Foundations of R Software

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

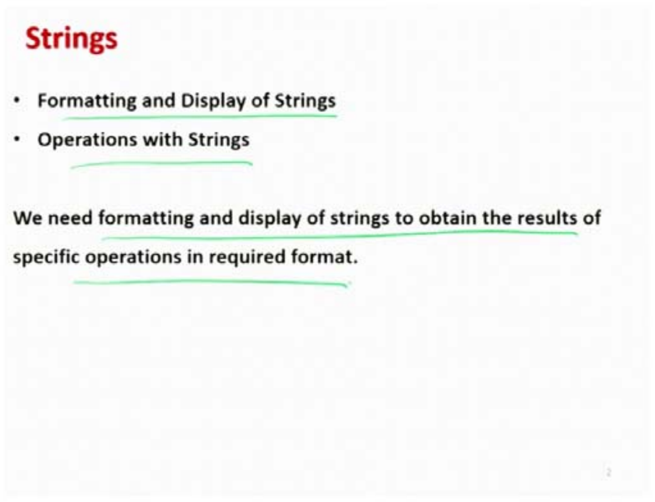
Lecture - 35
Strings - Display and Formatting
Print and Format Function

Hello friends, welcome to the course Foundations of R Software. And, now from this lecture we are going to begin with a new topic, with a new concept that how to implement it in R. You see whenever you are trying to write a program, the program will have some output and you want to write the output in a particular way. For example, you know that many software are popular because they can generate the reports in a very good format.

So, in R also you can format your reports the way you want, means you want this statement, this number on this line, here there and so on. So, how to get those things done in the R software is the broad topic that we are going to consider in this lecture and in the forthcoming lectures. For example, you have learnt about one function which is print, many times we have used it. What is print? Print is trying to show you the outcome on your screen, right and you want to write some output in a particular way. For example, suppose you want to write ok the sum of 2 and 2 is now; whatever is the sum that will come here, which is an output of the program.

So, now, how to write that the sum of 2 and 2 is 4, where the output is coming from the program and if you wish you can also insert the 2 and 2 also from the program. So, how to get all this thing that is exactly what we are going to learn in this lecture and in the forthcoming lectures and for that we have a couple of commands. So, let us begin our lecture and try to understand all these things.

(Refer Slide Time: 02:27)



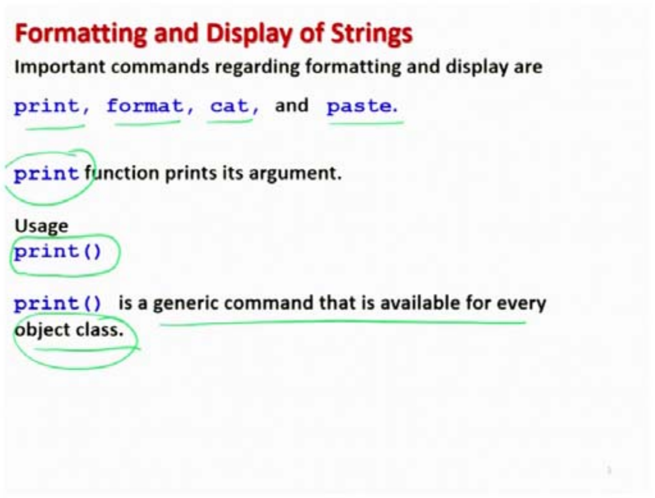
Strings

- Formatting and Display of Strings
- Operations with Strings

We need formatting and display of strings to obtain the results of specific operations in required format.

So, you know that whenever you are trying to work with strings then you would like to present them in a formatted way and you want to control the display of the strings. Sometimes you want to operate on those strings also so, that you get a required outcome. So, we need the formatting and display of strings to obtain the results of a specific operation in a required format.

(Refer Slide Time: 02:54)



Formatting and Display of Strings

Important commands regarding formatting and display are

`print`, `format`, `cat`, and `paste`.

`print` function prints its argument.

Usage

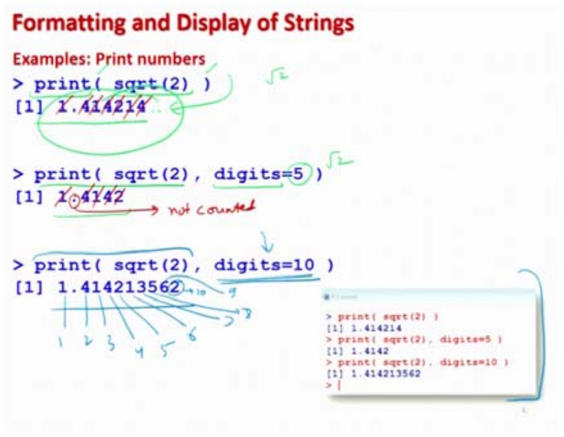
`print()`

`print()` is a generic command that is available for every object class.

So, in R software we have couple of commands which helps us in doing so, for example: print, format, cat, paste etc., right. So, we are going to learn about all these options one by one and I will try to take couple of examples so, that I can explain you and that how the outcomes of this function will look like. So, first let us try to consider here the function print. Well, that we already have used many times, right. But, this is the place where I am going to formally introduce it, because earlier I always used to say ok, we are going to talk about in the future lectures, further lectures, right, ok.

So, the way you would like to use the print function is p r i n t and then inside the parenthesis, you try to write down whatever you want to print. So, this is a generic command and that is available for every object class. Now, you will understand that what is the meaning of this object class. This is the class or this is the class of objects which we use to find using the function class ok.

