

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

Basics of Calculations
Lecture - 19
Conditional Executions - If and If-Else

Hello friends, welcome to the course foundations of R Software. And in this lecture, we are going to begin with the new topic, this is about the Conditional Executions. The first question comes here what are these conditional executions. So, you know that suppose we take a call that in case a student gets more than 50 percent marks, then the student is going to be declared as pass and in case if the student gets marks less than 50 percent, then the student is going to be classified as fail. So, now in this case, how are you going to execute this command?

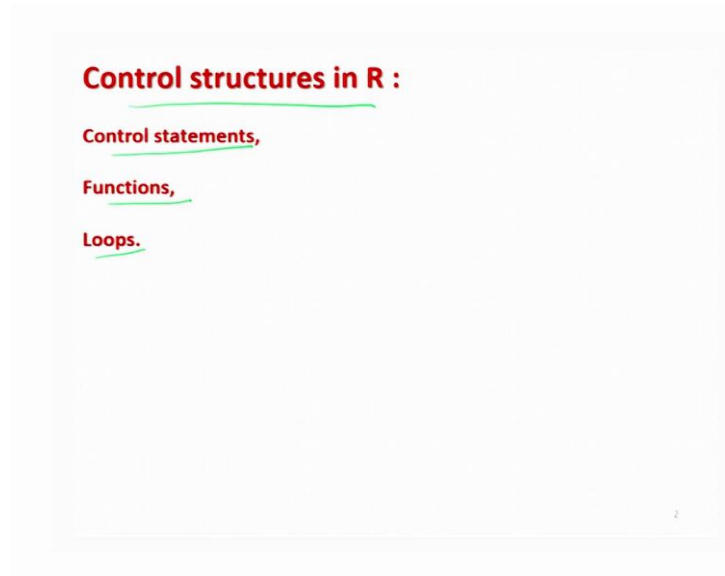
First, you want that some condition has to be checked. What is that condition? The condition is to check whether the student has got marks more than 50 percent or is smaller than 50 percent. Based on that you are going to take a call; what is going to be the outcome, whether pass or fail.

So similarly we have many more such conditions where the complication increases or the need of the situation changes. So, in order to fulfil those things, we have different kinds of conditional executions. And essentially, we are now beginning a topic on the control structure. Control structure means you are trying to control the control different parts of your programming.

So, the first part which we are going to talk about under the heading of, under the broad heading of control structure is the conditional execution. So, today in this lecture, we are going to talk about the if and if-else conditional execution commands, and we will try to see that how these conditional executions can be done in the R software. And in particular, if and if-else conditions, what are these conditions and how they have to be executed.

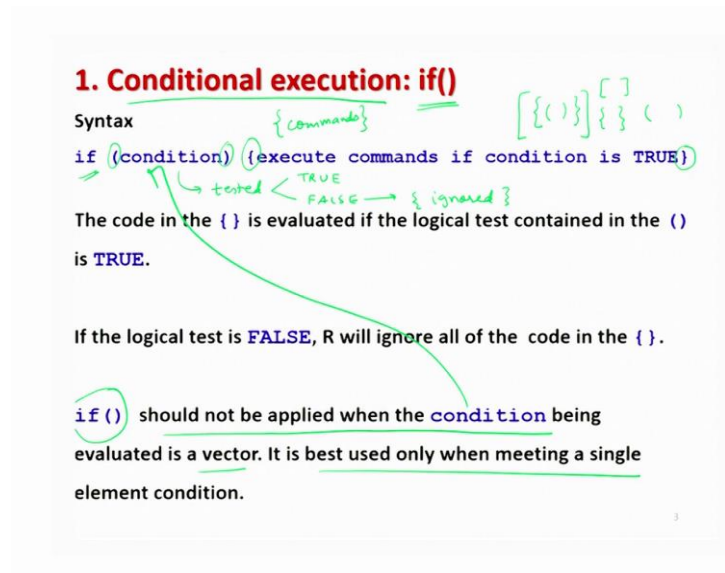
So, in order to explain you I will try to take here couple of examples, and through the good examples, I will try to show you that how these conditional executions are implemented in the R software. So, let us begin our lecture.

(Refer Slide Time: 02:36)



So, as I said, now we are going to begin with the topic of control structures in the R software. So, first we are going to discuss about the control statements, then functions, and after that loop. So, this will continue in the next couple of lectures, and we are going to cover all these topics one by one, right.

(Refer Slide Time: 02:55)



So, as I already have explained you that why do we need the conditional executions. The conditional executions are needed when we want to execute a command when certain conditions are TRUE or they are satisfied. Now, these conditions can be one or more

than one. So, based on that we have different types of contacts and commands in the R software and among them, we are going to discuss first the if condition, right. That is the popular name, that is how we try to address it.

