

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

Introduction
Lecture - 07
Some More Basic Operations in R

Hello friends, welcome to the course Foundations of R Software and you can recall that in the last lecture we started a small discussion on Some Basic Elementary Operations in the R Software. And in this lecture, we are going to continue on the same lines. So, we are going to demonstrate here and we are going to understand here some more basic fundamental operations to understand the functioning of the R software. So, let us begin our lecture and we try to learn some more aspects ok.

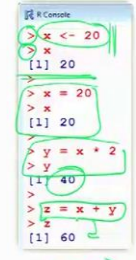
(Refer Slide Time: 00:48)

Assignment operator

- **>** is the prompt sign in R.
- The assignment operators are the **left arrow with dash <-** and equal sign **=**.
- **> x <- 20** assigns the value 20 to x.
- **> x = 20** assigns the value 20 to x.

Initially only <- was available in R.

- **> x = 20** assigns the value 20 to x.
- **> y = x * 2** assigns the value $2 \times x$ to y.
- **> z = x + y** assigns the value $x + y$ to z.



The slide includes a screenshot of an R console window. The console shows the following sequence of commands and outputs: `> x <- 20` (no output), `> x` (output: `[1] 20`), `> x = 20` (no output), `> x` (output: `[1] 20`), `> y = x + 2` (no output), `> y` (output: `[1] 40`), `> z = x + y` (no output), `> z` (output: `[1] 60`). Handwritten green annotations include a circle around the prompt sign '>', a circle around the '<-' operator with the note 'S-Plus', and a circle around the '=' operator. A note 'multiplication' is written above the second example, and '20 + 40' is written below the final output.

So, now, the first basic fundamental command which I am going to tell you is about this greater than sign. Although, I already have told you it many many times and you know it but in order to complete this lecture I need to inform you formally also. And I need to tell you some more stories also behind this thing. So, this is the prompt sign that you know that when you are coming in the R console, you can see this symbol.

So, this is the place after which you try to write your commands. Now after this when we try to consider the problem that how to assign a value to a variable, then the assignment operator in the case of R software. There are two assignment operators one is like this left arrow and dash this and this and second operator is the usual equality sign like this one. So, there are two assignment operators in the R software- one is less than and dash and another is equal to sign.

Actually when R is started then in the beginning there was only one assignment operator which was like this, less than and dash and this was the same operator which was also available in the S plus software. So, that is why possibly in the R the same operator continued, but recently couple of years back when this R software was updated then along with this the equality sign was also incorporated as an assignment operators.

Well in case if you try to ask any hard core programmer then possibly, they can explain you that what will be the difference between the use of these two symbols. But for users like us practically it will not make any difference whether we are going to use less than and dash symbol or equality symbol. So, now, onwards most of the time I will try my best to use equality operator, but in many many resources and books you will also see this operator- less than dash. So, please do not get confused both are the same thing. And I have explained you the reason also, so you need not to get confused.

So, for example, in case if I want to assign the value 20 to the x I have to write x less than hyphen 20. So, this is going to assign the value 20 to the x and similarly on the other hand, in case, if you write x equal to 20 this is also going to do the same thing and this will also assign the value 20 to x.

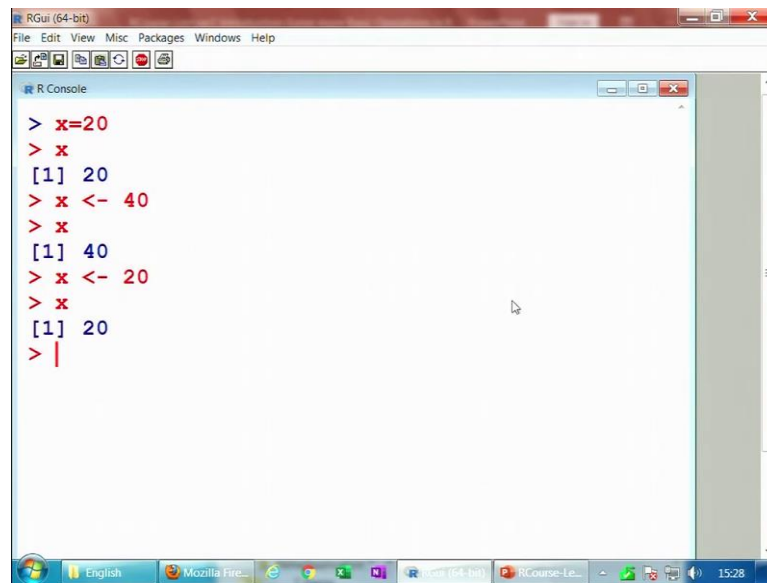
So, you can see here in the R console, I can show you that if you try to write down here, x less than hyphen 20 this also gives you 20 and when you write x equal to 20 this also gives you 20. And once you have assigned a value to a variable then it is also possible to assign a variable to another variable. For example, in case if I choose x equal to 20 and then I define a new variable x into 2 this star sin is the multiplication sign.

So, this 2 into x is going to be assigned to a new variable y. And after that the value of x plus y which is also a variable, this is going to be assigned to a new variable here z. So,

now, with these three statements, I am able to show you that numerical value can be assigned to a variable as well as a variable can also be assigned to another variable.

So, if you try to see here if I try to write down here y is equal to x into 2 and then the y value become 40. And if I try to write here z is equal to x plus y then x plus y is 20 plus 40 which is here 60. So, I try to show you on the R console also, so that you feel more confidence.

(Refer Slide Time: 04:54)

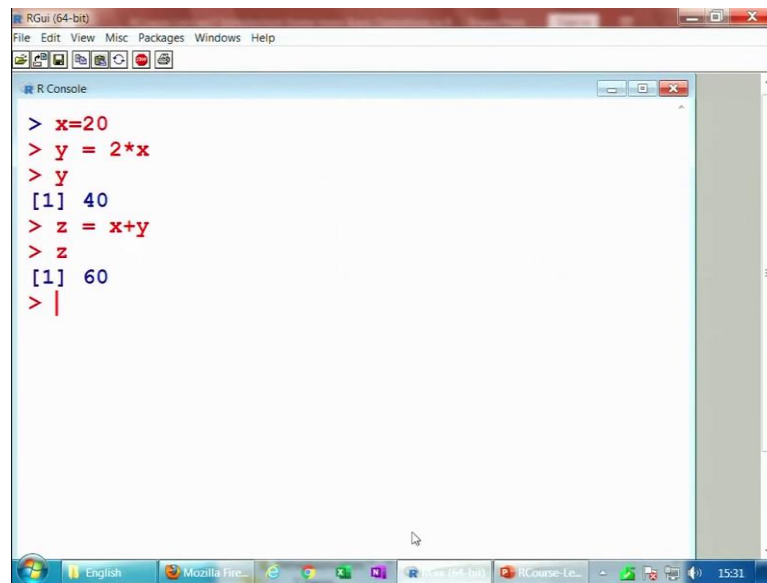


```
> x=20
> x
[1] 20
> x <- 40
> x
[1] 40
> x <- 20
> x
[1] 20
> |
```

So, if I try to say here suppose if I write here x equal to 20. You can see here now the value here is assigned to be here like this x equal to 20 and you already have done it many many times, but now I am telling you formally.