(Refer Slide Time: 04:00)



Formatting and Display of Strings

Examples: Print numbers

```
> print( sqrt(2) )
[1] 1.414214

> print( sqrt(2), digits=5 )
[1] 1.4142

> print( sqrt(2), digits=10 )
[1] 1.414213562
```

```
> print( sqrt(2) )
[1] 1.414214
> print( sqrt(2), digits=5 )
[1] 1.4142
> print( sqrt(2), digits=10 )
[1] 1.414213562
> |
```

So, now, let me try to take here some examples and through which I try to show you how the print command works. So, suppose I want to print the value of square root of 2. So, I try to use here the command print p r i n t and then within the parenthesis I try to write down here sqrt and inside parenthesis 2.

So, you can see here, this outcome will be shown on the computer screen like this 1.414214 and you know that this number can go further also. But, suppose you also wish to control the number of digits in this output. So, for that we have one more option here, that we try to write down here print within the parenthesis square root of 2 and comma then we try to write down here option d i g i t s digits in this lower case alphabets.

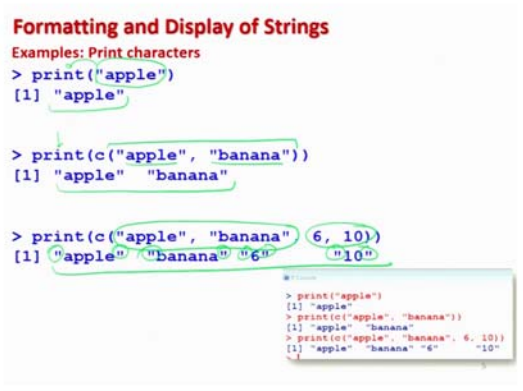
So, you know what is the meaning of this word is that total number of or numbers what you want in the square root of 2. So, suppose you say I want 5 degrees in the value of square root of 2. So, now, you can see here and try to compare what is the difference between this outcome and the earlier outcome, here you get here like this 1.4142.

Now, if you try to see how many numbers are there 1 2 3 4 and 5, right and, you can see here this point decimal, this is not counted. Well, you cannot argue why? Because that is the way the R software works, right. And, in the earlier command if you try to see you have more than five numbers, 1 2 3 4 5 then 6 and 7. Similarly, if you want that these number of digits should be 10; so, I can simply use here the same command print square root of t 2.

And, then I try to add here the option d i g i t s, all in lower case alphabets equal to 10. So, you can see now here this is here the outcome. So, how many values are there?

This is the 1st value, this is the 2nd value, this is the 3rd value, this is here the 4th value, now this here 5th is the this 2 is the 5th value, 1 is the 6th value, 3 is the 7th value, then 5 is the 8th value, 6 is the 9th value and 2 is the 10th value, right. So, you can see here now there are 10 digits in this outcome. So, this is how you can control the outcome with this command. And, this is here the screenshot of the same operation which I shown you on the which I will try to show you on the R console.

(Refer Slide Time: 06:43)



Formatting and Display of Strings

Examples: Print characters

```
> print("apple")
[1] "apple"

> print(c("apple", "banana"))
[1] "apple"  "banana"

> print(c("apple", "banana", 6, 10))
[1] "apple"  "banana" "6"       "10"
```

```
> print("apple")
[1] "apple"
> print(c("apple", "banana"))
[1] "apple"  "banana"
> print(c("apple", "banana", 6, 10))
[1] "apple"  "banana" "6"       "10"
```

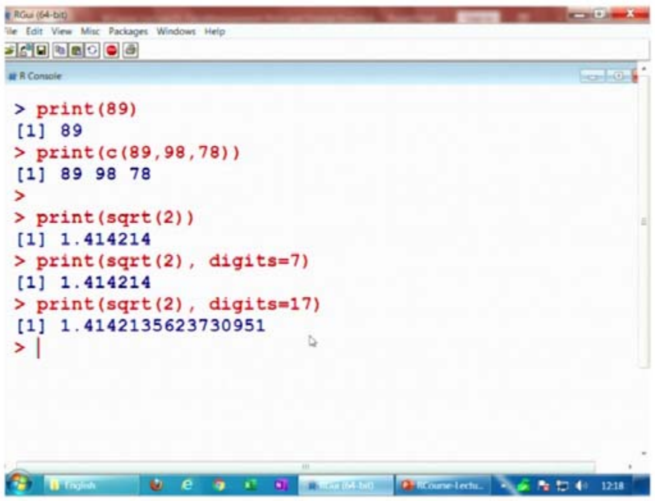
So, here I have talked about the print command with number, with some numeric values. Now, I try to use the same command print with some characters, right. So, suppose I try to say here take a word apple that is our popular example in this course; apple, banana and cake, right. So, I try to take here the apple within the double quotes and I try to use here the command here print. So, this will here print like this apple, right.

Now, you cannot talk that how many alphabets or how many letters you want here because yeah these are the strings. So, similarly if you want to print here a data vector which has suppose two characters, two strings, right. So, apple and banana then if I use here the command print, you can see here you are getting here apple and banana.

And, now even if you try to take a data vector consisting of the characters like as apple and banana and some numbers like as here 6 and 10, in that case you know the mode of this data vector will be character. And so, everything is going to be printed here as a character only, you can see here. Now, this apple is inside double quote, banana is inside double quote, 6 and 10 they are also inside the double quote.

Because, now this is 6 and 10 they are going to be taken as a character and whereas, earlier if you try to see, means if you are trying to take here 1.414. This square root of 2, this is not inside the double quotes, right. So, let me try to first show you these operations on the R console and then I will try to introduce you with more options.

