# Foundations of R Software

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

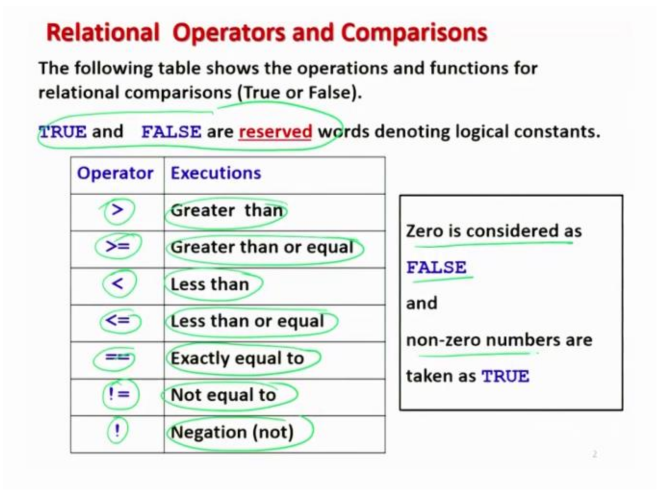
## Basics of Calculations
## Lecture - 17
## Relational and Logical Operators

Hello friends, welcome to the course Foundations of R Software and you may recall that in the last lecture, we started a discussion on the logical operators and we had understood that what are the different types of logical operators and how do they work. Now, in this lecture also, we will continue with the Logical Operators and Relational Operators and we will try to see how do they work. And we will try to investigate some more applications of these numbers and we will try to learn some more operations.

For example, in case if you come to know that you have got some variable. Now, you want to know, whether this variable is a logical variable or not. Because unless and until you would know that this variable is logical, you cannot make the logical operations. So, in this lecture we are just going to learn about these type of very simple observations and we will try to see how do they work in the R software.

So, we begin our lecture and try to understand and before that I will give you a quick review of all the operation that we had learnt in the last lecture. So, we are going to talk here about the relational and logical operators in this lecture.

(Refer Slide Time: 01:34)



### Relational Operators and Comparisons

The following table shows the operations and functions for relational comparisons (True or False).

TRUE and FALSE are reserved words denoting logical constants.

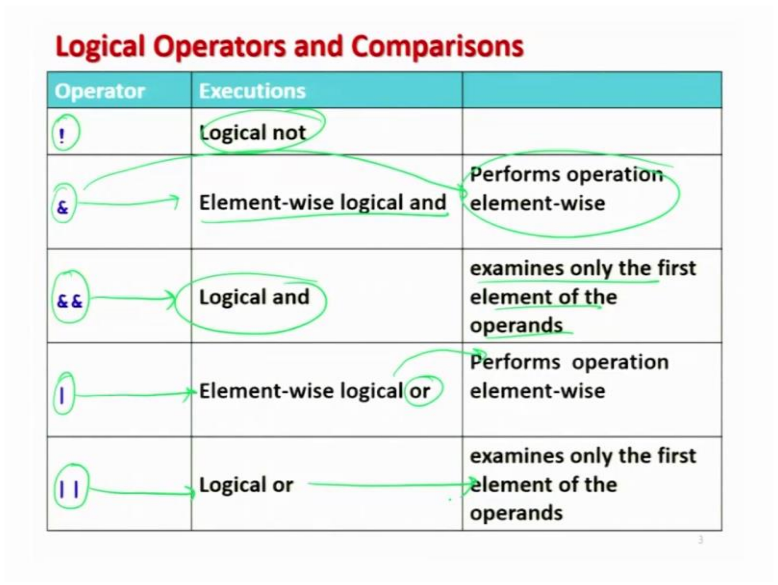
| Operator | Executions |
|---|---|
| > | Greater than |
| >= | Greater than or equal |
| < | Less than |
| <= | Less than or equal |
| == | Exactly equal to |
| != | Not equal to |
| ! | Negation (not) |

Zero is considered as FALSE and non-zero numbers are taken as TRUE

So, you had learnt in the last lectures couple of logical operators and relational operators like as greater than which is indicated by greater than sign, greater than or equal which is indicated by greater than and equality sign, less than which is indicated by less than sign, less than or equal to which is indicated by less than and equal to sign, exactly equal to which is indicated by two equality operators, not equal to which is indicated by an exclamation sign and equality operator, and negation is indicated by exclamatory sign like this.

And one thing you have to just keep in mind that TRUE and FALSE, these are the two reserved words and they are the logical constant. And zero is considered as FALSE and non zero numbers are taken as TRUE, ok.

(Refer Slide Time: 02:32)



