

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

**Lecture – 25**  
**Print and Format with Concatenate**

Welcome to the next lecture on the course introduction to R software. You may recall that in the earlier lecture, we were discussing about different types of option to display and format the outcome or the output. And we had discussed about the print function and at the end, we had discussed that there are certain limitations in the print function; that if you want to print more than one statement in a line that is not possible, but it will print one statement at a time.

And for that we had suggested that there are some other options which can be used in those situations. For example, we had taken the example of printing of the train ticket in the earlier lecture. So, if you really want to print anything at particular places then print command may not be very helpful, but in that case we have to use another command what is called `cat`; cat function, which is actually concatenates the strings. What is the meaning of concatenate? That means, it is trying to print the sequence or the strings in the same order in which you are asking it.

So, the meaning of concatenate is very simple; that it is printing with the same order in which you are defining it; that was not happening with the print command. So, in this lecture we are going to talk about the printing and formatting using the concatenate command. So, we have a function what we call here as say cat function `cat`.

(Refer Slide Time: 02:01)

### Formatting and Display of Strings

**cat function**

- The function `cat()` concatenate (link in the same sequence) and print the arguments in strings, concatenates them and prints the entire string in the command window.
  
- `cat` is useful for producing output in user-defined functions.
  
- It converts its arguments to character vectors, link them together in the same sequence to a single character vector, appends the given `sep = string(s)` to each element and then outputs them. 2

And this function concatenates that is; it links the outcome in the same sequence in which we are trying to give it and print the argument; whatever it is the strings concatenates them; that means, is trying to maintain the same sequence and then prints the entire string in the command window.

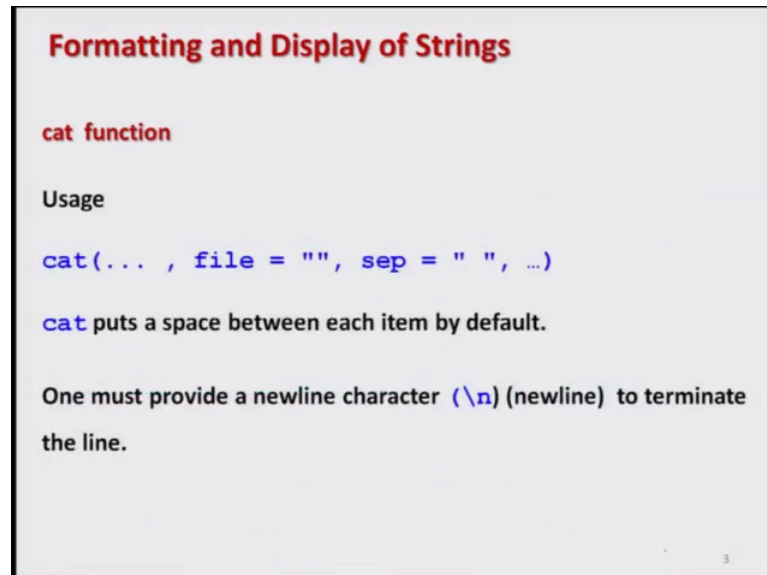
So, you can see that concatenates is a combination of print and formatting, but there is an additional thing that it can format in the required output; that is the difference between print and cat commands. So, this cat function is more useful when you want to produce the output in a user defined functions; the way we want the output to be, then use this cat function and basically this cat function converts its argument to character vectors; then link them together.

And then it linked them together in the same sequence and in a single character vector. For example, in case if you want to print three arguments in print command; they are coming as command 1, command 2, command 3; column wise, but here in this cat command they can come as call say statement 1, statement 2, segment 3; row wise, in the same sequence. And there is also an option that how these different strings can be separated by a comma, by a hyphen or by something else that option is also there

So, for that we have a option here `sep`; `s e p` that is separator that the short form of separator is `s e p` and then we try to give here is string by which we want to separate it.

And this separator is used to separate each of the element or each of the string that we want to concatenate.