On the other hand, in case if you try to suppose write x less than hyphen say 40. So, you can see here what happens, now the value of x is 40. But in case if you try to write down here once again x less than hyphen 20, you can see here the value of x will change here to x equal to 20 right.

(Refer Slide Time: 05:28)



```
> x=20
> y = 2*x
> y
[1] 40
> z = x+y
> z
[1] 60
> |
```

So, now, in case if I try to define here x equal to 20 then suppose I define here another variable here y as say here 2 into x. So, now you can see here the value of y becomes here 40. So, the value of x is coming from the first variable and then the value of y is obtained and similarly if you try to obtain here z is equal to suppose x plus y, you can see here the value of z comes out to be 20 plus 40 which is 60. So now, you can see here that the values as well as variables can be assigned to any variable, right.

(Refer Slide Time: 06:05)

Assignment of numbers and characters

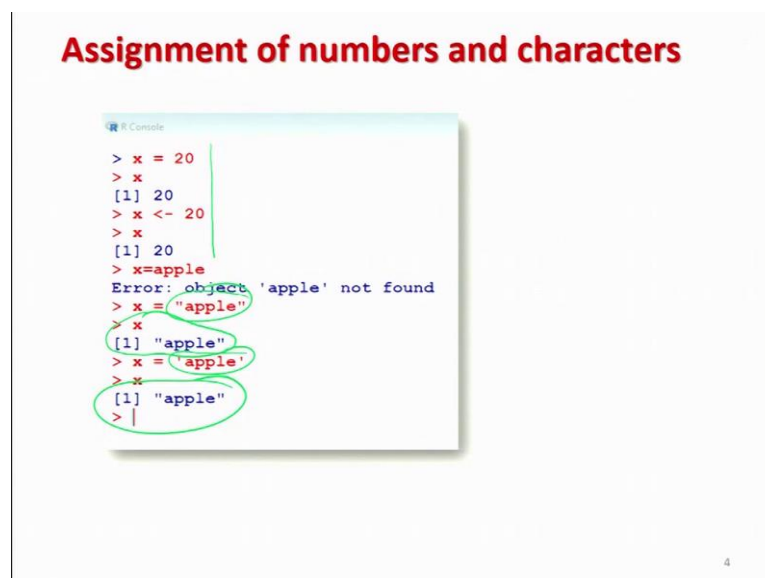
- The numbers are assigned in the usual way
 - > x <- 20 assigns the value 20 to x.
 - > x = 20 assigns the value 20 to x.
- The characters assigned within double or single quotes
 - > x = "apple" assigns the apple to x.
 - > x <- "apple1" assigns the apple to x.
 - > x = 'apple' assigns the apple to x.
 - > x <- 'apple' assigns the apple to x.

3

Now, when you are trying to assign a value to a variable then there are two types of values one which are numerical and other are some characters. So, the numbers, I already have discussed that the numbers are assigned to any variable in the usual way as we said x less than hyphen 20 or x equal to 20. But when you are trying to assign the or assign a character or some word or characters to a variable then they have to be assigned within the double quotes or within the single quote.

For example, suppose I want to define a word apple a double p l e and I want to assign it to a variable here x . So, I will write here double quotes and then within double quotes, I will write the value of the variable for example, it is here apple. So, this will assign the value apple to the variable x and similarly here also you can use this operator x less than hyphen. And suppose if I say that here you want to define here apple1, you can write down here x less than dash within double quotes, you write apple1. So, this will assign the apple1 to the variable x and beside those things, if you want to assign the value here with only the single quote also, that is also possible. Instead of using the double quote you can also use here the single quotes and either you try to use the equality operator or less than and a dash operator, they are going to assign the word apple to the variable x , right.

(Refer Slide Time: 07:48)



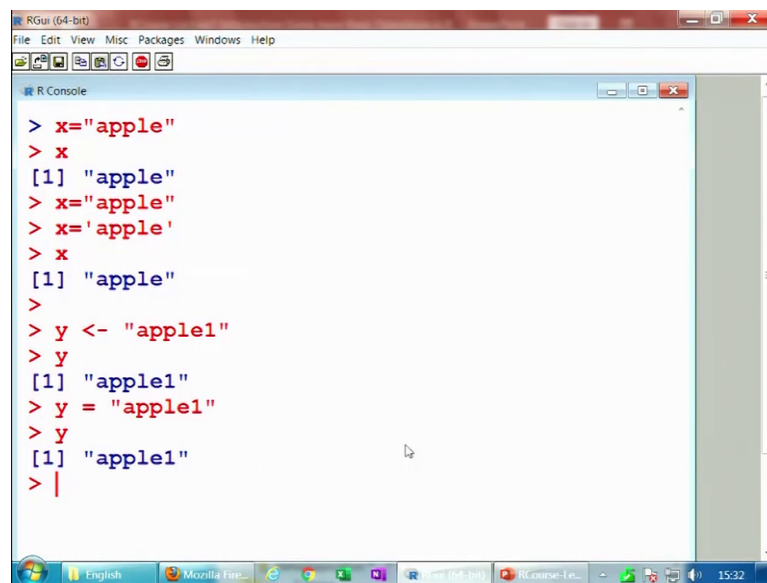
```
# Console
> x = 20
> x
[1] 20
> x <- 20
> x
[1] 20
> x=apple
Error: object 'apple' not found
> x = "apple"
> x
[1] "apple"
> x = 'apple'
> x
[1] "apple"
> |
```

So, here you can see here I have defined here like this x equal to 20 and here you can see here within double quotes, I am defining the variable and with single quote also I am

trying to assign the variable. And in both that cases, there is no issue both are acceptable, but when you try to look into the outcome will always have double quotes that is what you have to keep in mind.

So, by looking at the outcome you cannot judge whether the value was assigned through a single quote or a double quote. Now, I try to show it on the R console, so that you get more confident.