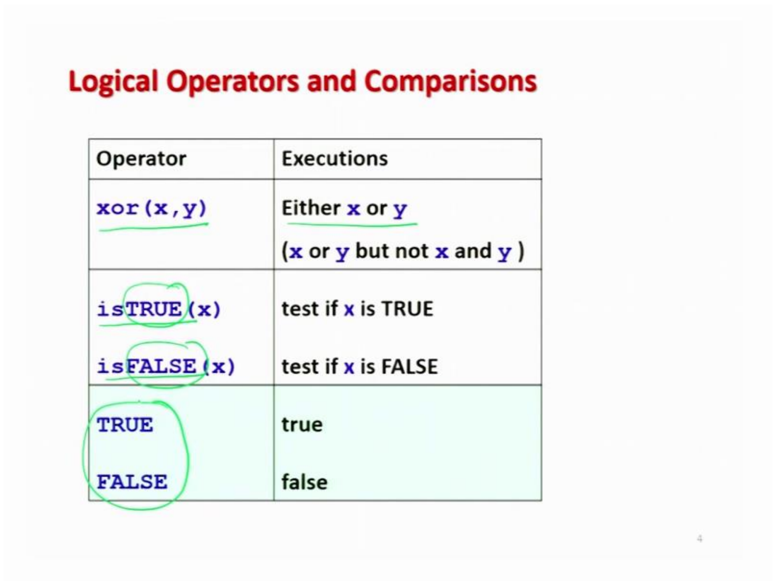
## Logical Operators and Comparisons

| Operator | Executions | |
|---|---|---|
| ! | Logical not | |
| & | Element-wise logical and | Performs operation element-wise |
| && | Logical and | examines only the first element of the operands |
| \| | Element-wise logical or | Performs operation element-wise |
| \|\| | Logical or | examines only the first element of the operands |

So, now and then you also had learnt about the & and or operator and we had seen that this exclamation sign this is a logical not and when we are trying to use this and operate then we have two options one is to use the single sign or single & or we try to use double &. So, when we are trying to use the single & then it gives element wise logical operations related to and operator and when we are trying to use double & sign then this is a logical and executes only the first element in the data vector whereas, this single & works for the entire data vector element wise.

And similarly this or operator which is indicated by this vertical line, single vertical line or double vertical lines. So, they are the logical operator for the operation or and this single operator gives you element wise logical operation and it works element wise; that means, for the entire data vector whereas, if you try to use double vertical lines, this is also logical or but it works only on the first element of your data vector.

(Refer Slide Time: 03:47)

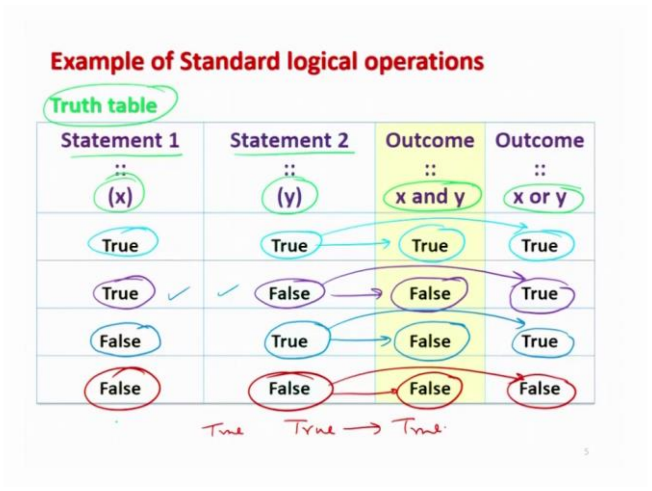


## Logical Operators and Comparisons

| Operator | Executions |
|---|---|
| `xor(x,y)` | Either **x** or **y**<br>(**x** or **y** but not **x** and **y** ) |
| `isTRUE(x)` | test if **x** is TRUE |
| `isFALSE(x)` | test if **x** is FALSE |
| `TRUE` | true |
| `FALSE` | false |

And after that you also had learnt about the one more operator like as xor and inside the parenthesis x and y. So, this is giving you either x or y or type of logical operations and then you had learn isTRUE and isFALSE to know, whether a statement x is TRUE or FALSE respectively. And here you have to notice that this TRUE and FALSE in these two statement that has to be given in the capital letters and then you had learnt about the TRUE and FALSE which are the simply true and false.

(Refer Slide Time: 04:22)



## 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 |

True   True → True

Now, based on it I try to give you the idea of the truth table. Truth table for example, if you try to see we had made a different types of operation in the last lecture and sometime I was saying that ok, true and true is true and false and false is false true or false etc. So, now, this is the table which is called as truth table and this gives us an idea that when we are trying to operate with the true and false using the & and or operator then what will be the outcome.

So, suppose I consider here two statements statement 1 and statement 2, which are indicated by x and y, respectively. Now, I try to consider here two logical operations-one is here x and y and another is here x or y. So, these are the two outcome that we are going to consider.

So, in case if you try to take here the first statement whose outcome is true and the second statement whose outcome is true then in case if you try to operate here x and y; that means, true and true this is going to be true and in case if you try to consider the outcome x or y that is true or true then this outcome is going to be here true.

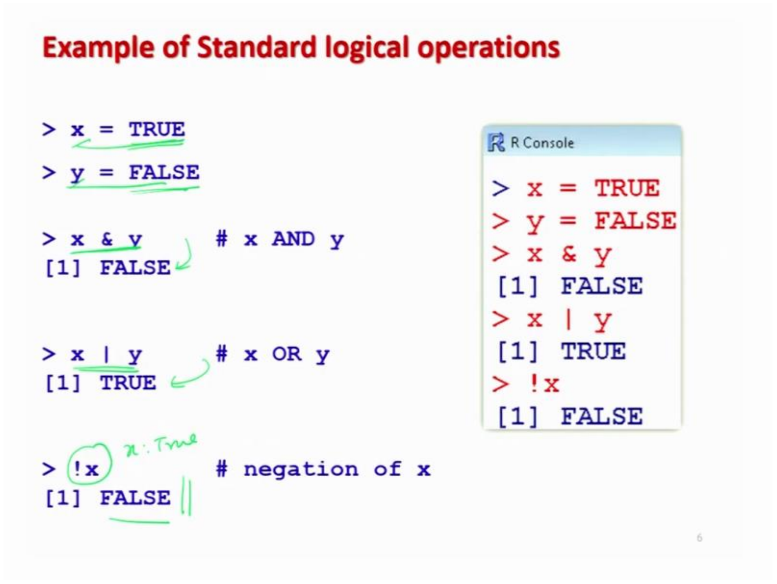
Similarly, in case if you try to consider one more statement whose outcome is true and the next statement whose outcome is false and in case if you are interested in considering the outcome like x and y then its outcome is going to be false. And in case if you are interested in an outcome like x or y, that is true or false then this outcome is going to be true.

