

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

Strings - Display and Formatting
Lecture - 39
Manipulation with strings and alphabets

Hello, friends. Welcome to the course foundations of R software. And, you can recall that in the last couple of lectures, we are trying to learn different types of manipulations with the string. And, we had learnt earlier that how you can join different strings using the paste function and then how you can split them using the string split function.

And, now in this lecture I will try to continue on the same lines and we will try to learn some other types of operations very small operation which are related to the string manipulations and alphabets. So, I will try to take here couple of examples and through those examples I will try to explain you that how these small functions are working when we are trying to deal with different types of characters and strings. So, let us try to begin the lecture and try to understand that what type of commands are we going to learn in this lecture, ok.

(Refer Slide Time: 01:10)

Operations with Strings: Counting characters

nchar takes a character vector as an argument and returns a vector whose elements contain the sizes of the corresponding elements of x. *number of characters*

nzchar is a fast way to find out if elements of a character vector are non-empty strings. *numeric*

TRUE & FALSE

So, the first function which I want to explain you here this nchar. Actually you see many times what happens that you are writing a statement or a string and you want to count that how many characters are there, right. For example, if you try to recall that when you are trying to work with some word document like as in the MS Word and when you are trying to write something.

Then on the left hand side in the bottom you always get how many characters you have used, how many words you have used etcetera type of information. So, that type of information how you can obtain in the R software for example, if you are given a statement and you want to count that how many characters are there then how are you going to use it, right. For that we use here the option or the function here say nchar which is simply the short form of the say number of characters, right.

So, what it does it takes the character vector inside it is argument and tries to return vector whose elements contain the size of the corresponding elements which are inside the x, right and similarly we have here one more function here nzchar. And, the difference between nchar and nzchar is that nchar this gives us the output which is numeric. And, this command nzchar it is gives us the output in terms of TRUE and FALSE.

So, what it does actually you see whenever you are trying to write something the string can be empty or the string can have some characters. So, this nzchar command helps us in finding out if the elements of a character vector are non empty strings or not, right. If they are not empty string or they are empty string then it will try to indicate it using the logical output TRUE and FALSE.

(Refer Slide Time: 03:05)

Operations with Strings: Counting characters

Usage

```
nchar(x, type = "chars", allowNA = FALSE, keepNA = NA)
```

```
nzchar(x, keepNA = FALSE)
```

Arguments

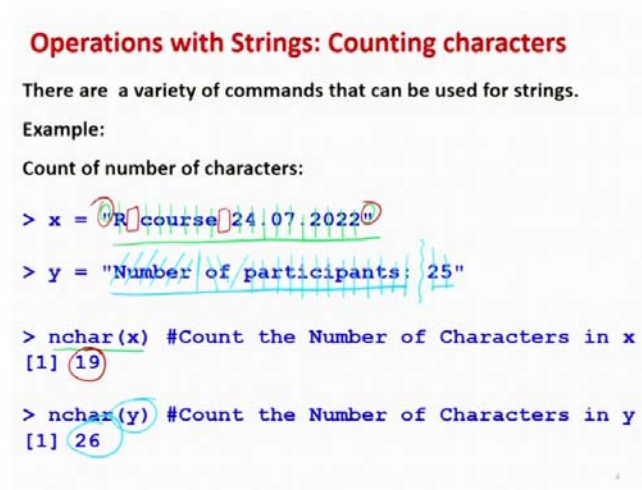
- x** character vector, or a vector to be coerced to a character vector.
- type** character string: partial matching to one of c("bytes", "chars", "width").
- keepNA** logical: should NA be returned when x is NA?

Well, let us try to take some example and then I will try to show you that what is the difference between the use of these two functions. But, before that when you are trying to use this function here nchar then how to use it you write nchar then within the parenthesis you have to give the string in which you want to count the character and then you have couple of options.

For example, it is here c it is a type. So, this type is a character string and it gives you a partial matching to one of the option like is here bytes, characters or say width, right and yeah. So, you have here different types of option well, I am not going to take here all the option because this is a very simple command which you can understand, but I will leave it up to you that you try to use this is a different types of option and try to see what it gives, right, ok.

And, then after this here we have a function here allowNA. So, it is here a double low capital NA. So, this is also a logical variable TRUE and FALSE and as the name suggests that whether you want to allow the NA values or not in the output that you can control from here. Similarly, you have here one more option here keepNA, k e e p is in lowercase alphabet and NA is in the upper case alphabet. So, it is also a logical variable and it tries to answer the question should NA be returned when x is NA, right?