(Refer Slide Time: 08:25)



```
> x="apple"
> x
[1] "apple"
> x="apple"
> x='apple'
> x
[1] "apple"
>
> y <- "apple1"
> y
[1] "apple1"
> y = "apple1"
> y
[1] "apple1"
> |
```

So, suppose if I take here a variable here x is equal to apple, you can see here now this x becomes here apple. And instead of this thing in case if instead of this one, you can see here and now I am using here single quote. So, you can compare, now I see here apple. So, you can see here apple and instead of this if I try to define here y less than hyphen dash say here apple1.

You can see here this is here again apple1 or I can also use here the equality symbol like this then you can see here this is again apple1, right. So, this is the basic structure what we try to use in the R software. Now we try to understand another aspect in the R programming.

(Refer Slide Time: 09:19)

Knowing numbers and characters

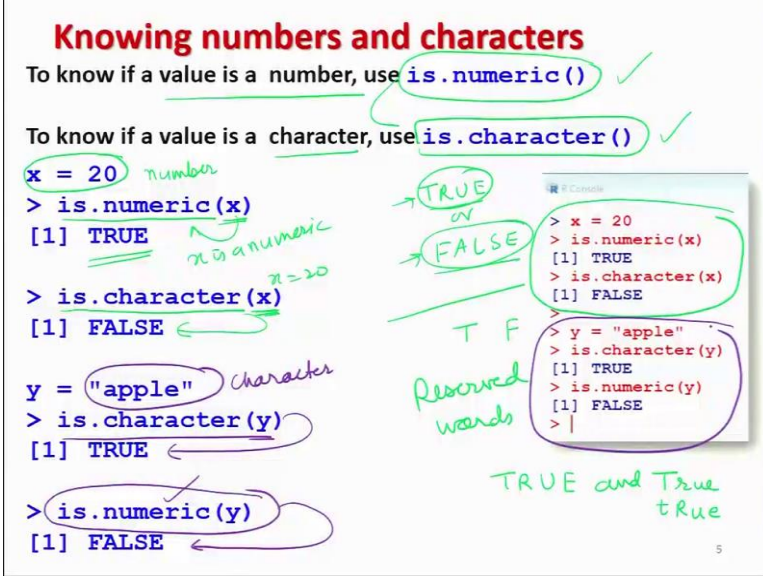
To know if a value is a number, use `is.numeric()` ✓

To know if a value is a character, use `is.character()` ✓

```
x = 20
> is.numeric(x)
[1] TRUE
> is.character(x)
[1] FALSE

y = "apple"
> is.character(y)
[1] TRUE
> is.numeric(y)
[1] FALSE
```

Handwritten notes: *x is a number*, *x is numeric*, *x=20*, *Character*, *TRUE or FALSE*, *T F Reserved words*, *TRUE and True true*



The image shows a slide with R code and console output. The code defines x=20 and y="apple", then uses is.numeric() and is.character() to check their types. The console output shows TRUE for is.numeric(x) and is.character(y), and FALSE for is.character(x) and is.numeric(y). Handwritten notes in green and purple explain the results and mention reserved words like TRUE and True.

Whenever you are trying to write down a program you are trying to handle some values. Now as you have seen these values can be numerical values or these values can be characters. So, now the question is that whenever you are trying to entertain any value within the program using some command, you would like to know whether this value is a numerical value or this is a character.

So, how to get it done? These type of situations are very common when you are trying to write down a program. For example, when you write a program you might be getting the data is filed from somewhere else and you have no idea. You have absolutely no idea whether the data is in the form of numerical characteristics or some data frame or some list etcetera. So, how to know you cannot look inside the file to know whether all the values numerical or some values are character or some values are numerical you did a way by which you can find it out.

So, in order to know that if a value is a number we have a command here is dot numeric and within the parenthesis you have to write down the variable. And in case if you want to know if a value is a character then the command is dot character i s dot c h a r a c t e r. And for is dot numeric this is i s dot n u m e r i c and in both the cases everything is in the lower case.

The outcome of these two commands is dot numeric or is dot character is going to be in terms of TRUE or FALSE although. I will take up this issue after some lectures, but let me tell you here that this TRUE and FALSE when they are written in the capital letters, capital T capital R capital U capital E or FALSE in all capital letter. Or instead of TRUE you can also write capital T and in place of FALSE you can write only here capital F.

These are the reserved words. Reserved words means these words are reserved to indicate only the TRUE and FALSE which are the logical variable. So, we will talk about the logical variable after some time, but at this moment I can explain you that you cannot use these two words TRUE and FALSE all in capital letters for any other thing.

And TRUE means TRUE and FALSE means FALSE one thing what you have to keep in mind that capital T capital R capital U capital E and capital T and all other things in a small letter small t r capital small letter u e etc. any other thing they are different. So, only the capital letters are the reserved words.

So, the answer to these two commands is going to be in terms of a logical variable as either TRUE or FALSE and then you have to understand what it is trying to say. For example, if I try to take here x equal to 20. Now you know that this is a number, so this is a numeric value.

So, now, if you try to use here the command is dot numeric and inside the parenthesis you write this value x, the answer is going to be here TRUE, so that means yes, the x is a numeric. And now in case if you try to use this command is dot character and within the parenthesis you write x.

So, x here is 20. So, do you think that x equal to 20 is a numeric or a character this is a numeric and it is not a character. So, when you try to use this command is dot character inside parenthesis x then it will give you the answer FALSE. That means x is not a character well one thing you have to keep in mind beside this number.

So, that you have to be careful when you are trying to interpret it. When you are trying to say is dot numeric is true, that means it is numeric. And when you are trying to say is dot character is FALSE. So, whatever is false that is FALSE, but then what is TRUE that we

do not know that can be character or that can be something else, that can be number that can be something else.

And if you try to execute these commands on the R console, you will get here the same thing what I shown you here. Now I try to take one more example, now I try to take here a character and I write here apple within the double quotes. So, now, you can see here, this is a character. Now I use the command here is dot character y answer is TRUE yes apple what you have given here in the variable y, this is a character.

And when you try to use here the command is dot numeric within the parenthesis y then this is indicating, it is FALSE. Obviously, y is apple which is a character, so when you are trying to check whether this is numeric it is telling you- No it is not numeric and your statement is FALSE. But now if you ask me what is this? Whether it is matrix data frame list etcetera that I do not know. So, you have to interpret the result in this particular way and if you try to do it on the R console also, you can see here, you will get the same outcome.

