

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

Lecture – 29
Replacement and Evaluation of Strings

Welcome to the lecture on the course introduction to R software, you may recall that in the earlier lecture we started a discussion on the topic that how to find and search a particular expression or a character inside a string and for that we have discuss the 2 function sub and g sub. The objective of these 2 function was to find something form an expression and them to replace it by something what we want, now there is another aspect that suppose we only want to search for an expression that I want that where this type of expression is occurring in the strin g; that means, in simple words we are only interested in matching, matching of an expression in the string. So, how to do it? That is the first objective of this lecture and after that we will discuss 2 more simple function.

So, now we start this lecture. So, we have seen that R has got several functions which can match and replace a particular expression inside any string and for that we have use the 2 functions here sub and g sub. So, sub and g sub have 2 steps to work first step match.

(Refer Slide Time: 01:42)

Operations with Strings

R has various functions for regular expression based match and replaces.

Some functions (e.g., `grep`, `grepl`, etc.) are used for searching for matches and functions whereas `sub` and `gsub` are used for performing replacement.

2

Second step replace, where as there are some function which do only the matching and these type of situations are pretty common whenever you are trying to deal with any database. Suppose I want to know whether this expression is occurring where inside the entire document or say entire file also entire data set, usually we have done such such as a by using the command control fine or say control f.

So, now, we are going to discuss a command that is used only for search, search means please try to find the match of the letter or character or an expression in the given expression in which I want. So, for that we have actually several commands, one is here grep and another is here grepl and so on, but just to give you a flavor of such things we are discussing here only one function that is grep and I understand that if you need more you can just read yourself whenever you will need it in the future.

(Refer Slide Time: 03:02)

Operations with Strings

grep function:

The **grep** function is used for searching the matches.
(**sub** and **gsub** are used for performing replacement.)

grep : Globally search regular expression and print it

grep (**pattern**, **x**) search for matches to argument **pattern** within each element of a character vector **x**.

It returns an integer vector of the indices of the elements of **x** that yielded a match

Handwritten notes:
- "that is to be searched" points to **pattern**
- "the search needs to be done" points to **x**

3

So, the objective of grep function is very simple, it is used for searching them matches, searching the matches means what? I have got an expression and I want to see where it is occurring in the file or in the given data set. So, this will only find and it will say these are the places where this match is occurring right.

So, the meaning of g r e p, grep is that globally search regular expression and print it. So, it will only search what only search, but only search is not sufficient for us we also need to be informed. So, for that it will print the outcome that is the simple use of this grep function. Thus syntax of for grep function is g R e p and inside the arguments you have

to write down the pattern, which pattern? The pattern which is to be search or that is to be searched. The next question is the I would like to know; yes I will search for you, but from where. So, for that we will see this is here the say statement or x vector where the search needs to be done. So, the idea is very simple this function grep will search for the matches which are given inside this pattern and this will try to search in the each element of the character vector x.

Now, we have 2 options, when we it is trying to search and match then it needs to inform us that where the matches occurring, we have 2 options either this can tell us the expressions or the strings where this matches occurring and inform us the entire word, where the matches occurring. Or second option is that it can only tells the position of the word where the matches occurring, now with this grep function we have 2 options we want to search the required pattern and we want the output in terms of number or in terms of expression. Based on that we have an option to use the option of say using true and false and based on that I can get a required outcome so why not to take some examples and try to understand what I am trying to say.

So, now first I am going to take an example where I will be using the true option.

(Refer Slide Time: 06:29)

Operations with Strings TRUE FALSE | Logical

to be searched
where to be searched
scanned

`grep(pattern, x, value = TRUE)` returns a character vector containing the selected elements of x.

```
> str <- c("R Course", "exercises", "include  
examples of R language")
```

1 *2* *3*

```
> grep("ex", str, value=T)
```

```
[1] "exercises" "include examples of R language"
```

Report the strings where match occurs

So, I write my syntax like g r e p and inside the arguments I would write here the pattern which is to be searched and then where to be searched, this is here in x, where to be searched it is here and suppose I want my outcome in terms of expression that it should

not give me the number, the address in terms of location in terms of number, but it should give me in terms of expression. So, for that I will use here another option which is value is equal to true, this value can have 2 option this can be here true or this can be here false.

