

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

**Lecture – 11**  
**Truth Table and Conditional Executions**

Welcome to the next lecture on the course Introduction to R Software. In the last two lectures, we had talked about the logical operators. And in the last lecture I had concluded with a topic on truth tables, and I told you that that all these logical operations are based on something called truth table. So, first we need to understand what is this truth table and how are acts on the truth table. And after that we will do something about the conditional equations means that I want to execute something based on certain conditions.

So, these are the two topics which I am going to cover in this lecture. So, let us try to start here. We have seen in the logical operations that we are trying to take a decision in terms of true and false. In the logical operators we had taken several examples where we have combined two statements, and then we had checked whether each of the statement is true or false, and based on that we had taken the final call whether the combination of them is true or false, and this combination was made using the operators AND or OR.

So, what are the logics behind this operation, that when we try to combine two statements then if both the statements are true, if both the statements are false and if any one of the statement is true and other is false then how do we take a conclusion. That is what is denoted in this truth table.

(Refer Slide Time: 02:12)

**Example of Standard logical operations**

**Truth table**

Statement 1 :: (x)	Statement 2 :: (y)	Outcome :: x and y	Outcome :: x or y
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

2

So, let us now to concentrate on the truth table. Suppose I have two statements: a statement number 1 and a statement number 2. And suppose we denote this statement one by a variable  $x$  and the statement two by a variable  $y$ .

Now, suppose first situation is that the statement 1 or  $x$  is true, then the statement number 2 which is denoted by here  $y$  suppose, this is also true. Now I have two operators one is AND and another is OR. So, when I try to use the AND operator over the two statements 1 and 2 that is through  $x$  and  $y$ , or in simple words I want to take a judgment what will happen to  $x$  and  $y$  in this case. That means, in more simple words; what will happen to  $x$  and  $y$  when  $x$  is true as well as  $y$  is true. In this case the answer comes out to be true.

So, in case if the statement 1 is true, a statement 2 is true then their combination with AND operator will also be true. Now I want to test if a statement 1 is true and statement 2 is also true then, what will happen to my statement  $x$  or  $y$ ; that means, both the statements are true then, what is the outcome when they are combined with the OR operator or means either of them is true. In this case the outcome is true; that means,  $x$  or  $y$  will give us the outcome true.

Similarly, if I take another case where a statement 1 is true a statement 2 is false, then the question is what will happen to a statement 1 and a statement 2. This answer will come out to be false. Then what will happen to  $x$  or  $y$  or a statement 1 or a statement 2; that will come out to be true, because at least one of them is true. And in the case of  $x$  and  $y$  I

am trying to see whether both are true simultaneously. So, that is why x and y is giving me false and whereas, when I am trying to say x or y; that means, at least one of them is true.

So that means, the answer comes out to be true. And the same thing happens in the reverse case that is, when the statement 1 is false a statement 2 is true then, x and y will give us an answer false and x or y will give an answer true. And the case number 3 is simply the complement of case number 2. Now the last question is where both the statements 1 and 2 are false: a statement 1 is false, a statement 2 is false.

Now, I want to check what will happen to a statement 1 and the statement 2 together simultaneously. So, this will come out to be false. Now I want to check what will happen to OR operator. That means, if a statement 1 is false or a statement 2 is false then what will be the outcome; the outcome comes out to be here false. And this table is called truth table and all the logical operation they are based on this truth table.

Now let us try to do this over r and see whether R follows such statements or not.

(Refer Slide Time: 05:55)

**Example of Standard logical operations**

```
> x = TRUE ✓
> y = FALSE ✓

> x & y # x AND y
[1] FALSE

> x | y # x OR y
[1] TRUE

> !x # negation of x
[1] FALSE
```