So, the syntax for the conditional execution using if is like this. First, we try to write down here if 'if' then inside the parenthesis you can see here we try to write the condition which we want to test. So, this condition is to be tested. Now, this condition that is going to be a sort of logical statement, its outcome is going to be either TRUE or FALSE.

So, in case if the condition is TRUE, then whatever commands we want to execute that we have to write inside this curly bracket, right. So, I will write down here these brackets and here I will try to write down all the commands which are going to be executed when the condition is TRUE. And in case if the condition is FALSE, then whatever is written inside this curly bracket, all these conditions are ignored. So, this is how the if condition works in the case of R software.

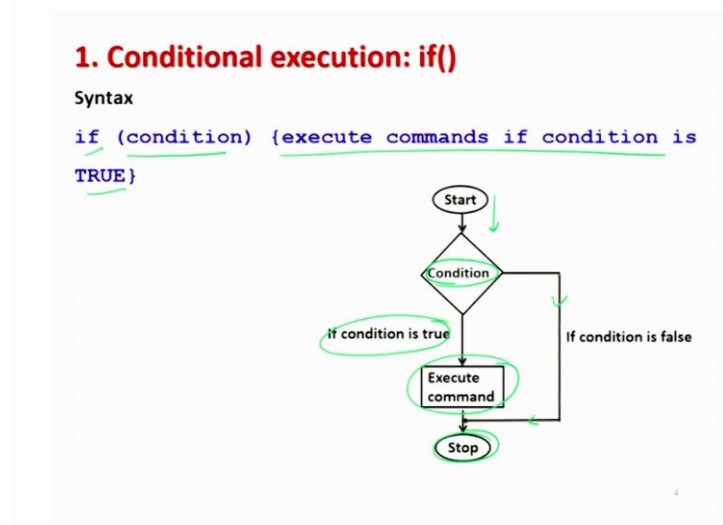
So, now, you can see here couple of things. Number 1, when you learned about the logical statements, now you can see one application of those logical statements that you want to first test and this outcome is going to be a logical variable in terms of TRUE and FALSE. So, for that you would also like to check whether the outcome of the condition is logical or not. So, all the commands which we have learnt in the earlier lectures, they are now going to be useful as we proceed further.

And after that I had discussed in the beginning that in mathematics, we have three types of brackets, this simple bracket, then curly bracket, and then square bracket. And I had explained you that in R software these brackets have got different roles. And in mathematics I had requested you to use only this parentheses, small brackets, these simple brackets.

And I had earlier explained you one utility of this square brackets, and now I am I have explained you here one utility of this curly brackets. If you recall at that time I had told you that in R software, this parenthesis curly and this square bracket they have got different types of utilities, and this is one of the utility where it is going to be used, right.

So, what you have to keep in mind that when you are trying to use this if condition, try to use it only when there is a scalar, and this if condition should not be applied when the condition whatever this condition here that we want to evaluate is a vector. So, this is best use only when we are trying to meet a single element conditions. So, what is this single element condition, I will try to show you that there is only one condition which has to be met and so on.

(Refer Slide Time: 06:40)



So, the basic syntax for this if condition is very simple, if then inside the parenthesis write down the condition and inside the curly bracket you try to write down the statements or the commands which need to be executed. In case if you try to look at the flow chart of this if conditional execution, it goes like this. That you start you try to test here the condition, now there are two options, the condition is TRUE or the condition is FALSE.

In case if the condition is FALSE here like this, then we come and we stop here. And in case if the condition is TRUE then we try to execute the commands and then the program is stopped. So, that is how the if condition works.

(Refer Slide Time: 07:31)

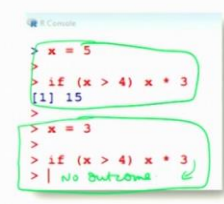
1. Conditional execution: if()

Example 1:

Suppose we want to multiply all the values $x > 4$ by 3.

$x > 4 \rightarrow \text{TRUE} \rightarrow 3 \times x$
 $\rightarrow \text{FALSE} \rightarrow$ Nothing will happen

```
> x = 5 >4 → True → 3x5  
> if (x > 4) { x * 3 }  
[1] 15  
> x = 3 >4 → FALSE  
> if (x > 4) x * 3  
No response is obtained.
```



So, now let me try to take here some example and try to show you that how it works. Suppose, we want to multiply all the values in some program which are greater than 4 and we want to multiply such values by 3, right. So, somebody will enter some value. The condition whether the value is greater than 4 or smaller than 4, that is going to be checked.

And in case if this condition that x is or the number what you have entered is greater than 4, if this is found to be TRUE, then whatever is the number this is going to be multiplied by 3. And if this is FALSE, the input number is not greater than 4 then nothing will happen, right. So, that is the thing which we want to do it and you can see here now that using the conditional execution concepts and using the if commands we can do it very easily.