So, I will try to show you the example of both true and false and you may also recall there is true and false they are the logical values and they are the reserved words. So, in case if you use here value equal to true then this is going to return a character vector that is containing the selected element of x. What does this mean now take an example and try to understand, suppose I try to create a vector here by using the c command, combined command and I have here 3 strings first string is here. So, number 1, second string is here exercises and third string is here number 3. So, first string is saying R course second string is saying exercises and third string is saying include examples of R language. So, essentially I had a sentence R course exercises includes example of R language.

So, I have broken this sentence into 3 strings and which I have numbered here I said 1, 2 and 3 and suppose I want to know that where the expression ex is occurring, in this say this vector which I have denoted by the name str that is the sort of a string I will say . So, now, if you try to look over here this pattern, which is given inside the double quotes where this is occurring, is it occurring here, no. Is it occurring here, yes this is occurring here after this you can see here no is not occurring here, it is not occurring here nowhere ex no it is occurring here say ex in the examples and after that it is not occurring here nowhere.

Now, I am using here the pattern ex and I am requesting R to please search this pattern ex in the vector str which is given here and my value is equal to here true, means I can use capital T or I can use capital TRUE both, both are acceptable and the answer comes out to be like this you can see here. So, that is trying to say that ex is appearing in the word exercises or the string exercises and ex is also appearing in say another string which include examples of R language. So, as you can see here this is happening here and here. So, ex is being matched and then printed, but you can see here that this is giving us the outcome in which it is trying to report the strings report the strings where match occurs.

Now, suppose I want to have the output in terms of the numbers only, what do you mean by number?

(Refer Slide Time: 10:46)

Operations with Strings

`grep(pattern, x, value = FALSE)` returns an integer vector of the indices of the elements of `x` that yielded a match.

`value = FALSE` is default.

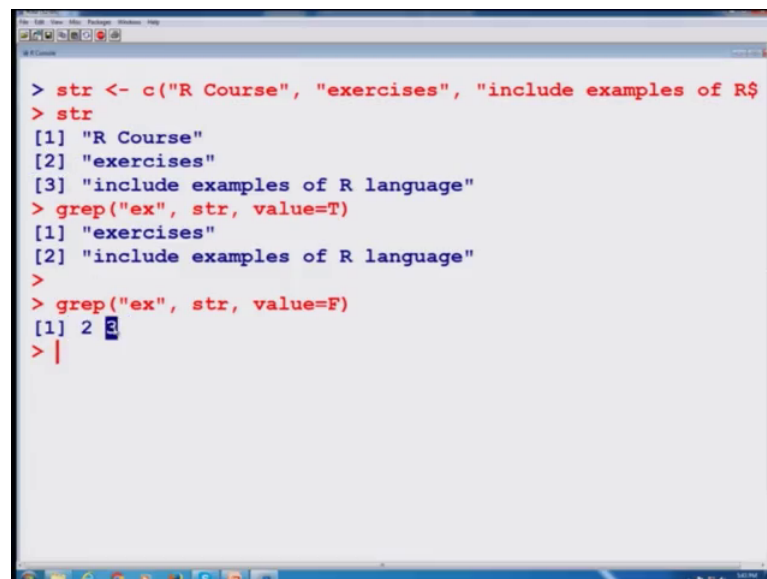
```
> str <- c("R Course", "exercises", "include  
examples of R language")  
> grep("ex", str, value=F)  
[1] 2 3
```

So, I will let us try to take this statement by using here false and try to understand what I am trying to say. So, now, I try to use the same example, but the syntax is now change a little bit here, in this grep pattern x there is no change, but I try to change only here this place that instead of true I have here false. So, the rule is this when I am using here the option value equal to false, then it returns an integer vector of the indices of the elements of x that yielded a batch, what you really understand by this thing? So, try to take that same example and try to use this command and see what happens.

So, here I am using the same vector say str in which my sentence R course exercise this included examples of R language is broken into 3 strings, number 1, number 2 and here this is here number 3, entire is number 3. So, now, when I use here the same command this is a grep ex inside the double quotes; that means, please find the pattern ex in just vector str which is given over here and my value here is false. So, I try to give here in capital f or say ca say capital f a l s e both are acceptable and you get here the outcome here is a 2 and 3 what does this mean. 2 means that ex is appearing in the string number here 2, this 2 and this here the 3 3 means here that this is occurring here in this address here 3, you can see here the ex is appearing in number 2 string number 2 here and string number 3 here.