(Refer Slide Time: 08:26)

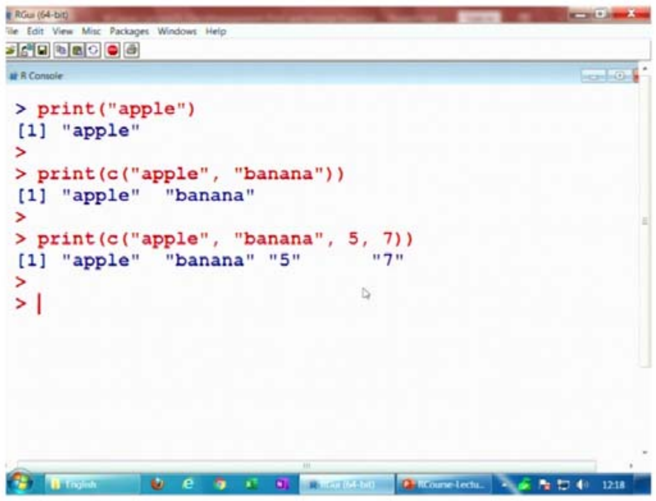


```
> print(89)
[1] 89
> print(c(89,98,78))
[1] 89 98 78
>
> print(sqrt(2))
[1] 1.414214
> print(sqrt(2), digits=7)
[1] 1.414214
> print(sqrt(2), digits=17)
[1] 1.4142135623730951
> 
```

So, if you try to see here, I can say here print say 89 and this is here like this. Similarly, if you try to take here a data vector say 89, say 98, 78 etcetera; it will be printed here as just like this, 89 98 78, right. Similarly, if you want to print here square root of 2 here so, this will be like this and if you want to add here the command d i g i t s, digits is equal to suppose 7.

So, now, you can see here this is here 1.414214 and if you try to make this digits equal to 17, you can see here now you have here 17 numbers in this or 17 digits in this outcome, right. So, that is not difficult that you can see, ok.

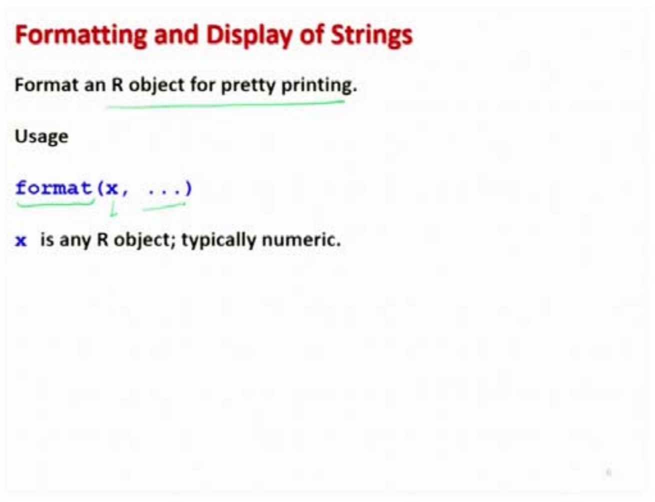
(Refer Slide Time: 09:22)



```
> print("apple")
[1] "apple"
>
> print(c("apple", "banana"))
[1] "apple"  "banana"
>
> print(c("apple", "banana", 5, 7))
[1] "apple"  "banana" "5"      "7"
>
> |
```

So, now let me try to take here some examples related to the characters. So, if you try to see here, I try to print here apple. So, here this is here apple and then if you try to print here data vector of this characters so, I say here banana. So, this will be your here print and here apple and banana. And, now if you try to add here some values which are say number say 5 and here 7, then you can see here they are going to be printed as here 5 and 7, right.

(Refer Slide Time: 10:00)



Formatting and Display of Strings

Format an R object for pretty printing.

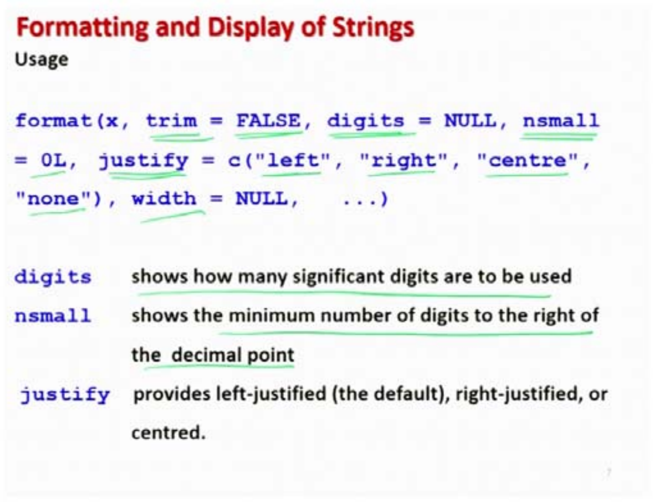
Usage

```
format(x, ...)
```

`x` is any R object; typically numeric.

And, now I try to show you here some more examples here with the using the next command here format. So, as you understand that what is the meaning of a format, you want the outcome in a particular way, in a formatted way. So, this command here format is used for say some nice printing the way. So, that you can control the way the outcome is coming and for that the command here is format, f o r m a t and inside the parenthesis you have to write the object and typically it is a numeric value.

(Refer Slide Time: 10:42)



Formatting and Display of Strings

Usage

```
format(x, trim = FALSE, digits = NULL, nsmall
= 0L, justify = c("left", "right", "centre",
"none"), width = NULL,    ...)
```

`digits`	shows how many significant digits are to be used
`nsmall`	shows the minimum number of digits to the right of the decimal point
`justify`	provides left-justified (the default), right-justified, or centred.