So, suppose I give here an input x is equal to 5 and then I try to write down if condition here like this if, and inside the parenthesis you try to write down here x greater than 4 and after that you write here $x * 3$; that means, the number is going to be multiplied by 3, right. Here in this case, either you write here curly bracket or not this will not make much difference because there is only one condition one an element. But later on you will see that when we are trying to increase such conditions, then this curly bracket will be necessary.

So, here on the R console, I will try to show you the application of actually both. So, in this case since you have taken the value to be 5 and this 5 is greater than 4. So, the condition is here TRUE and so this number is multiplied by 5, and you get here 3 into 5 as 15. Now, in case if you try to take here another number here x equal to 3, then what will happen? 3 is less than 4, this is FALSE.

So, now in this case, no outcome will be obtained. If you try to execute this condition, if greater than 4 and then x into 3 no response will be obtained here. So, you can see here when I am trying to take it x equal to 5, then I get this outcome, but when I try to take here x equal to 3, then you can see here as soon as I enter on the R console there is no outcome, right. So, let me try to show you.

(Refer Slide Time: 10:07)

1. Conditional execution: if()

Example 1: Same with using {}

Suppose we want to multiply all the values $x > 4$ by 3.

```
> x = 5
> if (x > 4) {x * 3}
[1] 15
> x = 3
> if (x > 4) {x * 3}
No response is obtained.
```

The slide also shows a screenshot of an R console window with the following output:

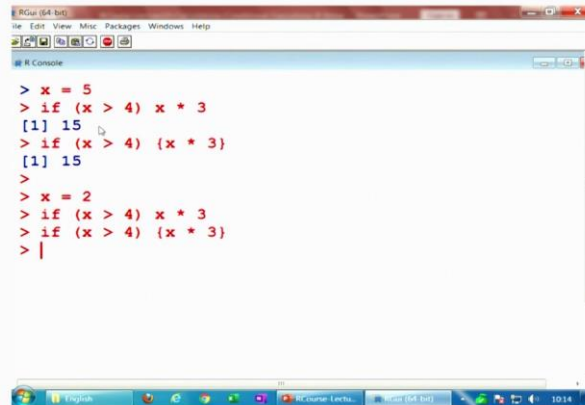
```
> x = 5
> x
[1] 5
> if (x > 4) {x * 3}
[1] 15
>
> x = 3
> if (x > 4) {x * 3}
> |
```

Green circles and arrows highlight the curly brackets in the code and the corresponding output values (15) in the console window.

And now I would like to show you the same example curly bracket for this execution, so that you do not get confused, right. So, you can see here that this is the same program I take x equal to 5, this is the same condition if x is greater than 4, but now I use here these curly brackets. And you can see here this gives you the same outcome.

And similarly, if you try to take here this x equal to 3 and if you use the same condition here, but now I am using here this curly brackets, then it is giving you no response. You can see here this is the same condition, but here I have used the curly bracket and in the earlier example I had not used here the curly brackets, right. So, as I said it will not make much difference as long as you are working with the scalar single conditions. Now, let me try to show you this execution on the R console, so that you get more confident.

(Refer Slide Time: 11:06)



```
RGui (64-bit)
File Edit View Misc Packages Windows Help

R Console

> x = 5
> if (x > 4) x + 3
[1] 15
> if (x > 4) {x + 3}
[1] 15
>
> x = 2
> if (x > 4) x + 3
> if (x > 4) {x + 3}
> |
```

So, you see when you are trying to execute such statements on the R console directly, then first you have to give the value of the variable. So, for example, if I say here x equal to 5 and then I have to give here this condition, and as soon as I give this condition you can see here that this condition is going to give you the value 15, right.

And here in case if you try to take here the same condition, but if you try to now put here the curly brackets you will see here you get the same outcome. But in case if you try to take here x equal to suppose here 2, then you can see here; what happens to this condition? This will give you no output because there is nothing, because the condition is FALSE, so there is no outcome. And even if you try to write down here these curly brackets also, this will not give you anything. So, this is what I meant when I was trying to explain you on this example.

(Refer Slide Time: 12:22)

1. Conditional execution: if()

Example 2:

Suppose we want to print if a value is more than 3.

```
x = 6 > 3 → TRUE
if (x > 3) {
  print("The value is more than 3")
}
```

[1] "The value is more than 3"

```
x = 6
> if(x > 3)
+ print("The value is more than 3")
+ }
[1] "The value is more than 3"

x = 2
> if(x > 3)
+ print("The value is more than 3")
+ }
[1] no output
```

So, now let us try to take one more example and try to see what happens. So, you know there is a command here print. So, print is used to print something on the screen, right. So, well, we are going to discuss about different types of such options in the forthcoming lectures, but here I would like to show you something which is based on the use of print. So, now I want to print a value, in case if the value is more than 3. So, suppose I give here the value x equal to 6 and then I try to write down this requirement in the form of R contacts. So, I try to give here the condition.

