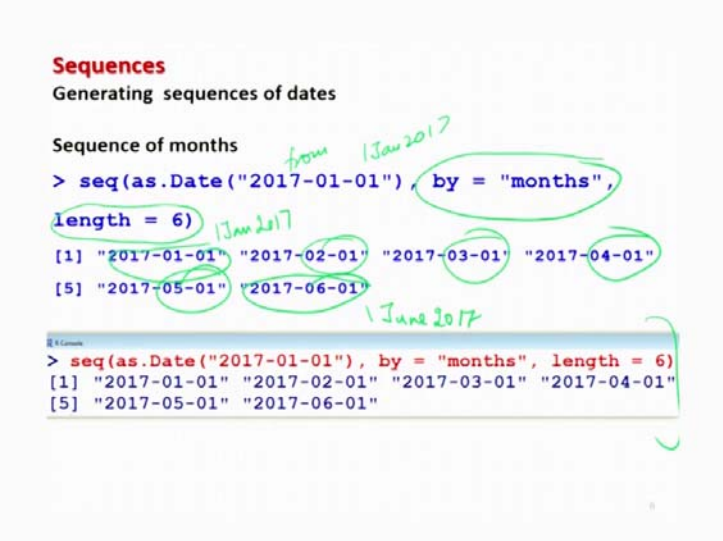
So, in order to do that thing you can recall in the case of sequence you had learnt that when you know the length of the sequence then how you can use the from command to generate the sequence of the required length. So, the same command I am going to use

here and I write down here seq then I try to write down here inside the parenthesis as as dot Date. So, this is going to be your here from.

So, this is the same format as I explained you, right and then you have to write here by is equal to "days" and then you have to write down here length is equal to 6. So, you can recall that earlier you had used this command length to inform the R that how many values in the sequence are required.

So, the same thing I am trying to do here that I am trying to inform the R that ok I need 6 dates which are beginning from 1st January 2017 and you can see here this is here "1st January 2017", then "2nd January 2017", "3rd January", "4th January", "5th January" and finally, "6th January 2017" and this is here the screenshot, right. So, you can see here that we have the same command sequence, but now it can be used in a different way for different job.

(Refer Slide Time: 09:33)



The screenshot shows an R console window with the following content:

```
Sequences
Generating sequences of dates

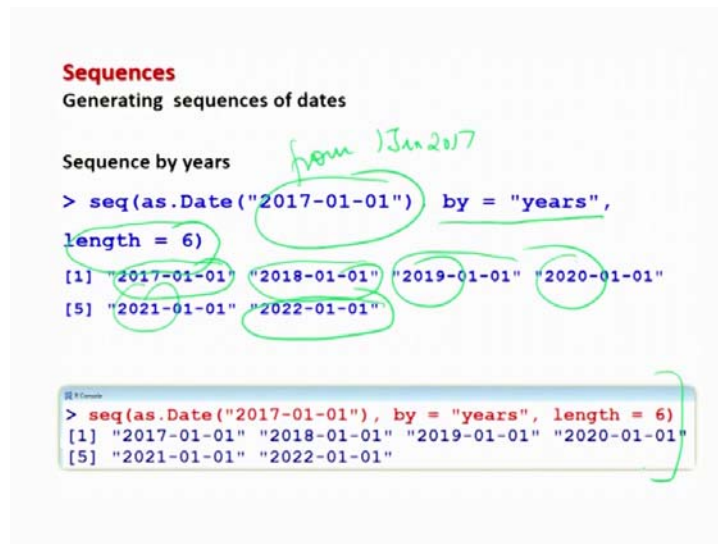
Sequence of months
> seq(as.Date("2017-01-01"), by = "months",
length = 6)
[1] "2017-01-01" "2017-02-01" "2017-03-01" "2017-04-01"
[5] "2017-05-01" "2017-06-01"
```

Handwritten green annotations include: "from 1 Jan 2017" above the first parameter, "1 Jan 2017" above the first date, and "1 June 2017" above the last date. Green circles highlight the "by = \"months\"", "length = 6)", and the date strings. A blue box highlights the second command and its output.

