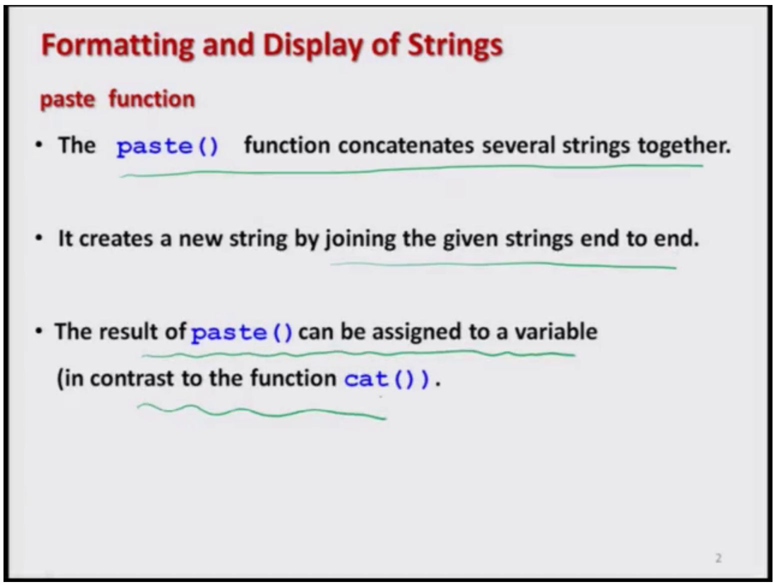
Introduction to R Software

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

Lecture - 26
Paste Function

Welcome to the next lecture on the course introduction to R software. You will call that in the earlier 2 lectures, we had talked about how to manipulate the outcome in terms of display, how the outcome should look like and then we had discussed several commands one was spring and another was cat. Now we will try to continue on the same lines and we will try to learn another function; what is called as paste function. Paste function also helps in manipulating the display outcome.

(Refer Slide Time: 01:02)



Formatting and Display of Strings

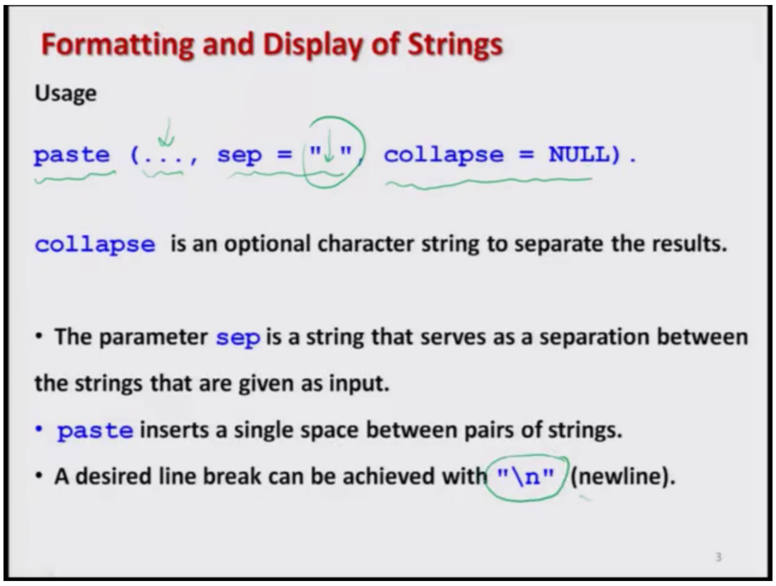
paste function

- The `paste()` function concatenates several strings together.
- It creates a new string by joining the given strings end to end.
- The result of `paste()` can be assigned to a variable (in contrast to the function `cat()`).

2

So, we try to understand what is this? This paste function concatenates several strings together and this function creates a new string just by joining all the given string from end to end, but if you try to recall the same thing was being done by the cat function also, then what is the difference between the 2? This is the main difference; the results of paste function can be assigned to a variable and that was not possible in cat function that is the major difference between. The 2 the syntax and other details of the paste functions are as like this that; we try to write down here paste, then whatever we want to paste here that will come here.