(Refer Slide Time: 04:19)



**Formatting and Display of Strings**

**cat function**

Usage

```
cat(... , file = "", sep = " ", ...)
```

**cat** puts a space between each item by default.

One must provide a newline character (`\n`) (newline) to terminate the line.

So, the syntax for the cat function is something like say here cat here whatever we want to print and then I can specify here a file; which we want to print. And we can also give here the separator inside this double quotes; whether it can be hyphen or a dot or a full stop or a multiplication sign or anything else.

And in case if you do not give anything over here; the default here is blank space. Blank spaces is automatically taken as a default separated with the cat function. And after having a one output; if you want to change the line, then for that we have a newline character which is given as say here backslash n and with this new line command the output is terminated at that particular point and the next output is printed on the next line. We will try to see these things with some more examples.

(Refer Slide Time: 05:18)

### Formatting and Display of Strings

The only way to print multiple items is to print them one at a time  
*The zero occurs at 2π radians.*

```
> print("The zero occurs at"); print(2*pi);  
print("radians")  
[1] "The zero occurs at"  
[1] 6.283185  
[1] "radians"
```

The `cat` function is an alternative to print that lets you combine multiple items into a continuous output:

```
> cat("The zero occurs at", 2*pi, "radians.",  
"\n")  
The zero occurs at 6.283185 radians.
```

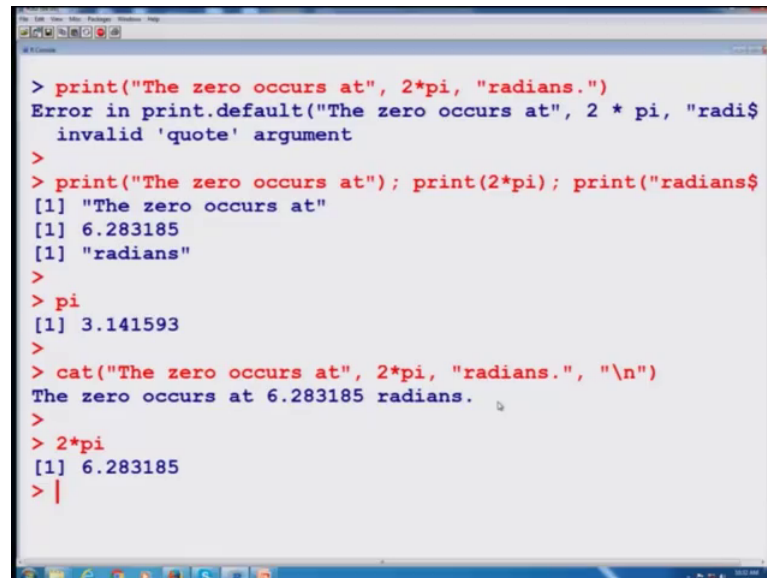
4

So, let us try to take first the same example where we concluded our last lecture. If you remember, we wanted to print the 0 occurs at 2 pi; whatever is the value of pi that should come radians. And we were unable to do it and when we try to have the print command writing all the strings in the same argument, we had an problem and we had got an error message. Then we tried to print the statement by giving it three different print functions separated by semicolon, but this outcome was like this; not in the way we wanted to have it.

And we had discussed that this cat command can help us in getting the output in the required format. So, how do we do it? Now, you can see here I am writing here c a t cat inside the arguments, inside this bracket. First of all I am writing this statement; the first string and this is the character so, I am trying to write it inside the double quotes; the 0 occurs at.

Now, this is the value which has to be computed by the R function. I will simply say here two times p i; pi is the default function for getting the value of pi in R and then inside the double quotes I am trying to write down here radians; that means, this is a character this is a string and once I try to do it here and then I want to have a next line; that means, I want to stop and after this radiance, I want that the outcome should break here and the control should come to the next line. So, for that I use here a command backslash here n. So, let us try to execute it and try to see what happens with this thing over here.

