

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

Lecture - 40
Strings - Display and Formatting
Substitution and Replacement of Setting

Hello friends. Welcome to the course Foundations of R Software. And you can recall that in the last couple of lectures, we were talking about different types of operation related to the strings like as partitioning, pasting, etc. So, in this lecture we are going to continue with the same topic, and today we are going to learn about a new aspect of string operation that is find or replace.

You know that whenever you are trying to work with any data file or any document. Usually, there is a command like find or say control f. So, what do you do? For example, if you want to find that, if there is some string, some character, or some number, you want to find that where it is inside the document.

So, you will go to the find option and then you will type the number, string, whatever you want to search and then you say ok. So, after that it will search the entire document and it will show you that where is this word, letter, or say any string.

Similarly, you have one more operation that is replace. In replace, what do you do? That suppose in the document if there any number or character or a string and you want to replace it by something else. So, then, there is an option for replace. So, there are two boxes, in one box you write replace and then by what you want to replace.

So, the first box relates to the find and second box relates to the replace. That means, first you are asking that first you try to find out the word where it is, and then you are asking that please replace this word by this new word.

So, similar is the case in the R software also. That we have here two types of operation, find and replace. So, this operation is very useful when you are trying to work in any document, any data file or anywhere.

So, today, we are going to learn about these two types of commands. And we will try to take here couple of example through which I will try to explain you that how you can find and replace a particular string in the bigger string or a character in a string.

So, let us begin our lecture and try to see how we can do it, ok.

(Refer Slide Time: 03:04)

Operations with Strings: Substitution

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, when you are trying to find a word and you want to replace it in the R software, then we have here two functions, one is here sub and another is here gsub. So, you can see here sub means substitution. So, these are the first 3 letters which are possibly indicating the meaning of this function.

So, sub is indicating the substitution; that means, like replace. And after this there is here one more option gsub, this is also the function for a substitution, but there is a difference between sub and gsub, the way they give us the outcome and the way they work. So, this sub is working when you want to replace only the first instance where the matching happens and gsub is something like you want to replace all the strings or characters in the entire document. And you can compare it with your any other document also.

For example, when you are trying to replace something, and as soon as you write the old word and then you write the new word like this. Say old word and then you write here the new word by which you want to replace, after that if you try to enter, it gives you a different types of options say replace and say here replace all, right.

So, what really happens in the case of replace? In the replace, only the first matching will be replace. But when you are trying to say replace all, then all the string wherever is the match in the entire document that is going to be replace. So, similar is the role of here sub and gsub. So, gsub you can say here it is a sort of global replace, right or say global substitution.

So, now we try to understand these two command. So, as I told you that sub and gsub are used to replace the first instance of a substring, right. So, in order to use here this sub, what you have to do?

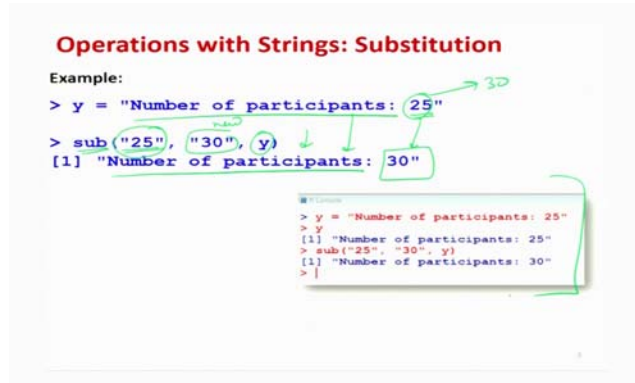
First you have to write down the word that you want to replace, right. It is something like here the old word that you want to replace. And then, after this separated by comma you have to give here new. New means, this is the new word by which you want to replace the old word.

And where you want to do? You have to give it here. This is the string under which you want to replace the word, right. So, this sub function finds the first instance of the old substring within a string and replaces it with the new substring, right. So, this old will try to search in this string that where is this string is matching, and then it will try to replace it with the new string, right. And this function here gsub, that is global substitution, this also does the same thing.

But what happened? It is like replace all. So, it will replace all instances of the substring, not only the first one. And the way it is going to be used, it is here the same, that you write down here gsub and within parenthesis, the old, the new and here the string. So, old is the old word, new is the new string, and string is the string in which you want to search and replace, right.

(Refer Slide Time: 06:11)

```
Operations with Strings: Substitution  
Example:  
> y = "Number of participants: 25"  
> sub("25", "30", y)  
[1] "Number of participants: 30"
```