(Refer Slide Time: 14:58)

Converting numbers and characters

To convert a value as a number, use `as.numeric()`

To convert a value as a character, use `as.character()`

Converting a number into a character:

```
> x = 20
> is.numeric(x)
[1] TRUE
> y = as.character(x)
> is.numeric(y)
[1] FALSE
> is.character(y)
[1] TRUE
> y
[1] "20"
```

Handwritten notes: $20 + 30 = 50$, *apple + banana* or *apple + 50*, $x = 20$, $y = "20"$

```
> x = 20
> is.numeric(x)
[1] TRUE
> y = as.character(x)
> is.numeric(y)
[1] FALSE
> is.character(y)
[1] TRUE
> y
[1] "20"
```

And then the question comes over here. In case if you have got a number which is either numeric or character, can you interchange its role, why not? Means suppose I am getting a value here for example, x equal to 20. Is this a number for that I use command here is dot numeric it is true yes this is a number. But somehow I want to use this value as a

character. What is the difference between a number and a character. That in case if you have two numbers, 20 plus 30, both are numbers, so their sum is going to be 50.

But if something is character apple plus say banana or apple plus 50, you cannot find the sum. But somehow in some value in some experiment 20 is going to indicate some category. So, I want to use this value as a character. But the default value inside the R software or any mathematical software that is always taken as mathematical. So, the question is how can you convert a numeric value into a character?.

So, for that in case if you want to convert a value as a number, the command here is as dot numeric and within parenthesis you give the variable name and in case if you want to convert a value to a character, the command here is as dot character and within parentheses you have to give the variable name. And both this command as dot numeric and as dot character they are going to be written in the lower case alphabets.

So now, in case if you try to take the same example, suppose I am trying to take here x equal to 20 which is a numeric, but now I operate the command here as dot character within parenthesis x and whatever is the outcome I try to store it here as a y. Then now after if you check, whether this y is numeric.

So, for that you use the command here is dot numeric y and you will get here FALSE, although x was numeric but now y is no more numeric. Now you would like to check what is this y? Is this character then the answer comes here yes, why? Because you have operated the command as dot character to change a number into a character. And so y has now become character and whose value here is like this, you can now see here your x was 20.

But y is now given as 20 within the double quotes, you can see here. So that means, this is now the character you can see here this is the screenshot of the same operation, right.

(Refer Slide Time: 17:52)

Converting numbers and characters

Converting a character into a number:

```
> y = "apple"
> is.numeric(y)
[1] FALSE

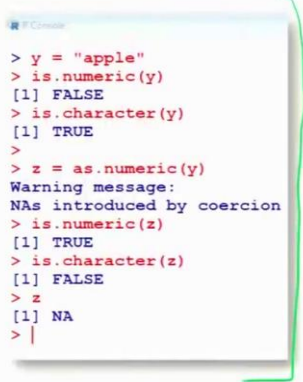
> is.character(y)
[1] TRUE

> z = as.numeric(y)
Warning message:
NA's introduced by coercion
> is.numeric(z)
[1] TRUE

> is.character(z)
[1] FALSE

> z
[1] NA
```

Handwritten notes: Error, Carefully, NA NULL



Now in case if you try to do the opposite suppose I have got here a character and I want to convert it into a number, what happens? So, suppose I take here the variable here apple which is a character and I store it here as a y. Now I try to operate the command here is dot numeric y. That means, I want to know whether this y is numeric or not.

The answer comes out to be here FALSE. Means obviously, y is apple it is character so it cannot be numeric. So, that is why the answer is coming out to be FALSE, but now I would like to check is dot character y. That means, whether y is a character, answer comes out to be here true yes, y is a character.

Now I try to convert this y into number. So, I try to use the command here as dot numeric and within parenthesis here y and then I try to store this value inside this new variable z. So, now, it is trying to give us a warning message. Whenever you are trying to work in the R software you will sometime get some messages. These messages are of primarily two types, one is here warning and second category is error.

So, what is the difference between warning messages and error messages? Warning is just like a warning. Warning means when you really want someone, you are not exactly going to do the same thing, right. And when you are getting a error message; that means, this is a mistake, this is an error and the program cannot execute after this. So, when you are getting the warning message; that means, the program will continue.

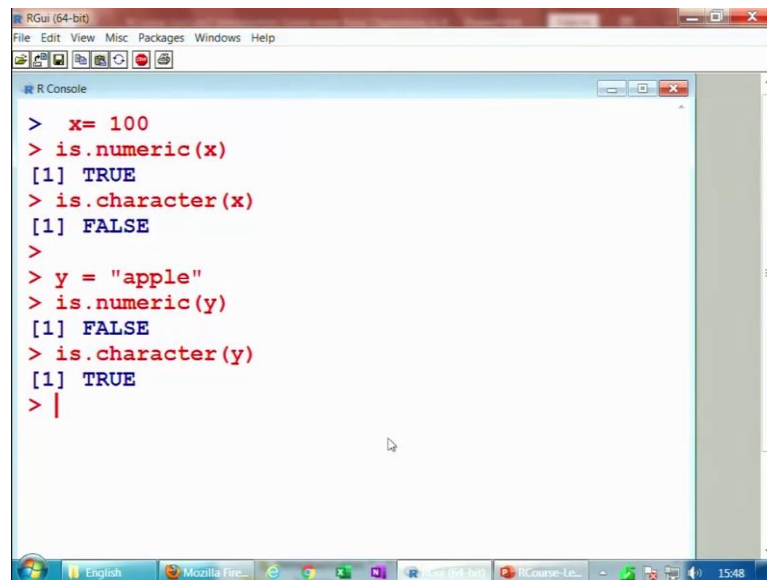
But it is warning you that something is happening please try to look, please be careful, please be watchful. But when there is error message then the program will simply stop there, it will not move forward. Because there is a mistake there is an error unless and until you correct it, the program will not run. So, it is now giving you here and is introduced by coercion means forcefully.

And NA is a sort of something like missing value- not available which we are going to discuss once again I will say in the further lectures in more detail. So, now, if you try to see here that is dot numeric z, it is telling you yes. Now z is numeric, but in case if you try to find out the value of the z, now tell me one thing. You are trying to convert apple into a number what do we expect? Is it possible- no?.

That is why when thing was possible, but when you are trying to do it forcefully then z is trying to replace this word by another number which is NA and in case if you try to see here is it a character answer is FALSE. Because NA is not actually a character it looks like a character NA at this moment, but after I give you the details of two aspects of two reserved works NA and NULL then you will understand that just like TRUE and FALSE these are also the reserved words.

So, this is how we are going to work in the R software when we are trying to convert the numeric into character or vice versa. And you can see here this is the same screenshot which I have explained you here, but then I would like to show you it on the R console also. So, let us try to understand these things. So, let me try to copy this command, so that I can save some time on the R console.