(Refer Slide Time: 01:46)



Formatting and Display of Strings

Usage

```
paste (..., sep = " " collapse = NULL) .
```

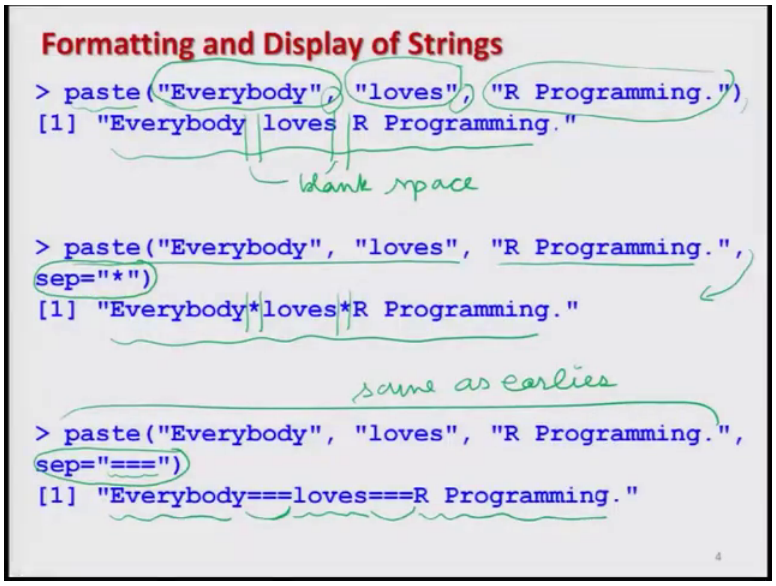
`collapse` is an optional character string to separate the results.

- The parameter `sep` is a string that serves as a separation between the strings that are given as input.
- `paste` inserts a single space between pairs of strings.
- A desired line break can be achieved with `"\n"` (newline).

And then how the strings are going to be separated that symbol or say anything; whatever you want; this has to be given here inside the double quotes and then there is another function here; collapse and this also has a rule that we will try to see with the applications and after you have pasted certain a strings, then in case if you want to change the line, then we try to give the command say backslash n inside the double quotes; that means, we want here a new line. So, let us now try to take some say example, but before that if you want to learn more about this paste function, I will say take the help from the R and you can see here that it comes to the website of the R software and here it gives you all the details.

Now you can see here; this collapse is given here that it is an optional character string to separate the results, right. So, all these details are here, I would request you that if you want to understand it more; please try to read it and I would like to take your several examples to explain you the usage of paste function.

(Refer Slide Time: 03:32)



Formatting and Display of Strings

```
> paste("Everybody", "loves", "R Programming.")
[1] "Everybody loves R Programming."

> paste("Everybody", "loves", "R Programming.",
sep="*")
[1] "Everybody*loves*R Programming."

> paste("Everybody", "loves", "R Programming.",
sep="===")
[1] "Everybody===loves===R Programming."
```

Now, let me take here several examples to explain the use of this paste function. For example, if you try to see here, I have taken a an example where there are 3 strings; one that is everybody, second is string loves and third is strings characters R programming and what we want; I want to write everybody loves R programming.

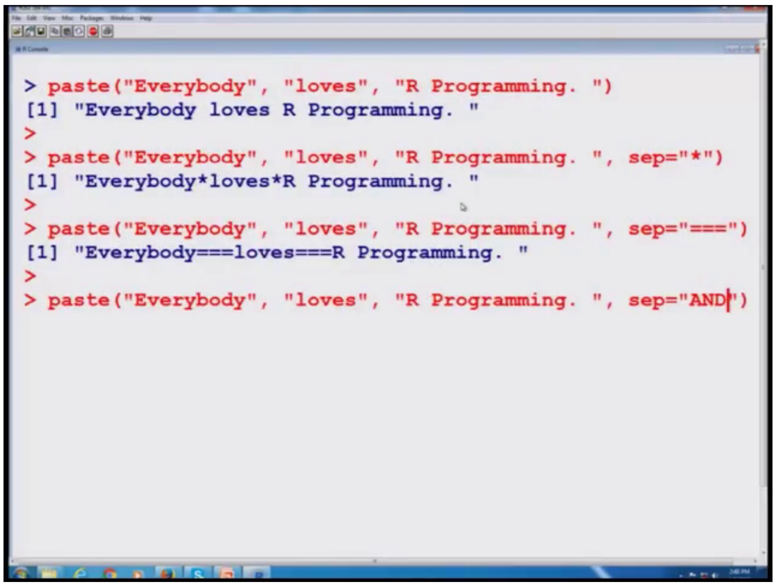
So, for that I can use here a function here paste and inside the arguments; inside the brackets, I have to give all these strings which are separated by the commas and here is the outcome that you can see over here everybody loves R programming.