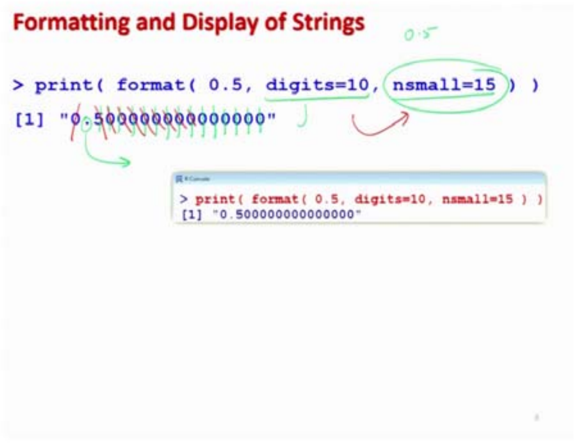
But, there are some other options also which I will try to show you here with this command and then I will try to show you some examples so, that you can understand how you can use this format command with the print command. So, you know whenever you are trying to format something, do you can you recall that when you are trying to type a document for example, one popular software to type that document is it a MS-Word, Microsoft Word.

When you try to type there something, then you have an option that the text which is printed there that is justified on the left hand side, right hand side or say centered etc. and then you can control the width etcetera. So, similar type of outcome, well it is not a word processor, but those types of things can be controlled using the format command and we have many options here. For example, if you want to trim an outcome, you can give here it is in terms of true and FALSE.

Similarly, you can use the option here digits. So, they are going to show how many significant digits are to be used, right. And, simply here if you want to control that how many numbers or how many digits should be on the, right hand side of the decimal point, then you have to use here the command here say nsmall. So, this shows the minimum number of digits to the, right of the decimal point then simply you have here justify here.

So, I mean for if you want to justify on the left hand side,, right hand side, centre or none that will be the default, then you can use this thing and similarly you have here width etc.

(Refer Slide Time: 12:22)



Formatting and Display of Strings

```
> print( format( 0.5, digits=10, nsmall=15 ) )
[1] "0.500000000000000"
```

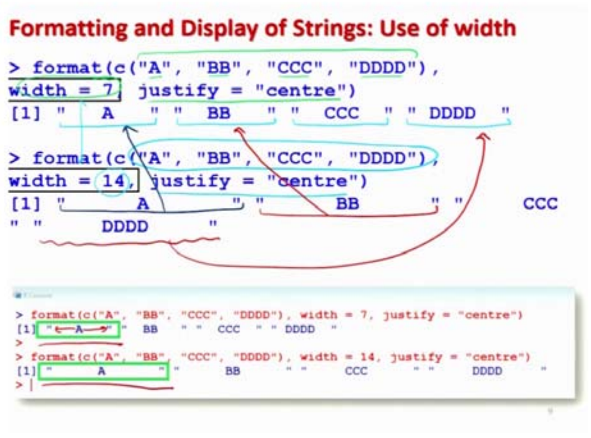
```
> print( format( 0.5, digits=10, nsmall=15 ) )
[1] "0.500000000000000"
```

So, there are many many options which are available. I will request you that you please try to look into the help and try to see the application of all such options. Well, I try to use here means some examples here to show you what really happens. Suppose, I simply want to print is 0.5 and what I want here that the number of digits should be equal to 10 and the number of this nsmall value is here 15.

So, you can see here what happens here, it will give you here this type of outcome so, if you try to count how many digits are there, how many values are there; 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15. So, you can see here you are trying to say the number of digits on the, right hand side of this decimal point should be 15. So, they are here 15 and total number of digits you are trying to say here 10.

But, then so, total number of digits are going to be counted here like this, this is here 1 2 3 4 5 6 7 8 9 10, but you also have given here nsmall. So, these values are increase further. So, that is you what you have to keep in mind. I will try to show you that when you try to use only one of the command on the R software, then what happens ok.

(Refer Slide Time: 13:45)



Formatting and Display of Strings: Use of width

```
> format(c("A", "BB", "CCC", "DDDD"),
width = 7, justify = "centre")
[1] "   A   " "  BB   " "  CCC  " " DDDD  "

> format(c("A", "BB", "CCC", "DDDD"),
width = 14, justify = "centre")
[1] "      A       " "      BB      " "      CCC
" "     DDDD     "
```

```
> format(c("A", "BB", "CCC", "DDDD"), width = 7, justify = "centre")
[1] "   A   " "  BB   " "  CCC  " " DDDD  "
>
> format(c("A", "BB", "CCC", "DDDD"), width = 14, justify = "centre")
[1] "      A       " "      BB      " "      CCC     " "     DDDD     "
> |
```

Now, after this I try to show you here the utility of the option width, what does this mean actually, right. So, I try to suppose I have here the data vector of some characters. So, we I have here characters like as here A, then double B, triple C and 4 times D. And, yeah I am trying to choose here the option here, justify is equal to centre. And, yeah how this justify work that I will try to show you in the next example.

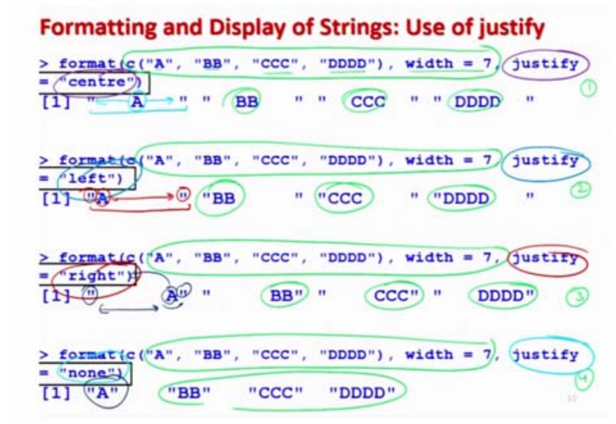
But, here you can say this that all the values are centered. And, now I try to use here the width equal to 7. Now, I use here the same data vector; A, double B, triple C and 4 times D and justify is also the same as earlier centre. But, I try to use here now with is equal to 14. So, if you try to see here in the first command and in the second command means everything is same, only the outcome of width is going to affect the outcome.