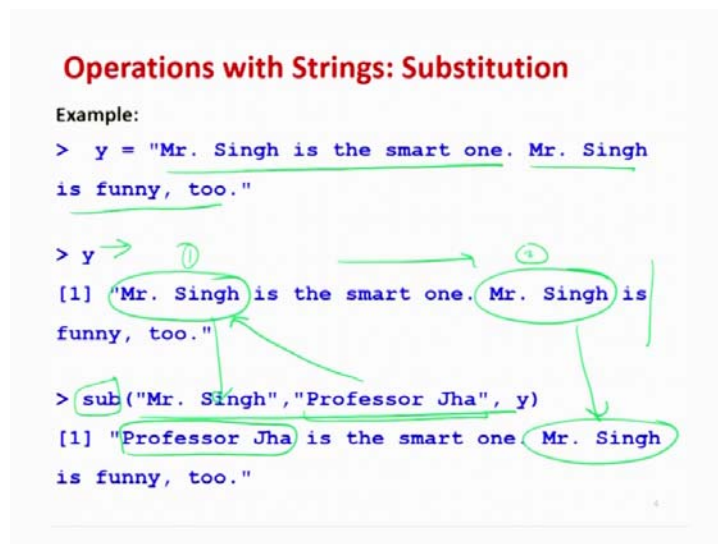


So, let me try to take here some example and try to show you that how it works. So, let me try to take here a string “number of participants colon 25” and suppose I want to replace this 25 by here 30, right. So, I can write down here like this sub and then because it is a string although “25” is a number, but it is inside the double quotes, so it is a string. So, you want to replace the string 25 by the new string that that is the new value “30”. Where? Here, here in y.

So, you can see here if you try to execute it, it will come out to be number of participant same as here, but this 25 will be replaced by here 30, right. And this is here the screenshot also.

(Refer Slide Time: 06:55)

```
Operations with Strings: Substitution  
Example:  
> y = "Mr. Singh is the smart one. Mr. Singh  
is funny, too."  
> 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, too."
```



So, I will try to show you, but let me try to take here some more example. Suppose, I take here one more string, Mr. Singh is the smart one and then I try to write once again a sentence which has Mr. Singh. So, then I write Mr. Singh is funny too, right. So, that is your string.

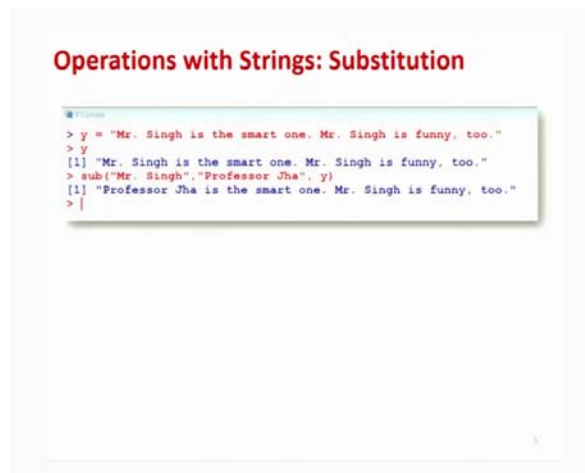
Now, what I try to do? That what I want is that I want to replace this Mr. Singh and this Mr. Singh by a new word Professor Jha, right. So, first I try to use here the command here sub and try to see that what happens.

So, when you are trying to use here the sub, then Mr. Singh is going to be replaced with the Professor Jha only at the first instance. Because the control will move from the first place, it will go in this correction. So, this is the first encounter where this sub command will find the matching with Mr. Singh, and then as soon as it finds the matching, it will replace it with Professor Jha.

And when you are trying to use here the command gsub, then it will move further. And it will also find out the occurrence of Mr. Singh at the place number 2 and there also it will replace Professor Jha.

So, in case if you try to see here the outcome of this sub command. So, you can see here this Professor Jha is coming here at the place number 1, but at the place number 2 Mr. Singh is still there. That is not changed. So, this is the role of sub, right. And this is here the screenshot of the same operation.

(Refer Slide Time: 08:18)



```
Operations with Strings: Substitution
> y = "Mr. Singh is the smart one. Mr. Singh is funny, too."
> 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, too."
> |
```

(Refer Slide Time: 08:20)

Operations with Strings: Substitution

Example:

```
> 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."
```

And now in this case if you try to use here the gsub that is global substitution, then what will happen? That here you have y Mr. Singh, Mr. Singh and this here y at two places, and now you are saying here gsub. And you want to replace Mr. Singh by Professor Jha. So, Professor Jha will be replaced at all the places in the string y, where it will encounter Mr. Singh or where it will find the word Mr. Singh, right.

So, you can see here, this is here the outcome and here, right. And if you try to recall in the sub, what was happening? It was replacing only at the one place. And at the second place the Mr. Singh was still there. There was no replacement.

(Refer Slide Time: 09:03)

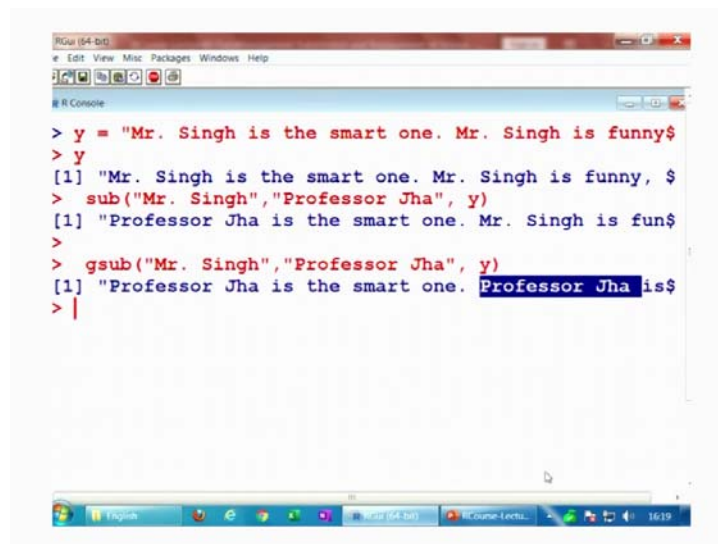
Operations with Strings: Substitution

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

So, that is the difference between the role of sub and gsub and this is here the screenshot. So, you can see here this is here Mr. Singh, this is Mr. Singh. And then in the case of gsub when you are trying to replace Mr. Singh by Professor Jha, Professor Jha is replaced here, Professor Jha is replaced here at both the places. But in the case of here sub, only the Professor Jha is replaced at the first instance. After that Mr. Singh is not replaced by Professor Jha.