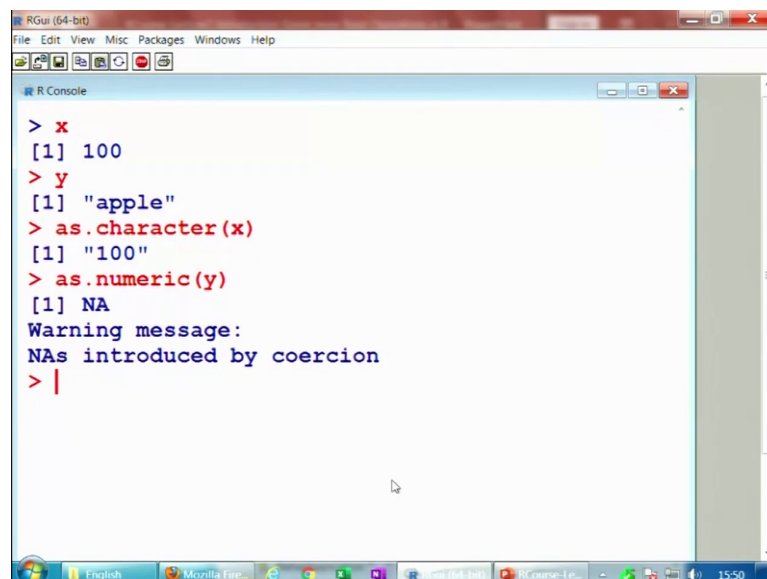
(Refer Slide Time: 21:44)



```
> x= 100
> is.numeric(x)
[1] TRUE
> is.character(x)
[1] FALSE
>
> y = "apple"
> is.numeric(y)
[1] FALSE
> is.character(y)
[1] TRUE
> |
```

So, let me try to remove these things by pressing control I now you know. So, if I try to take here x equal to 100 and if I try to do here is dot numeric x answer is TRUE. And if I try to say here is a character it is FALSE. But in case if I try to take here y as say here apple and now in case if I try to see here what is y? Is it at numeric? Answer is FALSE. But when I try to say here is it a character then it is coming as yes this is character.

(Refer Slide Time: 22:25)



```
> x
[1] 100
> y
[1] "apple"
> as.character(x)
[1] "100"
> as.numeric(y)
[1] NA
Warning message:
NAs introduced by coercion
> |
```

So, now because you already have seen that x equal to 100 is your number and y equal to apple is your character. So, I would like to convert them number into character and character into number. So now, you can see here the command here is in case if you try to convert a number into a character then the command here is as dot character. So, I try to use here as dot character x.

So, you can see here this is now within double quote it is 100. So, within double quotes as you are trying to say this becomes a character. So, by looking at this 100 and this 100 you can very easily identify that which of them is a numeric and which of them is a character. And similarly, in case if you want to change this character into a number then what you have to do- your y is here is apple and you want to convert into a numeric.

So, you can see here this is giving you the value here is coming out to be here NA, but it is giving you a warning message. That NAs are introduced by quotient means forcefully, means you are forcing R to do something which is not acceptable, but it is trying to do something to follow your orders and it is trying to replace the apple by the value NA, right. So, this is how the things work, when we are trying to work with this characters and numbers in the R software.

(Refer Slide Time: 23:58)

Comment operator

#: The character # marks the beginning of a comment. All characters until the end of the line are ignored.

> # mu is the mean

> # x <- 20 is treated as comment only

R Console

```
> x <- 20
> x
[1] 20
```

numerical value

Now after this one more aspect. Whenever you are trying to do any programming then as a good programmer you always like to give some information, which can be used either

by yourself or by somebody else. For example, suppose I am sitting at IIT Kanpur and I am writing a program in which I have assumed three variables, x is height y is weight and z is age.

Now when I am trying to send this program to somebody else outside IIT Kanpur, how do I communicate and if I try to communicate in a different file, that will be very confusing. So, what I will do, that within the program I would like to mention that x is this y is this and that is this. So, how to get it done?

So, in R software in case if you write this hash symbol in the first place on a line, then after that whatever you are going to write that will become only a comment. And no mathematical operations will be done over that. For example, in case if I am trying to write down here hash mu is the mean.

So, you can see here in case if you try to enter here, nothing will happen no warning error etcetera. That is accepted as soon as you write here hash all the characters after this up to the end of the line they are simply ignored and R will escape this line. R will come to know that I do not have to operate on this line.

And similarly in case if you try to write down here hash x equal to 20 or x hash less than dash 20, then it is going to be considered only as a comment. For example, if you try to write like here this. Then it is going to take it here as a numerical value, a numerical value of 20 is assigned to x. But when you are trying to write down here this hash then it is not going to be like this, right.

(Refer Slide Time: 26:00)

Case sensitivity in R

- Capital and small letters are different.

> **X** <- 20 and > **x** <- 20 are different

```
R Console
> x <- 20
> x
[1] 20
> X
Error: object 'X' not found
>
> X <- 30
> X
[1] 30
> x
[1] 20
```

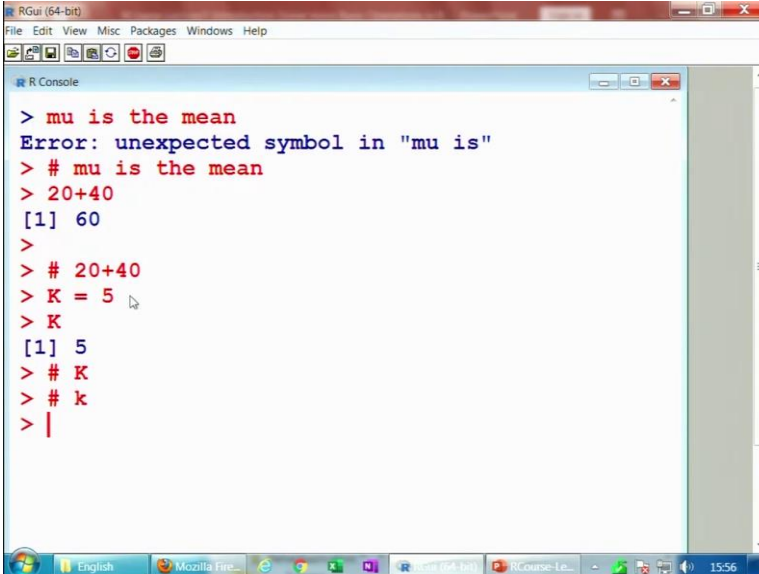
And one very important thing, R is case sensitive. The small letters and capital letters, they are different. For example, in case if you try to write down here X equal to 20 in capital letters and small letters they are going to be different for example, I can show you here on this screen output and then I will try to show you in the R console also when you are write trying to write down here x this is say lower case alphabet lowercase x.

You are trying to write down here lower case x is equal to 20 and you press here x it will come out to be here 20. But when you try to write down here capital X and enter it will say error object x not form because you have defined it to be small x, but you are trying to be here trying to find out the value of your capital X. But in case if you try to assign here the value capital X equal to 30 and then if you try to see the value of capital X, this will come out to be 30 and the value of small x will come out to be 20. So, I will try to show you this thing.