So, if you try to see here that how many values are inside this double quotes, try to see here. This is here the width and if you try to see this width in the next outcome, where you are trying to take the width is equal to 14. This is going to be here like this. So, if you try to compare this with this, you can see the difference. Similarly, if you try to take here the second width, second with here is like this which is compared to this width.

And, similarly if you try to take here this the fourth width D, that is going to be compared with this. And, if you try to yeah see into the screenshot that gives you a better outcome that in the first the width is here is like this and in the second option when you try to say with is equal to 14, this is like this.

So, this is actually trying to increase the width of the this outcome of every element in the data vector. So, that is the use of here width comma. So, you can see that with this example it is quite easy for me to explain you that how width is going to control the outcome.

(Refer Slide Time: 15:55)



Formatting and Display of Strings: Use of justify

```
> format(c("A", "BB", "CCC", "DDDD"), width = 7, justify
= "centre")
[1] "   A   " "  BB   " "  CCC  " " DDDD  "

> format(c("A", "BB", "CCC", "DDDD"), width = 7, justify
= "left")
[1] "A      " "BB     " "CCC    " "DDDD   "

> format(c("A", "BB", "CCC", "DDDD"), width = 7, justify
= "right")
[1] "      A" "     BB" "    CCC" "   DDDD"

> format(c("A", "BB", "CCC", "DDDD"), width = 7, justify
= "none")
[1] "A"    "BB"    "CCC"    "DDDD"
```

Similarly, I try to consider here 4 cases; case number 1, 2, 3 and here 4. In all the 4 cases, I have taken the same data vector and same width. So, the data vector is same as the earlier A, double B, triple C, 4 times D and the width is 7. So, this you can see, this is the same in all the 4 cases, right. And, now you can see here what I try to do is that I simply try to change here the option here justify.

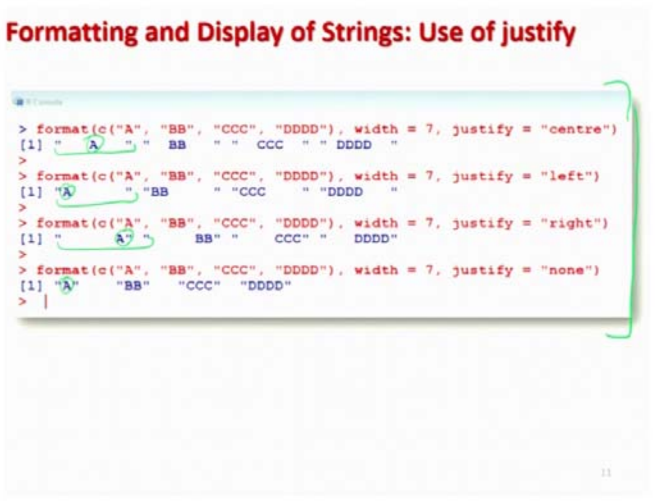
In the 1st case, I try to take justify is equal to here centre, in the 2nd case I try to take here justify is equal to the left. In the 3rd option, I try to take justify here as say, right and in the 4th option I try to take here justify is equal to here none and try to see the outcome, how the outcomes are going to vary. So, whatever is the effect in this outcome that is going to because of the justify option.

So, you can see here try to look into the 1st case and yeah I will try to look into the 1st outcome only. So, if you try to see here the location of here A which is in the middle of this double quotes, right. Then, if you try to see the location of this A in the 2nd case so, within this double quotes, the location of A is on the left hand side and this space is blank. Now, similarly if you try to see here that in the case number 3, within this double quotes the location of A is here on the, right hand side and this space is blank, right; because this is due to the option, right.

And, then in the 4th case because you have given the justify to be none so, it is simply trying to take the default here A. So, if you try to see here, when you are trying to use the options here; like as here centre, then all the values are in the centre. Centre of what? Centre of double quotes.

Similarly, if you try to use the take the 2nd case where you try to choose the option left, then all the values are on the left hand side of the double quotes. Similarly, if you try to take the case number 3 where you are trying to choose the option, right, then all the values are here on the, right hand side of this double quotes. And, in the 4th case you have not chosen any so, these are printed as a default, right.

(Refer Slide Time: 18:39)

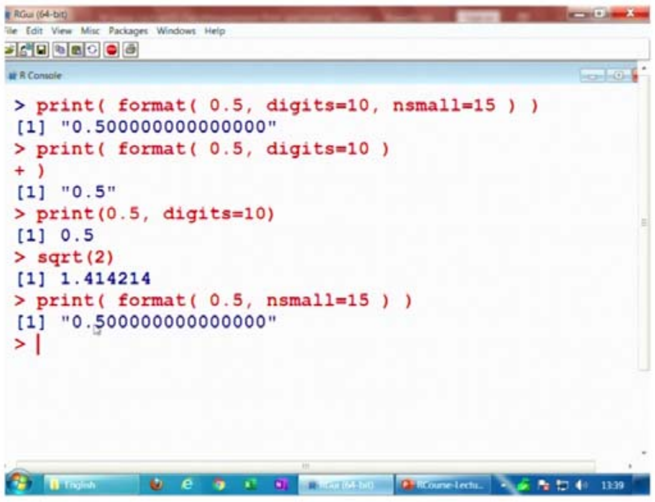


Formatting and Display of Strings: Use of justify

```
> format(c("A", "BB", "CCC", "DDDD"), width = 7, justify = "centre")
[1] "   A   " "  BB   " "  CCC  " " DDDD  "
>
> format(c("A", "BB", "CCC", "DDDD"), width = 7, justify = "left")
[1] "A      " "BB     " "CCC    " "DDDD   "
>
> format(c("A", "BB", "CCC", "DDDD"), width = 7, justify = "right")
[1] "      A" "     BB" "    CCC" "   DDDD"
>
> format(c("A", "BB", "CCC", "DDDD"), width = 7, justify = "none")
[1] "A"    "BB"   "CCC"  "DDDD"
> |
```

So, this is how you can see the outcome changes and you can see this change in this screenshot very clearly. Here you can see here this is here A, this is here A here like this and then this is here A and this is here A like this. And, similarly you can observe that what is happening to the other options, right. So, why not to do these operations first on the R console and try to see what do they do, right. So, let me try to take here this first this print command of this 0.5.