Similarly, in case if you try to consider here one more operation in which the statement 1 has got an outcome which is false. A statement 2 has got an outcome which is true. So, this is just like the earlier case like as here. So, in that case what will happen that if we are interested in the outcome like x and y, then this is going to be false and if we are interested in the outcome like x or y, then the outcome is going to be true.

And finally, in case if the outcome of both the statement is false, then the outcome of the statement x and y will also be false and the outcome of the statement x or y this will also be false like this. So, that is why now you can imagine that in the earlier lecture when I was trying to take two statement and sometime I was saying that ok, this is true and another statement this is true and then I used to say that their operation true and true this is going to be true like this.

So, this all those things were coming from the truth table and I had promised you the last lecture that I will explain you about the concept of truth table, but now since you have understood the application. So, it is not difficult for you to understand the application of this truth table.

(Refer Slide Time: 07:56)



## 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
```

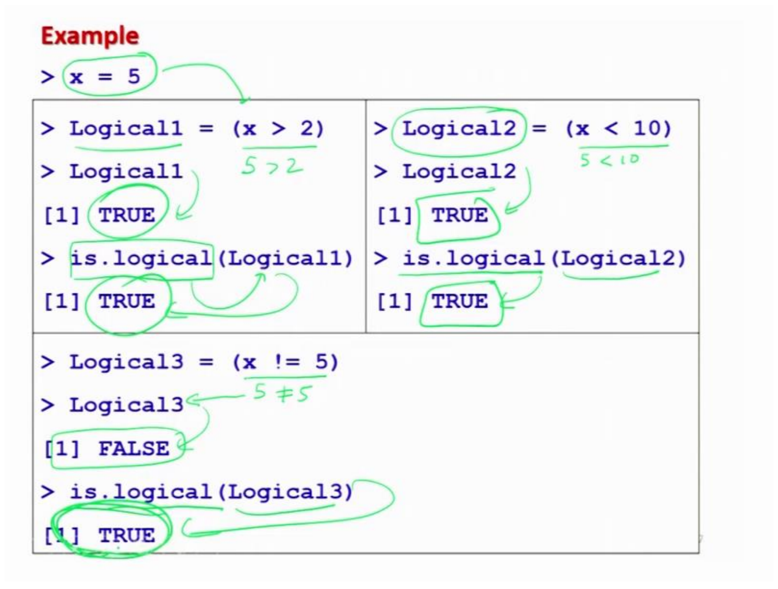
```
> x = TRUE
> y = FALSE
> x & y
[1] FALSE
> x | y
[1] TRUE
> !x
[1] FALSE
```

Now, I try to give you here some examples so that I can explain you what is the meaning and utility of the outcomes in this truth table. Suppose I try to take here two variables x and y and I take x as TRUE and y here as a FALSE. Now, I try to do here different type of this operation from the truth table.

So, if you try to see here I am trying to take here for example, case number here three one is true another here is false and then I want to operate here these two statements x and y and  x or y. So, you can see that x and y statement is here false and x or y will be true.

So, if you try to see here when x is TRUE, y is FALSE then x and y, it is giving you an outcome FALSE. And x or y which is here like this, this is giving you here an outcome TRUE. And in case if you try to consider here like as negation of x then you have to write down exclamatory sign and x. So, since x here is TRUE so, the negation of x is FALSE. So, you can see here this is here the outcome. So, you can see here this is how the truth table works and its concept are used, right.

(Refer Slide Time: 09:30)



**Example**

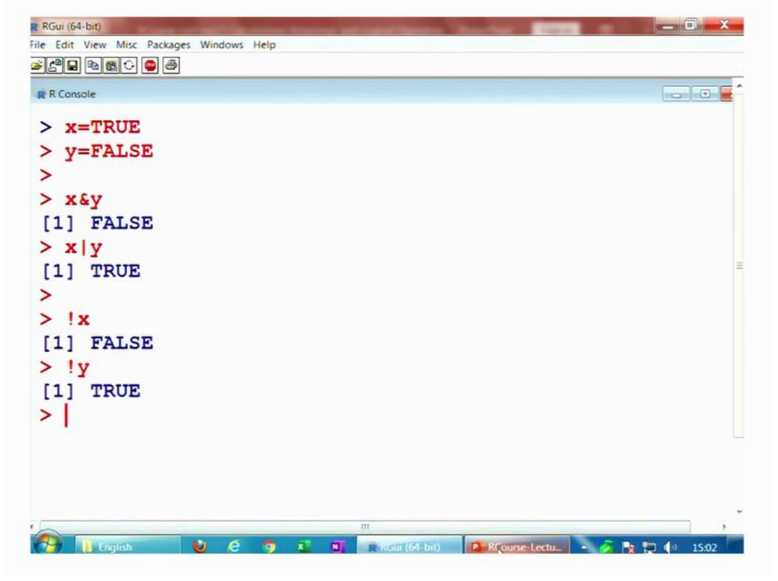
```
> x = 5
```

```
> Logical1 = (x > 2)
> Logical1
[1] TRUE
> is.logical(Logical1)
[1] TRUE
```

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