So, this is how it actually works. So, let me try to show you these operations on the R console also, so that you get more confidence, so, right.

(Refer Slide Time: 09:38)



```
RGui (64-bit)
File Edit View Misc Packages Windows Help
# R Console

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

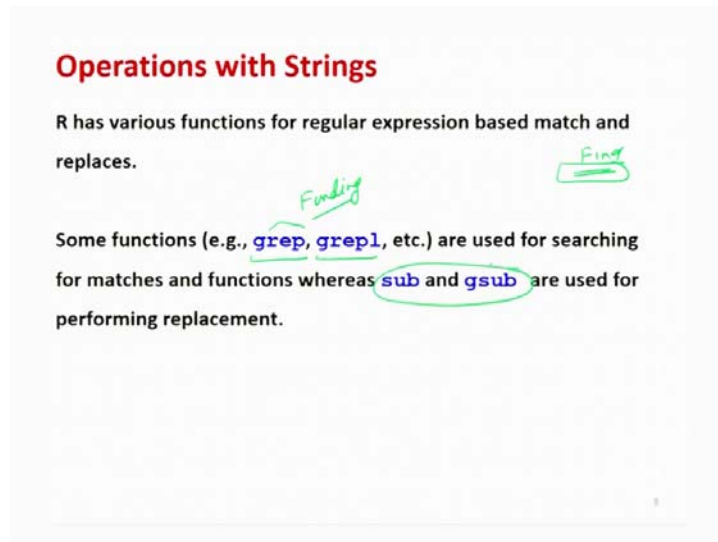
If I try to take here like this here y, and then you try to replace here 25 by 30, so you can see here this 25 is replaced by here this 30 and now you are getting here number of participant is 30, right. So, that is pretty simple, right.

And now you try to take here this here another command here and if you try to see here this is your here y, right. Mr. Singh is occurring here and Mr. Singh is occurring here also. And you try to use here the first here the command here sub, right.

And if you try to see here, now this Mr. Singh at the first instant is replaced by Professor Jha, but at the second instant Mr. Singh is also there. So, now, if you try to replace it sub by gsub. So, you can see here at both the places Professor Jha, Professor Jha, Mr. Singh. And at the second place Mr. Singh is replaced by the word Professor Jha, right.

Remaining is outside this range, but anyway that does not make any different. I have shown you here the screenshot also, right.

(Refer Slide Time: 10:44)



So, after looking into this screenshot here, you can come to now one more operation which is about find. So, this is the first command about the replacement and the global replacement. Similarly, you have here one more statement which is for here finding a match, right.

For example, if in a document if you simply want to find something that whether this number or this string or a character is occurring at what places in that document, there is an option there something like here find, right. And you try to write down these characters or string inside this box, and then you press enter. And after that it will search the entire document and it will show you that where this word is occurring with the batch.

So, now, here also you have two options. First option is that the control will start finding the word and it will stop at the first match. So, it will give you only the place where the first match occurs number one.

And second option is that it is trying to or the control is trying to search the entire document and it will inform you all the places in the document wherever is the matching occur, right. Wherever the matching occurs, it will simply try to give you the location.

Now, this location can be given in two formats, first format is that in the form of the string or in the form of some index. So, in order to do this thing, we have here two options here which is `grep` and `grepl`, right. Sometime people get confused with this word like as here, because it is here `grep`. So, sometime people get confused that `rep` is trying to indicate the replacement. But you please do not get confused. It is not replacement, but it is only finding for replacement you had this `sub` and `gsub` commands, right.

(Refer Slide Time: 12:29)

Operations with Strings: `grep`

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, ignore.case = FALSE)` 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.

The slide contains several green annotations: a circle around the 'R' logo, a circle around the 'x' parameter in the code example, a circle around the 'ignore.case' parameter, and a line underlining the return value description.

So, how to use this commands? Let me try to show you here. And they work almost exactly in the same way as you try to find something in a document in the usual way, right. So, the meaning of this `grep` is that it globally search the regular expressions, and it will try to show you the match on your computer screen.

So, the way it will work here is this you write `grep`, and then pattern you want to search, right. And where you want to search? In the string here `x`, right. And after that there is an important option which is here like this, `ignore dot case`, all in lower case alphabets.

So, now, you can see that after reading this statement, that `ignore dot case`, do you think that you can understand what it is trying to say? It is simply trying to say whether you want to consider the lower case or say upper case alphabets or you simply want to ignore.

For example, if somewhere it is written here capital R or somewhere it is small r. So, do you want to differentiate between the capital letter and a small letter or the uppercase or lowercase alphabets or you want to consider them both as simply R, right. So, this is the meaning of that ignore dot case.

So, if you try to see here ignore dot case is equal to FALSE, this will try to match and it will try to ignore the characteristic of upper case or lower case of any alphabet, right. And if you try to say here TRUE, then it will also match the case. And then it will try to return an integer vector of the indices of the element of x that yielded a match.

(Refer Slide Time: 13:59)

Operations with Strings: grep

ignore.case : if **FALSE**, the pattern matching is **case sensitive** and if **TRUE**, case is ignored during matching.

value : if **FALSE**, a vector containing the (integer) indices of the matches determined by **grep** is returned, and if **TRUE**, a vector containing the matching elements themselves is returned.

So, let us try to see what happens, right. So, what does this actually mean? That when you are trying to find something in a string, then you have two options. First option is that it will try to give you the location in terms of a string where the matching occurs or the second option is that it will try to give you the index of that string where the string is located, right.

So, as I said, means if you try to use here this ignore dot case, it will be related to the case sensitive behavior of the characters. And if you try to use here one more option here value, if this value is here FALSE, then there will be a vector of integer. Actually, that is indicating the location in terms of the indices of the matches which is determined by this grep, right.

And if this is here TRUE, then what will happen? Then, it will try to give you the whole string itself. So, if you try to put value equal to FALSE, then you will get the outcome in terms of some index, some numbers. And if you try to put a value equal to TRUE, then you will get the value of the string where the match is occurring.

(Refer Slide Time: 15:07)