And similarly, in this case suppose I want to generate six dates which are changing with respect to the month. So, here I have used here the by days. So, I can change here to by here "months". So, you can see here the starting date is here from which is 1st January 2017 and then after this I want to get here six dates and the dates are going to be changed with respect to the month.

So, I write down here length is equal to 6. So, you can see here the first date is here 1st January 2017, then after that 1st February, 1st March, 1st April, 1st May and finally, 1st June 2017, right and these are 6 values. So, you can see here this is not a very difficult thing the only thing is what you have to understand that how you are going to input the value of the dates and after that how are you going to use the command for the by different options for the by, right.

(Refer Slide Time: 10:26)



Sequences
Generating sequences of dates

Sequence by years *from 1 Jan 2017*

```
> seq(as.Date("2017-01-01"), by = "years",  
length = 6)  
[1] "2017-01-01" "2018-01-01" "2019-01-01" "2020-01-01"  
[5] "2021-01-01" "2022-01-01"
```

```
> seq(as.Date("2017-01-01"), by = "years", length = 6)  
[1] "2017-01-01" "2018-01-01" "2019-01-01" "2020-01-01"  
[5] "2021-01-01" "2022-01-01"
```

The screenshot shows two R console sessions. The first session includes handwritten green annotations: a note "from 1 Jan 2017" above the start date, and circles around the start date "2017-01-01", the "by = 'years'" parameter, the "length = 6" parameter, and each of the six date outputs. The second session is a clean screenshot of the same command and output.

And similarly in the same example suppose I getting the monthly dates I want to get here the annual dates in which I want to start from 1st of January 2017 and then I want to generate here 6 dates. So, the length is going to be here 6 and I want to increase the date with respect to “years”. So, I try to say here like this and the outcome is here like this 1st January 2017, 1st January 2018, 2019, 2021 and 1st January 2022 and you can see here this is the screenshot.

(Refer Slide Time: 11:05)

```
Sequences  
Generating sequences of dates  
  
To find sequence with defining start and end dates  
  
> startdate = as.Date("2016-1-1") 1 Jan 2016 seq(10, 2, by=-2)  
> enddate = as.Date("2017-1-1") 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"
```

So, you can see here this is not a very difficult thing to do, the only thing is this you have to just understand that how the things are going to happen. So, similarly actually whatever I have told you here whatever the output you are going to get for the dates that can also be saved in a variable and these values can be used in the for the programming. So, let me try to give you here an example suppose I want to generate the dates from beginning from 1st January 2016 and up to say 1st January 2017, right.

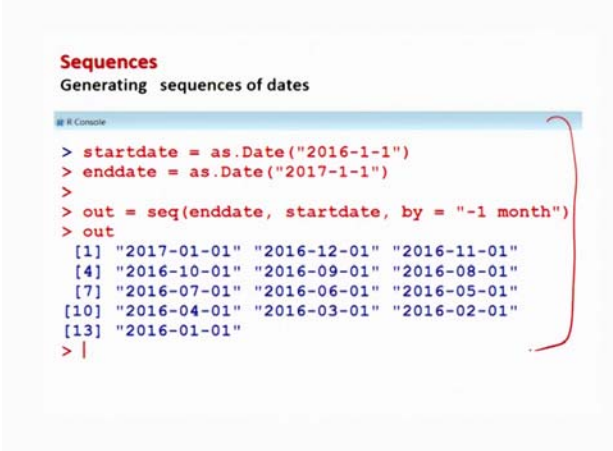
So, whatever is the date of a start I try to store it in a new variable start date and the final date that is stored in another variable end date, right. So, I try to write down here the sequence say end date start date and now I want to show you here one more command. If you try to see here in the by what I have done here, this is here "-1 month", right, this option is also there. So, now, if you try to say here by say "-1 month" and you try to give it within the double quote, see what happens.

The first value comes out to be 1st January 2017 then it is going into the reverse direction that 1st December 2016, 1st November 2016, 1st October 2016 and so on finally, up to 1st January 2016. So, this 1st January comes here starting date and this is here end date. So, what are you trying to do, do you think that when you done the sequence you had done something like you wanted to have a decreasing sequence.