The condition here is x greater than 3. So, it is if inside the parenthesis x greater than 3, and now in this curly bracket I am trying to write down the condition here. So, the condition here is print and whenever we are trying to print something which is a scalar we try to give it inside the double quotes. So, I try to write down here within double quotes the value is more than 3 and I try to include it inside the parenthesis.

So, now if you try to execute it because the value of x is 6. So, 6 is greater than 3, this condition is TRUE. So, whatever is there inside the curly bracket that will be executed and you get here this outcome. So, you can see here that if you try to take here x equal to 6 then it gives you here this outcome, but if you try to take here x equal to 2, then it will give you here no outcome. Why? Because 2 is greater than 3; no, this condition is FALSE, right.

(Refer Slide Time: 14:17)

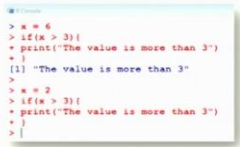
1. Conditional execution: if()

Example 3:

Suppose we want to print if a value is more than 3.

```
x = 2 > 3 → FALSE
if(x > 3){
  print("The value is more than 3")
}
```

No outcome is obtained.

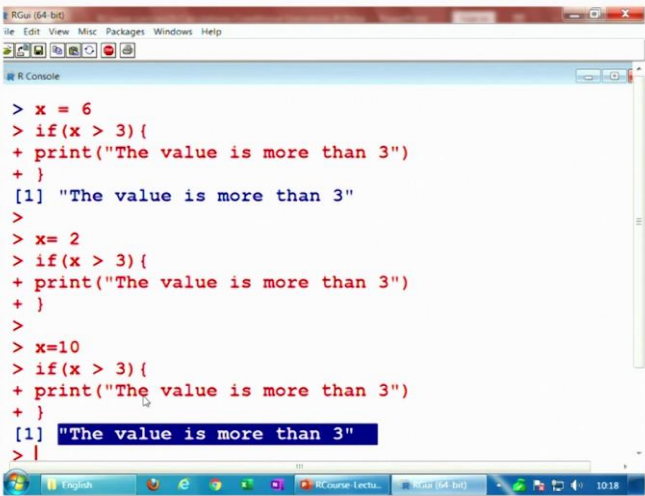


```
> x = 6
> if(x > 3){
+ print("The value is more than 3")
+ }
[1] "The value is more than 3"
>
> x = 2
> if(x > 3){
+ print("The value is more than 3")
+ }
> |
```

8

So, yeah, that is the same condition I want to show you here. So, let me try to show you here what happened with x equal to 2. This 2 is greater than, 3 this condition here is FALSE. And so this condition is FALSE, so whatever is written inside this curly brackets this will not be executed. So, if you try to execute it here, no outcome is going to be obtained. So, let us try to do this example on the R console and then we try to see what happens, ok. So, let me clear the screen and you know the command here is control l.

(Refer Slide Time: 14:52)



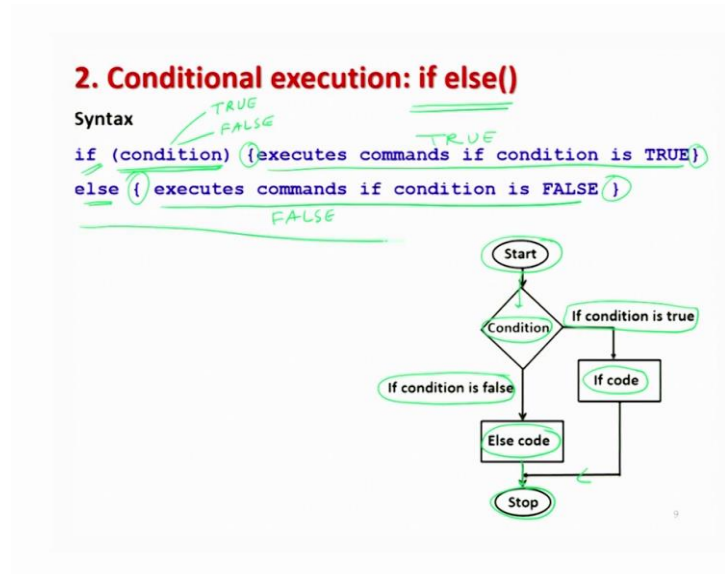
```
RGui (64-bit)
File Edit View Misc Packages Windows Help
R Console

> x = 6
> if(x > 3){
+ print("The value is more than 3")
+ }
[1] "The value is more than 3"
>
> x= 2
> if(x > 3){
+ print("The value is more than 3")
+ }
>
> x=10
> if(x > 3){
+ print("The value is more than 3")
+ }
[1] "The value is more than 3"
>
> |
```

So, now, if I try to take here some value here x equal to suppose 6, and then I try to give this statement like is here, you can see here the outcome is the value is more than 3. But on the other hand in case if I try to take here some value here 2, and if I try to repeat the same thing here, you can see here there is no outcome here, no outcome. But in case, if you just change it x to be here 10, and then if you try to repeat the command, you can see here the outcome comes here once again the value is more than 3, right, yeah. It is also possible that you can also write what is the value of 3.