So, now, you can see the difference between the two that one is giving the outcome in terms of entire say this word or string and another is giving the outcome in terms of the address in the form of an integer, right. What important thing the default value option here is false that if you do not give here anything then the grep function will choose value is equal to false as default. So, that is your choice what you want to get as an outcome. So, now, let us try to see some more example, but before that let us try to execute this thing and try to see do we get this thing or not.

(Refer Slide Time: 13:41)



```
> str <- c("R Course", "exercises", "include examples of R language")
> str
[1] "R Course"
[2] "exercises"
[3] "include examples of R language"
> grep("ex", str, value=T)
[1] "exercises"
[2] "include examples of R language"
>
> grep("ex", str, value=F)
[1] 2 3
> |
```

So, first let me try to take this create this string. So, I try to create this string over here. So, you can see here this is my here string and it is giving me here 3 numbers, but anyway I want to make here search, where I want to search that where this grep is occurring. So, you can see here that this is trying to say this is here ex this is occurring here in ex and here ex.

So, you can see here this outcome is coming here as say in the form of 2 strings and similarly in case if you try to give here this option here so false then the same command comes over here say 2 and 3. This 2 and 3 for example, this 2 this corresponding 2 this 2 and this 3 is corresponding to this 3 that you have to keep in mind and the same outcome I have given here that you can see. So, now, the same outcome I have given in this slide and we come back toward this slide.

(Refer Slide Time: 14:49)

```
Operations with Strings  
Example:  
> x <- "R course 24.07.2017"  
> y <- "Number of participants: 25"  
  
> c(x,y) # Combine the two strings  
[1] "R course 24.07.2017" "Number of  
participants: 25"  
  
> grep("our", c(x,y) )  
[1] 1  
  
"our" is in the 1st element (in the word "course"), therefore in x.  
There is no "our" in y.
```

So, now let us try to take some more example to understand the use of grep function. So, let me try to take here 2 separate vectors say x I m trying to say here R course 24 07 2017 and I try to take care another vector here y and I try to create a string number of participants colon 25. Now suppose I try to combine these 2 vectors together by the command c. So, you can see here as soon as I combined them this becomes here like this 1 and here this is here the second string and now I am saying here grep; that means, mister R please try to search for the pattern, who you are say our, where in cxy where cxy has been obtained here earlier as soon as you do here enter this will give here the number 1.

So, you can see here there are here 2 addresses 1 number and 2 number and that will also be clear to you when I try to show it in the R console, but before that you have to understand what is really happening you can see here this our, this is matching where here. So, in the word course there are 3 letter words o u r and then it is occurring nowhere else you can see it yourself. So, that is why this is saying that this is occurring at say address number 1.

(Refer Slide Time: 16:48)

Operations with Strings

Example:

```
x <- "R course 24.07.2017"
y <- "Number of participants: 25"
```

> c(x,y) # Combine the two strings
[1] "R course 24.07.2017" "Number of participants: 25"

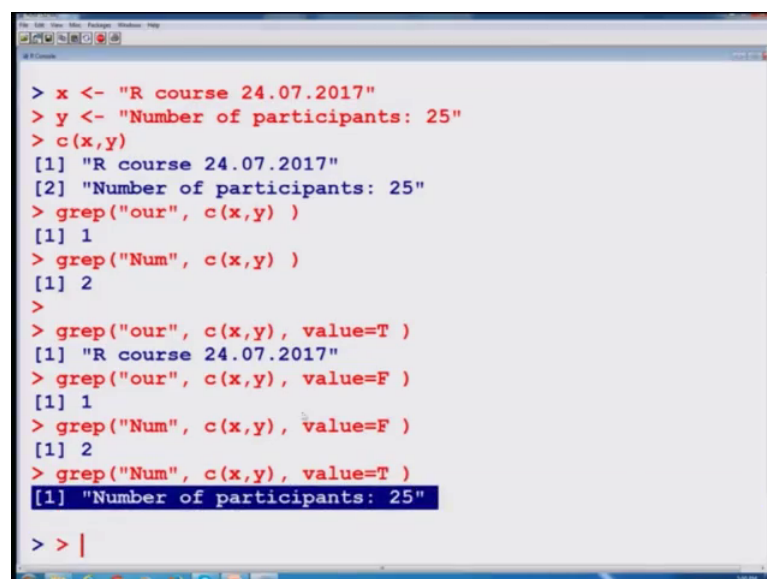
> grep("Num", c(x,y))
[1] 2

"Num" is in the 2nd element (in the word **"Number"**), therefore in y.