Now, here you can see that the separation between 2 words or 2 strings; this is only here a blank space. So, in case if you choose the default choice for the separation between 2 strings, then R chooses as blank space, but suppose you want different separator for example, I want to have a separator say star.

So, let me take here the same example paste and the 3 strings everybody loves and R programming and I try to write here one more option here, set equal to double quotes and then star set means separator. Now if you try to see the outcome, the outcome will come like this the same outcome everybody loves R programming, but the separation between 2 strings is being done by say star and similarly, to understand it better; suppose I want to have a separator which is 3 equality signs. So, you can see here I try to give it here and this is the same as earlier no change.

Now you see the outcome comes like this everybody first string love second string R programming third string and they are separated by the symbol 3 equality signs.

(Refer Slide Time: 06:13)

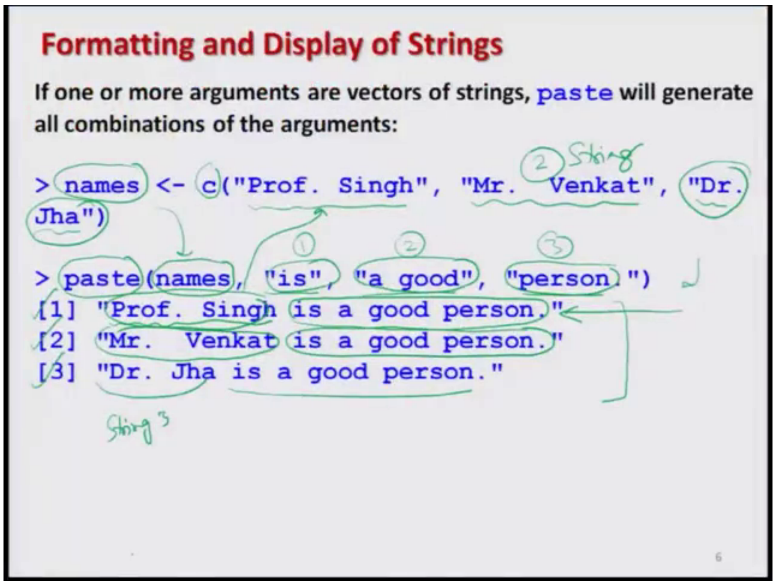


```
> paste("Everybody", "loves", "R Programming. ")
[1] "Everybody loves R Programming. "
>
> paste("Everybody", "loves", "R Programming. ", sep="*")
[1] "Everybody*loves*R Programming. "
>
> paste("Everybody", "loves", "R Programming. ", sep="===")
[1] "Everybody===loves===R Programming. "
>
> paste("Everybody", "loves", "R Programming. ", sep="AND")
```

Now, let us try to see that what happens when we try to execute these functions over the R console; you can see here; when I try to type the same command it gives me the same thing and here you can see that the 3 strings everybody loves and R programming, they are separated by the blank space. Now I want to change let us space and I want to give here space by star. So, I try to use here the command this separator or say sep is equal to say star and we get here this outcome where all the separators are say star.

Here you can see and similarly if I want to change this separator by this see here see here 3 quality signs, we can do it here by like this and even want we have here something here more whatever you want to do; say if I say here and then you can see here this is and so whatever you want to do; you can play with this paste function to get an required to get the required outcome. So, now, you let us come back to our slides and you can see here because of the screen shot of whatever we had obtained in this outcome.