So, these things we are going to learn in the forthcoming lectures that you can write that here that the value is more than 3, that is ok, but what is the value which you are trying to compare, so you can also print the value of x. But that we leave it at the moment for the future lectures, right.

(Refer Slide Time: 15:54)



So, now you have understood that how this if condition is going to work. And but now you have seen here that in this case you have a constraint that if the condition is TRUE only then whatever the condition you want that is executed. But in case if the condition is FALSE, then nothing is happening.

But now in case if your wish or your requirement is that if the condition is TRUE, then something should be printed or something should be executed and if the condition is FALSE then something else should be computed. For example, the example which I took in the beginning that if a student has got more than 50 percent marks then it should be for example, printed that the student has passed, and if the student has received the marks less than 50 percent, then it should be printed that the student is failed like this.

So, these type of conditions can be executed using the conditional execution with the help of the concept of if-else conditions. So, as the name suggests if-else, if it is TRUE then something should be executed and else, that means, if it is not TRUE then something else should be computed.

So, if you try to understand this situation by the flow chart of this if-else execution, then it goes like this. You start with something, then there is a condition which is here checked. Now, there are two options that the condition is TRUE or the condition is FALSE. In case if the condition is TRUE, then it will work just like your if condition and

whatever is the condition or the code you have written that is going to be executed and after executing it the program will stop.

But in case if the condition is FALSE, then in the earlier case there was no output, but now in this case you are going to write down here some more here code, that is called as else code. Means when the condition is FALSE then under else these syntax are going to be executed, and then after its execution the program will stop. So, this is how this if-else condition works.

So, in R the syntax for this if-else condition is like as follows. You try to write down here if or in lower case alphabets and then inside the parenthesis you try to write down the condition. Now, this condition is going to be either TRUE or this condition is going to be here FALSE. In case if the condition is TRUE, then whatever you want to execute that has to be written inside the curly bracket just after this.

So, this will go with TRUE condition. So, there can be commands here, they will be executed. And in case if the condition is FALSE, then what will happen? So, after that you write here else, e l s e all in lower case alphabets and then within the curly brackets you try to write down the commands which you want to execute when this condition is FALSE. So, that is a very simple thing, that is a sort of extension of the if condition that you are adding here one more here syntax for the else.

(Refer Slide Time: 19:42)

2. Conditional execution: if else()

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.
- `if else()` should not be applied when the condition being evaluated is a vector. It is best used only when meeting a single element condition.
- The condition may be a complex expression where the logical operators "and" (`&&`) and "or" (`||`) can be used.

10

So, in this if-else condition you have to be little bit watchful that the condition in the control statement may not be vector valued. And in this case only the first element of the vector is used, right. And means ideally if-else condition should not be applied when the condition which we want to judge or which we want to evaluate in terms of TRUE and FALSE is a vector.

And this if-else condition is best used when you are trying to have a single element, right. And when you are trying to use this condition means you want to judge whether this is TRUE or this is FALSE, this condition may be a simple expression or any complex expression where you can use all your logical operator like “and” like say “or” or, right. Whatever logical operators we have studied they can be used here in defining the conditions in the proper way, ok.

(Refer Slide Time: 20:54)

2. Conditional execution: if else()

Example 4:

```
x = 5
if ( x==3 ) { x = x-1 } else { x = 2*x }
```

Handwritten notes:
 $x = x - 1$
 Replace x by $x-1$
 $x = x-1$
 $x = 2x$
 Replace x by $2x$
 $x = 2x$
 $1 = 1$

x
 [1] 10

Interpretation:

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

Handwritten notes:
 $f(x) = \begin{cases} x-1 & \text{if } x=3 \\ 2x & \text{otherwise } x \neq 3 \end{cases}$
 $x=3$

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

```
> x = 5
> if ( x==3 ) { x = x-1 } else { x = 2*x }
> x
[1] 10
```

So, now let me try to take here one example and try to show you how the things are going to work and the more important part which I always say is that you have to understand what and how R is going to execute it. So, now, suppose I try to write down here a function like this, means I can write down here this function takes value x minus 1 if x is equal to 3, and it takes value $2x$ otherwise. So, that is the most simple language of mathematics in which we study.

So, now in case if you want to understand, what are you going to do? You are trying to say that in case if the input here is equal to 3, then this has to be executed that is x minus

1. And in case if number is something else, that x is not equal to 3 like this, then whatever the number is there that has to be multiplied by 2

So, now in case, if you want to write this statement then what you have to do? The first thing is this when you try to see here; what about this? x equal to 3 which I am writing here you are trying to say that if x is equal to 3 then execute x as x minus 1 and in case if x is not equal to 3 then execute x as twice of x . What is this x ? This is an outcome that will be obtained after giving the input value.