```
Operations with Strings: grep
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")

> grep("ex", str, value=T)
[1] "exercises" "include examples of R
language"
```

So, let us try to see the application of this function through this example. And after that I will try to show you that how you are going to work with the grepel, right.

So, here I am now going to use here the option here this value equal to TRUE. So, I will use here grep. And then within parentheses, I will try to give here the pattern. And then the string in which I want to correspond this pattern, and then I am trying to give here value is equal to TRUE, right.

So, in case of value equal to TRUE, what will happen? The outcome is going to be in terms of the string. So, if I try to take here this data vector of a string, say first is here R course, at a second place this is here exercise, third place it is here include examples of R language. So, you can see here, this is place number 1, this is place number 2 and this is here place number 3. So, there are 3 strings in this data vector. And I am calling it as a str.

And then now what I want here is that in this string str, I want to find where is my this “ex” I try to give this ex inside the double quotes and then I try to give the string in

which I want to find. And now I want to know the result in terms of the string, so I try to give here values equal to TRUE.

Now, first you try to see where is this ex is occurring. So, you can see here first ex is occurring here and then the next ex is occurring here. So, it is going to give you here this. So, if you try to see here that ex is occurring in second and third. So, these two are occurring here, but there is no ex in the string number 1 or the string which is at the position number 1. So, that is why it is not occurring here.

So, that is why now you can see grep is trying to give you here all the strings wherever this ex is occurring.

(Refer Slide Time: 16:51)

Operations with Strings: grep

`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
```

Now, I try to do here one thing, that I simply try to use here value is equal to FALSE, right. In the earlier case, you have used here value is equal to TRUE, right. You can see here. Now, I try to use here value is equal to FALSE. The same string `str` I try to consider here, right and if you try to see.

The answer is now coming here as a 2, 3. What is this 2, 3? As I said this is my location number here 1, this is my location number here 2, and this is my here location number 3. So, there are 3 strings in the data vector `str`. And this 1, 2, 3 their locations are indicated by this outcome.

So, this 2 is indicating because you have here ex, so it is indicating that ex is occurring at the string which is placed at the second index. And then you have here ex here. So, this 3 is indicating that then your ex is occurring at the string which is on the third position in your data vector str. So, this is the meaning of say using value equal to TRUE or FALSE.

And this here you can see this is here the screenshot of the same outcome.

(Refer Slide Time: 17:48)

```
Operations with Strings: grep

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

And now I try to show you here one more example here, where I am trying to use the option ignore dot case, right. That you are ignoring the property of lower case or upper case of the alphabets, right.

(Refer Slide Time: 17:53)

Operations with Strings: grep

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