So, you have to be very careful when you are trying to work in the R console that lowercase alphabets and uppercase alphabet, they are going to be different. And that was the reason that many times whenever I am trying to explain you, about the functions name and commands whenever there is a small letter or capital letter or lower case or upper case; I am informing you very clearly that, this is what you have to do.

So, before we move forward, let me try to show you these two commands on the R console and then we will move forward with some more commands. So, now, you can see here.

(Refer Slide Time: 27:49)



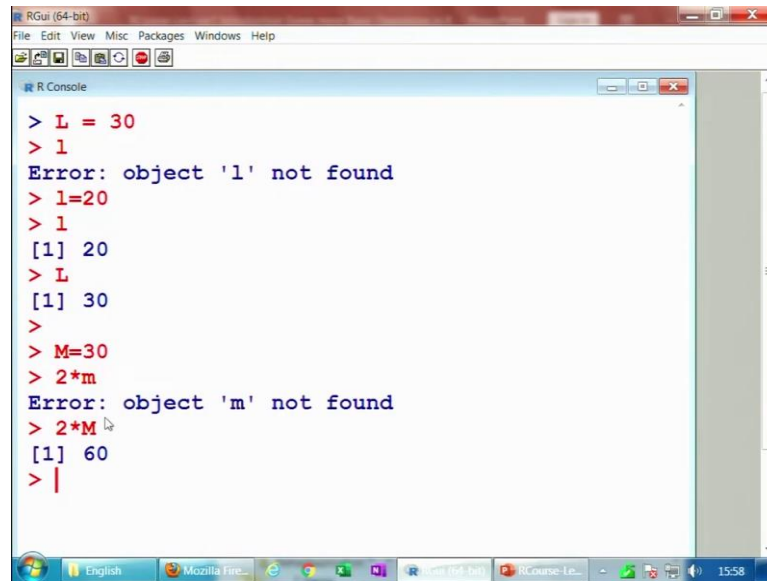
```
> mu is the mean
Error: unexpected symbol in "mu is"
> # mu is the mean
> 20+40
[1] 60
>
> # 20+40
> K = 5
> K
[1] 5
> # K
> # k
> |
```

Suppose if I write here mu is the mean, you can see here it will not understand it will say some error some unexpected symbol in mu is something like this. But in case if I introduce here the symbol hash then if I try to do it, it will, say it will just accept it that there is no issue.

Now, in case if I try to say here 20 plus 40 this will come out to be a 60. But in case if I try to put here a hash symbol before this 20 plus 40, let us see what happens? Nothing is happening, there is no value. So, that is the difference that when you are trying to put a hash at the first place then R is not really going to execute whatever is written after this right. Now suppose if I say here capital K is equal to 5.

And I try to write here, the value of here K like this and I try to write down here hash K there is no value there is no outcome, even if I try to write down a small k here it is there is no outcome, right. So, that is the thing what I want to show you here now. After this I try to show you, what is the difference between lower case and upper case alphabets. So, let me try to take it here.

(Refer Slide Time: 29:17)



```
> L = 30
> l
Error: object 'l' not found
> l=20
> l
[1] 20
> L
[1] 30
>
> M=30
> 2*m
Error: object 'm' not found
> 2*M
[1] 60
> |
```

Suppose if I try to take here capital L, this is equal to 30, right. And then I try to press here what is a small l, well I am not taking small x and capital X because you can see that it may be difficult for you to identify very clearly the difference between small and capital X, so that is why I have taken here l, right. But in case if I say here l is equal to 30 or say 20 you can see here.

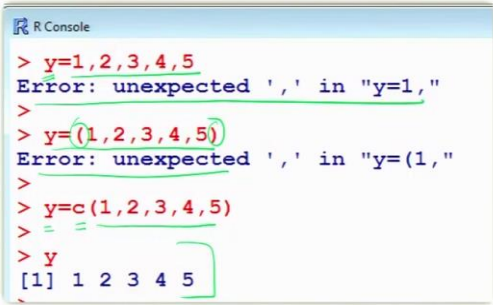
Now there is here l and if you want to press it here capital L it will give you this thing, right and similarly in case if you try to take here another variable here M is equal to say 30. And if you try to find out the value of a 2 into small m, it is saying no. It is not found because you have defined the value here as a capital M and you are giving here as a small m.

So but if you try to change it to capital M it will give you 60 that is what you have to keep in mind when you are trying to do the programming, these are very common mistakes and but sometime they take very long time to find it out. Similarly, there is another common mistake when people try to write 0 or o.

(Refer Slide Time: 30:26)

Combining values in a data vector

- The command `c(1,2,3,4,5)` combines the numbers 1,2,3,4 and 5 to a vector.



```
R Console
> y=1,2,3,4,5
Error: unexpected ',' in "y=1,"
>
> y=(1,2,3,4,5)
Error: unexpected ',' in "y=(1,"
>
> y=c(1,2,3,4,5)
>
> y
[1] 1 2 3 4 5
```

Once you write 0 and you want to write o or if you want to write o you write 0, it is very difficult to find. And particularly when you are trying to write down a mathematical expression where you have written o in place of 0, then it takes a considerable amount of time to find out the mistake, ok.

Now I come to another aspect. This is about combining the values in a data vector. Now you see I am using here the new terminology data vector. Do not get confused with the word vector we have vectors in physics, vectors in vector calculus etc. But this is here are data vector data vector means it is trying to combine more than one values.

For example, in case if you want to combine 5 values 1, 2, 3, 4, 5 and you want to give it as an input in the R software. Then the command here is that you write here small c. And then within the parenthesis, you try to write down all the values. So, this will combine all the values 1, 2, 3, 4, 5.

And in case if you try to follow something else I can show you with this screenshot here what will happen. Suppose if I want to combine 1 2 3 4 5 in our data vector y and if I simply give here y equal to 1 2 3 4 5 you can see here the outcome here is error. Similarly, if I try to give it here y is equal to 1, 2, 3, 4, 5 within the parenthesis, even then there is some error.

And in case if you write y is equal to a small c means lower case c and within parentheses 1, 2, 3, 4, 5 then you try to see this y here is like this. So, remember one thing very important for the R programming whenever you want to give more than one values in the R software, try to use this command lower case c and try to define the data vector using this command c.

(Refer Slide Time: 32:23)

Mode
Command `mode()` explains the type or storage mode of an object.

[Be watchful: It's not like mean, median, mode]

```
> x = 6
> x
[1] 6
> mode(x)
[1] "numeric"

> y = "apple"
> y
[1] "apple"
> mode(y)
[1] "character"
```

(Note: The slide also contains a smaller inset image of an R console window showing the same commands and outputs as above.)