*x: True*  
*!x: False*

**R Console**

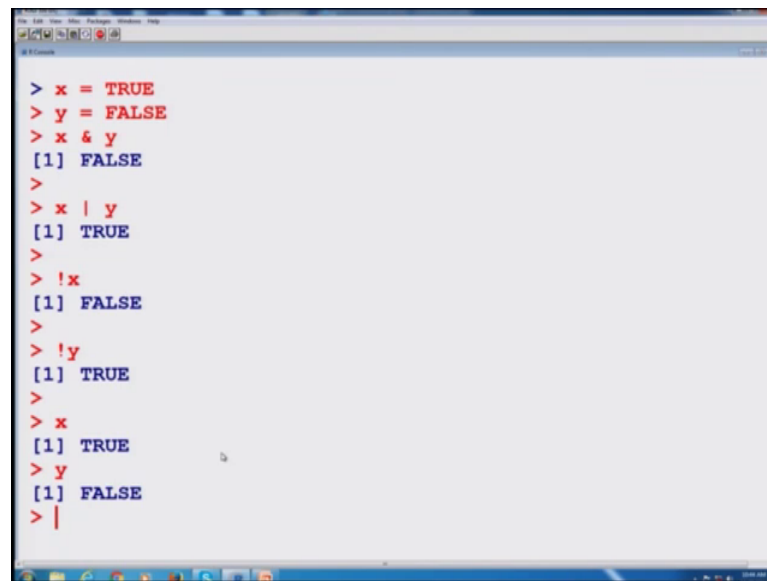
```
> x = TRUE
> y = FALSE
> x & y
[1] FALSE
> x | y
[1] TRUE
> !x
[1] FALSE
```

Suppose I define here two statements x as true and y here as a false. Now I try to say here x and y. So, x and y we have seen here in the earlier slide that was given here as a false. Now, you can see here the answer is coming out to be here false. Similarly, if I try to take here OR operator which is denoted by this symbol, then x or y is giving me here

true. That is again from the same operations, in the case number 2 like as here which I am is squaring it.

And, similarly if I try to say here the negation of x. Negation of x is denoted by exclamation sign and here x. so; obviously, here when x I have taken here is to say here to be true, then the negation of x is going to be here false. Then this comes out to be like this. I just try to do the same thing here over the R console.

(Refer Slide Time: 07:22)

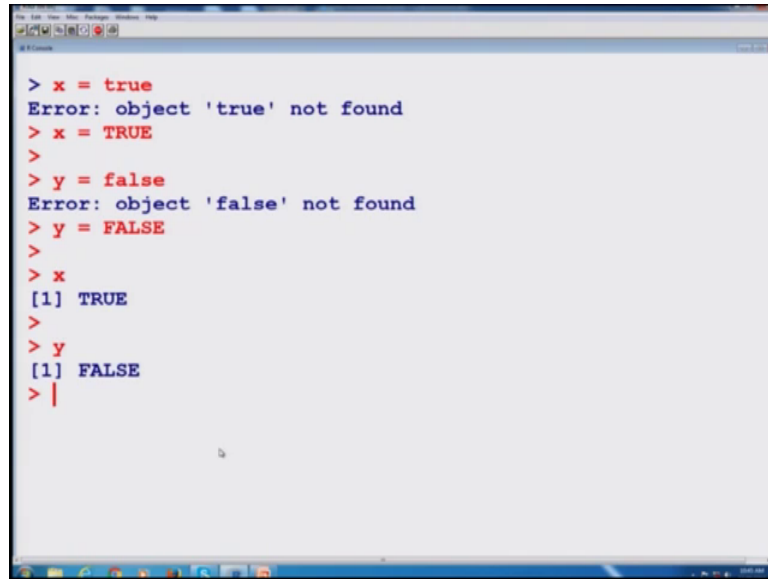


```
> x = TRUE
> y = FALSE
> x & y
[1] FALSE
>
> x | y
[1] TRUE
>
> !x
[1] FALSE
>
> !y
[1] TRUE
>
> x
[1] TRUE
> y
[1] FALSE
> |
```

Now I try to define here my here x as true and, y to be here false. Now you can see here I will try to do here all the operation x and y and x this thing and us try to see what happens.

For example, if I try to do here x and y you can see here this is false, if I try to do here for the OR operator this comes out to be here true. And if I try to see here negation of here x you can see here this is comes out to be false. And similarly if I try to take here negation of here y then this comes out to be here true, yes; obviously, when y is false then the negation of y will be true. One thing I want to show you here that when I try to find out, what is the value by x and what is the value of here y this gives me here true and false.