And after this if you try to see this equality operator, this is mathematical? No, this is logical. So, that is why when you try to write down the condition, you have to use the logical operator which consists of two equal to sign and that is indicating the exactly equal to sign under the logical operations. So, now if you try to see here how are we going to execute it; so, the inputs here suppose is x equal to 5 and now I write down here if and inside this parenthesis I try to write here x equal to 3 that is exactly equal to using the double equality operator.

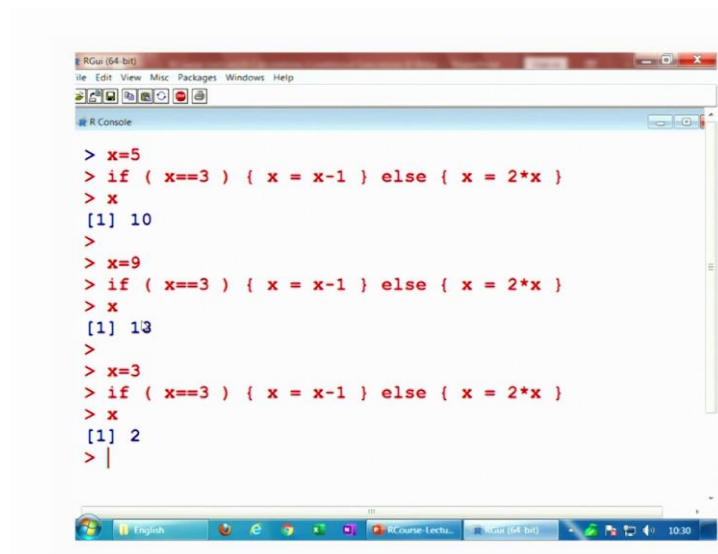
And now I am saying that in this case x should be replaced by x minus 1. So, I try to write down here this curly brackets, and within this curly bracket I write here x equal to x minus 1. And in case it is not TRUE; that means, x is not equal to TRUE, then I try to write down here 1 and then within the curly bracket I write to I write down here x is equal to twice of x .

One thing I would like to have your attention here I mean and my advice is do not misunderstand this statement. For example, here I am trying to write down here x is equal to x minus 1, please understand this is not a mathematical operation, but we are trying to say is that replace x by x minus 1. And some people may write it, think it mathematically that x is equal to x minus 1 and they try to cancel x , it will be simply rubbish, it has no meaning.

And similarly if you try to look at here, this is here x is equal to twice of x , this means replace x by twice x . This is the meaning it is not like that you write x is equal to $2x$ and then try to cancel x and x , and then try to prove that one is equal to 2 which is wrong, right. So, this is also wrong, this is also wrong. So, please do not misunderstand this expression, ok.

So, now, if you try to see here what will be the output; so, if I try to take here x equal to 5 then in this case x is not equal to 3 and so x will be operated as 2, 2 into x. So, the value will come out to be here 10. And you can see here this is here the 10, and this value here is coming out to be here 10, right. So, let me try to execute this expression first on the R console and then I will try to move further. So, let me try to copy this condition, so that I can save some time.

(Refer Slide Time: 25:41)



```
RGui (64-bit)
file Edit View Misc Packages Windows Help
R Console
> x=5
> if ( x==3 ) { x = x-1 } else { x = 2*x }
> x
[1] 10
>
> x=9
> if ( x==3 ) { x = x-1 } else { x = 2*x }
> x
[1] 18
>
> x=3
> if ( x==3 ) { x = x-1 } else { x = 2*x }
> x
[1] 2
> |
```

And now if I try to take here x equal to here 5, yeah, remember first you have to give the value of x as input and then I try to write down this condition and then I try to write down the value of here x. So, now you can see here you had given the value of x to be as 5, but now the value of x is transformed to 10. Similarly, if you try to take it here x equal to here 9, so what will happen here? This condition is going to be operated once again and then if you try to print out the value of here x this is here 18, that is 2 into 9 is 18.

But now I try to take it here x equal to 3 and then if you try to repeat this condition. And if you try to see, what happened to the value of x? This becomes here x minus 1, that is 3 minus 1 which is equal to 2, right. So, this is how you have to operate with this if-else conditions, right.

(Refer Slide Time: 26:38)

2. Conditional execution: if else()

Example 5:


```
x = 3
if ( x==3 ) { x = x-1 } else { x = 2*x }
x
[1] 2
```

Handwritten notes: "TRUE" with an arrow pointing to the condition `x==3`. `x = 3 - 1 = 2` with an arrow pointing to the assignment `x = x-1`.

Interpretation:

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

In this case, $x = 3$. Thus $x = 3 - 1 = 2$



12

So, exactly this is the same thing which I just explain you on the R console, that you try to take here x equal to 3 and then if you try to operate because x equal to 3 is now TRUE. So, this is going to be operated. And x will be replaced by 3 minus 1 is equal to 2 and then you get here an outcome 2, right, ok.

