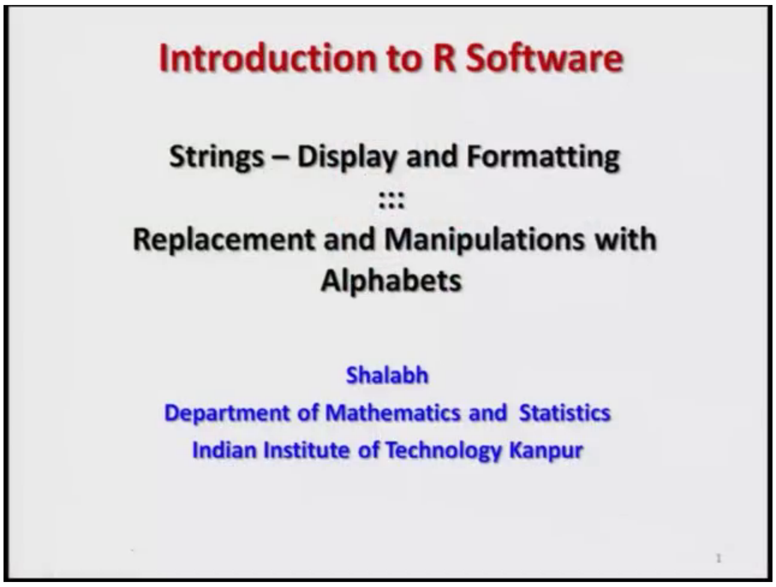
Introduction to R Software

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

Lecture – 28
Replacement and Manipulations with Alphabets

Welcome to the next lecture on the course introduction to R software, in this lecture we will continue with our earlier topic that that we wanted to learn different types of commands function and aspects of displaying an output. So, now, in this lecture we are going to deal with 2 aspects one is a replacement of a string and second thing is this manipulations with the alphabets like as you want to change the lowercase alphabets into uppercase or vice versa.

(Refer Slide Time: 00:40)



Introduction to R Software

Strings – Display and Formatting
:::
Replacement and Manipulations with Alphabets

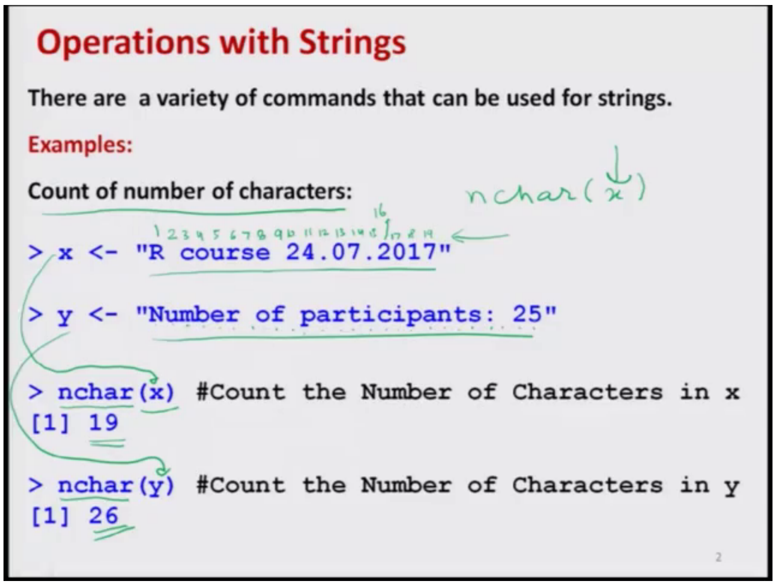
Shalabh
Department of Mathematics and Statistics
Indian Institute of Technology Kanpur

1

So, first of all we talked about that in case if we have a string then how to count the numbers of characters in the string for example, you have got a sentence and you want to know that how many letters are there or how many characters are there. for example, in say many software you see for example, in say ms word there is always a number in the bottom on the left side which counts that how many letters you have typed or how many characters you have typed.

So, similarly in any programming language in different places you may need to count the number of characters, sometimes you have seen that whenever you are trying to fill up an online form then that always says that you have a limit of say 200 characters and as soon as you type more than 200 characters it will say no this cannot type or it will give you an option that now 20 characters left 18 character characters left and so on. So, this type of option is useful in such situations.