```
> str = c("R Course", "exercises", "include examples of r language", "in R software.")
> grep("R", str, ignore.case=F, value=T)
[1] "R Course"          "in R software."
> grep("R", str, ignore.case=T, value=T)
[1] "R Course"          "exercises"
[3] "include examples of r language" "in R software."
```

The slide includes handwritten annotations: blue circles around "R" in the pattern and "R" in the strings; blue circles around "exercises" and "include examples of r language" in the second output; a green arrow pointing to the second output with the text "Output as string"; and blue circles with numbers 1, 2, 3, and 4 around the output elements.

So, when you are trying to use this function here `grep`, then you try to write down here the pattern that we want to search, where in the string `x`. And then, you are using the option here that ignore case is equal to `2`. That means, you please ignore the case. I am not bothered whether the matching is happening with respect to the lower case or with respect to the upper case.

So, I try to consider here an example which has here 4 strings, like as first here is “R course”, then “exercises”, then “includes example of r languages”, “in R software”. And yeah means you can see here I am trying to use here is small `r`, lowercase alphabet and here I am trying to use here capital `R`, right. And here also in the first string also we have here capital `R`.

Now, I want to replace the capital “`R`” by this in this string `str` which is here and I want to ignore dot case is equal to here `FALSE`, right. That means, you are not allow to ignore the case, right. Ignore case is `FALSE`, that means, you cannot ignore the case. And then value is equal to here `TRUE`. So, value equal to `TRUE` mean, that means, it is going to give you the output as a string, right. So, I want to show you here both.

So, now, if you try to see the control comes here and see here capital `R`, and then it finds the capital `R` here at two places, this and here this. And this here is lower case alphabet. So, when you try to use here the command `grep` and you say here please replace capital `R` in `str`, then it will give you here these two outcome. This is here `R course` and in `R software`. So, `R course` is coming from here and in `R software` it is coming from here, right.

Now, on the other hand, in the same screen I try to show you one more option that now I try to change this option, and I try to write down here ignore dot case is equal to `TRUE`, `T`. So, now, if you try to operate the same command here, and now it is asking that please find capital `R` in the string, `str`, and you have to ignore the case because it is `TRUE`, right. And value is equal to `TRUE` means that you want the answer in terms of here is string, otherwise only the index of the positions of the strings is given.

So, now if you try to see here this here is a matching here “`R course`”, which is here happening `R`, then you have here another here `R` where you can see here like this, right.

So, this is coming here in this “exercise”. And then you have r here, and this is coming here in the string number here 3. And then in the 4 here this is here R.

So, all the 4 strings are coming here as in the output. When you are trying to say here that ignore case is equal to TRUE and when you are trying to say ignore case is equal to FALSE, then only two outcomes are there, right. So, this is how you have to understand the use of ignore dot case.

(Refer Slide Time: 20:58)

```
Operations with Strings: grep

> grep("R", str, ignore.case=T, value=F)
[1] 1 2 3 4

> grep("R", str, ignore.case=F, value=F)
[1] 1 4
```

```
> str = c("R Course", "exercises", "include examples of r language", "in R software.")
> grep("R", str, ignore.case=T, value=F)
[1] "R Course"
[2] "in R software."
>
> grep("R", str, ignore.case=T, value=T)
[1] "R Course"
[3] "include examples of r language" "in R software."
>
> grep("R", str, ignore.case=F, value=F)
[1] 1 2 3 4
>
> grep("R", str, ignore.case=F, value=T)
[1] 1 4
>
```

And in the same outcome if you try to use here value is equal to FALSE. That means, you need the outcome in terms of the location indices. So, you can see here when you are trying to ignore the case, then this R is present in all the strings at all the locations. So, all the location 1, 2, 3, 4 are given here. It is here like this, this is here 1, this is here 2, this is here 3 and this is here 4.

And similarly in the case of this first example, this R course and in R software, they are occurring only at first and fourth location, right. So, now, if you try to see here all these outcomes, I can give you in brief from this screenshot that you are trying to find out here R, when you are trying to say here ignore case F, then R course is here where you have here R and then in there is a R in the R software.

And if you try to change this here as value is equal to TRUE to value equal to FALSE, then it is giving you the same thing here as a 1 and 4. And in case if you try to take the

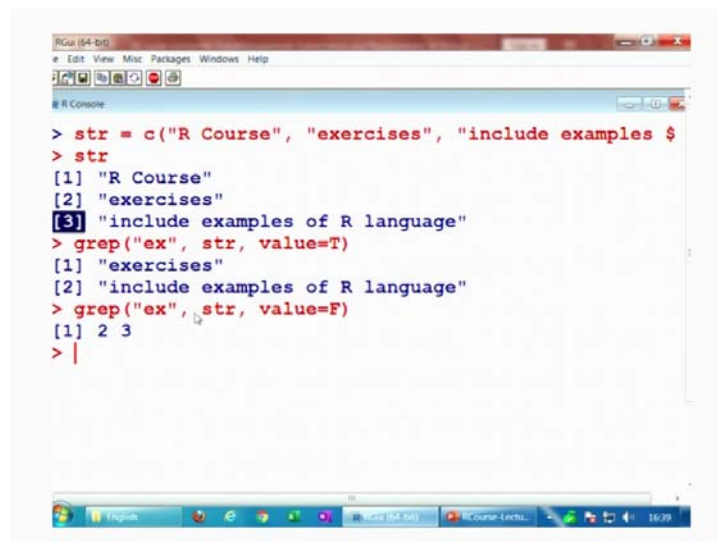
second case, where you are trying to say ignore dot case is equal to TRUE, and the value is equal to TRUE and FALSE, then in the case of first, you are getting here all the values 1, 2, 3, 4. And then in the second case you are getting, the same result in terms of numbers.

(Refer Slide Time: 22:14)

```
Operations with Strings: grep  
Example:  
> x = "R course 24.07.2021"  
> y = "Number of participants: 25"  
  
> c(x,y) # Combine the two strings  
[1] "R course 24.07.2021" "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, you can see here this is not a very difficult thing. So, let me try to first show you these outcomes in the R software, so that you become more confident. And then I will try to something else, right. So, I try to create here this string, right.