So, you wrote from 10 to say 2 and by you had written by here -1 or -2 like this. Well, in case if you try to write down sequence start date and end date and by say year or month

the outcome can be saved in another variable for example, here I have taken the variable to be here out, right.

(Refer Slide Time: 12:56)

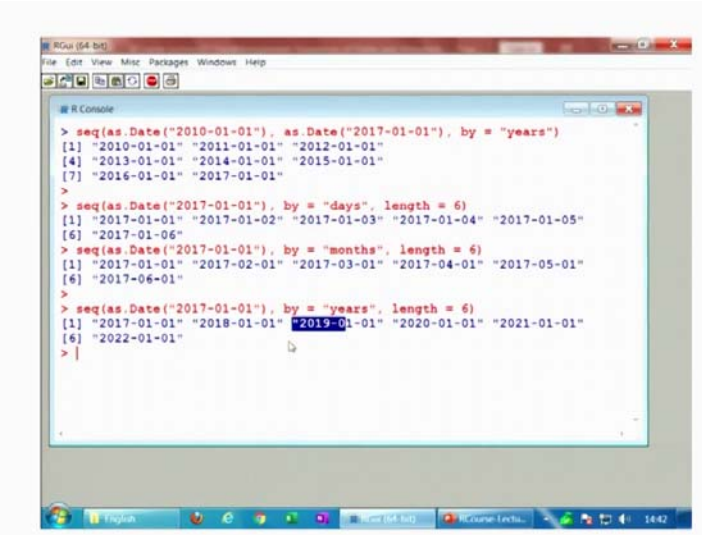


```
Sequences
Generating sequences of dates

R Console
> 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"
[4] "2016-10-01" "2016-09-01" "2016-08-01"
[7] "2016-07-01" "2016-06-01" "2016-05-01"
[10] "2016-04-01" "2016-03-01" "2016-02-01"
[13] "2016-01-01"
> |
```

And this is here the screenshot of the same operation which I just shown you. So, now let me try to show you these operations on the R console. So, that you become more confident that these things are really working. So, first I try to compute these dates which are from 1st January 2010 to 1st January 2017.

(Refer Slide Time: 13:16)