(Refer Slide Time: 02:02)



Operations with Strings

There are a variety of commands that can be used for strings.

Examples:

Count of number of characters:

nchar(x)

```
> x <- "R course 24.07.2017"

> y <- "Number of participants: 25"

> nchar(x) #Count the Number of Characters in x
[1] 19

> nchar(y) #Count the Number of Characters in y
[1] 26
```

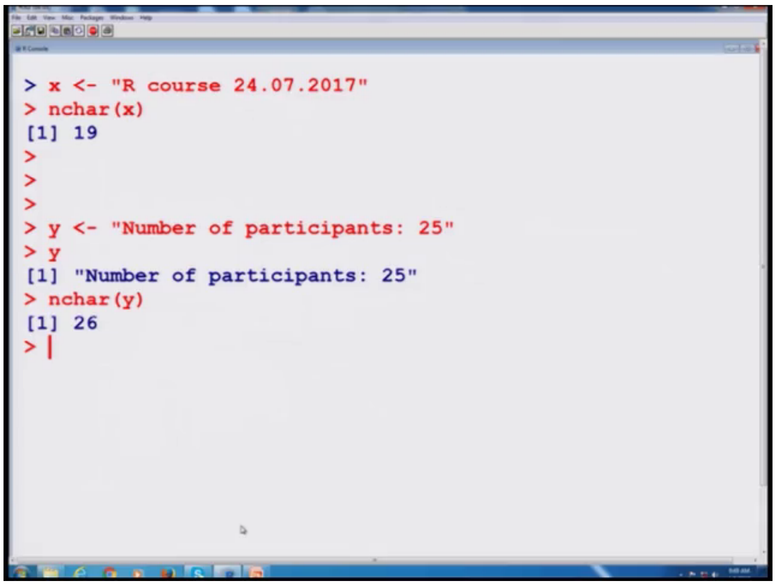
So, now, the option is this, suppose I want to count the number of characters in a string, the option or the command or the function for this is n char and inside this argument you have to write down the string of which you want to count the number of characters. Let me take here an example and try to understand, suppose I write see here 1 sentence R course 24 july 2017 right. So, if you try to count how many characters are here, 1 then a blank a space 2 then c 3, o 4, u 5, r 6, s 7, e 8 then a blank space 9 to 10, 4, 11 this full stop dot 12, 0 to 13, 7, 14, full stop 15 this is 16, 0 is 17, 1 is 18 and 7 is here 19.

So, you have to note down that I am also counting the blank a spaces, blank a space is also taken as a character. Now when I try to a use here a command say nchar and inside the argument inside the bracket I write x; that means, I want to count the number of characters in this statement. In this string and you can see here that we have counted here 19 characters and here you get here the answer s say 19 and similarly if you try to take another string, suppose I write number of participants colon blank space 25.

So, similarly instead of counting the characters by 1, 2, 3, 4 in this expression I can use here the command nchar that is the number of characters n y and it gives you here the answer say 26. So, there are 26 characters in this say string and you can count it 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21 22, 23, 24, 25 and 26 so there are 26 dots.

So, let us try to do this thing over the R console and see what happens.

(Refer Slide Time: 04:53)



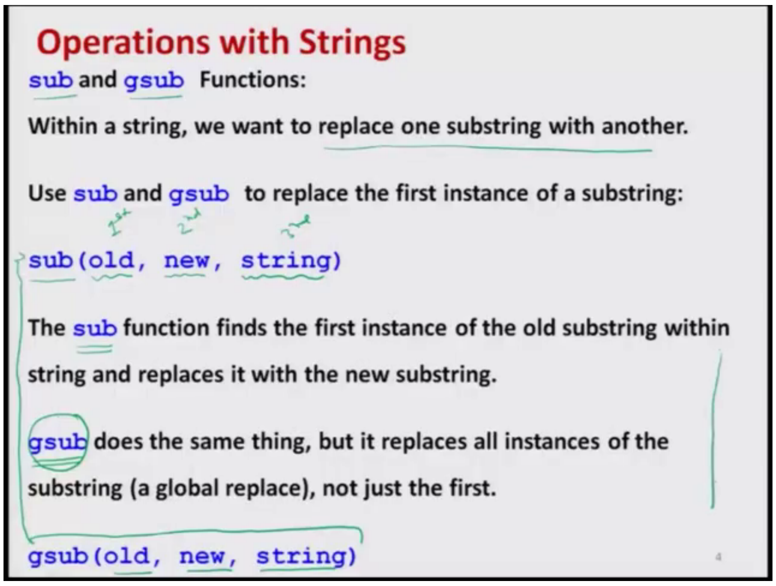
```
> x <- "R course 24.07.2017"
> nchar(x)
[1] 19
>
>
>
> y <- "Number of participants: 25"
> y
[1] "Number of participants: 25"
> nchar(y)
[1] 26
> |
```