(Refer Slide Time: 19:16)



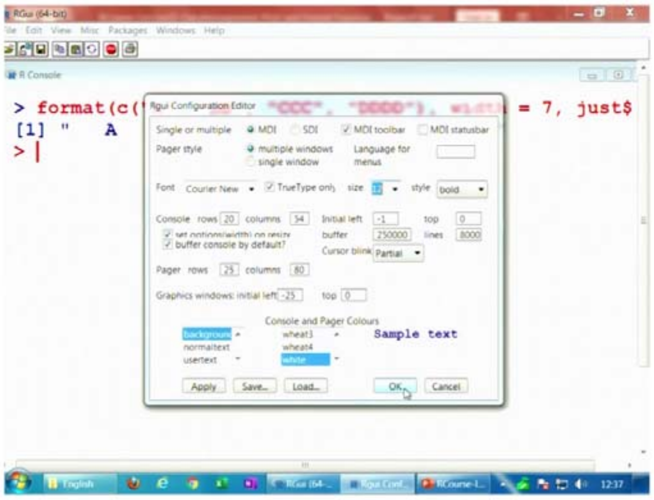
```
> print( format( 0.5, digits=10, nsmall=15 ) )
[1] "0.500000000000000"
> print( format( 0.5, digits=10 )
+ )
[1] "0.5"
> print(0.5, digits=10)
[1] 0.5
> sqrt(2)
[1] 1.414214
> print( format( 0.5, nsmall=15 ) )
[1] "0.500000000000000"
> |
```

So, now, let us try to first see this option that I am trying to print here 0.5 with the this is equal to 10 and nsmall equal to 15 and you get here this outcome. So, now let me try to show you here that if you try to use one of the options here what happens. So, if you try to see here, if you are trying to use here only this 0.5 with the format command and using the digits inside the format command what happens?

It will give you only here the value 0.5. There is a reason, I will try to show you that if I if you try to write down here only here pin 0.5 and with digits is equal to here say here 10. Even then it will give you 0.5, why? Because, 0.5 is an exact value whereas, if you are trying to use here square root of here 2, this is here like this in which the values are here more. So, here it will try to control the number of digits.

But, if you want to print here 0.5 in which you want to increase the number of digits like here this. Then instead of using here the digits, try to use here the option, nsmall and it will give you the value which is required here. So, this is here like this and you can see here there are 15 values after the decimal point. So, this is how you can control it.

(Refer Slide Time: 20:53)



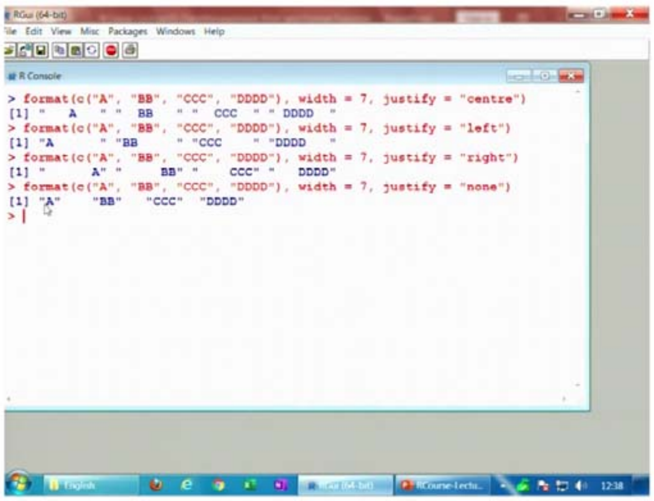
```
> format(c(                  "CCC", "DDDD"), width = 7, just$
[1] "   A
> |
```

Rgui Configuration Editor

```
Single or multiple   (• MDI) ( SDI)   [✓] MDI toolbar   [ ] MDI statusbar
Pager style          (• multiple windows)   Language for menus
                     ( single window)
Font   Courier New   [✓] TrueType only   size   style bold
Console   rows 20   columns 54   Initial left -1   top 0
  [✓] set options(width) on resize   buffer 250000   lines 8000
  [✓] buffer console by default?     Cursor blink Partial
Pager   rows 25   columns 80
Graphics windows: initial left -25   top 0
Console and Pager Colours
  background   wheat3   Sample text
  normaltext   wheat4
  usertext     white
```

But, I try to now add here the command here, justify, right. So, you can see here, if I try to give here the justify here as say centre, it is coming out of here like this. I can reduce the font size so, that you can see it very clearly, right.

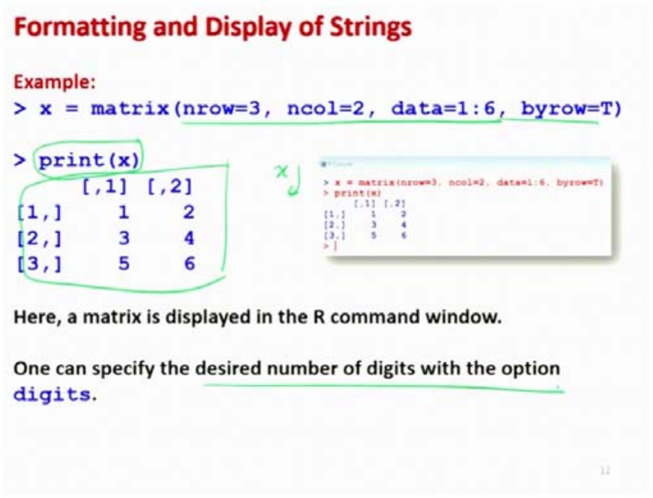
(Refer Slide Time: 21:02)



```
> format(c("A", "BB", "CCC", "DDDD"), width = 7, justify = "centre")
[1] "   A   " "  BB   " "  CCC  " " DDDD  "
> format(c("A", "BB", "CCC", "DDDD"), width = 7, justify = "left")
[1] "A      " "BB     " "CCC    " "DDDD   "
> format(c("A", "BB", "CCC", "DDDD"), width = 7, justify = "right")
[1] "      A" "     BB" "    CCC" "   DDDD"
> format(c("A", "BB", "CCC", "DDDD"), width = 7, justify = "none")
[1] "A"    "BB"   "CCC"  "DDDD"
> |
```