```
> Logical3 = (x != 5)
> Logical3
[1] FALSE
> is.logical(Logical3)
[1] TRUE
```

(Refer Slide Time: 09:40)



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

So, before going into the R console let me try to show you these operations on the R console itself. So, you can see here I take here two variables here. x is equal to here TRUE and y equal to here FALSE, right. And now, if you try to see here x and y this is FALSE, x or y this is here TRUE.

And if you try to take here negation of x this is FALSE, because x is TRUE and if you try to take a negation of y, the y is FALSE , so the negation of y will be TRUE. So, you can see that these are the operation which are giving you the same result which you have just considered right.

After this I try to give you another context or command, which is helpful in finding out whether the outcome of an operation or a variable is a logical variable or not. So, you know, that in the earlier lectures we had used different types of such commands like this dot numeric is dot character etc.

Similarly in case if you want to check whether a variable or a value or an outcome is a logical variable or not, the command is dot logical i s dot l o g i c a l right. And then; obviously, it is going to give you an answer which is either TRUE or FALSE and based on that we can take the correct decision.

So, we will take here a couple of example and try to see how this operation works. So, I try to take here a variable whose value is 5 and then I try to define here a variable Logical1 equal to x greater than 2. So, x is coming from here. So, the outcome of this statement that 5 is greater than 2 is going to be stored in the variable Logical1. So, 5 is greater than 2, you can see here there is no issue. So, the outcome of this Logical1 is TRUE.

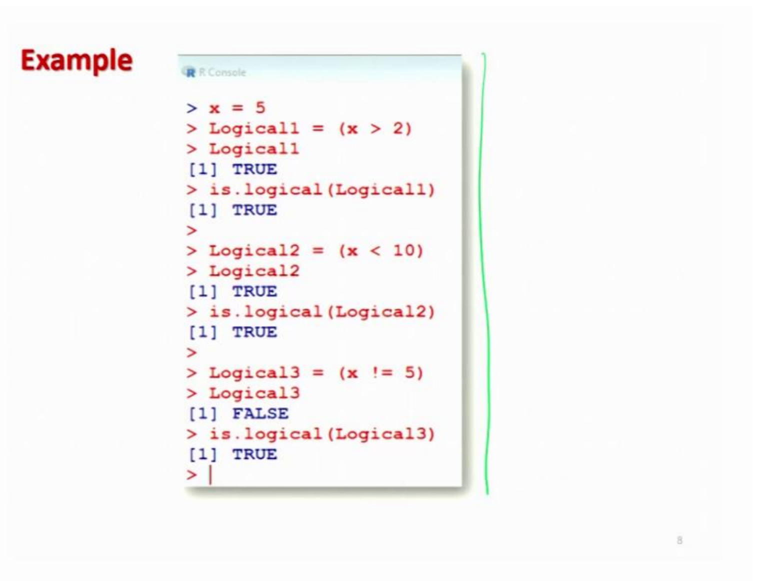
And now, if you try to see the character of this Logical1 or the mode of this Logical1, whether it is a logical variable or not. So, you can see here is dot logical and inside the parenthesis you have to give Logical1 and this is giving you an answer TRUE. And similarly, if you try to take here means another operation which is smaller than.

So, I try to take here the variable x smaller than 10 and I try to assign its outcome in another logical variable Logical2. So, x here is 2 and five here is less than 10. So, this is TRUE. So, if you try to see the outcome of Logical2 is TRUE.

So, in case if you try to see the behavior of this Logical2 using the command is dot logical and inside the parenthesis you say Logical2 the name of the variable, it gives you here TRUE; that means, this variable is a logical variable. And similarly, if you try to take here one more operation that x is not equal to 5. So, 5 is not equal to 5. The outcome of this operation is stored in the Logical3 variable, whose outcome comes out to be here FALSE; obviously, 5 is not equal to 5 is a FALSE statement.

So, now if you try to see the output of is dot logical and inside the parenthesis Logical3, then this comes out to be TRUE. So, do not get confused, that because here it is TRUE. So, it is here TRUE here it is TRUE. So, this is also here TRUE, no these are two different outcome they are trying to represent two different things. So, here it a Logical3, value here is FALSE, but this TRUE is indicating that the outcome of Logical3 is a logical variable which is correct.

(Refer Slide Time: 14:10)

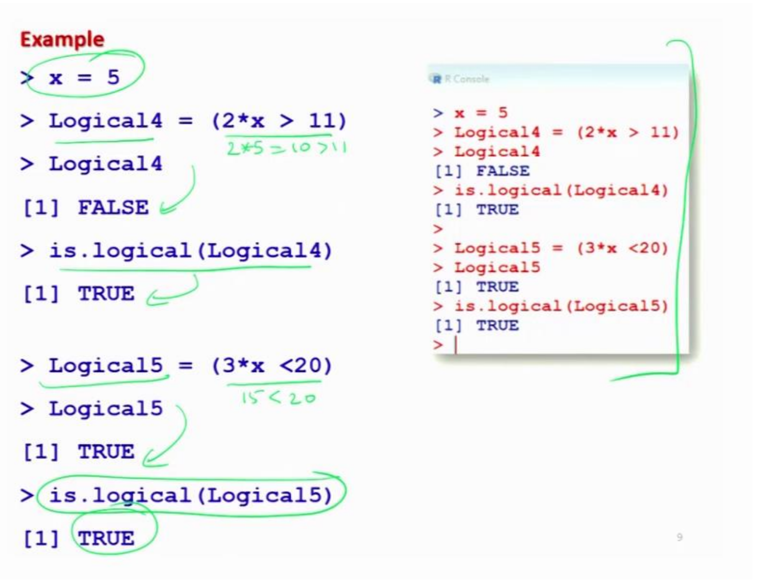


## Example