(Refer Slide Time: 08:28)

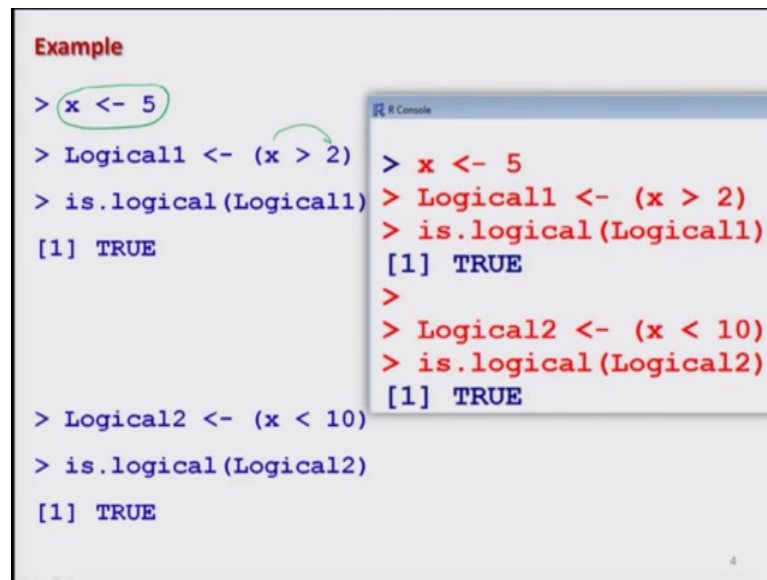


```
> x = true
Error: object 'true' not found
> x = TRUE
>
> y = false
Error: object 'false' not found
> y = FALSE
>
> x
[1] TRUE
>
> y
[1] FALSE
> |
```

Now, do here like this. If I try to define here x here as a true: why this is happening because I am using the small letters true and we had discuss that when I am trying to use the word true, this has to be given only with capital letters. And same is to with here with the word here false, if I try to do, if I try to type it with the small letters this will give me false, but on the other hand if I try to type y here with capital letters it gives me the same thing. So, you can see here now x is true and y here is false.

So, that is the thing which we have to keep in mind, whenever you are trying to deal with the logical operators using say here true and false right. Now I take some more example and try to see what happens, for example, if I assume a particular value x equal to 5, and I try to compare.

(Refer Slide Time: 09:45)



The image shows a slide titled "Example" with R code and its output. The code is as follows:

```
> x <- 5
> Logical1 <- (x > 2)
> is.logical(Logical1)
[1] TRUE

> Logical2 <- (x < 10)
> is.logical(Logical2)
[1] TRUE
```

An inset window titled "R Console" shows the same code in red text with the following output in blue text:

```
> x <- 5
> Logical1 <- (x > 2)
> is.logical(Logical1)
[1] TRUE
>
> Logical2 <- (x < 10)
> is.logical(Logical2)
[1] TRUE
```

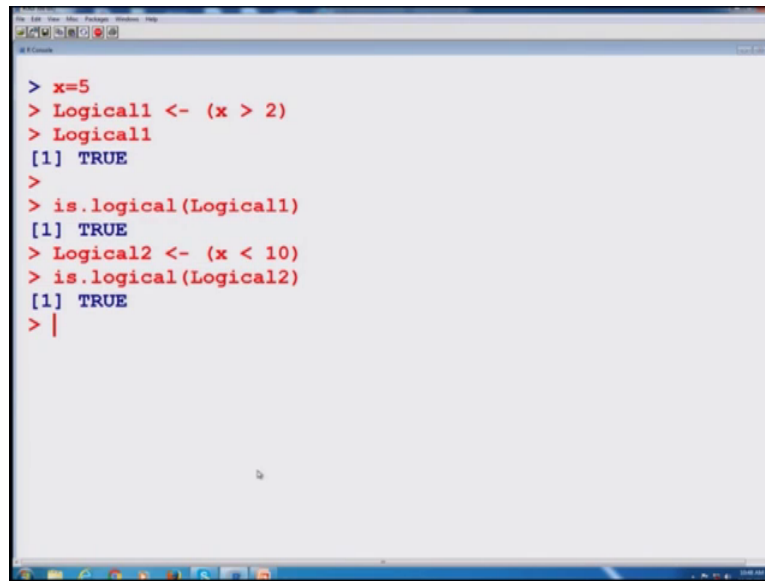
A green circle highlights the assignment `x <- 5` in the main code, and a green arrow points from it to the `x` in the `(x > 2)` comparison in the next line.

Whether this x is greater than two or not, and I try to assign whatever be the result of this thing, inside a variable calling it to be logical 1, logical one because I am taking 4 example, logical 1, logical 2, logical 3 in logical 4 will give us a sort of understanding.