So, you can see here and then I will count here in character number of characters in x this is 19 and similarly here if I try to count down the number of characters in y then, this is my here y you can see here and which I want to find out here the number of characters in y this is 26. So, that is a very small application, but it is very useful in programming in several situations.

So, here you can see this is the screenshot of the outcome that we have just performed, now if we consider another application you have seen that whenever you are typing a document in any word processors like s ms office word or so on, sometimes you have typed something and you want to replace that word with something else. So, you go to menu and say replace and then it ask that what you want to replace and in the second option it ask to buy what you want to replace and then you say ok and then it goes either a step by step or if you say that no replace all then it will replace all the words with the new word in the entire document.

So, we can have a situation that we are writing a program in which we want to get an outcome in which some string, some characters, some words, some numbers they are going to be replaced by some new word, character, string or same number. So, how to get it done? how to do a replacement and in order to do this replacement in R with a strings we have 2 options. First option is that it will replace only the first match that it encounters or second option is that that it will replace all the matches in the entire document that is globally and in order to do such a replacement of a strings we have 2 commands here one is s u b sub and another this g sub. These are the 2 functions which helps us in replacing one substring with another within a string.

(Refer Slide Time: 07:34)



Operations with Strings

sub and gsub Functions:

Within a string, we want to replace one substring with another.

Use sub and gsub to replace the first instance of a substring:

```
sub(old, new, string)
```

The sub function finds the first instance of the old substring within string and replaces it with the new substring.

gsub does the same thing, but it replaces all instances of the substring (a global replace), not just the first.

```
gsub(old, new, string)
```

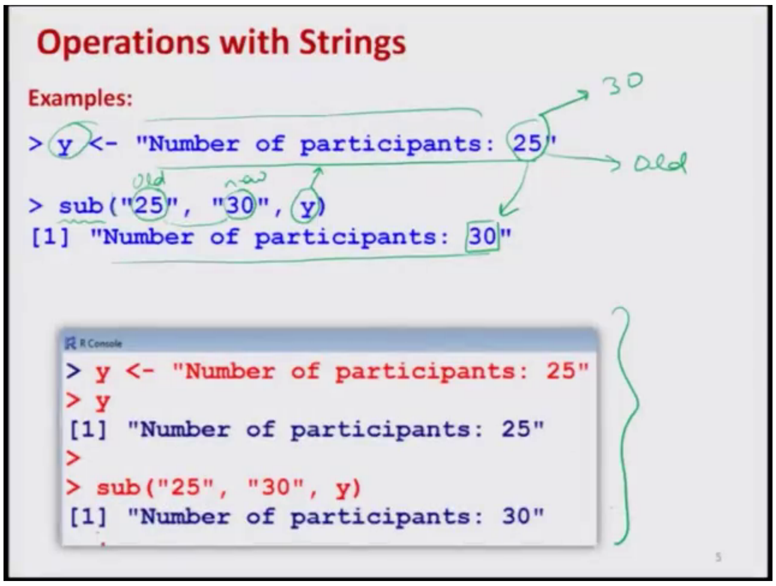
So, we have got a string and we want to replace some words which are occurring in the string by some new words or say new characters. So, the syntax for using sub function is that we write here the sub then in the first place we write whatever is the old string, whatever is existing in the string that we want to replace and then separated by comma we write another, this option which is the new string this is always at the second place. New string means the string by which you want to replace the earlier string and in the third place we have string. So, here we have to write the string in which we want to do the operation in which we want to replace the older characters by a some new characters and similarly we have say here another function g sub and g sub also has a similar syntax, say older is string, newer string and the string in which you want to do the replacement.

So, you can see here the structure of sub and g sub that is same, but then the question is what is the difference between the two the difference is in their outcome, when we are trying to use sub function then it replaces only at the first encountered place. For example, if I have a statement then the R control will start from the first character it will move from left to right and as soon as it finds the first encounter the first place where it can match the character it will just replace it and then it will stop.