```
> x = 5
> Logical1 = (x > 2)
> Logical1
[1] TRUE
> is.logical(Logical1)
[1] TRUE
>
> Logical2 = (x < 10)
> Logical2
[1] TRUE
> is.logical(Logical2)
[1] TRUE
>
> Logical3 = (x != 5)
> Logical3
[1] FALSE
> is.logical(Logical3)
[1] TRUE
> |
```

8

(Refer Slide Time: 14:16)



## Example

```
> x = 5
> Logical4 = (2*x > 11)
                 2*5 = 10 > 11
> Logical4
[1] FALSE
> is.logical(Logical4)
[1] TRUE

> Logical5 = (3*x <20)
                  15 < 20
> Logical5
[1] TRUE
> is.logical(Logical5)
[1] TRUE
```

```
> x = 5
> Logical4 = (2*x > 11)
> Logical4
[1] FALSE
> is.logical(Logical4)
[1] TRUE
>
> Logical5 = (3*x <20)
> Logical5
[1] TRUE
> is.logical(Logical5)
[1] TRUE
> |
```

9

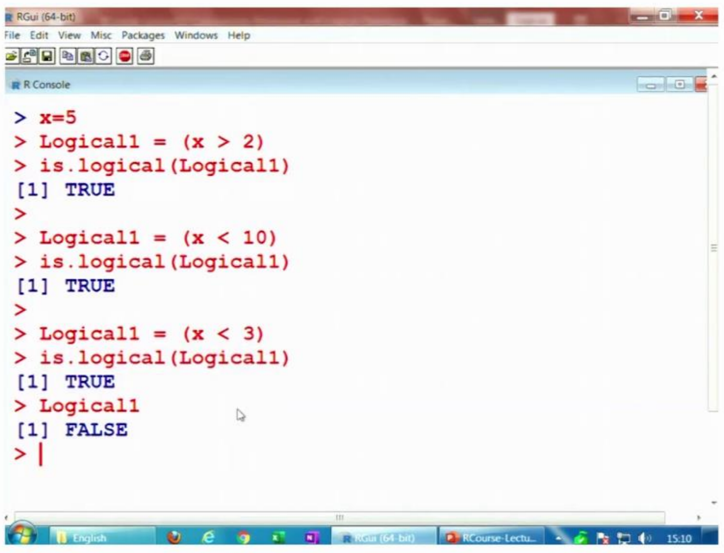
And this is here the screenshot of these operation which I just explained you, I will try to show you it on the R console also, but before that let me try to take here two more examples. So, now I try to show you that if in if you are trying to take some mathematical operations on these variables then what happens.

So, I try to define here one more variable here as a Logical4, which is saying 2 into x is greater than 11 and x here is 5. So, 2 into x which is here 2 into 5, 10 is greater than 11 or not this is a FALSE statement. So, this outcome comes out to be here FALSE and if you try to see is it a logical variable answer comes out to be here TRUE.

And similarly, if you try to take here similar one more operation that 3 into x is less than 20 and you store the outcome in variable Logical5, then the outcome of this will be 3 into 5 is 15 little is less than 20 answer is yes.

So, that is why this statement is TRUE and if you try to see whether this is a logical variable or not using the operator is dot logical then the answer comes out to be here TRUE and this is here the screenshot. So, this is how the R works with these logical operators. So, before I try to move forward let me try to give you these operations, I try to show you on the R console also, right.

(Refer Slide Time: 15:48)

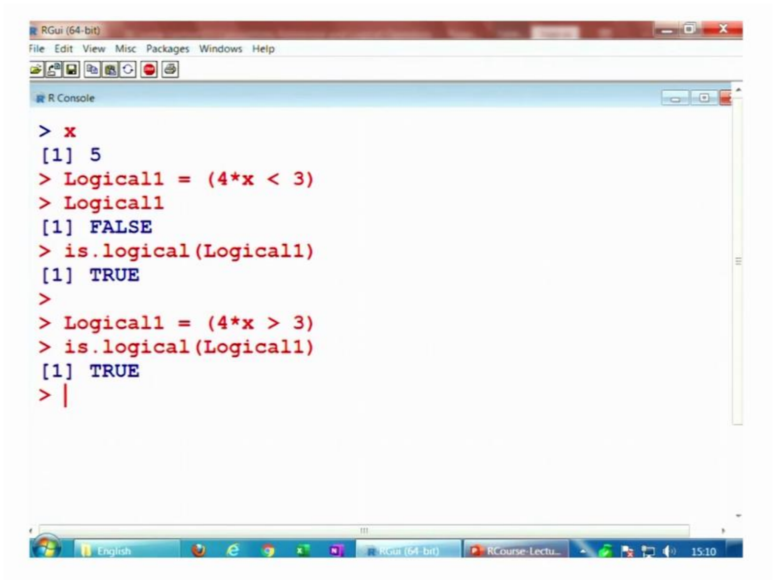