(Refer Slide Time: 04:21)



Operations with Strings: Counting characters

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

Example:

Count of number of characters:

```
> x = "R course 24.07.2022"
> 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, I will try to show you these options and other type of outcome through the example. So, let us try to consider here a string say R course 24 dot 0 7 dot 2022 – this is my string you can see here, these are my double quotes and I want to count here that how many characters are used in this string x.

So, I try to write down here nchar and inside the parenthesis I simply write down here x and you can see here how it is trying to give us the value. So, please try to count how many characters are there. See here 1, now suppose I am now confused whether I should include here the blank space or not. So, at this moment I say ok blank space should not be counted. So, I am skipping blank.

Later on we will see whether the R is counting the blank space or not because as I always say towards the end that you have to understand how the R is working. So, this is my here first value R, then c is second third fourth fifth 6 7 8 9 10 11 12 13 14 15 16 17. And, so, now, you can see here there are here two blank spaces which are also to be counted. So, this will give you the number 19 here.

So, the rule here is this command nchar this is also counting the blank space n. Obviously, they are inside the double quotes. So, this is a whole string. So, black space is automatically going to be counted as a one of the character. Similarly, I try to take here one more example and I try to write down here number of participant colon and then 25.

So, let us try to first count that how many numbers are there and now, you know how to count it. So, this is here 1 2 3 4 5 6 and then blank space is 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 and then blank space 24 25 26. And, you can see here this is here the outcome of nchar and inside the parenthesis I gave here y, right. So, you can see here it is simply trying to count the number of characters in the string, right.

(Refer Slide Time: 06:27)

Operations with Strings: Counting characters

Example:

The `nzchar(x)` function takes the character vector `x` as a parameter. Its output is either `TRUE` or `FALSE`.

```
> x = "R course 24.07.2022"  
> y = "Number of participants: 25"  
> nzchar(x)  
[1] TRUE  
> nzchar(y)  
[1] TRUE
```

The vectors are non empty

And, similarly if you try to use the command here `nzchar` on the same variables then let us try to see what it does, right. So, its output is going to be either logical `TRUE` or logical `FALSE`, right. So, if you try to see here in this string is there anything which is empty? No, you already have counted that it has 19 characters and the same here is in the `y` also there are 26 characters.

So, how it can be non empty? So, when you try to use the same statement `x` and `y` here as you use earlier the outcome of this `nzchar` and `nzchar y` that is going to be `TRUE` this means the vectors are non empty yeah after that I will try to take an example where I try to take an empty vector and then I will try to show you what it gives, right.

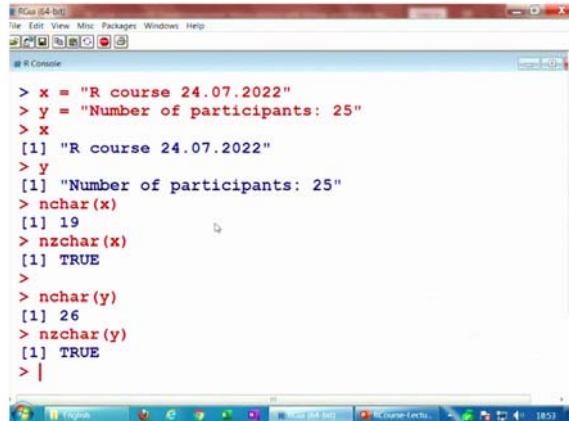
(Refer Slide Time: 07:15)

Operations with Strings: Counting characters

```
> x = "R course 24.07.2022"  
> x  
[1] "R course 24.07.2022"  
> y = "Number of participants: 25"  
> y  
[1] "Number of participants: 25"  
> nchar(x)  
[1] 19  
> nchar(y)  
[1] 26  
>  
> nzchar(x)  
[1] TRUE  
> nzchar(y)  
[1] TRUE  
> |
```

So, this is here the screenshot of the same outcome. So, firstly, let us try to see how the outcome will look like in the R console.

(Refer Slide Time: 07:25)