(Refer Slide Time: 27:01)

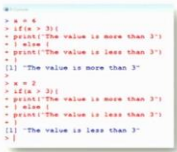
2. Conditional execution: if else()

Example 6:

Suppose we want to print if a value is more than 3 or less than 3.

```
x = 6
if (x > 3) {
  print("The value is more than 3")
} else {
  print("The value is less than 3")
}
[1] "The value is more than 3"
```

Handwritten notes: `x = 6` is circled. `> 3` is annotated with `T` and `F`. A diagram shows `x < 3` leading to `T` (most than 3) and `F` (less than 3). `> 3` is annotated with `T` (most than 3) and `F` (less than 3). The `else` block is circled.



13

So, now I try to take here one more example. This is the same example or a sort of extension of the earlier example which we consider, that we want to print if a value is more than 3 or less than 3. So, earlier we had printed the value is more than 3, but what if the value is smaller than 3 then there was no outcome, right. So, now I try to extend it and I try to show you here how you can do. So, you can see, here you have a two

condition that you try to give here an input x and this condition is that that x is going to be checked whether this value is more than 3 or smaller than 3.

And based on that there will be an outcome TRUE or FALSE, and based on that there will be two outcome. If the condition is 2, then you have to print. The value is more than 3, right like more than 3 and if the value is here FALSE, the outcome here is FALSE, then you have to print here less than 3, ok. So, now how to get it done?

So, I try to write down here this statement if inside the parenthesis x greater than 3 and then within this curly bracket, first set of curly brackets I write here print and then within the parenthesis and within the double quotes I write down here whatever I want to type, the value is more than 3. And if this condition is FALSE, because this x greater than 3 will have here two outcomes one is here TRUE.

So, when this TRUE is there this is going to be operated, and if this outcome here is FALSE then I have to write down here else, and after that I have to write another pair of curly brackets and I have to write here what we want to do. So, here I want to print the value is less than 3. So, I try to use here this parenthesis and I try to write down the statement within the double quotes.

So, now, in case if you consider here this x equal here 6 because 6 is greater than 3, this condition is TRUE. So, whatever is the condition under the TRUE part that the value is more than 3, that is going to be printed here, right.

(Refer Slide Time: 29:23)

2. Conditional execution: if else()

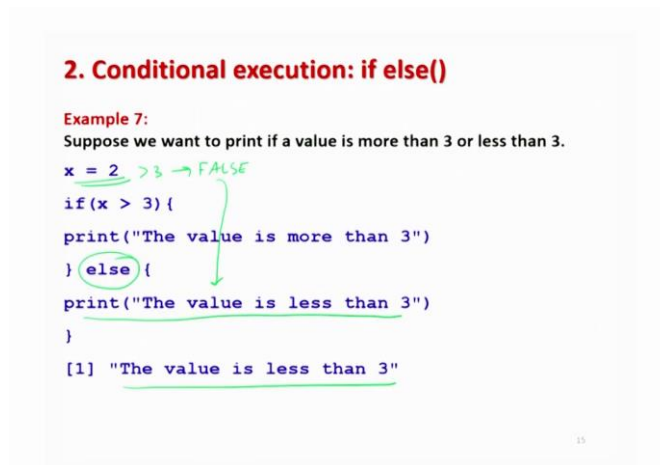
Example 6:

```
> x = 6
> if(x > 3){
+ print("The value is more than 3")
+ } else {
+ print("The value is less than 3")
+ }
[1] "The value is more than 3"
>
> x = 2
> if(x > 3){
+ print("The value is more than 3")
+ } else {
+ print("The value is less than 3")
+ }
[1] "The value is less than 3"
> |
```

14

And similarly, if you try to take here say here one more value here that x equal to here 2, then what will happen? 2 is greater than 3 this condition here is FALSE. When this condition is FALSE, then you are going to print whatever is written after the else command and it is here print the value is less than 3, right. So, you can see here the outcome will come here like this. And you can see here this is here the screenshot of both these operations, but let me try to show you these operations on the R console to make you more confident, right.