(Refer Slide Time: 22:33)

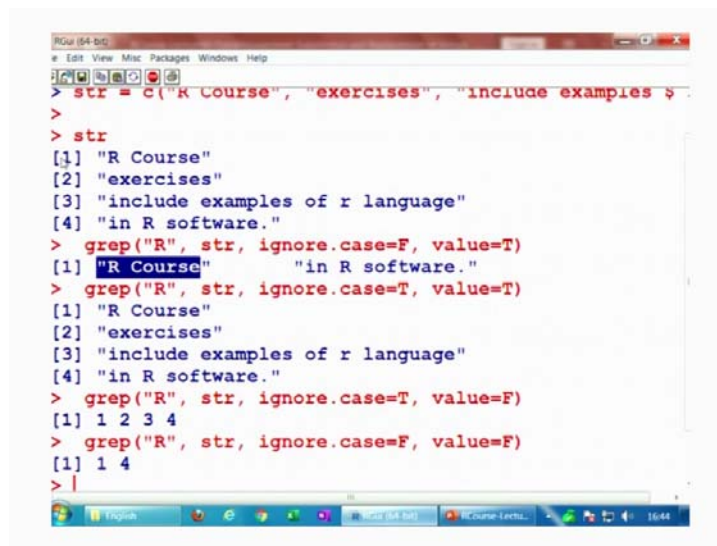


```
R Console  
> str = c("R Course", "exercises", "include examples of R language")  
> str  
[1] "R Course"  
[2] "exercises"  
[3] "include examples of R language"  
> grep("ex", str, value=T)  
[1] "exercises"  
[2] "include examples of R language"  
> grep("ex", str, value=F)  
[1] 2 3  
> |
```


So, this is your here string. And then you try to replace here, and then if you try to see here it is coming out to be here. If you are trying to find out here ex, so ex is here and ex is here. So, in second and third. So, these two are coming.

And if you try to give here value is equal to here FALSE, then you can see here it is giving you here the location number 2 and 3. So, this is here this 2 and here this 3 that is coming, right. So, that is how it is working, right.

(Refer Slide Time: 23:05)



```
RGui [64-bit]
File Edit View Misc Packages Windows Help
> str = c("R Course", "exercises", "include examples of r language", "in R software.")
> str
[1] "R Course"
[2] "exercises"
[3] "include examples of r language"
[4] "in R software."
> grep("R", str, ignore.case=F, value=T)
[1] "R Course"      "in R software."
> grep("R", str, ignore.case=T, value=T)
[1] "R Course"
[2] "exercises"
[3] "include examples of r language"
[4] "in R software."
> grep("R", str, ignore.case=T, value=F)
[1] 1 2 3 4
> grep("R", str, ignore.case=F, value=F)
[1] 1 4
> |
```

Similarly, if you try to take here this example here, where I am trying to take here this rep command with this string here is like this. And if you try to see here this grep, so you are trying to see here consider here the capital R and try to find where this capital R is occurring.

So, it is giving you here these two values. So, you can see here this capital R is occurring here and capital R is occurring here. And since you are trying to say that ignore dot case is equal to FALSE. So, here also there is lower case r, but that is ignored.

Now, in the same example, if you try to make it here TRUE, then you can see here capital R is occurring here, small r is occurring here in the second position, then R is occurring in the third string also and R is occurring in the fourth string also. So, you can see here this is here coming like this.

And in case if you try to make it here value is equal to here FALSE, so you can see here that it is coming out to be this 1, this 2, this 3 and this 4 this is coming here as 1, 2, 3 4, right.

And in case, if you try to take here his command and you try to make it here value is equal to here TRUE, then instead of now you take a value is equal to here FALSE, so you can see here this is coming here 1 and 4, right. So, this is the value at the first location and this is the value at the fourth location which is coming here as say 1 and 4. So, you can see here this is how grep actually works.

So, now, I try to take here one more example here. And in which, what I am trying to do? I am trying to consider here two strings, right. One here is x and y, x here is “R course 24 dot 07 dot 2021” and y here is say “Number of participants colon 25”.

And then I am trying to combine these two strings which is giving me here like a c x, y. So, this will become here like this, that the value at x will be here and the value at y will be here.

Now, if you try to see, I want to find “our” in this c x, y. So, I try to find out here this “our” in this c x, y. So, now, can you find here where is this “our”? There is no our, but if you try to see very carefully, this inside the course there is a word here o u r and that is what is giving you here as say 1, right. And there is no o u r in here y. You can see here. So, this is how actually it works.