```
> x = "R course 24.07.2022"
> y = "Number of participants: 25"
> x
[1] "R course 24.07.2022"
> y
[1] "Number of participants: 25"
> nchar(x)
[1] 19
> nzchar(x)
[1] TRUE
> nchar(y)
[1] 26
> nzchar(y)
[1] TRUE
> |
```

So, let me try to take it here x here is like this and let me try to take here y here as like this. So, you can see here x is this and y is this and then I try to use here nchar x. This is here 19 and if I try to use here nzchar, this is TRUE and similarly, if you try to take here the nchar in the y that mean the number of characters in the y, this is giving you here 26 and if you try to use here nzchar it is giving you here TRUE, right.

So, you can see that it is not very difficult to use these commands and yeah, many times in the data manipulations and when you are trying to do different types of operation, they actually help us.

(Refer Slide Time: 07:59)


Operations with Strings: Counting characters

Example: `nzchar(x)`

```
> x = c("Apple", "Banana", "Cake")
> x
[1] "Apple" "Banana" "Cake"
> nzchar(x)
[1] TRUE TRUE TRUE
> y = c("Apple", "", "Cake")
> y
[1] "Apple" "" "Cake"
> nzchar(y)
[1] TRUE FALSE TRUE
```

Handwritten annotations:

- Red arrows point from the numbers 5, 6, and 4 above "Apple", "Banana", and "Cake" in the first output to the corresponding elements in the `nzchar(x)` output.
- Red text: "Each of few string is non-empty" with arrows pointing to the TRUE values in `nzchar(x)`.
- Blue text: "no blank space" with an arrow pointing to the empty string "" in the second output.
- Blue text: "0" with an arrow pointing to the FALSE value in `nzchar(y)`.



So, now, I try to show you here that how this nzchar command will work if there is an empty argument, right. So, let me try to take here first an example where I am trying to take here three strings apple, banana, cake – our popular example. So, these are 3 characters which are stored in the data vector here x and this is here x and if you try to see here if you try to use here nzchar, x it is giving you here TRUE, TRUE, TRUE.

This is obviously, apple has how many characters? 1 2 3 4 5. Banana has how many characters? 1 2 3 4 5 6 and cake has 1 2 3 4. So, definitely they are non empty. So, it is trying to give here this TRUE and TRUE for apple banana and cake respectively, right; that means, each of the string is non empty.

Now, you see how do I modify the same example. I try to take here the apple I try to take here the cake, but for the banana I drop it, but I simply try to write down here only double quotes and there is no blank space also that is what you have to keep in mind if you try to give the blank space, then it will give you that the string is non empty.

Now, if you try to see here this is your here outcome first is here apple, then it is here blank; blank means you know nothing and then it is here cake. But, since you are you have given here double quotes, so that means, it is going to be a string means if you do not give the double quotes then it is say taken only here as a blank space. So, that is the reason that now you have created a string which is blank and then if you try to use here nzchar it will give you here TRUE FALSE and TRUE.

So, this TRUE is correspond to here TRUE to here apple which has 5 characters, then FALSE here is like this because there are no characters. So, there are 0 characters and then you have here four characters in the cake. So, it is corresponding to TRUE. So, this is a utility of this nzchar, right. So, let me try to show you first this operation on the R console so that you can understand and you are confident, ok.

(Refer Slide Time: 10:02)

```
RStudio [64-bit]
File Edit View Misc Packages Windows Help
R Console
> x = c("Apple", "Banana", "Cake")
> nzchar(x)
[1] TRUE TRUE TRUE
> nchar(x)
[1] 5 6 4
>
> y = c("Apple", "", "Cake")
> nzchar(y)
[1] TRUE FALSE TRUE
> nchar(y)
[1] 5 0 4
> |
```

So, if you try to see here like this. So, if you try to see here nzchar this is here x, this is here like this TRUE and if you try to simply use here only nchar, see what it gives? It will give you that there are 5 6 4 values, right. So, similarly now I try to modify my here this x and you try to see what I am trying to do here I am simply to delete all the values and I have only here just these two double quotes, right. Now, this is and I suppose I try to write down here as say y.

So, now, if you try to find out here what are the nzchar inside this here y you can see here this is giving you here TRUE, then FALSE and then here TRUE. And, if you try to use here the command here nchar y this will give you here 5 0 4. So, you can see here. Now, this has 0 character so, it is trying to give you a reply in terms of the logical FALSE, but the more important part is that you have to understand how to interpret the result, right.

(Refer Slide Time: 11:00)

Operations with Strings: Counting characters
Example:

```
> x = c("Apple", "Banana", "Cake")
> nchar(x)
[1] 5 6 4

> y = c(2, 4, 6)
> nchar(y)
[1] 1 1 1

> z = c(11, 222, 3333)
> nchar(z)
[1] 2 3 4

> z1 = c(11, 2, 22, 3, 333)
> nchar(z1)
[1] 3 4 5
```

```
> x = c("Apple", "Banana", "Cake")
> nchar(x)
[1] 5 6 4
> y = c(2, 4, 6)
> nchar(y)
[1] 1 1 1
> z = c(11, 222, 3333)
> nchar(z)
[1] 2 3 4
> z1 = c(11, 2, 22, 3, 333)
> nchar(z1)
[1] 3 4 5
```


So, if you try to see here now I try to give you here some more example that how you can count the numbers and the or the characters. So, I have shown you here that if I try to take the string here let us say apple, banana, cake then nchar is going to give you the outcome 5 6 4. So, 1 2 3 4 5, 5 it is alphabets or 5 characters in apple; banana 1 2 3 4 5 6 characters in banana; c a k e cake which has 4 characters. So, this answer is coming here as say 5 6 4.

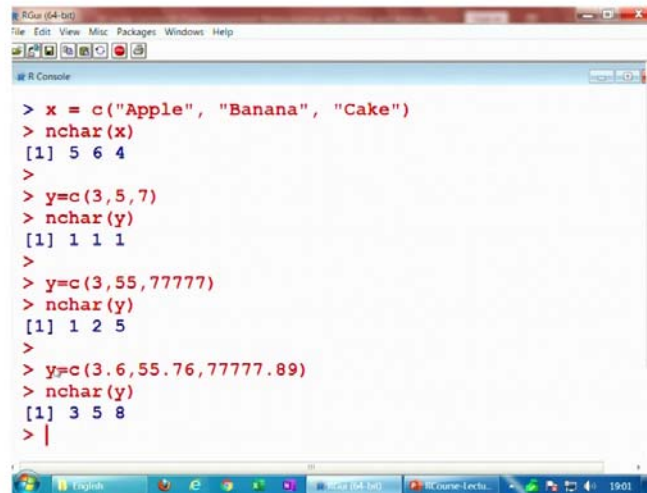
Now, similarly if you try to take here three numbers 2, 4 and 6, what do you expect? What the command nchar over this y will do? It will give you the answer here 1 1 1; that means, it is here 1 character, 1 character and 1 character if you try to see it is not trying to read the number, but it is trying to read the number of characters.

Similarly, if you try to modify this example and try to take care these 3 number such that they are 11, 222 and 3333 such that the first step this 11 has 2 characters; the second number 222 it has 3 characters and 3333 it has 4 characters. So, if you try to operate your command here nchar over here z it will give you here 2 3 4, right; so, 2 in 11, 3 in 222 and 4 in 3333.

Now, in case if you try to take here any other number with decimal point then what happens. So, you have to remember that the decimal points are also taken as a character. So, if you try to take here this data vector where I have taken here these 3 values. So, the first value is here 1.1, this is a 1 2 and 3. So, the number of characters in 1.1 is 3, then my next value is 2.22. So, in this case 1 2 3 4. So, I have 4 characters in 2.22 and similarly here in 3.333 it is 1 2 3 4 5, there are 5 characters in 3.333.

And, if you try to see here the outcome of this nchar over z1 which is the this vector here it has the outcome 3 4 5 and this is here the same outcome which I will try to show you on the R software, right. Now, so, before I try to move forward let me try to show you these things over here. So, let me try to yeah.

(Refer Slide Time: 13:10)