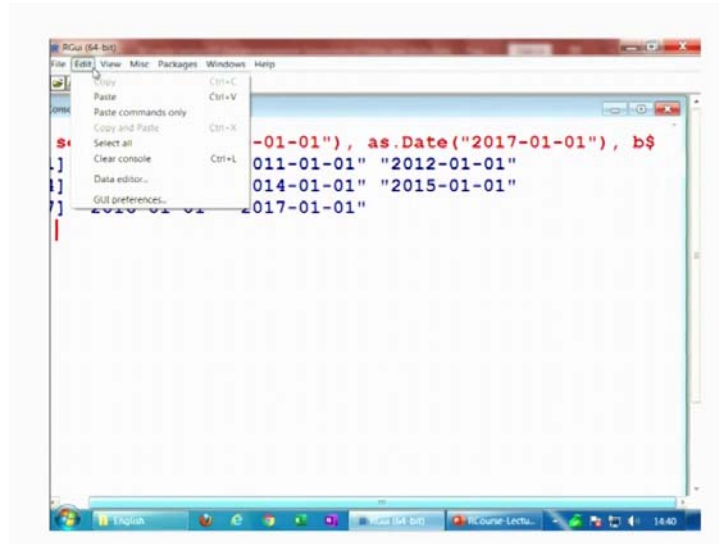


```
RGui (64 bit)
File Edit View Misc Packages Windows Help

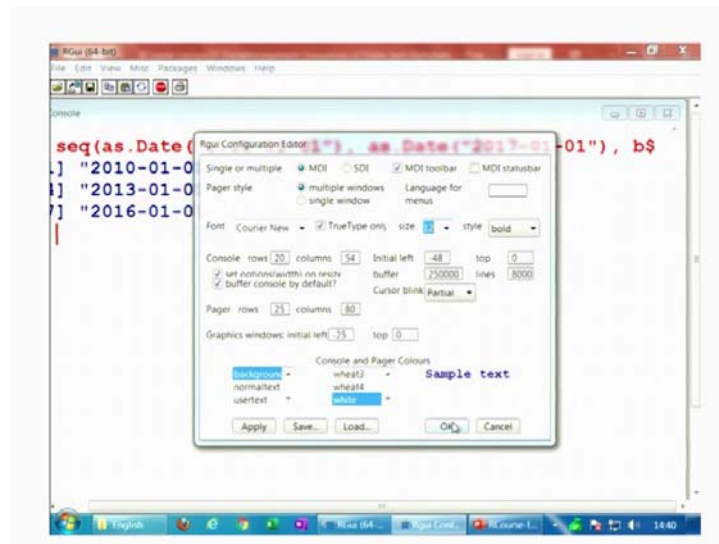
R Console
> seq(as.Date("2010-01-01"), as.Date("2017-01-01"), by = "years")
[1] "2010-01-01" "2011-01-01" "2012-01-01"
[4] "2013-01-01" "2014-01-01" "2015-01-01"
[7] "2016-01-01" "2017-01-01"
>
> seq(as.Date("2017-01-01"), by = "days", length = 6)
[1] "2017-01-01" "2017-01-02" "2017-01-03" "2017-01-04" "2017-01-05"
[6] "2017-01-06"
> seq(as.Date("2017-01-01"), by = "months", length = 6)
[1] "2017-01-01" "2017-02-01" "2017-03-01" "2017-04-01" "2017-05-01"
[6] "2017-06-01"
>
> seq(as.Date("2017-01-01"), by = "years", length = 6)
[1] "2017-01-01" "2018-01-01" "2019-01-01" "2020-01-01" "2021-01-01"
[6] "2022-01-01"
> |
```

And the dates are increasing annually. So, we can see here that this is going to happen.

(Refer Slide Time: 13:23)



(Refer Slide Time: 13:23)



Well, I can reduce here the font size, so that you can see it easily, right. You can see here this is how you can change the location of your screen and similarly if you try to come here for this command that you want to generate here the days which are changing by days and you want six days. So, you can write down here starting from 1st of January it gives you here six days.

And similarly if you want to do here by months or by year, whatever you want you can write down here by month then and then you can see here what happens in months. And

similarly if you want to do it by years so, you can see here you can change here by as equal to “years” and it is giving you here six days 1st January 2017, 18, 19 etc. So, you can see here this is not a very difficult thing to do; the only thing is this you have to understand how are you going to do. So, let me try to just show you here.

(Refer Slide Time: 14:22)

```

> 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" "2016-09-01"
[6] "2016-08-01" "2016-07-01" "2016-06-01" "2016-05-01" "2016-04-01"
[11] "2016-03-01" "2016-02-01" "2016-01-01"
>
> out = seq(enddate, startdate, by = "months")
Error in seq.int(r1$mon, 12 * (to0$year - r1$year) + to0$mon, by) :
wrong sign in 'by' argument
> seq(startdate, enddate, by="months")
[1] "2016-01-01" "2016-02-01" "2016-03-01" "2016-04-01" "2016-05-01"
[6] "2016-06-01" "2016-07-01" "2016-08-01" "2016-09-01" "2016-10-01"
[11] "2016-11-01" "2016-12-01" "2017-01-01"
> out1=seq(startdate, enddate, by="months")
> out1
[1] "2016-01-01" "2016-02-01" "2016-03-01" "2016-04-01" "2016-05-01"
[6] "2016-06-01" "2016-07-01" "2016-08-01" "2016-09-01" "2016-10-01"
[11] "2016-11-01" "2016-12-01" "2017-01-01"
> |