(Refer Slide Time: 25:20)

```
Operations with Strings: grep

Example:
x = "R course 24.07.2022"
y = "Number of participants: 50"

c(x,y) # Combine the two strings
[1] "R course 24.07.2021" "Number of
participants: 25"

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

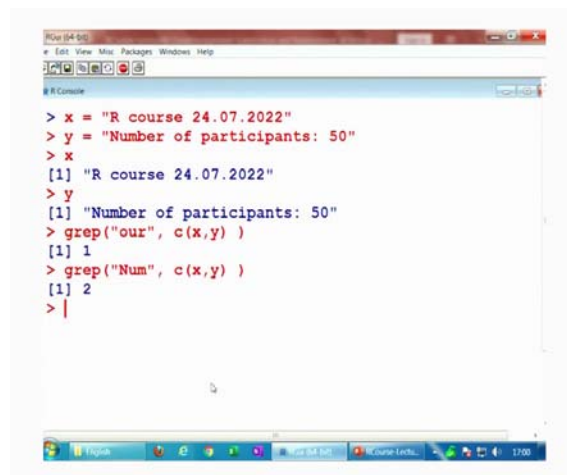
"Num" is in the 2nd element (in the word "Number"), therefore
in y.

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

And similarly if I try to repeat the same example, but now suppose I try to find out here another word here “Num” n u m and you can see here that in this first string there is known num, but in the second string there is here a num in the number and so this is head position number 2, “Number of participant colon 25”.

So, when you are trying to write down here grep and say here “Num”, you can see here it is trying to give you say here answer 2, right. So, this is how it actually works.

(Refer Slide Time: 25:58)

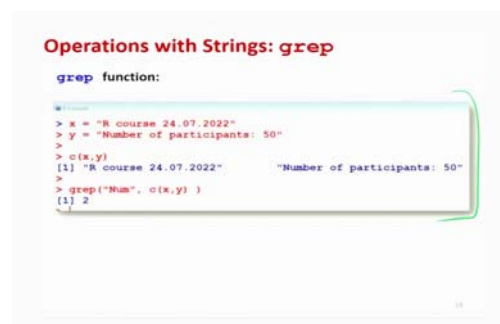


```
R Console
> x = "R course 24.07.2022"
> y = "Number of participants: 50"
> x
[1] "R course 24.07.2022"
> y
[1] "Number of participants: 50"
> grep("our", c(x,y) )
[1] 1
> grep("Num", c(x,y) )
[1] 2
> |
```

So, let me try to show you this operation on the R console also, and then I will try to give you one more example. So, let me try to create here this my here x and y.

So, x here is like this, y here is like this, and after that first I try to see our word here “our”, right. So, you can see here this is here “our” at here like this, and if you try to see here “Num” num is occurring at 2, right.

(Refer Slide Time: 26:20)



Operations with Strings: grep

grep function:

```
> x = "R course 24.07.2022"
> y = "Number of participants: 50"
>
> c(x,y)
[1] "R course 24.07.2022"      "Number of participants: 50"
>
> grep("Num", c(x,y) )
[1] 2
```

So, that is pretty straightforward; that here I wanted to show you that you can combine the two strings together and then you can work also. And this is here the screenshot of the same operation that I just shown you, ok.

(Refer Slide Time: 26:27)

Operations with Strings: grepl

`grepl(pattern, x)` returns a character vector containing the selected elements of `x` and the outcome is in terms of **TRUE** and **FALSE**. Indicating if the matching is available or not.

```
> str = c("R Course", "exercises", "include  
examples of R language")  
  
> grepl("R", str)  
[1] TRUE FALSE TRUE
```

Now, I would like to give you here the details about the command `grepl`. So, `grepl` also works like a `grep`, but the only difference is that in this case the outcome is coming in terms of **TRUE** or **FALSE**, logical **TRUE** or logical **FALSE**, right. So, when you are trying to find out something, then for example, in the case of `grep` you are getting number or the string in which the value is present.

Now, it will try to give you the outcome in terms of logical **TRUE** and logical **FALSE**, and this will be for all the strings in your string in which you want to start, right. For example, if data vector has 3 different string, then this **TRUE FALSE** will be there 3 times.

And when the answer is **TRUE**, that means, that character is present, the matching is done successfully. And if the answer is **FALSE**, then that means, the matching was not successful.

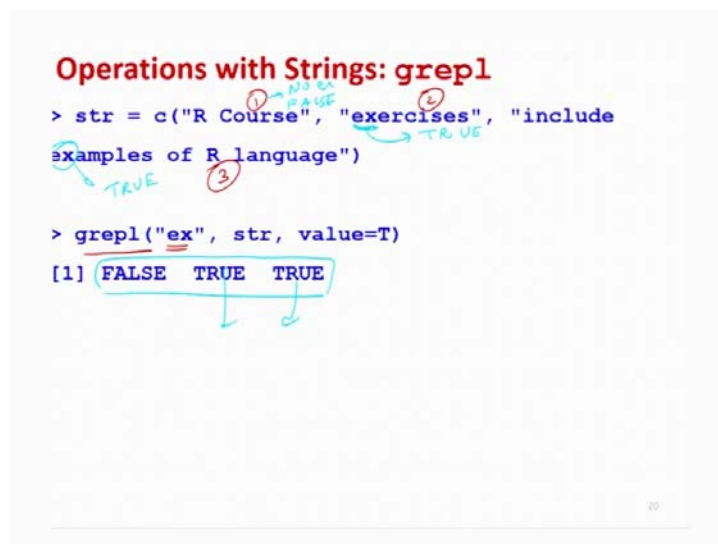
So, let me try to take here one example and try to show you here. Suppose I try to take here the same example, right that your string is `str`, “R course”, “exercises”, “includes examples of R language”. And now if I try to use here `grepl` and I try to look here for the

word R or the alphabet R. So, you can see here now I am trying to give it here R where in the str, now if you try to see this is here like this string number 2. Here you have the R is present. So, it will try to give you answer here TRUE.

Then you come to here, the secondary string, here there is no R, means capital R, so this is here FALSE. Then in the third one which is here like this there is here R. So, this R here is present. So, it will say here TRUE and this is here the outcome. You can see a TRUE, FALSE, and TRUE, right.

So, now, after looking into this statement TRUE. FALSE. TRUE. you have to identify where the R is occurring. So, wherever you have an answer TRUE, the R is occurring there.

(Refer Slide Time: 28:20)