```
> x=5
> Logical1 = (x > 2)
> is.logical(Logical1)
[1] TRUE
>
> Logical1 = (x < 10)
> is.logical(Logical1)
[1] TRUE
>
> Logical1 = (x < 3)
> is.logical(Logical1)
[1] TRUE
> Logical1
[1] FALSE
> |
```

So, let me try to take here x equal to here 5 and then the Logical1 here is say x greater than 2. So, if you try to see here is dot logical and then inside the parenthesis logical say here 1 and if you try to see here this comes out to be a TRUE. And in case if I try to change my logical variable as x say smaller than say here 10 then what happens comes out to be here if you try to see here the outcome comes out to be TRUE, yes that is correct. And if you try to change it to be x is say, less than 3 then what happens, if you try to see the same operation it will again come out to be TRUE.

Why? Because if you try to see the outcome of this Logical1 here is like FALSE; means 5 is smaller than 3, no this is FALSE. So, this is FALSE, but is this a logical statement answer is yes.

(Refer Slide Time: 17:09)

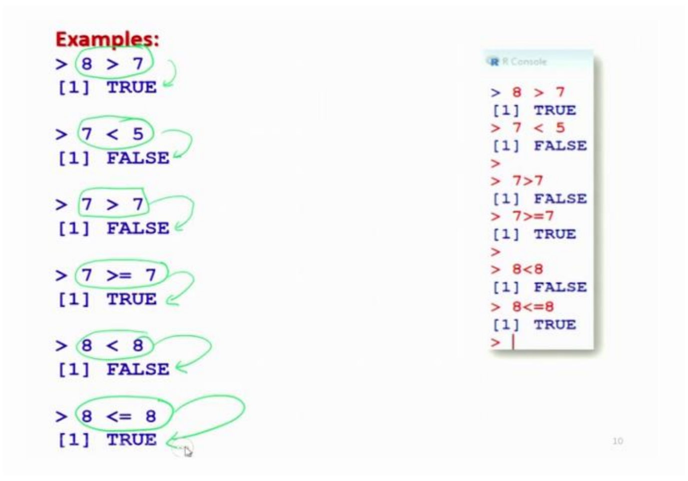


```
> x
[1] 5
> Logical1 = (4*x < 3)
> Logical1
[1] FALSE
> is.logical(Logical1)
[1] TRUE
>
> Logical1 = (4*x > 3)
> is.logical(Logical1)
[1] TRUE
> |
```

So similarly, if I try to consider here more operations here like as I try to take here say 4 into x is 3 and if I try to see the value of this logical variable is like as here FALSE, but if you try to see here is dot logical is this variable answer is TRUE. And similarly, if you try to change this variable as a 4 into x is greater than 3 yes. This is also a logical variable which is TRUE.

So, you can see here it is not a very difficult thing to find whether a variable is logical variable or not, right. And the answer is always going to be in terms of only TRUE and FALSE.

(Refer Slide Time: 17:55)



**Examples:**

```
> 8 > 7
[1] TRUE

> 7 < 5
[1] FALSE

> 7 > 7
[1] FALSE

> 7 >= 7
[1] TRUE

> 8 < 8
[1] FALSE

> 8 <= 8
[1] TRUE
```

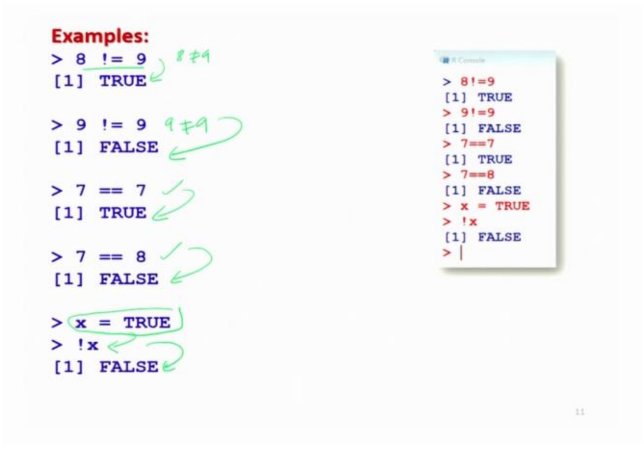
```
> 8 > 7
[1] TRUE
> 7 < 5
[1] FALSE
>
> 7>7
[1] FALSE
> 7>=7
[1] TRUE
>
> 8<8
[1] FALSE
> 8<=8
[1] TRUE
> |
```

Now, in case if you try to take here some more examples which will give you a query made and handy solutions without much efforts. I am trying to write down here 8 greater than 7 is this TRUE? Yes. This is TRUE and the answer here is TRUE once again.

Now, my next statement is 7 is smaller than 5? The answer is no. That is why answer comes out to be here FALSE, then I try to take here is 7 greater than 7? Answer is no. So, the answer comes out to be here FALSE, but if I try to rephrase this question as is 7 greater than or equal to 7? Yes. Answer is correct.

So, this comes out to be here as a TRUE. Similarly, if I try to take here a statement like 8 is smaller than 8? No. So, the answer here is FALSE. And when I try to rephrase it 8 is smaller than or equal to 8? It is correct. So, that is why the answer comes out to be here TRUE, right.

(Refer Slide Time: 19:08)



**Examples:**

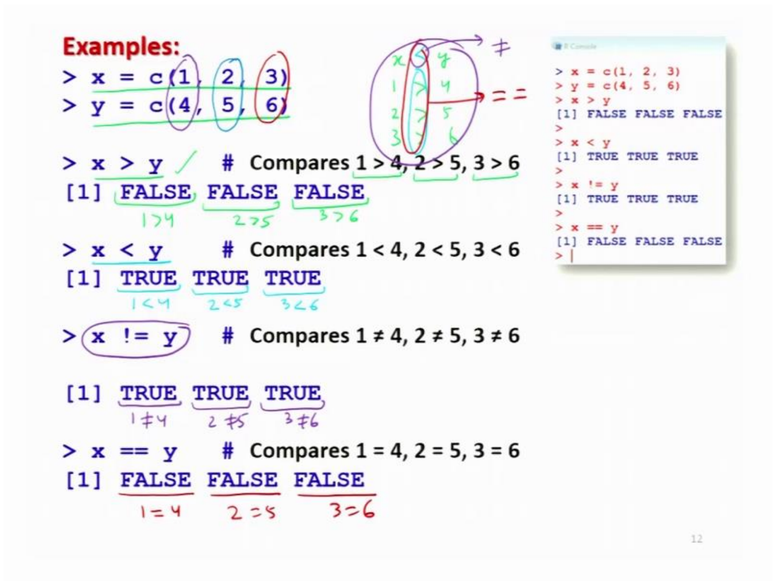
```
> 8 != 9
[1] TRUE