```

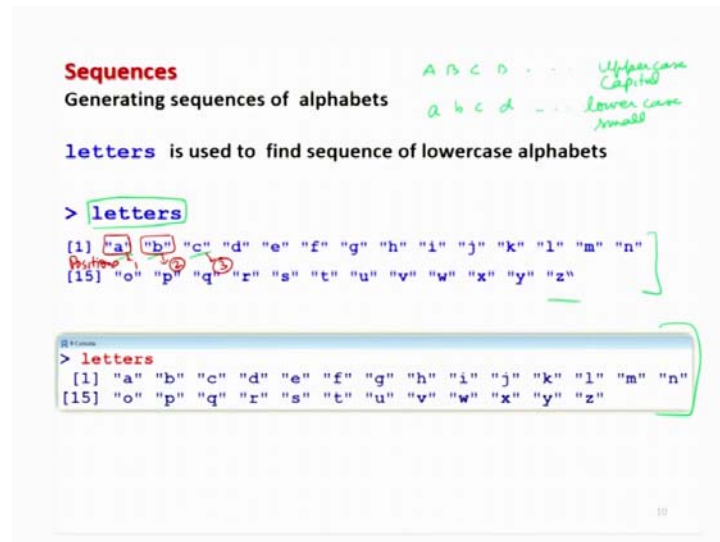
This example which is very interesting so, you can see here, I store here this variable start date as 1st January 2016 and then I try to store here end date as 1st January 2017 and suppose I try to have here this outcome that I want to store here the values starting from end date to start date by -1 by “-1 month” which so you can see here, now this is here out like this.

So, you can see here this is the same outcome, right and now if you try to see here that in this case if you try to rewrite this thing and suppose I try to give it only by month, then you see what happens? What do you expect, what will happen? There is going to be an error, why?

You see; this end date is coming earlier and start date is coming later. So, do not do this type of mistake, but if you really want to do it you have to write down here like this that sequence start date then to end date. And then you have to write down here by say here same months and you will get here these values.

So, you can see here and if you want to store these values also you can give it under the new variable name say out l and you can see here out l is here like this, right same thing. So, that is how this our R actually works, right. So, you can see here that generating the dates is not a very difficult thing and let me try to increase my font, so that when I come back here next I can show you things more clearly.

(Refer Slide Time: 15:52)



```
Sequences
Generating sequences of alphabets

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"
```

The screenshot shows an R console window with the following content:

- Section header: **Sequences**
- Section header: **Generating sequences of alphabets**
- Text: **letters** is used to find sequence of lowercase alphabets
- Command: `> letters`
- Output: `[1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n"`
- Output: `[15] "o" "p" "q" "r" "s" "t" "u" "v" "w" "x" "y" "z"`

Handwritten annotations in green and red are present:

- Green text: "A B C D . . ." above "a b c d . . ."
- Green text: "Upper case Capital" above "a b c d . . ."
- Green text: "Lower case small" below "a b c d . . ."
- Red circles around "a", "b", "c", "p", "q", "r" in the output.
- Red text: "Position" next to the first row of the output.
- Green boxes around the command and the output.

So, I hope you have understood that how you can generate different types of sequence for the dates. Now I come to another aspects you know that you have alphabets like as A B C D etc. So, they are upper case alphabets or sometime you call it in common letter as say capital letter. And similarly you have here alphabet small a small b small c small d etc. these are your here lower case alphabets they are called in common language as a small letters.

So, in this R you can also generate the sequence of alphabets. So, how to get it done that is what we are aiming here to learn. So, the first command here is `l e double t e r s` all in lower case alphabets, this function is going to give you all the alphabets in the lower case you can see here a small “a” small “b” up to here “z” and this is here the screenshot.

(Refer Slide Time: 16:42)

Sequences
Generating sequences of alphabets

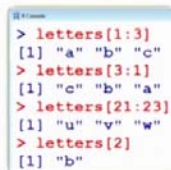
`letters[from_index:to_index]` is used to find sequence of lowercase alphabets from a particular index to a specified index.

`> 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 notes: "Starting point" above "from_index", "end point" above "to_index", "location" above "1:3", and "a b c" with "1 2 3" below it.