But when we are using the command g sub, then g sub also starts from the first character and it will move from left to right as soon as it encounters the first match it will replace it by a new string new character. But it will not a stop here it will move forward and then it will try to find the next match and as soon as it finds the next match it will replace the new string with the older string and this process will continue till the control reaches to the end of the string.

So, the difference between sub and g sub commands is that sub is a local one, local in the sense that it will replace only the first match wherever it occurs whereas, g sub is the global command it will replace the characters in the entire string right. So, that is the basic difference which I have written here, now let us try to take some examples and we try to understand how it happens.

(Refer Slide Time: 11:19)



Operations with Strings

Examples:

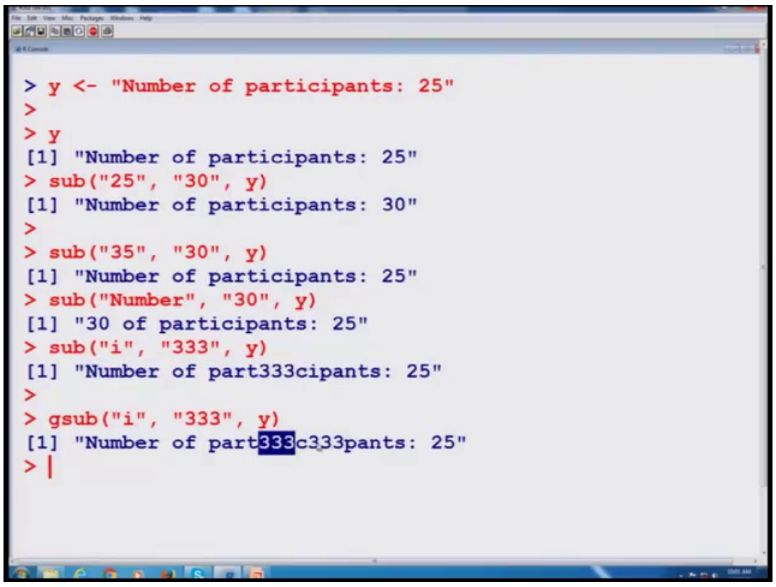
```
> y <- "Number of participants: 25"

> sub("25", "30", y)
[1] "Number of participants: 30"
```

```
> y <- "Number of participants: 25"
> y
[1] "Number of participants: 25"
>
> sub("25", "30", y)
[1] "Number of participants: 30"
```

So, suppose I try to take here a statement here or a string number of participants colon 25 right, now what I want here is that I want to replace this 25 by 30. So, what I would do that I would try to write down here sub and inside this argument I will write the older one, what is here old that is 25. This is our old string which is existing in the a string y, what is here y? Y is the a string in which I have stored this sentence number of participants and this is separated by say here new string. What do you mean by new string? The older existing character in the string y is 25 and I want to replace it by 30 in that cell 30 is the new set of characters and 25 is the old set of characters and this I have to give it here in this y that is the string that is the number of participants colon 25 and here you can see the outcome this will come out to be that this 25 is a replace by here this 30.

So, here is the screenshot, but we would like to do it over the R console. So, that you can have some confidence yes that things are really happening. So, you can see here why is my disc statement and that is my function.

(Refer Slide Time: 13:27)



```
> y <- "Number of participants: 25"
>
> y
[1] "Number of participants: 25"
> sub("25", "30", y)
[1] "Number of participants: 30"
>
> sub("35", "30", y)
[1] "Number of participants: 25"
> sub("Number", "30", y)
[1] "30 of participants: 25"
> sub("i", "333", y)
[1] "Number of part333cipants: 25"
>
> gsub("i", "333", y)
[1] "Number of part333c333pants: 25"
> |
```