So, this is here centre and similarly if you try to make it here left. So, you can see what is happening, everything is coming on the left hand side. And, similarly if you try to make it here, right, then this is coming here like this and if you try to make it here none, then you can see here it is the default. So, this is what I wanted to show you that when that how you can control all these operations in the R software without any problem, right ok.

(Refer Slide Time: 21:36)



Formatting and Display of Strings

Example:

```
> x = matrix(nrow=3, ncol=2, data=1:6, byrow=T)

> print(x)
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
```

```
> x = matrix(nrow=3, ncol=2, data=1:6, byrow=T)
> print(x)
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
> |
```

Here, a matrix is displayed in the R command window.

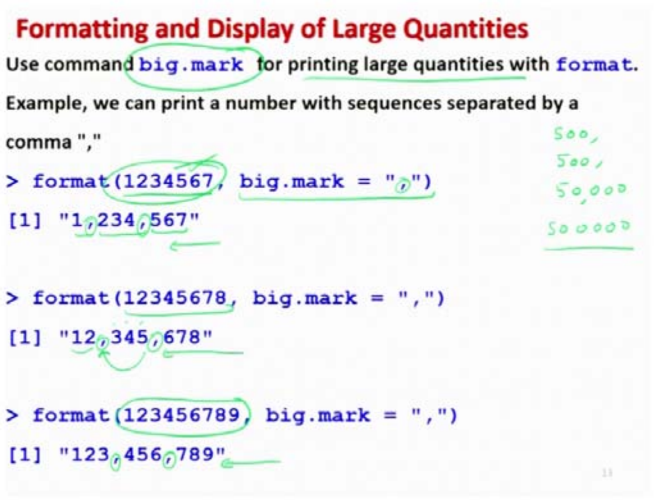
One can specify the desired number of digits with the option `digits`.

So, after this I try to show you that what happens when you are trying to play with the different say R objects. For example, I try to consider here a matrix. So, I try to consider here a matrix of order 3 by 2 in which the data is arranged by rows and the data values are 1 to 6, right. So, this is your here matrix x. So, now, if you want to see this matrix; so, matrix has a pattern. So, if you want to print the matrix also, you simply have to write down here print x and it will give you exactly in the same way as the matrix look like, right.

And, if you try to see the same command we used to get when we used to write only here x and enter. But, remember one thing, there is a difference; x is trying to give you the value which is stored inside it and print is trying to show you the value which is stored inside it. In a simple case, it will not make any difference because you are trying to take only the numerical values.

But, when you try to take the numerical and characters, then you will see this difference very clearly that how these things change. If you want to control here the digits so, that you also you can control here, right.

(Refer Slide Time: 22:52)



Formatting and Display of Large Quantities

Use command big.mark for printing large quantities with format.

Example, we can print a number with sequences separated by a comma ","

```
> format(1234567, big.mark = ",")
[1] "1,234,567"

> format(12345678, big.mark = ",")
[1] "12,345,678"

> format(123456789, big.mark = ",")
[1] "123,456,789"
```

So, after this I try to give you here one more option in the format. Sometime, you want to print the large quantities. Large quantities means, if the number of digits are quite large. For example, you know that when you are trying to write down the rupees means Indian rupees. So, we try to write down here if I have to write down here say 500, I will write down like this, if I have to write 5000 I will try to write down this.

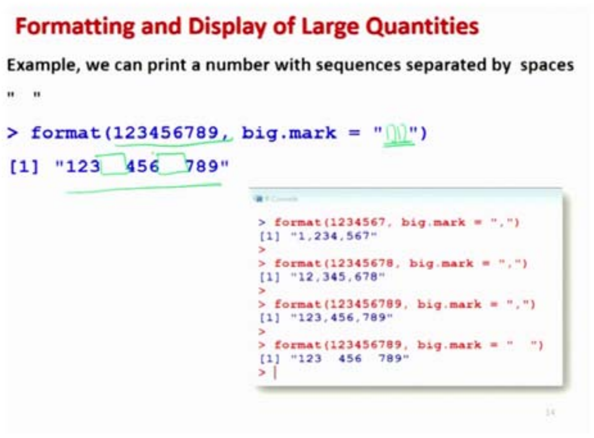
But, if I try to write down here 50,000, then you have seen that we try to make here a comma, right. And, similarly after this if I want to write here more digits here, we try to give a comma at suitable places, right. So, similarly if you have any requirement where you want to print such large quantities, where you want to control this type of spacing or any symbol inside this number, then how to get it done, right.

So, in this case what exactly do we want? We want to print the number with the sequence, which are separated by some value including the blank space suppose, I want to write down here 1 2 3 4 5 6 7, as here like this 1 comma 234 comma 567. Then, I have to use here a command here big dot mark; b i g dot m a r k. So, this command is used along with the format command and it will control the formatting that the values are separated by the given symbol.