There is no **"Num"** in x.

So, now let us try to see what happens if I try to take another choice of letters. So, I try to take the same example, but now I am looking for say say here n u m, num where in this vector c x y and the answer comes out to be here 2. What does this mean you can see here this is my number 1 and this is my here by 2. So, you can see here num is occurring here in the word number and it is giving me here 2 and here is the say screenshot of this thing, but let us try to do it ourselves and try to see what do we obtain here.

(Refer Slide Time: 17:50)



```
> x <- "R course 24.07.2017"
> y <- "Number of participants: 25"
> c(x,y)
[1] "R course 24.07.2017"
[2] "Number of participants: 25"
> grep("our", c(x,y) )
[1] 1
> grep("Num", c(x,y) )
[1] 2
>
> grep("our", c(x,y), value=T )
[1] "R course 24.07.2017"
> grep("our", c(x,y), value=F )
[1] 1
> grep("Num", c(x,y), value=F )
[1] 2
> grep("Num", c(x,y), value=T )
[1] "Number of participants: 25"
```


So, first let me try to create here these 2 vectors say x and then here y and now I will try to combine them with cxy I can see here that there are here 2 things joined together and that is why there are 2 addresses here 1 and here 2 and now when I am trying to go for this here search and match for the value o u R string, you can see here this is coming out to be here. Say here one and similarly when I try to do it here say here num you can see here that is occurring here at 2 num is occurring in number 2 and our is occurring here in number 1.

And similarly now if I try to use here another function just for the sake of illustration if I try to see here value is equal to here true, again what we get here this thing and in case if I you try to use your value is equal to here false you get here the same thing. So, you can see here that this outcome is based on the use or the value option right and similarly if you want to go for here num you can see here value is equal to here false will give you the same outcome that you obtained earlier here, but in case if you try to give here value is equal to here true, you get the first sentence the entire string is here. So, that is the use of this grep command. So, let us now come back to our slides and we move see here further. So, that is the outcome.

(Refer Slide Time: 19:44)

Operations with Strings

eval function: *evaluate*

eval function evaluates an (Unevaluated) R expression in a specified environment.

Example:

```
> eval(2 ^ 2 ^ 3)
[1] 256
```

Calculator

$$2^{(2^3)} = 2^8 = 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 256$$

R Console

```
> eval(2 ^ 2 ^ 3)
[1] 256
```

11

So, now I will stop with this grep function and would like to take up another function which has got some utility when you are trying to do some data searching or data analysis or something like this. There is one function which is called as eval, eval is the

short form of evaluate this means what this eval function evaluates an R expression in a specified environment, what does this mean? So, to understand it let me take here some examples and we try to see what it is really doing that will give you a better idea then I explain you the theory part.

For example I write here eval(2^2^3), this means what this is here is power of here and its power of a 3. So, what it is trying to do here this will be solved in the first step here I say for this part and this will come out to be 8, 2 cube is 8 and then once I try to multiply it by a 2 into 2 into 2, 8 times this will come out to the a 256, 2 5 6 that is 256 what shall I doing, dont you think that this function is working just like a calculator. So, when I am writing the syntax like eval and inside the argument if I am writing any mathematical expression then it is evaluating the expression, but then the next question come why should I use is what is the difference between then the calculator and this eval function.

So, let me take it another, suppose if I write here eval(6+8) what do we expect the answer will come out to be 14 the addition of 6 plus 8 is 14.

(Refer Slide Time: 22:03)

Operations with Strings

eval function:

Example:

```
> eval("6+8")  
[1] "6+8"
```

```
> eval(6+8)  
[1] 14
```

The `eval()` function evaluates an expression, but `"6+8"` is a string, not an expression whereas `6+8` is not an expression.