```
> x = c("Apple", "Banana", "Cake")
> nchar(x)
[1] 5 6 4
>
> y=c(3,5,7)
> nchar(y)
[1] 1 1 1
>
> y=c(3,55,77777)
> nchar(y)
[1] 1 2 5
>
> y=c(3.6,55.76,77777.89)
> nchar(y)
[1] 3 5 8
> |
```

So, although I have shown you, but still I will try to show you here nchar over x. So, this is going to be here 5 6 4. Now, if you try to take here is a data vector of sub number 3, 5, and 7 it will be your here if you try to see the number of characters in y will be here 1 1 1 and if you try to modify that first value is 3, second value becomes here 55 two digit and third value become here suppose five digit 77777.

Now, in case if you try to see your here nchar y this will be here 1 2 and 5. So, there is one character in 3, two characters in 55 and five characters in 77777, right. So, this is the outcome here now in case if you now try to convert these numbers into some digit with decimal point like a 3.6, 55.76 and 77777. say here 89.

And, then if you try to say here nchar y. So, this will be your here say 3 5 and 8. So, there are 1 2 3 and then 1 2 3 4 5 and then 1 2 3 4 5 6 7 8. So, this is here the outcome 3 5 8, right. So, these very simple operation by which you can count the number of characters in any string or any number.

(Refer Slide Time: 14:21)

Operations with Strings: Lower and upper cases

`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.

`aA -..`

The slide contains handwritten annotations: a green arrow points from the text 'lower case' to the `tolower(x)` function, and a red box highlights the example string `aA -..`.

Now, after this I try to give you here one more operation which is related to the alphabets and that is a very simple operation that suppose you have some characters or some strings which are based on some alphabet. And, you want to change the lower alphabets into upper case alphabet or say upper case alphabets into the lower case alphabet.

And, if you try to see if you have used the software like MS Word, then if you try to see on the left hand side on the top there is one option here where there is an option like something like this say small a capital A etcetera and it gives you here different option change to lower case change to upper case change to contents case etcetera.

So, similar type of operation many times are needed when we are trying to deal with different types of a strings and particularly, when you are trying to prepare a format for generating a report these types of operations are sometimes needed. For example, if I take a very simple example suppose 5 people are trying to prepare a report and you want the heading should be in the capital letters.

Now, suppose different people have used different formats. Somebody has used the sentence case like as the first alphabet as in the capital letter and all other remaining in the lower case alphabet or somebody has used only the lower case; somebody has used the upper case etcetera. So, if you want to generate the report which is uniform for all then you can simply use here this type of commands.

So, whatever characters are there they will be converted into either lower case or to the upper cases. So, how to get it done? This is I would like to show you with the command `tolower(x)` and `toupper(x)` and to upper `t o u` double `p e r`. So, all in lower case alphabets they are written. So, as the name suggest to lower; that means, whatever is there try to convert it into the lower case alphabets, right.

So, this to lower what it will do? It will try to convert all the lowercase alphabet means if you try to give any string and if it has any characters in the upper case all the characters are going to be converted into the lower case. And similarly, this command here to upper if you try to give any string here in the `x` which has suppose some lowercase alphabets. So, they are going to be converted into all uppercase alphabet, right.

And, yeah so, the first question also comes in the mind that if there are some numbers like as non alphabetic characters like a symbols or numbers, then they will be remain intact they are not going to be changed.

(Refer Slide Time: 16:45)

```
Operations with Strings: Lower and upper cases
tolower(x) and toupper(x) Functions:

Examples:
> x = "R course will start from 24.07.2022"
> toupper(x)
[1] "R COURSE WILL START FROM 24.07.2022"
> z = "INDIAN INSTITUTE OF TECHNOLOGY"
> tolower(z)
[1] "indian institute of technology"
```

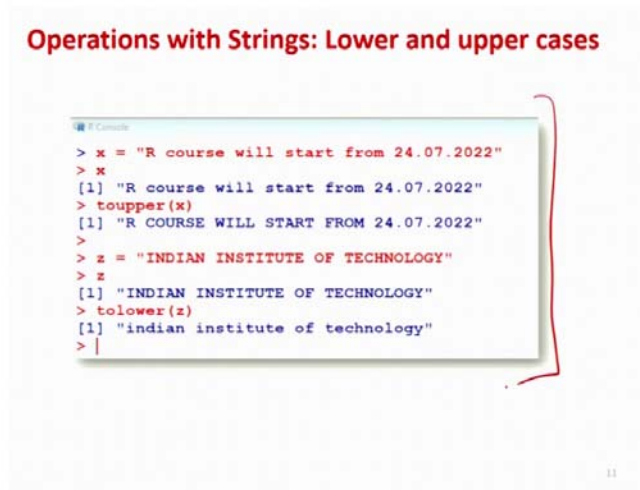
Handwritten annotations:
- Green circles around "R" and "24.07.2022" in the first example.
- Green arrow pointing from "Lowercase" to "R".
- Green arrow pointing from "uppercase" to "R COURSE WILL START FROM".
- Red arrow pointing from "uppercase" to "INDIAN INSTITUTE OF TECHNOLOGY".
- Red arrow pointing from "lowercase" to "indian institute of technology".

So, let me try to take here some example and try to show you the application of these two commands. So, let me try to take here this first example where I try to take here as string R course will start from 24 dot 07 dot 2022. So, you can see here this is here in capital letters whereas, everything here is in lower case alphabet and after this I have here numbers or some special characters.

Now, when I try to use here this command here to upper all in lower case alphabets and I write here x inside the parenthesis, then you can see here what will happen. R is already in the capital letter. So, it will remain as upper case and, but all other letters they are going to be converted into the upper case you can see here and these numbers they will remain intact as such.

Now, similarly I try to take here one more example and I try to take here a string z in which I am trying to write INDIAN INSTITUTE OF TECHNOLOGY and all are written in the upper case alphabets. Now, I try to use here the command here tolower t o l o w e r and inside the parenthesis I write the string here z we can see here now everything is converted into the lower case alphabets, right you can see here. So, this is how this actually works, right.

(Refer Slide Time: 17:59)



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

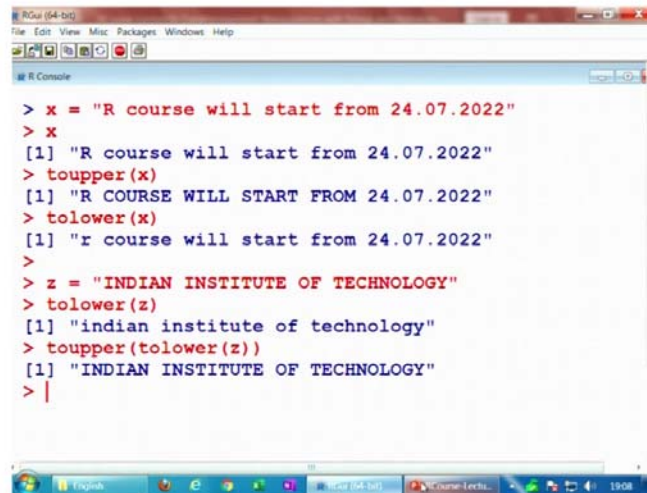
```
Operations with Strings: Lower and upper cases