So, suppose I want to separate them by comma. So, I will write down here the format, the number and then I will add here big dot mark is equal to within double quotes this comma. So, this comma will be start from the, right hand side and after third place it will be put here, then again after third place it will be put here. Now, in case if you try to increase the number of digits in this value. So, now I try to take here 8 values.

So, you can see here, then it starts from the, right, it will put it at the third place, then it will put at the means another third place which is means after 54 and 3 and then it will be here 2. And, similarly if you try to take here 9 digits 1 2 3 4 5 6 7 8 9, then the comma is going to be put which will begin from the, right hand side. And, it will be first comma after the third place, then the second comma once again after the third place from the first comma. So, you can see here and then it is 1 2 3. So, this is how you can control these outcomes in the R software also.

(Refer Slide Time: 25:45)



Formatting and Display of Large Quantities

Example, we can print a number with sequences separated by spaces " "

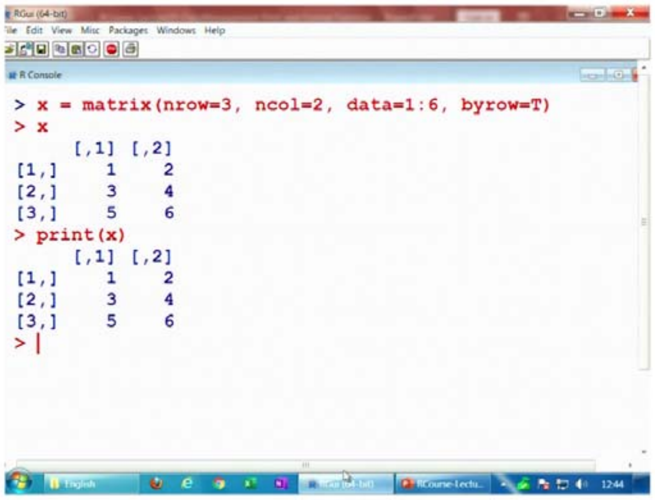
```
> format(123456789, big.mark = "  ")
[1] "123  456  789"
```

```
> format(1234567, big.mark = ",")
[1] "1,234,567"
>
> format(12345678, big.mark = ",")
[1] "12,345,678"
>
> format(123456789, big.mark = ",")
[1] "123,456,789"
>
> format(123456789, big.mark = "  ")
[1] "123  456  789"
> |
```

And, suppose if you want to choose any other symbol that also you can give within the double quote quotes. Say for example, if I simply want to separate the numbers by a blank signs.

So, you see here I am trying to give it here see here 1 and 2 blank signs and then you can see here this 1 2 3 4 5 6 7 8 9, this will be printed here like this where you have here these blank signs, right. So, let me try to first show you these options on the R console so, that you can operate them. And, you can see here well this is the command which you have used many times.

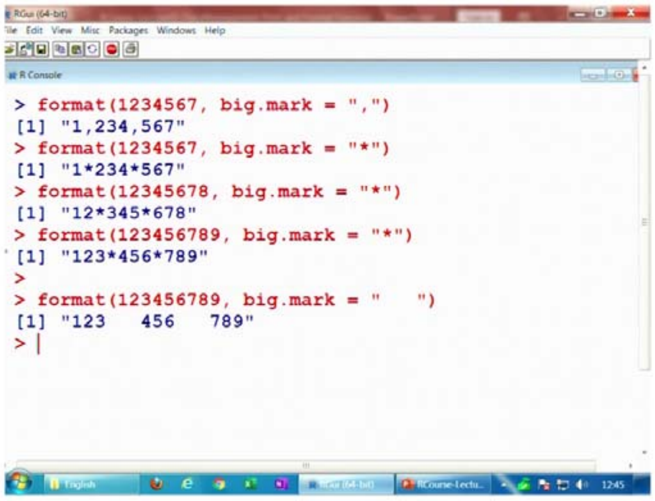
(Refer Slide Time: 26:26)



```
> x = matrix(nrow=3, ncol=2, data=1:6, byrow=T)
> x
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
> print(x)
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
> |
```

This is the matrix here that x will look like this, but if you try to see here print say here x, this will also give you the same outcome. But, definitely I will try to clarify this different that whatever will be the advantage of then using the print command in the next lecture, when I try to consider one more command that is cat.

(Refer Slide Time: 26:53)



```
> format(1234567, big.mark = ",")
[1] "1,234,567"
> format(1234567, big.mark = "*")
[1] "1*234*567"
> format(12345678, big.mark = "*")
[1] "12*345*678"
> format(123456789, big.mark = "*")
[1] "123*456*789"
>
> format(123456789, big.mark = "   ")
[1] "123   456   789"
> |
```

And, now similarly if you try to take here this example, that you want to use here the big mark so, you can see here when your big mark here is like this comma and if you want to use here some other star here, even then that can be done here. So, that depends on you and if you try to increase here the number of digits here and suppose I want to use the star. So, you can see here the star is coming at every third place beginning from the, right and if you try to make it here say here 9, this will be here like this, right.

So, this is how this R software works with the print and format command ok. So, now, we stop here and that was a pretty simple lecture, it was very easy to understand. And, I have just demonstrated the use of print command with the format and in format there are many options actually. So, you can see here that there are two types of command. One command which work within the parenthesis of print and there are another command which I am trying to give within the parenthesis of format.

And, which of the option does what, unless and until you practice it you will not able to learn it. I have taken here only some collected simple example to motivate you that you try to explore further. So, now, this is a very easy job, but it depends on you that how easy you want to make it. So, try to take some example and try to experiment with different combinations and try to see what happens.

You have to write your statement for formatting to get a desired output following the way in which R operates at this stage; unless until you try to write your own function and own program. So, at this stage I would request you try to look at the utility of all these options and try to incorporate, try to take different combination and try to observe the output and see how R works. So, you try to practice it and I will see you in the next lecture.

Till then good bye.