(Refer Slide Time: 07:37)



```
> print("The zero occurs at", 2*pi, "radians.")
Error in print.default("The zero occurs at", 2 * pi, "radi$
  invalid 'quote' argument
>
> print("The zero occurs at"); print(2*pi); print("radians$
[1] "The zero occurs at"
[1] 6.283185
[1] "radians"
>
> pi
[1] 3.141593
>
> cat("The zero occurs at", 2*pi, "radians.", "\\n")
The zero occurs at 6.283185 radians.
>
> 2*pi
[1] 6.283185
> |
```

So, first of all you can see here what we had discussed in the earlier lecture that when we are trying to print here something like this; this is giving us some error. Then we tried the same thing by dividing the entire sentence into three different print function, but it was giving one sentence or one string in a row; first row second row and then third row and the value of pi was something like this.

But now I am trying to use here another command here cat and you can see here this input or the contents inside this argument; they are the same what we had used in the print function but when we use it in the print function that was giving us an error, but now if I try to simply replace the print function, which I have highlighted here. Now, by this cat function; let us see what happens, you get the same output and how this value is coming out; 6.283185; this is the value of here two times pi, which is something like here 2 say star pi; you can see here this is the value.

So, you can see here that we have here some characters which are appearing here as just like a characters; this value 2 into pi that has to be computed by the R; this is computed here. And this computation follows from here please try to follow where I am highlighting it and then in the third step; we wanted to have the radians as a character and this is appearing here again as a character. Now, this backslash n that is now changing the line and the outcome or the next outcome will start from the next line right.

So, now let us come back to our slide and try to say take some more example. Well this is the screenshot of the outcome that we have just done. So, now we take some more examples and try to understand about this cat functions.

(Refer Slide Time: 10:08)

```
Formatting and Display of Strings
> d <- date()
> cat("Today is:", d, "\n" )
Today is: Wed Jan 04 23:49:52 2017

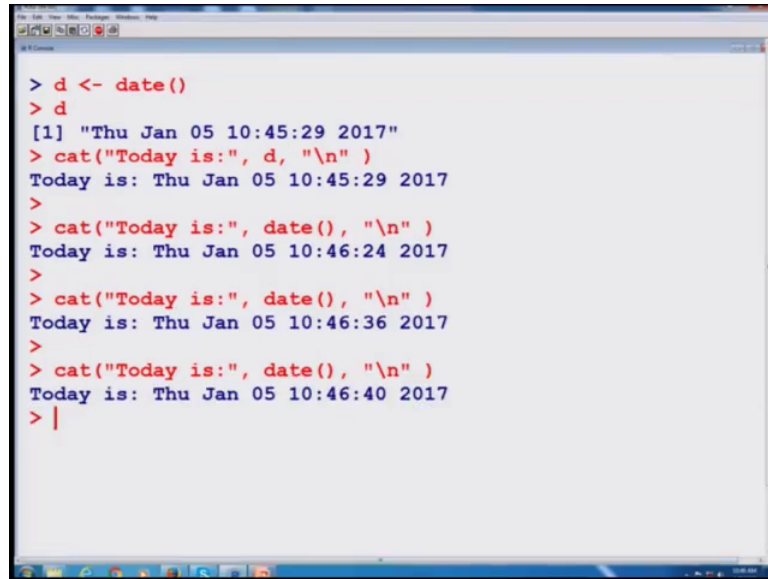
R Console
> d <- date()
> d
[1] "Wed Jan 04 23:49:52 2017"
> cat("Today is:", d, "\n" )
Today is: Wed Jan 04 23:49:52 2017
```

Suppose, I have given an assignment to my students and I also want to know when they have done their job. One option is this whenever they are executing a program; I can ask them that please write the date over there. So, one option is this they can write a statement like today is; and after that whatever is the time and date when they are executing the program for that assignment will also be printed. So, how to get it done? So, for that if you remember we had discussed a command date and time. So, out of those two I can use here the function date and whatever is the current date whenever the a execution has been done; this will be saved in a variable say here d.