Now, whatever is the outcome of this thing that we do not know, but I really want to know, that whatever is the outcome of this logical 1: is it really, logical and what is the outcome. In that case similar to the earlier statement I can use here is dot logical, is dot logical all in small letters, and inside the bracket I can write down the variable name. That means, I want to check whether, the value contained in the variable logical 1 is a logical value or not.

Let us try to do it here and try to see it.

(Refer Slide Time: 11:04)

A screenshot of an R console window. The window title is "R Console". The console shows the following commands and their outputs:

```
> x=5
> Logical1 <- (x > 2)
> Logical1
[1] TRUE
>
> is.logical(Logical1)
[1] TRUE
> Logical2 <- (x < 10)
> is.logical(Logical2)
[1] TRUE
> |
> |
```

So, I try to take here  $x$  equal to here 5, and then I try to define here logical 1 as variable. And you can see here, if I want to see what is the value of here logical 1, this can come out to be here true, but I do not want to do this thing, but I want to use this operator is logical to see whether this value is really true or false. And the answer comes out to be a true right and.

Now, similarly if I want to check here say this, another statement about  $x$  less than 10. Whether the answer of this  $x$  less than 10 is logical or not, I can assign this value, inside a new variable say logical 2 and I try to store it here, and then I try to check here whether this variable is really logical answer comes out to be here true. And here you can see here that I have given here the screenshot of this operation, so that you can verify it yourself.

Similarly, if I try to click here another example here, where I try to check whether, twice of  $x$  is greater than 11 or not, and I try to assign it inside a new variable say this logical 3. Then, I want to check it here, whether this is logical or not. So, I am trying to use the same thing over and the answer comes out to here true. Similarly, I try to check whether three times  $x$  is smaller than 20 or not, with  $s$  equal to 5 and whatever is the outcome this I am trying to assign inside a variable logical 4, and then I am trying to check here whether this is true or false, or whether this is really logical or not and the outcome here comes out to be here true, And the screenshot is given over here. So, I would request to that, you try to operate this logical 3 in logical 4 statements and try to check yourself.

Now, a bigger question comes, why I am doing all these things. Why I want to check whether this statement is true or false. Once I know that  $x$  is equal to 5, then I know that we whether 5 is greater than 10 or 5 is smaller than two, but you can think of a bigger program. Suppose there is a program that is describing the entire system. In that programming, there can be several steps where you would like to execute the thing, depending on whether something is true or something is false.

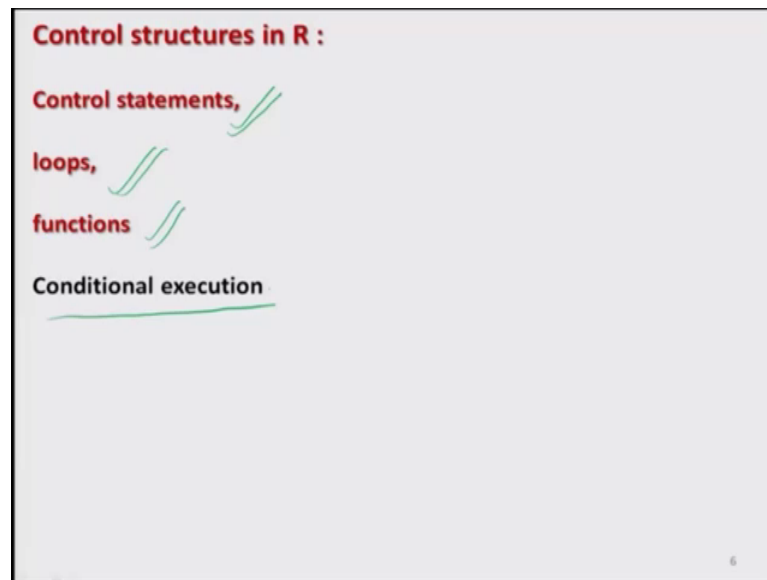
For example, in case if you are trying to, write down a program for issuing a ticket. Then the price of that ticket suppose depends on the age of the person. Then you would like to see whether the age is greater than or a smaller than any threshold value, for example, if you say that the price for all the children up to the age say 10 years is 0, the entry is free and the price of the ticket for anyone whose age is more than 10 that is, 20 rupees; that means, I would like to check as soon as, I enter the value of the age I need a conditional execution to tell me whether the statement whether age is greater than 10 or a smaller than 10 is correct or not, and based on that I have to take the next step to assign the value of a price to be 0 or say 20 rupees.