This is very important, but before that I try to show it to you on the R console also. So, that you are clear about it.

(Refer Slide Time: 32:33)

```
RGui (64-bit)
File Edit View Misc Packages Windows Help
R Console
> y = 1,2,3,4,5
Error: unexpected ',' in "y = 1,"
>
> y = (1,2,3,4,5)
Error: unexpected ',' in "y = (1,"
>
> y = c(1,2,3,4,5)
> y
[1] 1 2 3 4 5
> |
```

You can see here if I try to define here y is equal to suppose 1, 2, 3, 4, 5, you can see here it is not giving me any value. And even if I try to make it here parenthesis it is giving me error. But when I try to write down here small letter c , lower case c then it is assigning the values, right.

So, this is the way and whenever you are trying to do the mathematical operation and other types of operations, unless and until you give the values in this format using the c command there will be trouble things may not really work. So, you have to be very careful when you are trying to do it, ok. Now I would like to give you one more command this is about mode `mode`.

This command `mode` explains the type or storage mode of an object. For example in case if you try to say here x equal to 6. So, this is a number, so the mode of x is numeric. So, in order to find out the mode of a variable you have to use the command `mode` and within the parenthesis you have to write the name of the variable.

Now here I would like to caution, to give a caution to all of you that when I am trying to use the word `mode` sometime people get confused that ok, it is a mode like mean, median, mode, that we study in statistics. Because in order to find out the mean the command is `mean` in order to find out the median the command is `median`.

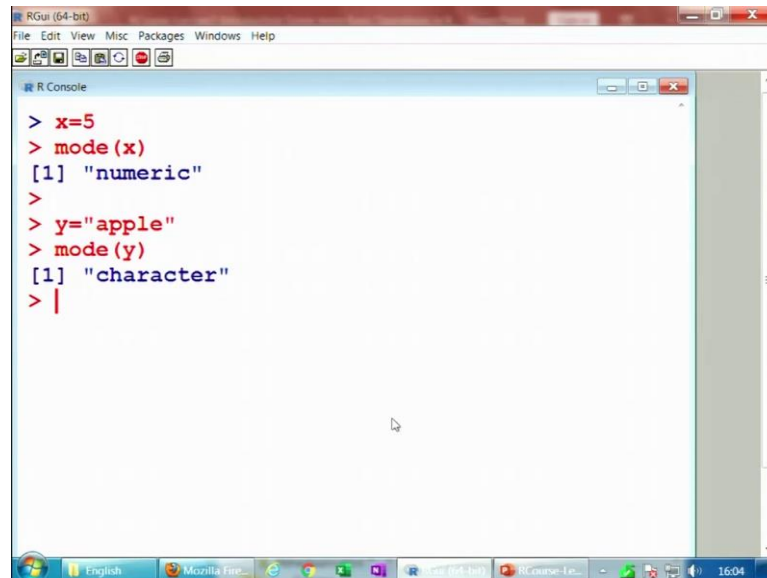
But in order to find out the mode the command is not the mode it is something else. So, be watchful in your childhood you have learnt the chapters like, modes of transportations you say that ok there are different ways for the transportation you go by ship, go by air, go by train etc. So, this mode is like that mode it is not the statistical mode.

Where you try to find out the value corresponding to the maximum frequency be careful. Similarly in case if you try to take a character like as here I try to take a variable y in which I sign the character as apple then the mode of this y mode of this apple will be character. So, this is another operation mode which helps you when you are trying to do a programming and when you are trying to get the data from some external sources.

Because you will be getting a file which will have say thousands and millions of values and it is not possible to go through with a each and every number in the file to know the

mode of the value. And based on that you have to do the mathematical operations and this is here the screenshot of this thing. I will try to show you on the R console also.

(Refer Slide Time: 35:24)



```
RGui (64-bit)
File Edit View Misc Packages Windows Help
R Console
> x=5
> mode(x)
[1] "numeric"
>
> y="apple"
> mode(y)
[1] "character"
> |
```

You can see here if I try to see here x equal to here 5 then the mode of here x here is numeric.

(Refer Slide Time: 35:41)

- Mode**
Following modes are available:
- "logical", ✓
 - "integer", ✓
 - "double", ✓
 - "complex", ✓
 - "raw", ✓
 - "character", ✓
 - "list", ✓
 - "expression", ✓
 - "name", ✓
 - "symbol" and ✓
 - "function". ✓
- 12

But in case, if I try to take here y here as a apple then mode of y here is character. But in R, we have some more modes that we will to use and learn whenever we need them, they are logical, integer, double, complex, raw, character, list, expression, name, symbol and

functions. So, we are not going to do it here, but then whenever you need such numbers you can always find it out.

(Refer Slide Time: 35:59)

Mode

Command `storage.mode()` returns the storage mode of its argument.

It is generally used when calling functions written in another language, such as C or FORTRAN, to ensure that R objects have the data type expected by the routine being called.

13

Now there is another command for this mode that is the storage mode this command returns the storage modes of its argument. That means whatever is written inside this parenthesis whatever is the mode of storage of this value that is informed through this command. Actually this command is usually used when we are trying to call the functions which are written in other language like as C or FORTRAN. And this actually ensures that the R objects have the data type as expected by the routine being called by those programming languages, right.

(Refer Slide Time: 36:38)

Mode

mode	storage.mode
logical ✓	logical ✓
numeric ✓	integer ✓
numeric	double ✓
complex ✓	complex ✓
character ✓	character ✓
raw ✓	raw ✓

14

So, we are not really going to use it here much in this course, but it is very important for you to know. For example, if the mode is logical then its storage mode is also logical when the mode is numeric then the storage mode is integer or double. When the mode is complex then the storage mode is also complex, when the mode is character then the storage mode is also character, when the mode is raw then the storage mode is also raw.