Handwritten notes: `eval(6+8)` ✓
14

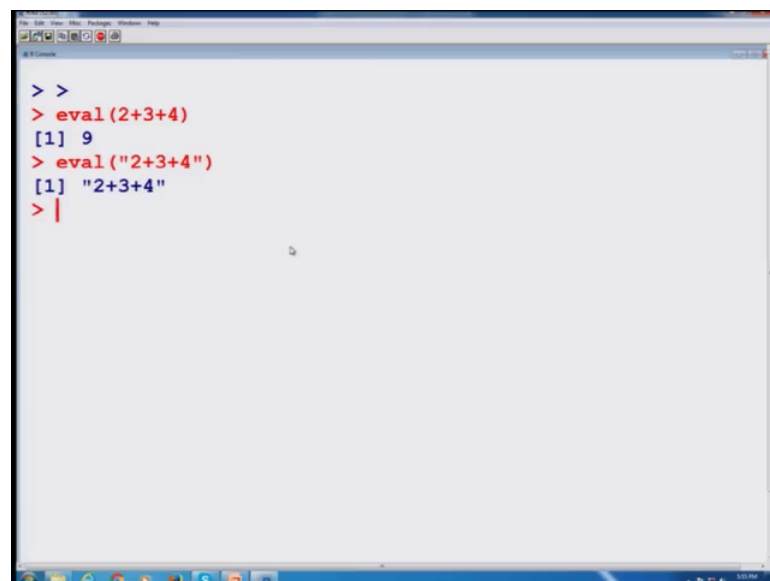
R Console screenshot:
> eval("6+8")
[1] "6+8"
>
> eval(6+8)
[1] 14 ✓

But now suppose in some programming at some places I want this 6 plus 8 to be printed just like 6 plus 8; that means, I am not requesting are to do any mathematical calculations on the expression 6 plus 8. So, if I want to print a mathematical expressions just as such then I can write down here see eval(6+8), but within the double quotes and then I

will get an outcome here just like 6 plus 8 what are the places where this can be used. Have you ever seen a mathematics book, where they try to type say in the first line they will write 6 plus 8 and in the next line they will write 14 so; that means, in the first line they just want to print 6 plus 8 as 6 plus 8 and in the second line they want the result of 6 plus 8.

So, similarly you could encounter several places when you are trying to analyze the data where you would like to do so in those situation this function is going to be useful and here you can see here the outcome. I will try to show you on the R console also, but the this the screenshot when you are writing here e v a l 6 plus 8 you are getting an answer here 14, but when you are writing 6 plus 8 inside the inverted commas you are getting it like this. So, let us try to do it on the our console and try to see it here.

(Refer Slide Time: 23:51)



```
> >
> eval(2+3+4)
[1] 9
> eval("2+3+4")
[1] "2+3+4"
> |
```

Suppose I will write here e v a l, suppose I want to write here 2 plus 3 plus here 4 n you can see here the answer will come out to be 9, but in case if I try to enclose it inside that double quotes the answer will come out to be like this.

So, that is the advantage of here e v a l function. So, whenever you want to print the mathematical expressions as such then use the mathematical expression inside the double quotes. So, they are going to be treat as just like a string they will not be treated like a mathematical function, let we take here another example.

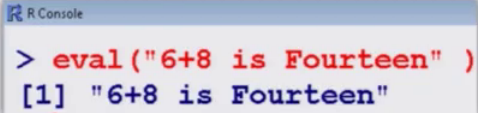
(Refer Slide Time: 24:33)

Operations with Strings

`eval` function:

Example:

```
> eval("6+8 is Fourteen" )  
[1] "6+8 is Fourteen"
```



```
> eval("6+8 is Fourteen" )  
[1] "6+8 is Fourteen"
```

13

Suppose I want to print something like this 6 plus 8 is 14. So, here we have some numbers and we have some here characters and, but I need the entire expression to be printed as such. So, I will try to give it double quotes and then I will try to use here this function evaluate and it will print us like this and you can see here this is the screenshot. So, you can try it yourself.

(Refer Slide Time: 24:58)

Operations with Strings

`parse` function:

`parse()` with `text=string` is used to change the string into an expression.

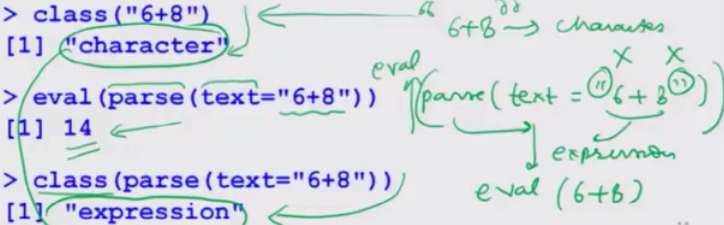
Example:

```
> eval("6+8")  
[1] "6+8"
```

```
> class("6+8")  
[1] "character"
```

```
> eval(parse(text="6+8"))  
[1] 14
```

```
> class(parse(text="6+8"))  
[1] "expression"
```