> x = "R course will start from 24.07.2022"
> x
[1] "R course will start from 24.07.2022"
> toupper(x)
[1] "R COURSE WILL START FROM 24.07.2022"
>
> z = "INDIAN INSTITUTE OF TECHNOLOGY"
> z
[1] "INDIAN INSTITUTE OF TECHNOLOGY"
> tolower(z)
[1] "indian institute of technology"
> |
```

The console output demonstrates the effect of the `toupper` and `tolower` functions on strings. The `toupper` function converts all lowercase letters in the string to uppercase, while the `tolower` function converts all uppercase letters to lowercase. The numbers and spaces in the strings remain unchanged.

So, if you try to see this is here the screenshot of the same outcome and let me try to first show you these operations on the R console so that you can be here more confident.

(Refer Slide Time: 18:12)



```
> x = "R course will start from 24.07.2022"
> x
[1] "R course will start from 24.07.2022"
> toupper(x)
[1] "R COURSE WILL START FROM 24.07.2022"
> tolower(x)
[1] "r course will start from 24.07.2022"
>
> z = "INDIAN INSTITUTE OF TECHNOLOGY"
> tolower(z)
[1] "indian institute of technology"
> toupper(toupper(z))
[1] "INDIAN INSTITUTE OF TECHNOLOGY"
> |
```

So, I try to take here say here x here x is like this and if I try to see here say toupper and here x you can see here everything is converted into means upper. And in the same if I try to take it here tolower, then what will happen? You can see here that only R is in the upper case alphabet. So, this will also be converted into a lower and rest everything was already in the lower case. So, it will remain as such.

And, similarly if I try to take care this next example where I am trying to take care z which is in the upper case alphabets and if I try to say here tolower. So, here z and now you can see here everything is converted here like this, right and in case if you try to use here say here toupper and then I you try to use here the same command which you have obtained say tolower and here z you can see what happens this is converted once again into this thing.

Yeah, first you are trying to lower case and then you are trying to do the upper case. So, this is quite obvious, right so, ok. So, now we come to an end to this lecture and you can see that in this lecture we have considered a very small operations which are very useful also when you are trying to generate different types of reports and you want to make such a small operations. So, they can be done automatically using this simple functions.

So, now once again, you try to see that what are the areas in which you can use such commands. I am sure that you are working in somewhere either as a student or as a

professional and you might be needing these types of operations in your work. So, try to see and then try to practice it that what really happens when you are trying to experiment with these operation. So, you try to practice it.

And, I will see you in the next lecture. Till then, goodbye.