And similarly, in case if you want to generate a sequence of this alphabet, so the question here is, how are you going to do it? So, first you try to understand let me try to show you here. Suppose, if you try to see this "a" is coming here, this "b" is coming here. So, if you try to understand what are the positions or location of this alphabet? The position here is 1 for "a", for the "b" the position here is 2, for "c" the position here is 3, and so on.

So, now, it is just like the index of these alphabets. So, this particular alphabets they can be called by their location and also a sequence can be generated by using their locations. So, if you try to see here in order to generate a sequence or you want to call a particular alphabet, what you can do?

You can simply use this command `l e double t e r s` all in lower case alphabets and then inside the square bracket you try to write from to means the starting point. Starting point in the index or the location and then here colon and then here to this is the end point, right.

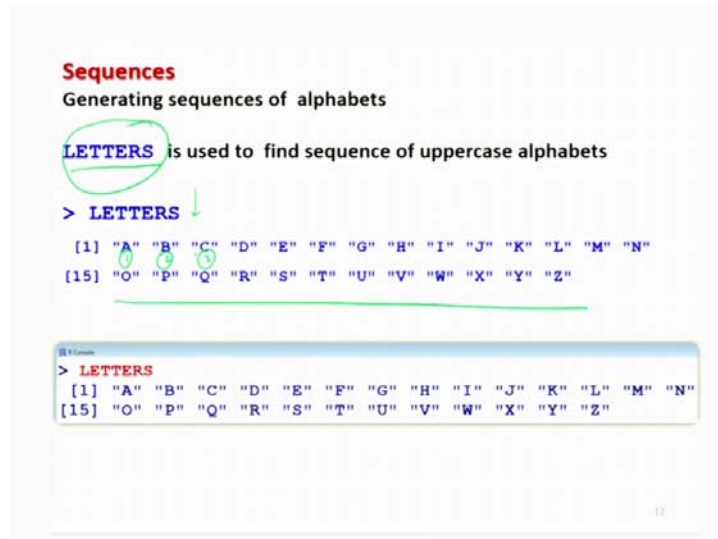
So, for example, means if I want to have the lowercase alphabets which are occurring at per second and third positions. So, I will try to write down here this lower case alphabets `l e double t e r s` then inside the square brackets you will write here 1 colon 3. So, these

are the locations of your alphabets. So, now, you can see here that the first 3 alphabet a b c their location is 1, 2 and 3 respectively they will come here, right.

Similarly, if you want to have a sequence of lowercase alphabets from the position number 3 to 1. So, you begin here writing lower case that l e double t e r s and then inside the square bracket you write 3 colon 1, right and this will give you 3 alphabets “c” “b” and “a”. Similarly, in case if you want to have any arbitrary positions also. For example, you want to know what are the alphabets at the location 21 to 23.

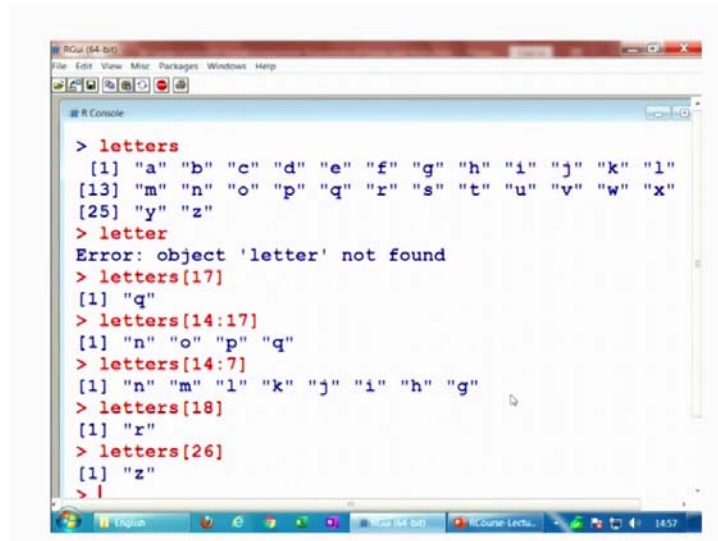
So, you can see here l e double t e r s you can write inside the square bracket you can write here 21 colon 23 and this will be coming here as a “u” “v” and “w”. And similarly, if you want to find out any particular alphabet at a given location then you can simply write down l e double t e r s in lower case alphabets and then within in the square brackets you write down here 2, right and this will give you here the second alphabet that is “b”.