So, these are the places where we try to use it. And as I said earlier, that here we are dealing with the basic fundamentals, it is the very smaller thing, which are use in writing a bigger program. Now after this logical operators and truth table. Let us try to start a new topic. This is about the control structures. What you mean by control structure? What is control? Control is something, that there is an officer, and depending on the conditions and requirement. He tries to control the staff, and based on that he gives different types of instruction depending on the conditions.

So, he is also trying to make a conditional execution, how to do these things inside the R that, we now try to learn.



(Refer Slide Time: 16:10)



There are several aspects of these control structures in R. First thing is that there are certain control statements, there are certain loops, and based on that we try to define some function. If we will try to take up all these topics one by one, but today, we are going to deal with the conditional executions. Conditional execution means something has to be executed based on a condition whether, the condition is true or not.

So, there are several ways. First we try to understand the first simple approach. Now, when you say conditional, conditional means what, if something happens: then please take action number 1, and if something else is happening: please take action number 2. And this can be extended further. In case if there is a condition number 1: then take an action number 1, if there is condition number 2: take an action number 2. In case if there is condition number 3: then take action number 3.

For example, when we go to a doctor, In case, if the person has got a headache, then doctors take action 1 and it gives the medicine for headache. And in case if somebody has pain in the back, he take action number 2 and he gives the medicine for the backache, and soon. So, that is the basic idea of the conditional execution.

(Refer Slide Time: 17:54)

**1. Conditional execution**

**Syntax**

```
if (condition) {executes commands if condition is TRUE}
if (condition) {executes commands if condition is TRUE}
else { executes commands if condition is FALSE }
```

**Please note:**

- The condition in this control statement may not be vector valued and if so, only the first element of the vector is used.
- The condition may be a complex expression where the logical operators "and" (&&) and "or" (||) can be used.

7

So, in R, in case if I try to operate such a command, then first we have to understand the syntax. The syntax is very simple. First, we write here if all in a small letters and inside this brackets: we try to write down the condition. The condition under which, the action has to be taken.

And, then inside these curly brackets: I try to write the command, which has to be taken when this condition is true. And in case if there is another condition, then I will try to continue with writing here if, and then here I will try to write down here another condition, and here I will try to write down the command that has to be executed; when this condition is true. And this is how we will keep on continuing. And at the end, we will write the statement else else and then inside this curly bracket, we can write the command which have to be taken when the condition is false.

That means at the end you have to do this thing, right. Few, caution about this syntax. This control statement is more useful when we are not dealing with the vector valued. When we are trying to deal with the vector values, then there is a problem that this condition will be evaluated, only with respect to the first element of the vector. So, we might be assuming that, the control is moving from first element to second element, second element to third element and, soon, but this will not be happening.

And, the other good thing is this, this condition whatever we are writing here that may be a complex expression, where even we might be using the statements like double and or

say double bar and, there we can use such statements. Now can I try to take certain examples to understand this thing right?

(Refer Slide Time: 20:30)

**1. Conditional execution**

**Example**

```
> x <- 5
```

*x = 5 == exactly equal to*

```
> if ( x==3 ) { x <- x-1 } else { x <- 2*x }
```

*True* (under x==3)      *False* (under x <- 2\*x)

**Interpretation:**

- If  $x = 3$ , then execute  $x = x - 1$ .      *x = 3 then x = x - 1*
- If  $x \neq 3$ , then execute  $x = 2 * x$ .      *x ≠ 3 then x = 2x*

In this case,  $x = 5$ , so  $x \neq 3$ . Thus  $x = 2 * 5$       *x = 5, x ≠ 3*  
*x = 2 \* 5 = 10*

```
> x
```

```
[1] 10
```

Now choose  $x = 3$  and repeat this example

```
> x <- 5
```

```
> if ( x==3 ) { x <- x-1 } else { x <- 2*x }
```

The first example: I try to assign a variable  $x$ , a value  $x$  equal to 5. Now I am trying to take a very simple example, where there are only 2 conditions and here we can recall that, this double equality sign is the sign for exactly equal to.