14

Now after this is small and simple use of this eval function, now let me try to introduce one more function. This function is called as here p a r s e, parse the advantage and

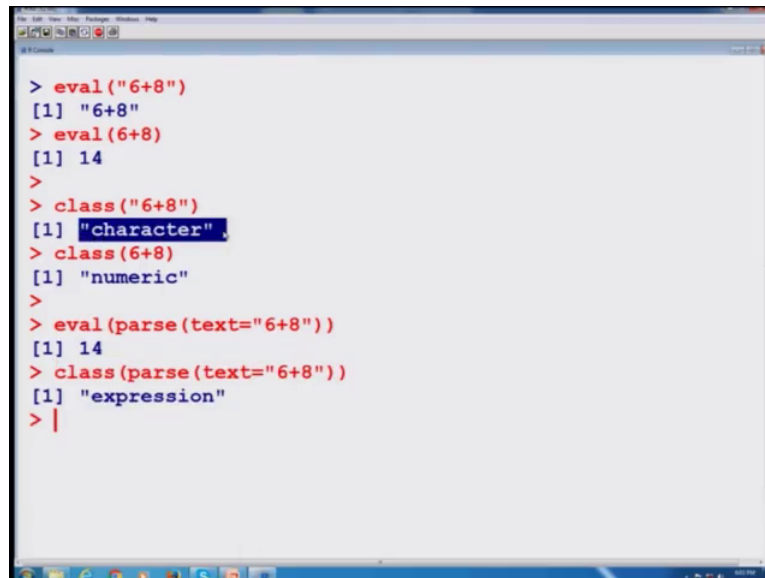
utility of parse function is that this is used to change the a string into an expression so; obviously, when you are trying to deal with data analysis where you have some numbers, where you have got some strings, you have got some characters and if you want to manipulate with the data set and if you want to change any string into an expression you can use this function. The syntax here is something like `parse` and inside the bracket inside the argument you have to give the details.

So, instead of going through with the details on the parse function why not to take here some example and we try to understand what this function is really doing. So, you have seen that here in this function here `eval` in case if you try to write 6 plus 8 inside the double quotes, it is pointing 6 plus 8 as such and in case if you want to find out the class of this string you can see here this will come out to be here as a character, but now I try to use here the parse function and I say here `text`, text is equal to this then you can see here what is happening the answer is coming out to be here 14. So, what is really happening? Here in this step this 6 plus 8 was written inside that double quotes and it was taken as a character, character means no mathematical manipulations were used.

So, as soon as I write down here `parse` and say inside that, what is my head text? text here is 6 plus 8 inside the double quotes as soon as this parse function enters inside this bracket this will delete these 2 inverted commas and this function will become a expression and then as soon as you try to use here the function `eval`. Now here this will be eval well 6 plus 8 and the answer comes out to here 14 and now in case if you try to see what is the class of this expression this will come out to be here as expression. So, you can see here this character is changed to here expression.

So, why not to do it over the R console and try to see what is really happening. So, you can see here.

(Refer Slide Time: 28:14)



```
> eval("6+8")
[1] "6+8"
> eval(6+8)
[1] 14
>
> class("6+8")
[1] "character"
> class(6+8)
[1] "numeric"
>
> eval(parse(text="6+8"))
[1] 14
> class(parse(text="6+8"))
[1] "expression"
> |
```

First I try to write down the same thing, see here this will give us like this and if I try to remove here the 6 plus 8 this will give us here 14 and if I try to. So, and if find out here the class of yes 6 plus 8 you can see here this comes out to be character, but in case if I try to find out here the class of say here 6 plus 8 you can see here this is numeric. So, now, what I try to do here that I try to use this here the function here eval and you can see here the answer comes out to be we have here again.

So, this function please try to observe where I am highlighting this expression and this expression they are giving us the same outcome right and in case if you try to find out the class of this, you can see here this is expression now it is no more R character. So, on the character we have implemented the function parse. So, that has changed its class and now I can do the mathematical manipulations because now it has become a numeric. So, and now here this is the screenshot of the same thing that you can see and you can try and now I would request all of you to please use these functions which we have discussed in different situations, try to take different examples and try to familiarize yourself with these functions and in the next lecture we will try to come up with some new topic, till then goodbye.