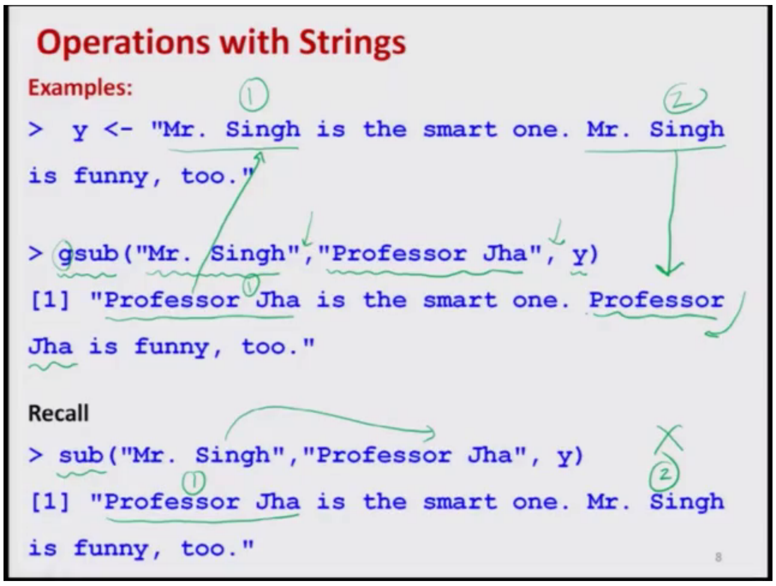
So, you can see here that this 205 is replaced by 30, now suppose I try to play with this thing which I always ask you to do it suppose I say I want to replace 35 by 30.

Let us see what happens it is not doing anything because there is nowhere 35 in the string y and suppose if I say here that I want to replace the number by 30 in the string y. You can see here here I had got the number, but here now I have got a 30 and suppose if I see here that I want to replace the character here I by say head say 333. Let us see what happens, you can see here wherever I have here the force encountered I, you see here in this number there is no I, 10 here in off there is no I and then there is up to here part there is no i.

First encountered I here is at this place which is replaced by 333 now there are some more is for example, here i, but this sub command will not execute because sub command what it does it will start from say here number from n it will move from left to right and wherever it encounters the first I this will match with this highlighted I and this I will be replaced by this 333 and that what is happening here that this is replaced and after this it will come back, it will not move forward and that is why this I is not a replaced. But I will try to show you in the next part that if I use g sub then this I will also will replace why not to do it here.

Let us see if I try to use here g sub you can see here that this I is also replaced by 333. So, you can see the difference between this outcome where there is only one I is replace and with this g sub both is are replaced first I second I right now let us come back to our slides ok.

(Refer Slide Time: 16:13)



Operations with Strings

Examples:

```
> y <- "Mr. Singh is the smart one. Mr. Singh
is funny, too."

> gsub("Mr. Singh","Professor Jha", y)
[1] "Professor Jha is the smart one. Professor
Jha is funny, too."
```

Recall

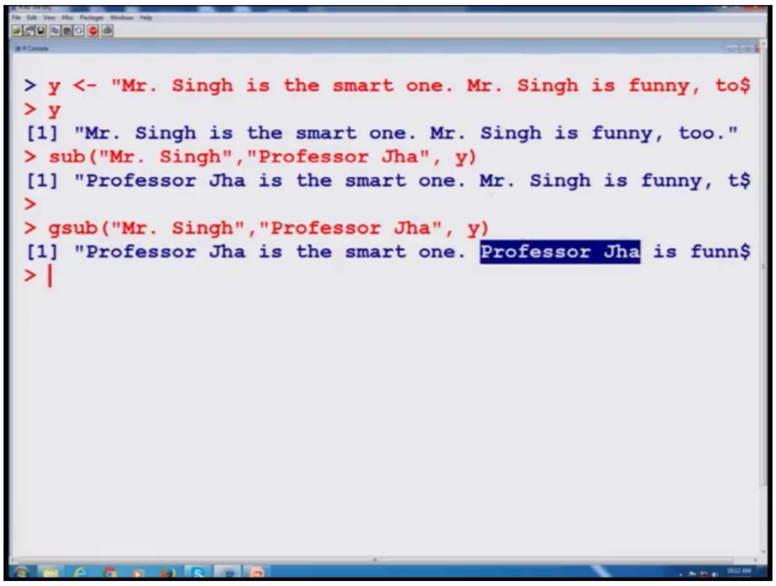
```
> sub("Mr. Singh","Professor Jha", y)
[1] "Professor Jha is the smart one. Mr. Singh
is funny, too."
```

Now, I take some more examples which will make this use of sub and g sub more clear, I take here one sentence mister Singh is the smart one and then mister Singh once again is funny too. Now my objective is that I want to replace this name mister Singh by professor Jha, now I have 2 options either I try to replace the first place where mister Singh is occurring mister Singh will be replaced by the name professor Jha or I want to replace the second place where mister Singh name is occurring also by professor Jha. So, in this situation first we try to see how the function sub box.