Now if you try to see how I am writing. First I am writing if, then inside this bracket I am writing the here this condition. The condition here is,  $x$  is exactly equal to 3. Now I am saying if this condition is: suppose true, then you operate this statement. And if this condition is false: that is else, then operate this condition. So, now I have here one a statement which has to be executed in two different ways depending on the conditions. And these conditions you can see here they are enclosed under the curly bracket.

So, now let us try to see, what I am trying to see here; if  $x$  equal to 3, then we have to execute that  $x$  is equal to  $x$  minus 1. And if  $x$  is not equal to 3, then we are going to execute that  $x$  is equal to twice of  $x$  that is, what we want to do. Now, since I have taken here the value  $x$  equal to here five so; obviously, here in this case we can see that  $x$  is not equal to 3, and in this case  $x$  will be operated as 2 into 5, which is equal to here 10. And now if you try to do, here is the screenshot and here is the operator, where we want to where we can see here that this is the answer is coming out to be here 10.

And, similarly if I try to take x equal to 3 here then, I would request you to follow my this slide and try to execute it, the same statement and try to understand.

(Refer Slide Time: 23:09)

**1. Conditional execution**

**Example**

```
> x <- 3
> if ( x==3 ) { x <- x-1 } else { x <- 2*x }
```

R Console

```
> x <- 3
> if ( x==3 ) { x <- x-1 } else { x <- 2*x }
```

**Interpretation:**

- If  $x = 3$ , then execute  $x = x - 1$ .
- If  $x \neq 3$ , then execute  $x = 2*x$ .

In this case,  $x = 3$ , so  $x = 3 - 1$

R Console

```
> x
[1] 2
```

R Console

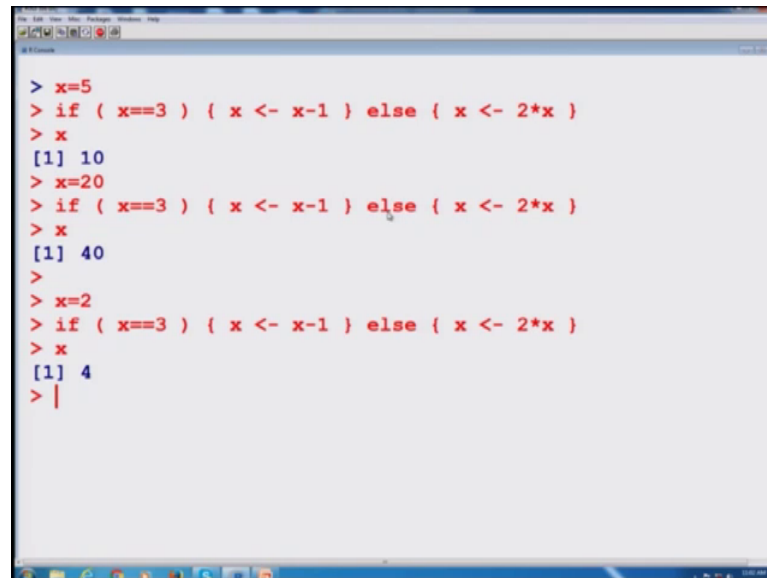
```
> x
[1] 2
```

9

That, if x is equal to 3, then operates this thing if x is not equal to 3, then operate this thing. And the outcome will come out to be here x equal to 2. Now let us try to operate this thing over the R console. Now, let us try to take x equal to 5 and I try to copy this condition over here. So, now, you can see that this condition has been executed over the value of x, and now, if I try to see the value of here x which comes out to be here 10 right.

So, now the value of x has been executed with the same value.

(Refer Slide Time: 23:31)



```
> x=5
> if ( x==3 ) { x <- x-1 } else { x <- 2*x }
> x
[1] 10
> x=20
> if ( x==3 ) { x <- x-1 } else { x <- 2*x }
> x
[1] 40
>
> x=2
> if ( x==3 ) { x <- x-1 } else { x <- 2*x }
> x
[1] 4
> |
```

Now, on the other hand, if I try to take here x equal to here say a 20, and then if I try to operate the same condition over here; we get here you can see here x equal to 40. And similarly if I try to take here x equal to here 2, and then if I try to operate this condition; then I get here x equal to 4. Please try to execute this statement yourself and try to see that whether are you getting the same thing and then try to understand what is really happening inside the R programming. I would like to stop here, and then on the next turn I will try to take some more example; on this conditional execution. And I would request you that please try to revise this lecture and try to understand, so, that in the next lecture when I continue you can understand it better; till then good bye.