(Refer Slide Time: 19:13)



So, you can see here that is not a very difficult thing to understand but, before going further let me try to show you these things on the R console.

(Refer Slide Time: 19:17)



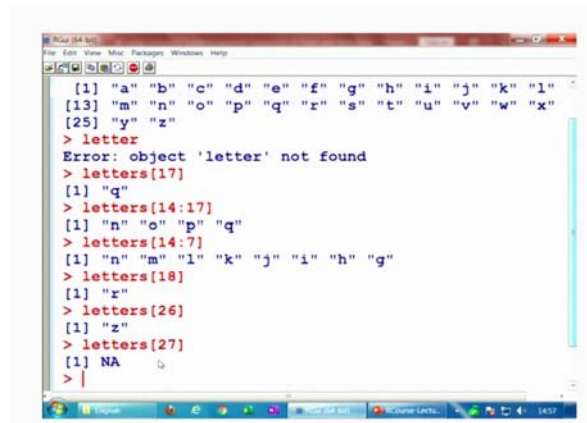
```
> letters
[1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l"
[13] "m" "n" "o" "p" "q" "r" "s" "t" "u" "v" "w" "x"
[25] "y" "z"
> letter
Error: object 'letter' not found
> letters[17]
[1] "q"
> letters[14:17]
[1] "n" "o" "p" "q"
> letters[14:7]
[1] "n" "m" "l" "k" "j" "i" "h" "g"
> letters[18]
[1] "r"
> letters[26]
[1] "z"
> |
```

So, if you try to see here this letters will give you this thing and remember one thing if you try to write down here only letter l e double t e r without s this will give you error because it does not exist. So, now, if you try to see here in letters if you want to know what are the alphabet at the 17th position, what will happen?

It is “q” and if you want to have the letters from say 14 position to 17 position you can see here these are “n” “o” “p” “q” and how you can find out this is the 13th position which is here “m” so, 14th is “n”, 15th is “o”, 16th is “p” and 17th is “q”. And similarly, if you want to have it from here 14 to 7 in the reverse order so, that can also happen here, right.

And suppose, if you want to know that what is in the letter at suppose here say 18 position, you can see here this is “r” and similarly if you try to see here what is the letter at 26 position this is here “z”. Now, what will happen if you try to give any other number?

(Refer Slide Time: 20:13)



```
RStudio [64-bit]
File Edit View Misc Packages Windows Help
[1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l"
[13] "m" "n" "o" "p" "q" "r" "s" "t" "u" "v" "w" "x"
[25] "y" "z"
> letters
Error: object 'letter' not found
> letters[17]
[1] "q"
> letters[14:17]
[1] "n" "o" "p" "q"
> letters[14:7]
[1] "n" "m" "l" "k" "j" "i" "h" "g"
> letters[18]
[1] "r"
> letters[26]
[1] "z"
> letters[27]
[1] NA
> |
```

Suppose I say 27, there is NA, do you know why? Because there are only 26 alphabets so, that is why if you are trying to give any index after this 26 which is 27 here it is saying that it is not available, right, ok. So, you can see here it is not a very difficult thing. So, similarly if you try to understand that whatever we have done for the lower case alphabet that can be done for the upper case alphabet exactly in the same way.


So, in order to generate the uppercase alphabets we have the command here L E double T E R S, but everything is written in the capital letters in the upper case alphabets and if you try to execute it on the R console you will get here all these alphabets capital A capital B and so on. So, now, the same thing is capital A is at the location number 1, capital B is at the position number 2, capital C is at the position number 3 and so on.