And then I can write down the cat function; say cat and as a character; I can write down today is and after that whatever is the value of the variable d; that is determined by the system that can be printed automatically. And after that the next command will be executed and the outcome will come on the next line using the command backslash n.

So, you can see here when I prepared this slide; this was the time and date. So, and this the same screenshot is given over here; now let us try to execute it and try to see what do we get over here; so this is my here d.

(Refer Slide Time: 12:01)



```
> d <- date()
> d
[1] "Thu Jan 05 10:45:29 2017"
> cat("Today is:", d, "\n" )
Today is: Thu Jan 05 10:45:29 2017
>
> cat("Today is:", date(), "\n" )
Today is: Thu Jan 05 10:46:24 2017
>
> cat("Today is:", date(), "\n" )
Today is: Thu Jan 05 10:46:36 2017
>
> cat("Today is:", date(), "\n" )
Today is: Thu Jan 05 10:46:40 2017
> |
```

So, you can see here d is my here; the date function and when I try to get here a printing of this outcome in this particular way suppose I can here; this thing. So, you can see here that this is coming out to be like this and you can match here; this January fifth is the date and time is 10.45; 10 taken over 45 minutes; 29 seconds. And this is the same time which is given on this here computer system; that you can see.

So, anyway you would come to know that when I had recorded this lecture and if you want to make it more general; instead of writing here d I can simply write here date whatever it is. So, you can see here with this date function; as soon as somebody is trying to execute it, this time is changed. For example, you can see here the time here is 10.45; 29; here it is 10.46; 24; then the seconds are changing 36 and then here 40.

So, you can see here I have got my output in the required format.

(Refer Slide Time: 13:26)

```
Formatting and Display of Strings

> x <- 7
> cat("The square of", x, "is", x^2, "\\n")
The square of 7 is 49!

7^2 = 49

R Console
> x <- 7
> cat("The square of", x, "is", x^2, "\\n")
The square of 7 is 49 !
```

And now we try to take another example; suppose I want to print the square of a number. So, I can write down here the square of some number is something and I want to fill this value. So, I can do like this suppose I give here one particular value  $x$  equal to 7 and I want to print using the `cat` command; I write thus the square of just as such as a character, then I write down here the value  $x$ ; which will come from here and then I write is and then I write here  $x$  square.

So, this  $x$  squared will be computed using this value of here  $x$  and the next output will come on the next line, but before that there is going to be an exclamation sign. So, if I try to write down this statement and when I try to execute it; we get here the answer the square of 7 is 49. The seven is coming from here and 49 is coming from 7 says square which is 49 and this is here the outcome. Similarly, if you try to execute it over the R console you can see here.



(Refer Slide Time: 15:03)

```
> x<- 7
> x
[1] 7
> cat("The square of", x, "is", x^2, "!\n")
The square of 7 is 49 !
>
> x<- 1234
>
> cat("The square of", x, "is", x^2, "!\n")
The square of 1234 is 1522756 !
> |
```

That I try to take here x equal to 7 and then 2 x is equal to 7; we have 7 and then I try to do this thing. The advantage is that suppose if I try to change my value of here from x to say 1, 2, 3, 4 and no issues then that is changed. So, you can see here that as soon as I am trying to change the value; the output is automatically changed and that is the advantage of using this cat function; that you get an outcome in the way you want. Now, let me take another example.

(Refer Slide Time: 15:48)

**Formatting and Display of Strings**

```
> cat("The square root of", x, "is approximately", format(sqrt(x), digits=3), "\n")
The square root of 7 is approximately 2.65
```