(Refer Slide Time: 07:36)



Formatting and Display of Strings

If one or more arguments are vectors of strings, paste will generate all combinations of the arguments:

```
> names <- c("Prof. Singh", "Mr. Venkat", "Dr.
Jha")

> paste(names, "is", "a good", "person.")
[1] "Prof. Singh is a good person."
[2] "Mr. Venkat is a good person."
[3] "Dr. Jha is a good person."
```

2 String
1
2
3
String 3

Now, let me take here another example and try to explore a situation where I want to paste the strings with certain vector; that means, I have got for example, list of students and I want to write that this student is excellent; what I want that R command should go to my list that is containing the names of the students and it should pick one name at a time and after every name, the outcome should be that the student name; whatever is picked is a good student.

So, we are trying to use the paste function using vector and we are trying to paste some characters with each of the element in the vector. So, let me take here this example. So, now, if you see what we want we have some number of values which are assigned to a vector. So, the vector is containing all the values and I want to paste a string with each and every element in that vector. So, let us try to take an example and try to understand how it can be done. So, we have taken here 3 strings, they are the 3 names Professor Singh, Mister Venkat and Doctor Jha and I have stored these 3 names as a string in this vector called as here say names using the c command

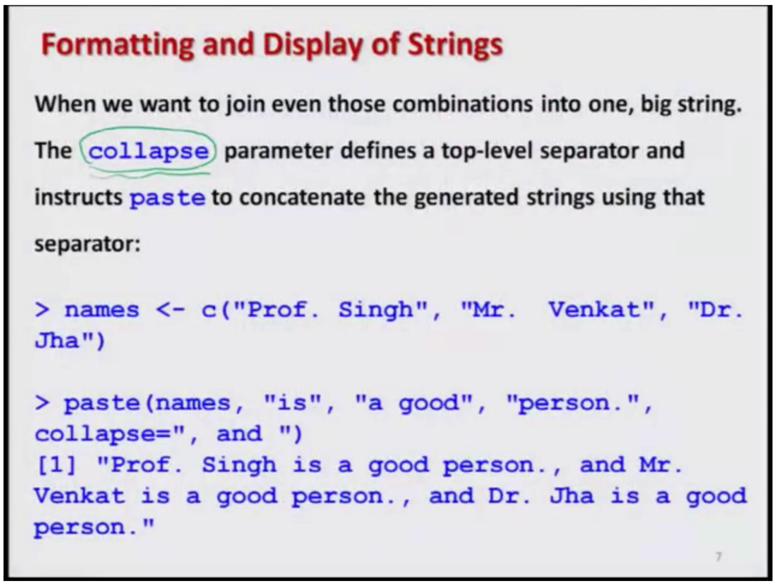
And now I am using here a paste function and my objective is that that I want to write Professor Singh is a good person, Mister Venkat is a good person and Doctor Jha is a good person. So, what I do here that I try to write down my here names variable vector here and then I write here is a good person and this is my string number one this is my string number 2 and this is my string number 3.

Now, once I try to execute it we get here this type of outcome, first we are try to understand; what is this outcome and how it is coming. You can see here; it is writing Professor Singh and then is a good person, this Professor Singh is coming, from here the first element in my vector names and then is a good person this is coming from is from here a good from here and a person from here and finally, it is giving me the complete sentence Professor Singh is a good person.

Now, in the next command; the R automatically goes to the second string which is here Mister Venkat; it chooses Mister Venkat and it writes is a good person just by repeating the same string is a good person, then in the next step, the control of R goes to the third string which is here Doctor Jha; Doctor Jha is my string number 3 and it goes to the third element in my vector names and pickups Doctor Jha and then pickups is a good person from the given a string. So, you can see here we have got here 1, 2 and 3 outcomes. Similarly if you have more you can do it yourself. So, now, let us try to do this example on the R console.