Now, first of all I have to give the command. So, I am saying that mister are please try to search for the name mister Singh and then wherever you are finding it in the first shot please replace it by professor Jha. Then are ask me where do you want to replace, then I say yes please use the string y which I already have informed here. So, now, R start from first character and it moves left to right, it will say I am going from here yes here I find sir mister Singh character and its simply replace mister Singh by see here professor Jha and it crosses it and the outcome comes over here you can see here that professor Jha is occurring here.

And now after this the R control will come back to its original position it will not move ahead and that is why you can see here this second day mister Singh is not replaced by professor Jha this will remain as such and here is the outcome in the screenshot, but let us try to do it. So, that you get confidence that yes the things are happening. So, I try to copy this command over here come back to R console create my here this y you can see here this is my here y and then I try to use the same command and let us see what happens this is happening.

(Refer Slide Time: 19:08)



```
> y <- "Mr. Singh is the smart one. Mr. Singh is funny, to$
> y
[1] "Mr. Singh is the smart one. Mr. Singh is funny, too."
> sub("Mr. Singh","Professor Jha", y)
[1] "Professor Jha is the smart one. Mr. Singh is funny, t$
>
> gsub("Mr. Singh","Professor Jha", y)
[1] "Professor Jha is the smart one. Professor Jha is funn$
> |
```

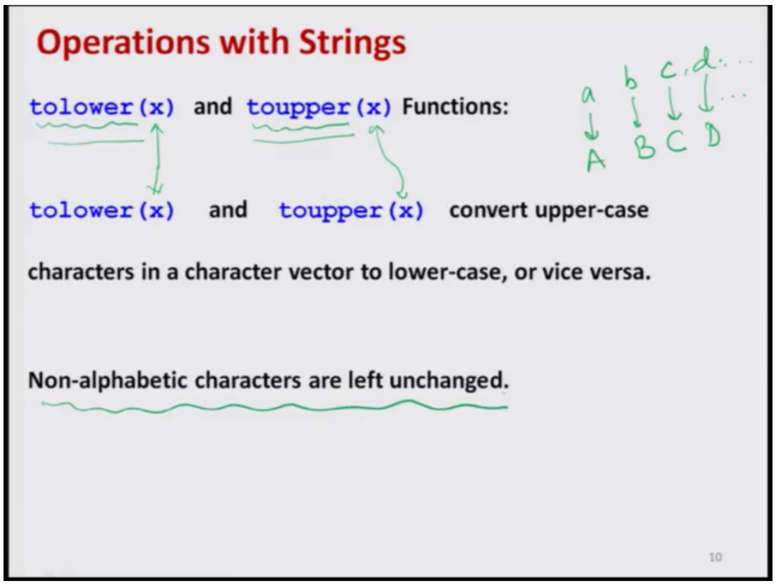
So, you can see here that mister Singh which was happening here in y is being replaced by professor Jha, but this is occurring at the first instance mister Singh name which was occurring at the second place in y this is not changed,, but you remain as such right, now we come back to our slides, this is the screenshot.

Now, I make another experiment, I take the same statement where mister Singh is occurring at 2 places place number 1 and place number here 2 and we had used just now the command sub and we had seen that in the outcome that mister Singh is being replaced by the name professor Jha is occurring only at the first place and second place here 2 that was not being replaced. So, I try to use here another command here g sub global substitution and I say mister R please try to look for the word mister Singh and please replace it by professor Jha in the string which is given by here y and all these commands are separated by this comma and now I am asking that here this is g so mister R please do it globally.

Now, this is here the outcome, this professor Jha this is at the one this is replaced by here mister Singh and now this m mister R Singh is also replaced here by this professor Jha and my sentence which was originally mister Singh is the smart one, mister Singh is funny too becomes professor Jha is the smart one professor Jha is funny too. So, these types of manipulations also you can do in this R and let us try to do this g sub in this one. So, I simply try to come back to my earlier command and I simply replace here sub by g sub we can see here now the things are happening. Both this mister Singh is replaced by professor Jha at first place and mister Singh at the second please this is also replaced by professor Jha in the second place ok.