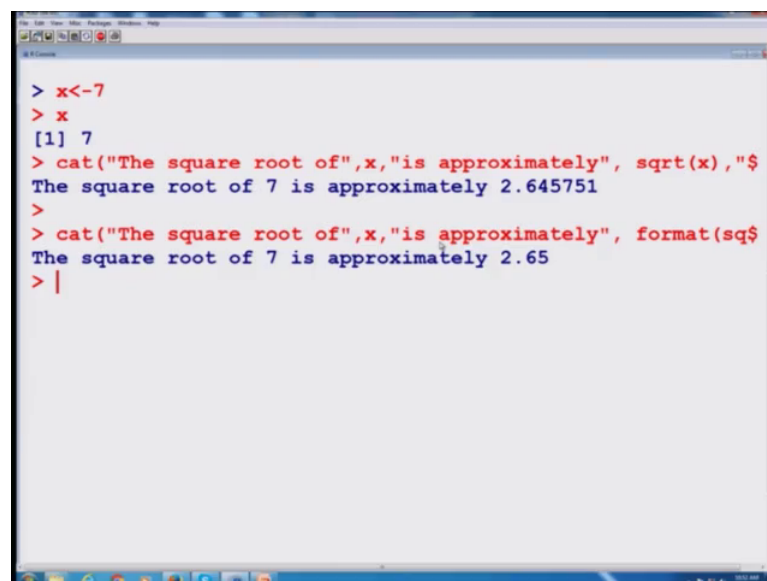
```
> cat("The square root of", x, "is approximately", format(sqrt(x), digits=3), "\n")
The square root of 7 is approximately 2.65
```

8

Same example but my requirement is now changed and I want this outcome in a particular format. Suppose I want that instead of square; the square root is computed and the number of digits are only; say for example, 3 or some other number. So, I try to use this cat command and if I write say; I say here the square root of the character and then here x; then here it is say here is approximately and then I am trying to find out here square root of x, but square root of x that is in a particular format; that is should be equal to 3.

So, for example, I am using here the earlier value x equal to 7. So, the outcome comes out to be the square root of 7 is approximately 2.65. So, the square of x; this is coming from here the square of this thing. The 7 is coming from here x and this then next sentences is approximately; this is coming from here is approximately and whatever are this value inside this; whatever other values of is square root of 7; in the format digit is equal to 3. So, let us try to do it over the R console and see what do we get over here.

(Refer Slide Time: 17:25)

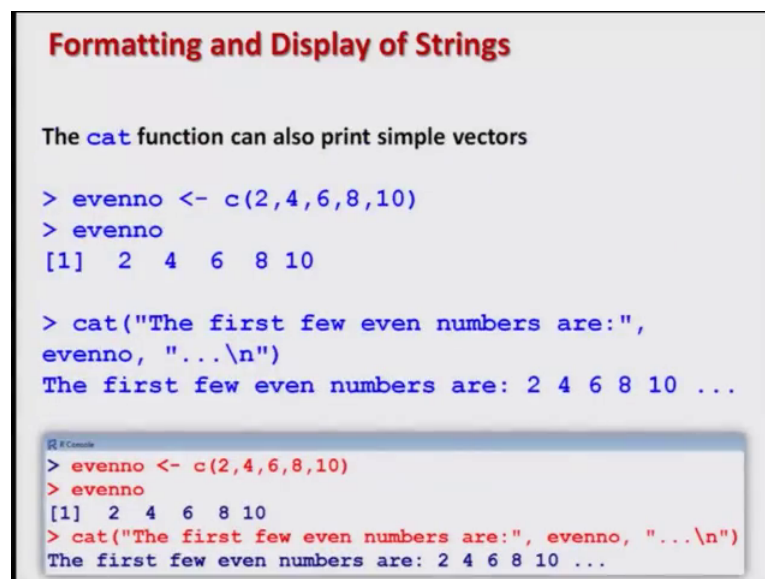


```
> x<-7
> x
[1] 7
> cat("The square root of",x,"is approximately", sqrt(x),"$
The square root of 7 is approximately 2.645751
>
> cat("The square root of",x,"is approximately", format(sq$
The square root of 7 is approximately 2.65
> |
```

So, you let me first define x equal to here 7 and I try to define here; first let me try to define the same thing or try to find out the square root of x without any format. So, I try to remove the format command and you can see here; this is coming out to be like this. I have not used here the format command, but simply the square root of x is whatever is the square root of x or the square root of 7.