```
Operations with Strings: grepl
> str = c("R Course", "exercises", "include
examples of R language")
> grepl("ex", str, value=T)
[1] FALSE TRUE TRUE
```

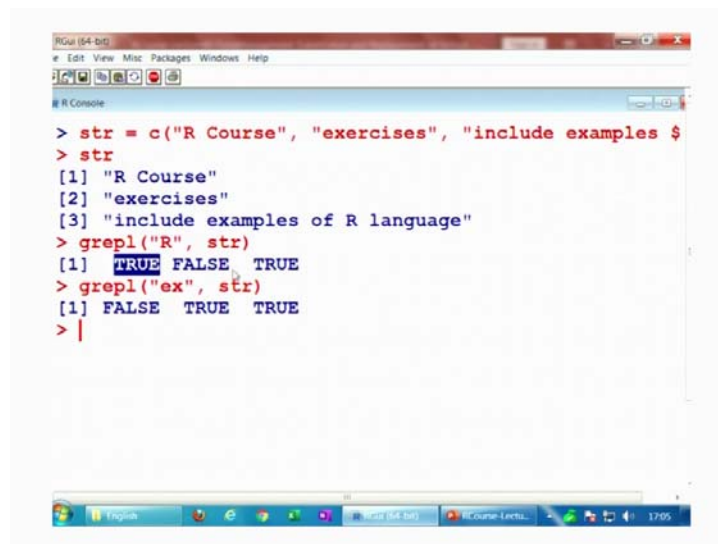
The screenshot shows an R console session. The first line is the title "Operations with Strings: grepl". The second line is the command `> str = c("R Course", "exercises", "include examples of R language")`. Handwritten annotations include a circled '1' above "R Course" with an arrow pointing to "FALSE", a circled '2' above "exercises" with an arrow pointing to "TRUE", and a circled '3' below "include examples of R language" with an arrow pointing to "TRUE". The third line is the command `> grepl("ex", str, value=T)`. The output is `[1] FALSE TRUE TRUE`, with a box around the output and arrows pointing from the circled numbers in the first line to the corresponding output values.

And similarly, if I try to take the same example, but I try to use here the grepl with the say word here ex, right. So, you can see here that this is here 1, this is here 2, and this is here 3.

Now, in this case in 1, there is no ex, so the answer is coming out to be here FALSE. And then there is here is an ex, so answer here is TRUE. And then there is here ex in 3, so the answer will come out to be here TRUE. So, this is you can see here FALSE, TRUE and TRUE, right.

So, now if you try to look at this answers, since because you have here TRUE at these two places, so that is going to indicate that the ex is present in the string number 2 and 3 in the data vector str.

(Refer Slide Time: 29:11)

A screenshot of the R GUI console window. The window title is 'RGui (64-bit)'. The menu bar includes 'Edit', 'View', 'Misc', 'Packages', 'Windows', and 'Help'. The console shows the following R code and output:

```
> str = c("R Course", "exercises", "include examples of R language")
> str
[1] "R Course"
[2] "exercises"
[3] "include examples of R language"
> grepl("R", str)
[1] TRUE FALSE TRUE
> grepl("ex", str)
[1] FALSE TRUE TRUE
> |
```

And in case if you try to do this operation on the R console also. So, you can see here this is your here string and now you are simply trying to use here, see here grepl for this string and you can see here this is here TRUE, FALSE, TRUE. And if you try to replace here the word here R by here ex, we can see here this comes out to here FALSE, TRUE, and TRUE.

Why? Because this R is here, here and here. So, it is giving you answer TRUE and TRUE for the first and third. And there is no R in the second, so it is giving you here FALSE. And then similarly, here for this ex in there is no ex in 1, there is ex in 2 and there is ex in 3, so it is giving you here FALSE, TRUE and FALSE, right.

And this is here the screenshot of the same operation which I shown you, ok.

(Refer Slide Time: 29:47)

Operations with Strings: grepl

```
R Console
> str = c("R Course", "exercises", "include examples of R language")
> str
[1] "R Course"
[2] "exercises"
[3] "include examples of R language"
> grepl("R", str)
[1] TRUE FALSE TRUE
> grepl("ex", str)
[1] FALSE TRUE TRUE
> |
```

21

So, now I come to an end to this lecture. And I stop here. So, you can see here that in this lecture, we had just learnt about two functions which are used for finding and replacement, finding and substitution, and they had two more variants. So, you have to understand that how these functions are going to work.

And it depends on you that what do you want, whether you want the terms of; whether you want the result in terms of strings or say number or TRUE or FALSE, etcetera, etcetera, based on that you can write the suitable function, with suitable options.

So, now I will request you as usual once again, that you please try to take some examples and try to execute this command. They are very simple commands, but the main thing is that you have to understand that how the output is going to be. If it is terms of numbers, then what it is trying to indicate, and if it is terms of TRUE and FALSE, then what it is trying to indicate in the original string.

So, you try to practice it. And I will see you in the next lecture.

Till then, goodbye.