> 9 != 9
[1] FALSE

> 7 == 7
[1] TRUE

> 7 == 8
[1] FALSE

> x = TRUE
> !x
[1] FALSE
```

```
> 8!=9
[1] TRUE
> 9!=9
[1] FALSE
> 7==7
[1] TRUE
> 7==8
[1] FALSE
> x = TRUE
> !x
[1] FALSE
> |
```

And similarly if you try to take here some more operations, you are trying to write down here 8 is not equal to 9? Yes, this is correct. So, it is TRUE. Now, you are trying to write down here 9 is not equal to 9, is FALSE. You are trying to write down here is 7 is equal to 7? Answer is; obviously, TRUE.

So, the answer comes out to be here logical TRUE right. Here is 7 is equal to 8? Answer is FALSE, 7 cannot be equal to 8. Now, in case if you try to take here one variable here say x equal to TRUE and then if you try to see here negation of x this comes out to be a FALSE.

(Refer Slide Time: 19:52)



**Examples:**

```
> x = c(1, 2, 3)
> y = c(4, 5, 6)

> x > y        # Compares 1 > 4, 2 > 5, 3 > 6
[1] FALSE FALSE FALSE

> x < y        # Compares 1 < 4, 2 < 5, 3 < 6
[1] TRUE TRUE TRUE

> x != y       # Compares 1 ≠ 4, 2 ≠ 5, 3 ≠ 6

[1] TRUE TRUE TRUE

> x == y       # Compares 1 = 4, 2 = 5, 3 = 6
[1] FALSE FALSE FALSE
```

```
> x = c(1, 2, 3)
> y = c(4, 5, 6)
> x > y
[1] FALSE FALSE FALSE
>
> x < y
[1] TRUE TRUE TRUE
>
> x != y
[1] TRUE TRUE TRUE
>
> x == y
[1] FALSE FALSE FALSE
> |
```

So, this is how you can see that these operation work on the R console, but in all this example I am trying to take only the scalars, only one value at a time. Now, the next question comes what happens if you are trying to consider more than when one values which are stored in the format of a data vector.

So, I try to consider here an example in which I consider here two data vectors x and y and x takes values 1, 2 and 3 and y takes three values 4, 5 and 6. So, now, in this case in case if I try to write the statement like x greater than y. Remember both x and y are the data vectors which have more than one values.

So, now the answer comes out to be the outcome comes out to be FALSE, FALSE and FALSE. What is this indicating? Actually this is indicating that the elements in the x and y, they are compared position wise. For example, the first element in the data vector x and the first element in the data vector y, they are compared after the first element, the control come to the second element and then the second elements of data vector x and y, they are considered and after that the control comes to the third element, and the operation is made over the values at the third positions in the data vectors x and y.

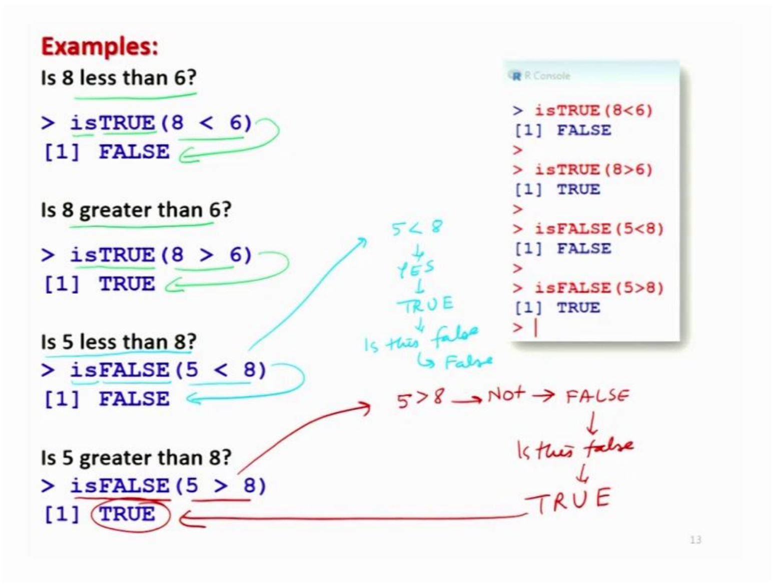
So, if you try to see what is really happening when you are trying to write down x greater than y. So, if you try to write down here x and here say here y, x is 1, 2, 3 and y here is 4, 5, 6 and the operation here is like this greater than. So, it is being compared here that 1 is greater than 4, 2 is greater than 5 and 3 is greater than 6. So, this FALSE, if you write this is an outcome of the operation 1 greater than 4. This is an outcome of 2 greater than 5 and this and the third value which is here the FALSE, it is an outcome of the operation 3 greater than 6 right.

And similarly, if you try to take here next operation say here is say x smaller than y then it tries to compare the three values and the first value in the outcome is an outcome of the operation 1 less than 4. The second TRUE is an outcome of the operation 2 smaller than 5 and the third TRUE is an outcome of the operation 3 smaller than 6, because now instead of this greater than sign. Now it is replaced by less than sign.

And similarly, in case if you try to take here the operation x is not equal to y. Then again it has three outcome and in this operation and this sign is replaced by not equal to. So, now this TRUE, the value at the first position this is an outcome of 1 is not equal to 4 which is TRUE. The second value is an outcome of 2 not equal to 5 and the third value is an outcome of the operation 3 is not equal to 6.

Now, after this I try to consider the equality operator. So, all these symbols they are replaced by equality sign double equal to sign. So, that it is a logical equality sign. So, if you try to see here this is here the outcome where the first outcome it is trying to consider and this is an outcome of comparison of 1 and 4 that 1 is equal to 2, second value is the outcome of 2 equal to 5 and third value is an outcome of 3 equal to 6 and so, this is how the operation is done when we are trying to do with the vectors data vectors right.

(Refer Slide Time: 23:54)



**Examples:**

Is 8 less than 6?