So, I try to create here a vector here; see here names and then I use my here paste function and you can see here we get this outcome know how we come back to our slides and try to make this example little bit more useful.

(Refer Slide Time: 12:49)



Formatting and Display of Strings

When we want to join even those combinations into one, big string. The collapse parameter defines a top-level separator and instructs paste to concatenate the generated strings using that separator:

```
> names <- c("Prof. Singh", "Mr.  Venkat", "Dr.
Jha")

> paste(names, "is", "a good", "person.",
collapse=", and ")
[1] "Prof. Singh is a good person., and Mr.
Venkat is a good person., and Dr. Jha is a good
person."
```

And for that; I am going to introduce here new function collapse; what do you really mean by this word collapse this is something like if you have a stag several thing one over other 1, 2, 3, 4 and so on and you see you say these things are collapsing lap just like this and the entire a string which are given a string one is string to string 3, they will collapse and then they will be written in a single line that is what is the meaning literal meaning of this word and then there can be a separation also.

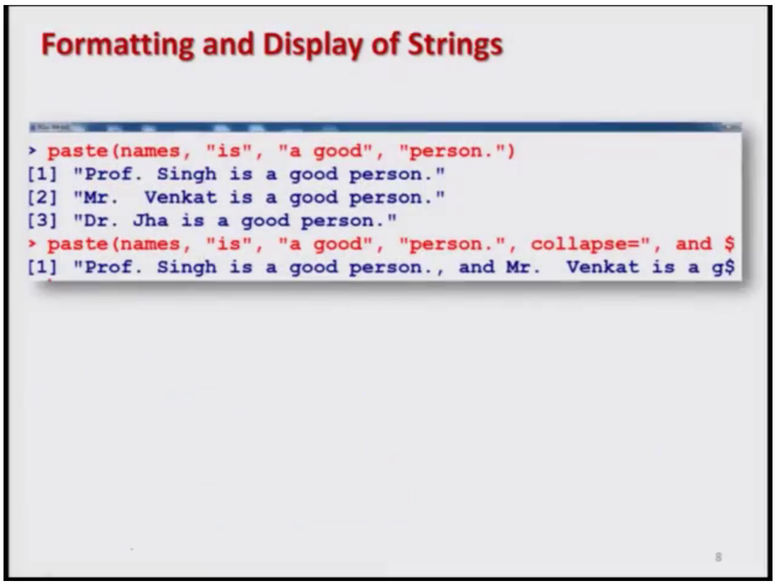
So, first we try to understand; what is the role of this function collapse. So, now, if your objective is that instead of writing 3 statements what we did in the earlier case that we have got 3 statements you can see over here 1, 2 and 3, but now I want to write all these 3 a strings or 3 statements in a sequence a statement number here one a statement number here 2 and statement number here 3 this is here; Professor Singh is a good person, followed by Mister Venkat is a good person and followed by Doctor Jha is a good person, but in a single line not in 3 lies.

So, for that I can use the function collapse. This is a function which joins the combination into one big string and it uses in site itself the paste function and it instructs that please concatenate all the a strings together by a given separator. So, let me take at the same example where I have 3 name; Professor Singh, Mister Venkat and Doctor Jha and these 3 names are contain in a vector names and now I am using here the same command as earlier which is the paste names is a good person, but I am adding here one more option which is collapse and then I am saying that this comma and is the separator.

Now, once you do this thing you will get here this type of outcome you can see here there is only one line Professor Singh is a good person and this comma this is coming from here this collapse Mister Venkat is a good person then comma and this again coming from this separator; Doctor Jha is a good person. So, you can see now the difference between the earlier outcome where you had 3 lines outcome and now you have here one line outcome, let us try to see over the R console and see the outcome. So, you can see here that here there are 3 statements which are highlighted here, but here when you try to use here the separator this collapse and then you can see here that we have got here one single sentence which is here starting from here one processing is a good person Mister Venkat is a good person and followed by and Doctor Jha is a good person; well since this statement is very long. So, it is going beyond the screen of the computer, but the outcome is the same.