So, now let us come back to our slides and here you can see the screenshot of the same operation.

(Refer Slide Time: 22:08)



Operations with Strings

`tolower(x)` and `toupper(x)` Functions:

`tolower(x)` and `toupper(x)` convert upper-case characters in a character vector to lower-case, or vice versa.

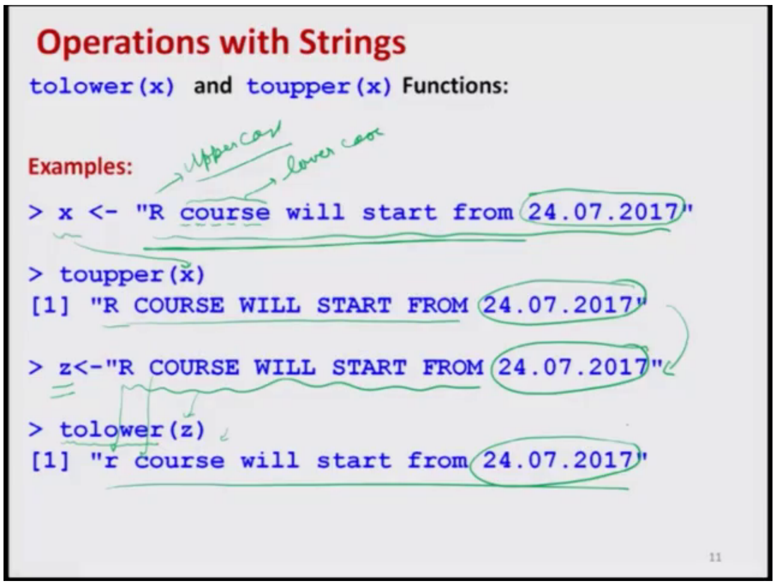
Non-alphabetic characters are left unchanged.

a b c, d...
A B C D
10

And now we come to another option that is available in R, sometimes you have done something and your output is coming in say lowercase letters that is small alphabets what we call in common language small a, small b, small c something like a, b, c and so on and suppose I want to change the lowercase in to uppercase; that means, I want to change the letter in say lowercase into uppercase; that means, I want to change small a into capital a, b into capital b and small c into capital c, small d into capital d and so on. So, I want to do this type of replacement that lowercase alphabets are replaced by uppercase alphabets or vice verse also; that means, the uppercase alphabets are changing to lowercase alphabets, this type of outcome can be controlled by 2 functions which are here to lower and to upper to the meaning of this function itself clarifies what we want. I want to change into lower case and with 2 upper I want to change into uppercase and what I want to change, whatever is my string in this say here x.

So, I will have a string x and if I use the command to lower or to upper they will try to change the lower into uppercase or say upper into lower case and in case if you have any non alphabetic character or any character which is not an alphabet then this will remain unchanged. So, this is an important point that you have to keep in mind that only the alphabets are going to be changed.

(Refer Slide Time: 24:17)



Operations with Strings

`tolower(x)` and `toupper(x)` Functions:

Examples:

```
> x <- "R course will start from 24.07.2017"

> toupper(x)
[1] "R COURSE WILL START FROM 24.07.2017"

> z<-"R COURSE WILL START FROM 24.07.2017"

> tolower(z)
[1] "r course will start from 24.07.2017"
```