(Refer Slide Time: 29:26)



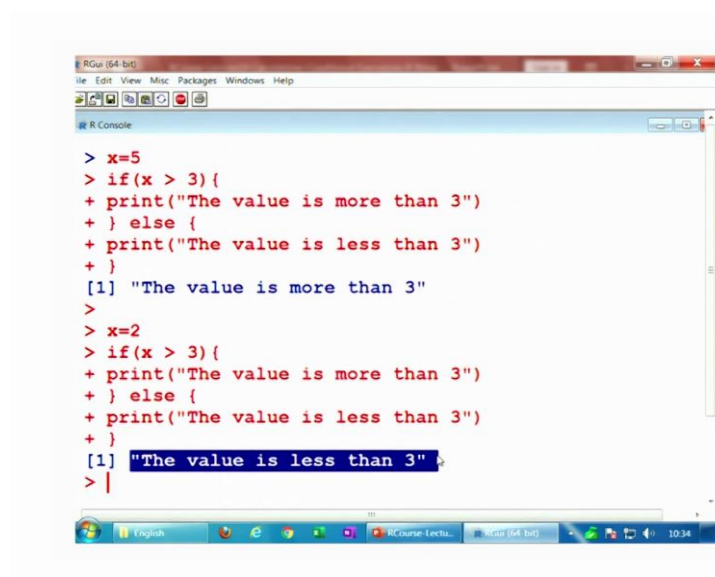
2. Conditional execution: if else()

Example 7:
Suppose we want to print if a value is more than 3 or less than 3.

```
x = 2 > 3 → FALSE
if(x > 3) {
  print("The value is more than 3")
} else {
  print("The value is less than 3")
}
[1] "The value is less than 3"
```

So, let me try to copy this command, so that I can save some time.

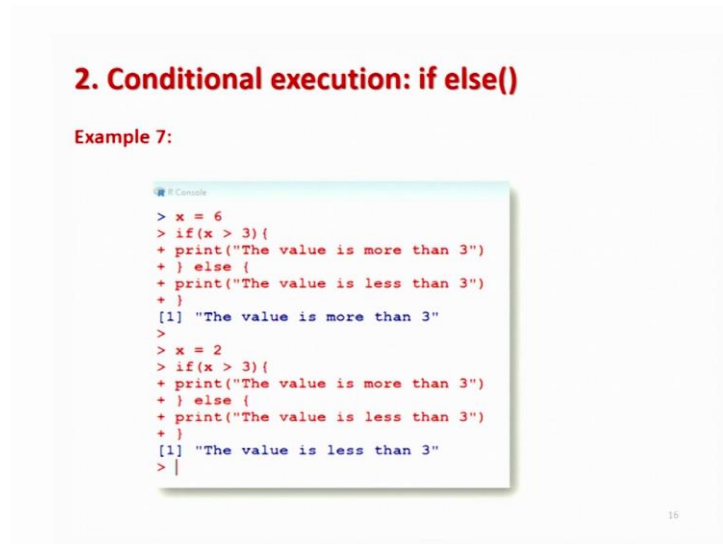
(Refer Slide Time: 30:07)



```
> x=5
> if(x > 3) {
+ print("The value is more than 3")
+ } else {
+ print("The value is less than 3")
+ }
[1] "The value is more than 3"
>
> x=2
> if(x > 3) {
+ print("The value is more than 3")
+ } else {
+ print("The value is less than 3")
+ }
[1] "The value is less than 3"
> |
```

And I try to take it suppose here x equal to 5, and if I press this command you can see here it is printed here the value is more than 3. And in case if I try to take here x equal to 2, then you can see here it is printed here the value is less than 3. So, you can see here it is not a very difficult command here; and yeah. So, this is the screenshot.

(Refer Slide Time: 30:32)



2. Conditional execution: if else()

Example 7:

```
# Console
> x = 6
> if(x > 3){
+ print("The value is more than 3")
+ } else {
+ print("The value is less than 3")
+ }
[1] "The value is more than 3"
>
> x = 2
> if(x > 3){
+ print("The value is more than 3")
+ } else {
+ print("The value is less than 3")
+ }
[1] "The value is less than 3"
> |
```

16

So, now, we come to an end to this lecture. And you can see here I have explained you two types of conditional execution, one is using the condition if and another is the if-else. So, these if and if-else, now you can very clearly understand they have got different types of role. The if condition is going to be used only when you have one condition, that if the condition is TRUE then this has to be executed.

But in case if you have suppose two outcome, and you want that in case of the if condition is TRUE, then this should be done, and if the if condition is FALSE then something else should be done. Then you have to use the if-else condition. And believe me these are very useful conditions because whenever you are trying to write down the programs basically in statistics at least I know, that you have these types of condition that this has to be evaluated only when the condition is True.

For example, many times you want to evaluate mathematical expression only in case of the value of that expression is greater than 0, right. Suppose, you want to find out the square root of some complicated expression. And when you are trying to evaluate it you

would like to find out the square root of that the outcome only when the outcome is positive, and suppose the outcome can be positive or outcome can be negative.

So, in this case this type of conditional execution help us and we always say please try to check this condition, if this condition is greater than 0 only then you try to find out the square root. So, these are the ingredients of a program. So, now, my request to all of you is that you try to take some example. Try to create the example yourself because when you are trying to create an example yourself you also know the answer.

And then you try to see that when you are trying to execute the same concept in the R software, are you getting the same outcome. And that will give you more confidence and after that you will become a good programmer. So, you try to practice it, and I will see you in the next lecture with more details on the conditional executions.

Till then goodbye.