So, now, you let us come back to our slide and here is the screenshot of this outcome.

(Refer Slide Time: 17:22).

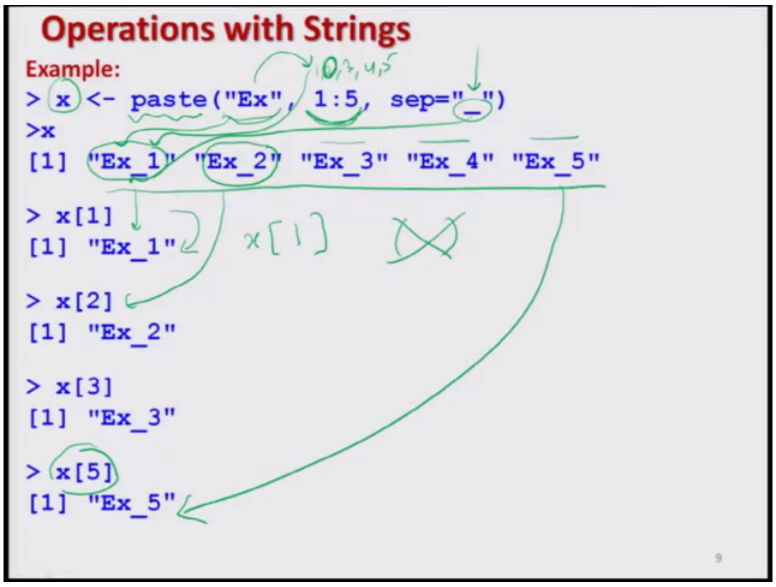


Formatting and Display of Strings

```
> paste(names, "is", "a good", "person.")
[1] "Prof. Singh is a good person."
[2] "Mr.  Venkat is a good person."
[3] "Dr. Jha is a good person."
> paste(names, "is", "a good", "person.", collapse=", and $
[1] "Prof. Singh is a good person., and Mr.  Venkat is a g$
```

Well, here it is giving a dollar sign which is indicating that the outcome is continuing beyond the screen.

(Refer Slide Time: 17:46)



Operations with Strings

Example:

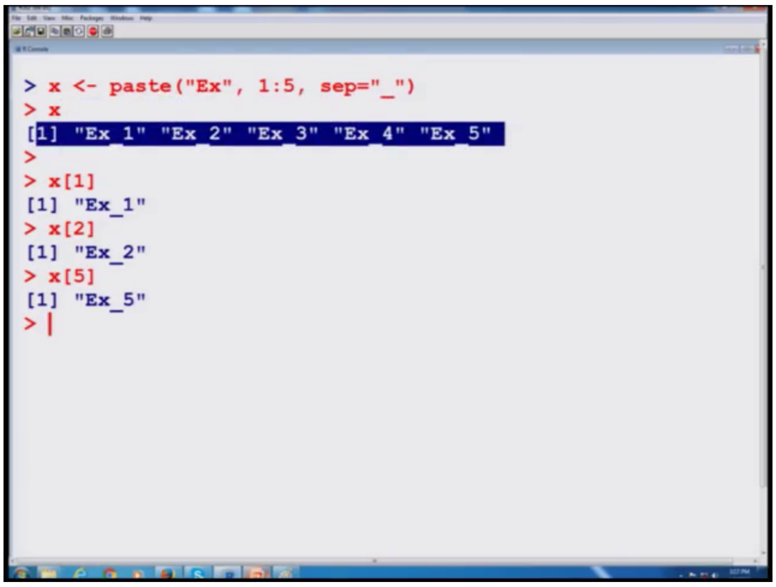
```
> x <- paste("Ex", 1:5, sep="_")
>x
[1] "Ex_1" "Ex_2" "Ex_3" "Ex_4" "Ex_5"

> x[1]
[1] "Ex_1"

> x[2]
[1] "Ex_2"

> x[3]
[1] "Ex_3"

> x[5]
[1] "Ex_5"
```

So, that is here like this. Now let me take some more example which are similar to this example just to make you more confident I am using here a paste function in which I have got here 1; a string say exercise and briefly I am writing at Ex, this is a character and then I am writing here a sequence 1, 2, 5; there are 5 numbers here; 1, 2, 3, 4 and 5 and then I am using here a separator which is underscore.