Now I try to use the format command; you can see here now the difference, means I have said that I need only 3 digits. So, there are here 7 digits, but here there are only altogether 3 digits. So, cat can also work with format command in the same way, the print was working with format command. So, that is the moral of the story which I wanted to give you from this example; that you can use the format function with the cat function also. Now we come to the; see another say example and this is also a sort of a similar example.

(Refer Slide Time: 18:46)



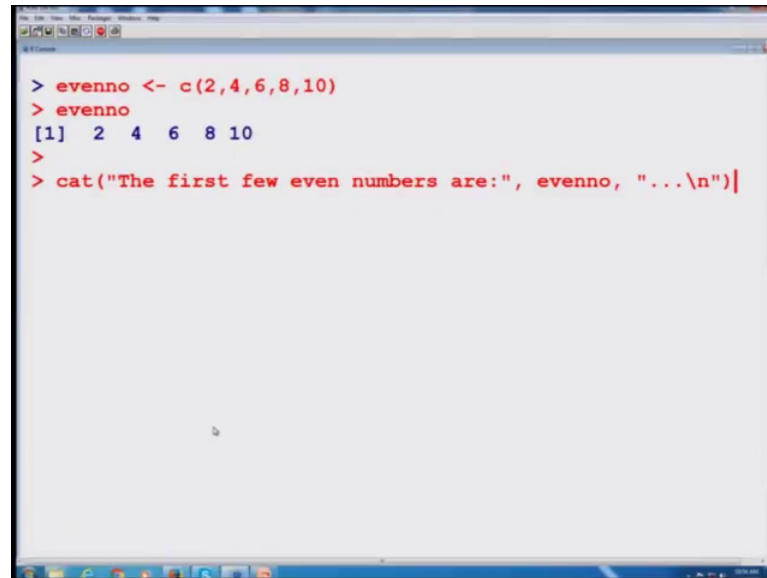
```
Formatting and Display of Strings  
  
The cat function can also print simple vectors  
  
> evenno <- c(2,4,6,8,10)  
> evenno  
[1] 2 4 6 8 10  
  
> cat("The first few even numbers are:",  
evenno, "...\\n")  
The first few even numbers are: 2 4 6 8 10 ...  
  
R Console  
> evenno <- c(2,4,6,8,10)  
> evenno  
[1] 2 4 6 8 10  
> cat("The first few even numbers are:", evenno, "...\\n")  
The first few even numbers are: 2 4 6 8 10 ...
```

But here I am trying to work only with a simple vector and I want to show you that cat functions can also work with simple vectors and it can print the simple vectors also. For example, I have say here some values which are sequence of even numbers. These values are combined in a vector using c command values are 2, 4, 6, 8 and 10 and I try to store them in a variable name say even numbers; even n o; short form.

This values are like this and now I use here the cat command and I say that the first few even numbers are and whatever are the numbers which are given inside this variable; even number, they will be printed over here from here. And then after this there will be a sort of dot dot dot; that means, the numbers are continuing further and then the outcome will be coming on the next line, using this backslash n.

And you can see here what is the here the outcome; this is the screenshot, but I will try to show you here on the screen also that what do we get here. So, if I try to define my here this vector comes like this; so, you can see here this is the outcome.

(Refer Slide Time: 20:31)

A screenshot of a terminal window with a white background and a blue title bar. The terminal shows the following R code and output:

```
> evenno <- c(2,4,6,8,10)
> evenno
[1] 2 4 6 8 10
>
> cat("The first few even numbers are:", evenno, "...\n")
```

This is my vector and I need the outcome in this particular format. You can see here the outcome is like this; so, the moral of the story in this example is simply to understand that cat command can work with simple function, with characters say strings and format command can also be embedded inside the cat function and we can get the outcome in a required format.

So, I would stop here and I would request you to just do some practice with this cat command and whenever you are trying to do something with cat command; try to do with the same job with print function also. And see; what is the difference between the two, which is more convenient? Which is more User friendly? And we will see you in the next lecture; till then good bye.