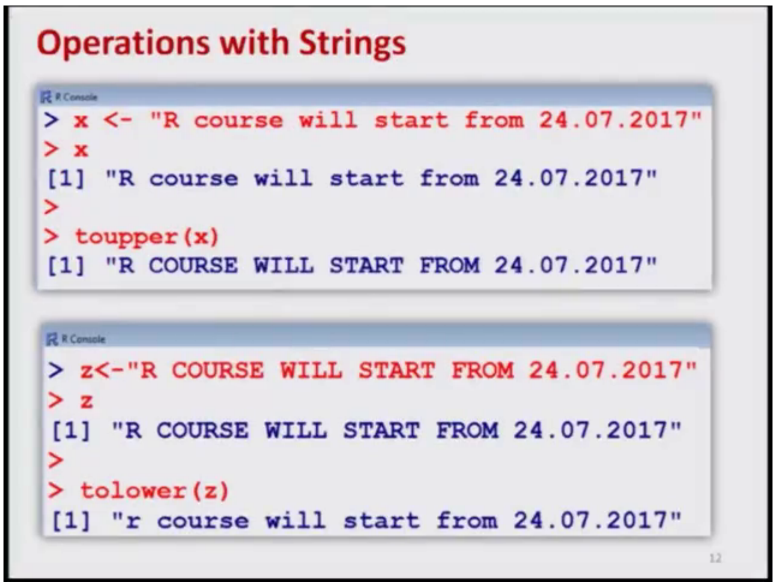
So, let us try to take one example and try to understand how things happen, suppose I have a string in which I have written a sentence some like R course will start from 24 07 2017 right and I have stored this statement in a variable say here x you can see here this is in the uppercase R. But this all these things are in the lower case a small alphabets have been used now I want to convert all the letters into upper case that this means small c becomes capital C is small o becomes capital O is small u becomes capital U is small r becomes capital R and so on and also you see here there are some here non alphabetic character which are numbers here.

So, now I try to use here the function to upper and to upper will convert all the characters contained in this here x from R course will start from into a an uppercase and so, I try to do it here to upward with this here x and this is here the outcome. You can see here that all the letters are converted into say uppercase and now suppose I store this thing in say another variable here z whatever is the outcome, now I have a new variable z in which all the letters are in say capital letters or say uppercase alphabets.

Now, suppose I want to change all of them into lowercase. So, I am saying here Mr. R please use the command to lower and change all the characters which are in the upper case into lowercase and you can see here as soon as I enter this outcome comes over here this even R is changing to small r, capital C is changing to small c and so on. But here you can see that these are the numbers they are not changing and these two function to upper and to lower they are not affecting the non alphabetic characters.

So, let us try to do it over the R console and can we can see what do we obtain over here. So, that is my here x and I copy here to upper x and you can see here that this is converted from a small letters into capital letters.

(Refer Slide Time: 27:11)



Operations with Strings

```
> x <- "R course will start from 24.07.2017"
> x
[1] "R course will start from 24.07.2017"
>
> toupper(x)
[1] "R COURSE WILL START FROM 24.07.2017"
```

```
> z<-"R COURSE WILL START FROM 24.07.2017"
> z
[1] "R COURSE WILL START FROM 24.07.2017"
>
> tolower(z)
[1] "r course will start from 24.07.2017"
```

12

And now I can start to make an experiment as I always say that why do not you play, first I try to write down here instead of upper suppose I type a to lower, let us see what happens you can see here even this R which was it capital letters this is converted into small letters.

So, the model of the story of this example is that in case if you have a statement in which some letters are in upper case and some letters or say characters are in a lowercase. So, when you are trying to say here to upper. So, only those characters are changed into a upper which are occurring in the say lowercase and similarly when you try to do a do to lower; that means, all the characters which are occurring here in the upper phase only they are converted, all other characters all other alphabetics characters they will remain unchanged.

Now, let me try to say store this outcome say take to operas of x into z. So, you can see here now this z is here this thing and now I try to make it here to lower means I want to change the all uppercase letters into lowercase letters. So, I use here the command to lowers and please try to convert all the letters contained in here this z into small case. So, you can see here that is converted into this lowercase.

So, these are very simple operations, but they are very very useful and these are the say screenshots of whatever we have done. So, now after this introduction to some elementary operations we will stop here and once again I would request you that you please try to play with this command, try to practice this command, try to take some example and try to solve yourself. This will give you more confidence and one thing you have to keep in mind as we are going further into this course it is possible that you might be forgetting the earlier commands please do not worry for these things.

We are going step by step at 1 step I am trying to expose you with 1 or 2 commands. Firstly, we try to understand the use and application of each and every command and once we have reached to a reasonable depth then we will try to use these commands together. One good thing in in programming is that in case if you do not use it for a long time you may forget these commands, but the good part is this as so as you start using it very quickly you will recall each and every command. So, that is why after every lecture I am requesting you to this practice.

So, that these commands are settle down in your mind and whenever we are writing a program these commands will come back to your mind and even if you have if you are confused with the syntax whether the whether there should be a comma or there should be a colon or there should be a square bracket or a simple argument do not worry for these things this help menu in R is always there to help you, you can always depend on it. So, you practice you enjoy and we will see you in the next lecture till then good bye.