Now first we try to look at the outcome and then I will try to explain you; what is happening if you see the outcome this is like here like this. So, if you try to see what is really happening that exercise is going to the first element one and it uses this separator underscore? So, this comes over here and this number one comes over here and this separator comes over here and we get this type of outcome exercise also Ex under the score 1; after completing the first iteration the control repeats to the number of times, the number of elements in the vector 1 to 5 in the second shot. This Ex comes to the second element here to and it prints here Ex under score 2 using the separator under score and this process goes here exercise 3 exercise 4 and exercise 5.

So, essentially what I am doing that I have now here one single a string, but the numbers which are to be repeated are some numbers; they are numerics; earlier what we had done that we had a vector of 3 characters and then we had joined there those characters with us which another variable and those variables were characters, but now this is a combination of a string with a numerical vector. Now in case, if you call any particular element that is possible here. For example, I have stored these values inside the vector x. So, if we want to call the first element this can be called here by x and inside a square bracket that you have to keep in mind not the arguments no not the this type of bracket and then you have to write down the number which you want to call suppose I want to call the first element this is coming here, the second element is coming here and similarly you will see here the fifth element is coming from here.

(Refer Slide Time: 20:48)

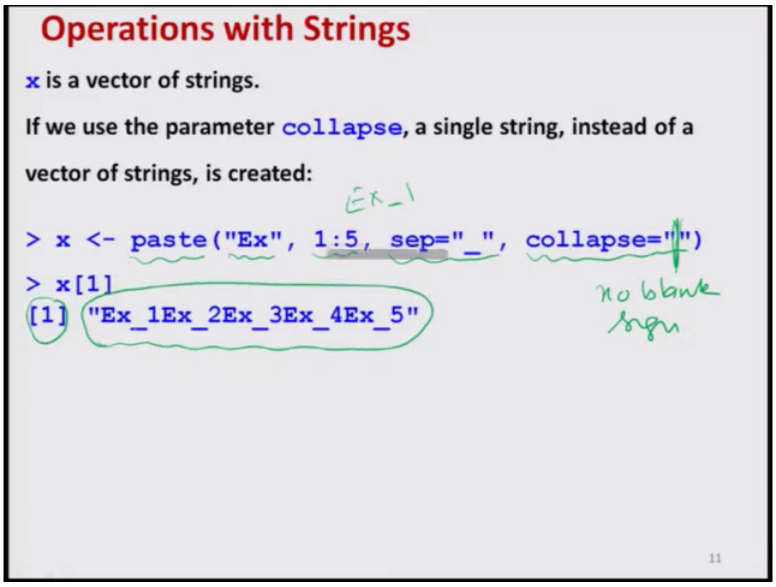


```
> x <- paste("Ex", 1:5, sep="_")
> x
[1] "Ex_1" "Ex_2" "Ex_3" "Ex_4" "Ex_5"
>
> x[1]
[1] "Ex_1"
> x[2]
[1] "Ex_2"
> x[5]
[1] "Ex_5"
>
```

Let us try to do this over the R console and see; what do we obtain. So, I try to write down here x. So, let us try to have this outcome over here x and now if you try to see here what is here x, sorry; x, this is here like this and now if you want to call any particular number, you have to write if you want to call first number inside the square bracket; you have to write x 1, you get first value. Similarly if you want to call the second value need to write here x 2, you will you get here the second value and similarly if you want to call suppose fifth; suppose the fifth value, this is x 5 and this is the fifth element in this outcome.

So, here you have an option that you can also call a particular input from this vector just like any other usual vector. So, now, let us come back to our slides and try to see other things well this is the screen shot of the same outcome.

(Refer Slide Time: 21:54)



Operations with Strings

x is a vector of strings.