(Refer Slide Time: 37:08)

Mode
Examples:

```
> x = 6 #numeric
> storage.mode(x)
[1] "double"

> x = TRUE #logical
> storage.mode(x)
[1] "logical"

> x = "apple" #character
> storage.mode(x)
[1] "character"
```

R Console

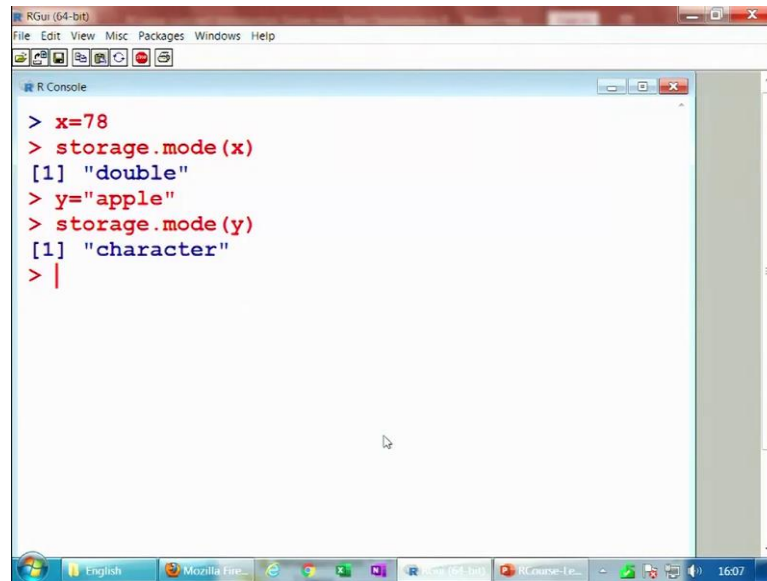
```
> x = 6 #numeric
> storage.mode(x)
[1] "double"
>
> x = TRUE #logical
> storage.mode(x)
[1] "logical"
>
> x = "apple" #character
> storage.mode(x)
[1] "character"
> |
```

15

So, this is for your information that you must know actually. For example if you want to know the storage mode for example, if I try to say here x equal to 6 which is a numeric then if you try to use the command here is storage dot mode and within parenthesis you write this x then it will give you the answer double.

Similarly, if you try to take a logical variable which I just explained you as small x equal to TRUE which is a logical variable, so the storage mode of this x is logical. And in case if you try to, take here a character like as x equal to here apple. Then the storage mode is here character and the same thing you can see this on the screenshot of this software also. So, before I go further let me try to show you these things on the R console also. So, that you are more convinced, right.

(Refer Slide Time: 37:59)



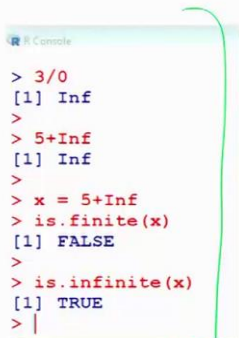
```
> x=78
> storage.mode(x)
[1] "double"
> y="apple"
> storage.mode(y)
[1] "character"
> |
```

So, if I try to take here x equal to here 78 for the this is a number, so the storage mode of x is double. But in case if I try to take here y is equal to here apple right, then the storage mode of this y here is character and so on. So, this is needed when you are trying to write down the programs many times for the mathematical operations, you need to do it and based on that you try to make your operations ok. Now whenever you are trying to do some calculations.

(Refer Slide Time: 38:36)

Infinity
Some calculations yield result as infinity (∞). For example, $3/0$. In R `is.finite()` and `is.infinite()` are used to know if an outcome is finite or infinite, respectively.

```
> 3/0
[1] Inf
> 5+Inf
[1] Inf
> x = 5+Inf
> is.finite(x)
[1] FALSE
> is.infinite(x)
[1] TRUE
```



```
> 3/0
[1] Inf
> 5+Inf
[1] Inf
> x = 5+Inf
> is.finite(x)
[1] FALSE
> is.infinite(x)
[1] TRUE
> |
```

16

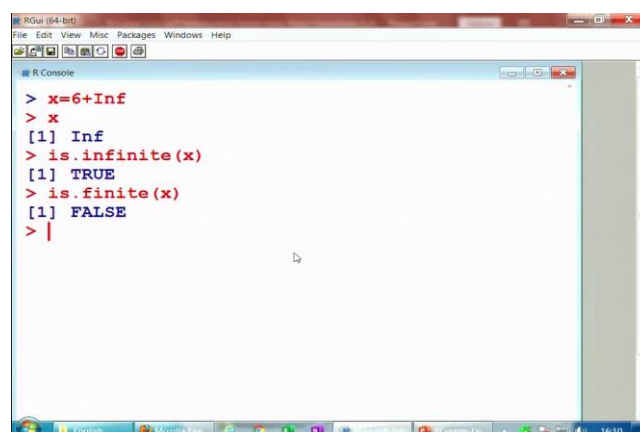
For that you write a programming sometime you get the answer as infinity and many times, this creates a problem and while you are working with a longer program. Then sometime in between at some place you are getting some value infinity and it is not giving you the output. At that moment we always would like to know that somehow if I can find out the outcome of this command is coming out to be finite or infinite.

Because it may happen that you are trying to divide some number by some number, but somehow during the computation that number has become so small that it is almost 0. So, when the number divided by 0 that will give you infinity. So, how to know all these things?. So, in R software in case if you want to know whether our number is finite or infinite, we have a command here is dot finite or is dot infinite. For example, you know that when you are trying to divide 3 by 0, the answer is infinity.

That you can see here in the R if you try to divide 3 divided by 0, the answer will come out to be Inf. So, which indicates that infinity right and you know that if you try to add 5 in infinity the answer will again be infinity or if you try to subtract divide multiply anything it will remain as infinity. So, if you want to know that there is a variable x is equal to 5 plus infinity 5 plus inf. Then I just want to know what will the outcome.

So, I am asking, is dot finite this x is obviously, we have given here infinity though x is not finite. So, it will give you the answer here as say FALSE and then if I ask, is dot infinite x it will give you the answer TRUE yes it is infinite, right. So, and you can see here this is the screenshot and if I try to show you these things on the R console and if I try to define here.

(Refer Slide Time: 40:43)



```
> x=6+Inf
> x
[1] Inf
> is.infinite(x)
[1] TRUE
> is.finite(x)
[1] FALSE
> |
```

The variable here I say x is equal to suppose 6 plus infinity, right. So, you can see here x is infinity and if you want to find out is, x infinite TRUE and if you want to know is x finite answer is FALSE. So, this is how you can use such commands while doing the programming. And that will help you, right, ok.

So now, we come to an end to this lecture and now you can see here in this lecture we have understood a variety of things. And these are a small commands which are trying to give you some information about the intermediate steps also. So, my request will be that you please try to go through these commands, try to execute them and try to understand, what are they trying to do, that is more important. And I am not saying that these are the only commands which are possible in the R software there are many more commands. But I expect that if you know these many commands after that if you want to know more commands and if you consult any book or any source, goods source, it should not be a difficult thing for you to learn.

At least you will understand what is really happening once you know what is this is dot character or as dot character after that if you come to know that there is a command like is dot matrix or as dot matrix, you will automatically know that it is trying to understand the character or the characteristics of this variable, whether this is a matrix or not.

So, that was my objective to choose and to give you some representative commands, so that your further learning in the R software become easier. So, you try to practice it and I will see you in the next lecture till then, good bye.