(Refer Slide Time: 21:04)

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

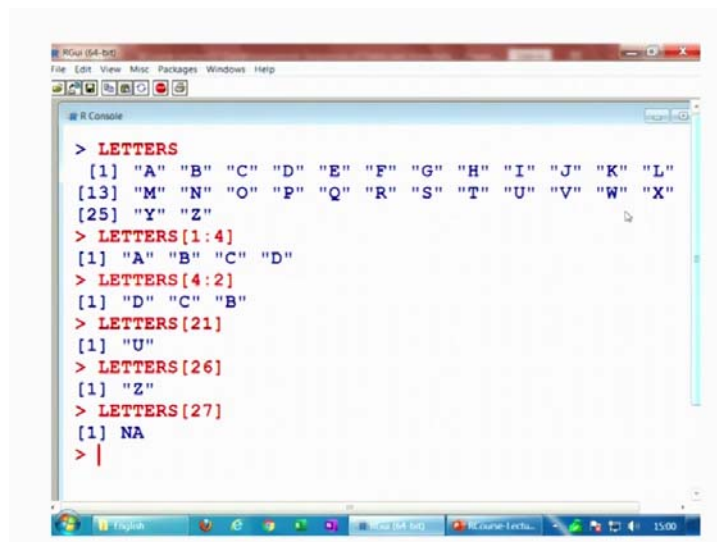


So, similarly, we can generate the sequence of upper case alphabets exactly in the same way as we have done in the case of lower case alphabets. So, suppose I want to generate uppercase alphabets which are located at the position number 1 to 3. So, I write here L E double T E R S all in upper case then square bracket and then I write down here 1 colon 3. So, that will give me a 3 letters “A”, “B” and “C”.

Similarly, if I want to go in the reverse direction then I try to type here L E double T E R S and from 3 to 1. So, this will give me here 3 values here, 3 alphabets which are in the reverse order at third position, second position and first position that is “C “B” and “A”. Similarly, if you want to know what are the alphabets at the location number 21 to 23, you can write down here L E double T E R S in the upper case and then you try to write down here 21 and 23 and it will give you this “U” “V” and “W”.

And similarly, if you want to know what are the alphabet at the location number 2, then you try to give here L E double T E R S inside the square bracket and then it will give you here the alphabet capital “B” which is at the second location. So, you can see here this is not a very difficult thing, but let me try to show you it on the R console. So, that you get more confidence.

(Refer Slide Time: 22:17)



```
> LETTERS
[1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L"
[13] "M" "N" "O" "P" "Q" "R" "S" "T" "U" "V" "W" "X"
[25] "Y" "Z"
> LETTERS[1:4]
[1] "A" "B" "C" "D"
> LETTERS[4:2]
[1] "D" "C" "B"
> LETTERS[21]
[1] "U"
> LETTERS[26]
[1] "Z"
> LETTERS[27]
[1] NA
> |
```

So, if you try to see here I try to write down here LETTERS. So, this gives you the 26 alphabets. So, now if you want to write down here what are the LETTERS from 1 to 4

this will be “A” “B” “C” “D” and similarly if you want to have what are the letters from the location number 4 to 2 that will be “D” “C” and “B”. And similarly, if you want to see here what is the location number say here 21, you can see here this is “U”, similarly if you want to see what is the alphabet location number 26, this is here “Z”.

Now, once again if you try to write down here what is the alphabet at 27 position this will give you NA because there are only 26 alphabets. So, this is how you can actually work with the sequence of alphabets also. So, now, we come to an end to this lecture and you can see here that we have now considered different types of sequences of numbers of dates and then alphabets also.

So, these sequences are going to help us in different ways that you will realize when you start doing the real life programming, right. But my objective was that I wanted to make you learn the basic ingredients of the programming that these are the different components.

So, if you try to learn all these components at the end whenever you need it you can use them. So, now, my request to you all is that now we have used the sequence command to generate different types of dates the sequence of dates which are monthly, annual, by days, etc. etc.

So, why do not you try to experiment with all the options which I had shown you in the slide, I have taken here only couple of them and then go back to your lecture in the when we had learnt about the sequence and we had used different types of options of say from, to, by, etc. Why do not you try to implement those concepts over here and try to see what happens. The more you practice the more you learn better you are. So, you try to practice it and I will see you in the next lecture, till then Goodbye.