If we use the parameter collapse, a single string, instead of a vector of strings, is created:

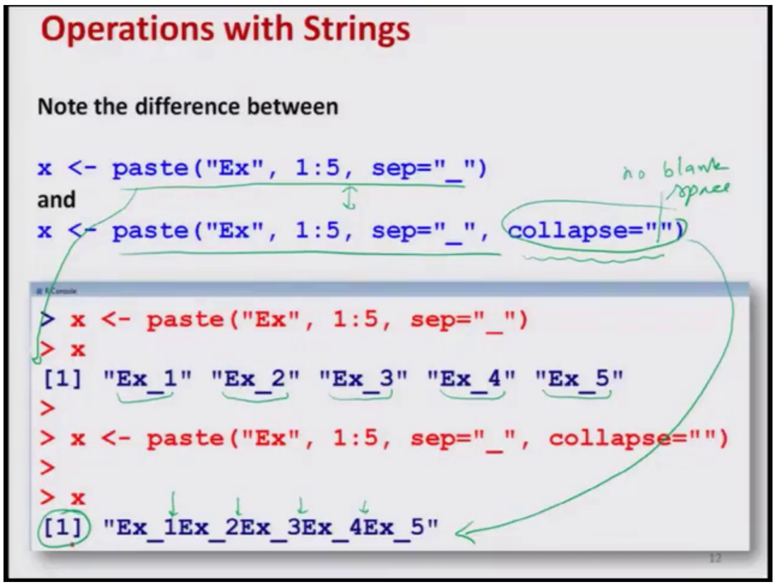
```
> x <- paste("Ex", 1:5, sep="_", collapse="")
> x[1]
[1] "Ex_1Ex_2Ex_3Ex_4Ex_5"
```

And now I am going to use the parameter collapse with this operation and let us to let us try to see what happens now if you try to see; what is really happening in this outcome you can see here there are 5 values which are separated by this blank space, right and when I want that all these 5 values should not be separated, but they must.

But they must collapse; that means, I want to remove the a spacing between these 5 values and if you try to see here I am using the same command here that paste exercise 1 to 5 and separator here this underscore. So, this is going to give me a value like exercise underscore 1, 2, 3, 4, 5, but now I am using here the collapse where there is no symbol here no blank sign there is no blank sign here and as soon as you say enter you get here this type of thing.

So, you can see here this is only here one single value which is continuously giving you all the outcomes. So, let us try to see the use of this collapse here now you can see here this here x. So, you can see the difference between this earlier output without collapse and this output with collapse let us try to say play with this thing and try to see if I try to put here a star; say double star as a collapse then what happens you can see here that the blank space has been changed by 2 stars, but there is no separation mean the separation is being now controlled by this double star. So, now, it depends on you whatever is your need and based on that you can have an outcome in order to make it more clear and to give you a better understanding that we try to have both these options together on this screenshot.

(Refer Slide Time: 24:46)



Operations with Strings

Note the difference between

```
x <- paste("Ex", 1:5, sep="_")
```
and
```
x <- paste("Ex", 1:5, sep="_", collapse="")
```

no blank space

```
> x <- paste("Ex", 1:5, sep="_")
> x
[1] "Ex_1" "Ex_2" "Ex_3" "Ex_4" "Ex_5"
>
> x <- paste("Ex", 1:5, sep="_", collapse="")
>
> x
[1] "Ex_1Ex_2Ex_3Ex_4Ex_5"
```

So, if you try to look at this slide. Here I am using this paste function and here I am using paste function with collapse and this and this they are exactly the same. Now if you try to look at this outcome here you can see the outcome here is same like this and this is coming from the outcome of this one, but when I am using here say another option collapse, I am giving here no blank space and the outcome of this is given over here.

So, you can see here that now there is no blank space here, here and here and here and all these 5 values they are coming here as a single a string like this one say only one. So, that is the basic difference between the use of this another option collapse. So, now, we stop here and once again, I would request you to understand the use of paste function with different type of separators and say all options and try to create an output what you want to have and do some practice and in the next lecture we will continue with the some more operations on the strings and you will try to learn some more options and functions; till then goodbye.