```
> isTRUE(8 < 6)
[1] FALSE
```

Is 8 greater than 6?

```
> isTRUE(8 > 6)
[1] TRUE
```

Is 5 less than 8?

```
> isFALSE(5 < 8)
[1] FALSE
```

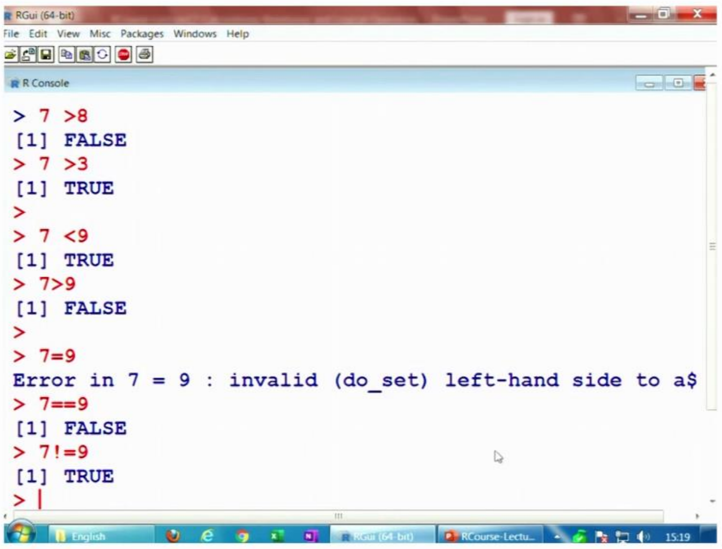
Is 5 greater than 8?

```
> isFALSE(5 > 8)
[1] TRUE
```

```
> isTRUE(8<6)
[1] FALSE
>
> isTRUE(8>6)
[1] TRUE
>
> isFALSE(5<8)
[1] FALSE
>
> isFALSE(5>8)
[1] TRUE
> |
```

R Console
5 < 8
YES
TRUE
Is this false
False
5>8 → Not → FALSE
Is this false
TRUE
13

(Refer Slide Time: 24:13)

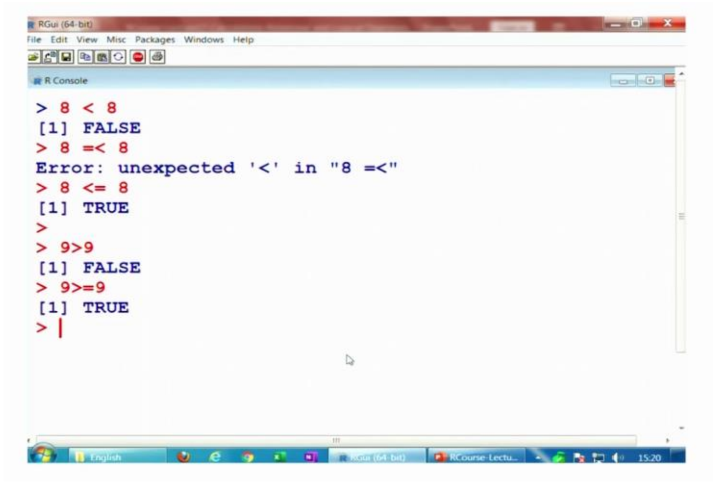


```
> 7 >8
[1] FALSE
> 7 >3
[1] TRUE
>
> 7 <9
[1] TRUE
> 7>9
[1] FALSE
>
> 7=9
Error in 7 = 9 : invalid (do_set) left-hand side to a$
> 7==9
[1] FALSE
> 7!=9
[1] TRUE
> |
```

So, let me try to show you first these values in the R software and then I will try to explain you the last slide. So, if you try to see here. If I try to consider here the outcome like 7 is greater than 8, it is FALSE, but if I say 7 is greater than 3, it is TRUE and if I try to say here 7 is less than 8 or 9, it is TRUE, 7 is greater than 9, this is TRUE, 7 is equal to 9, no why? Because you have not used the logical operator.

So, let me try to write down here. 7 double equal to 9 and this is going to be FALSE. And if you try to write here 7 is not equal to 9 this is here TRUE right. So, you can see here these are the operations here.

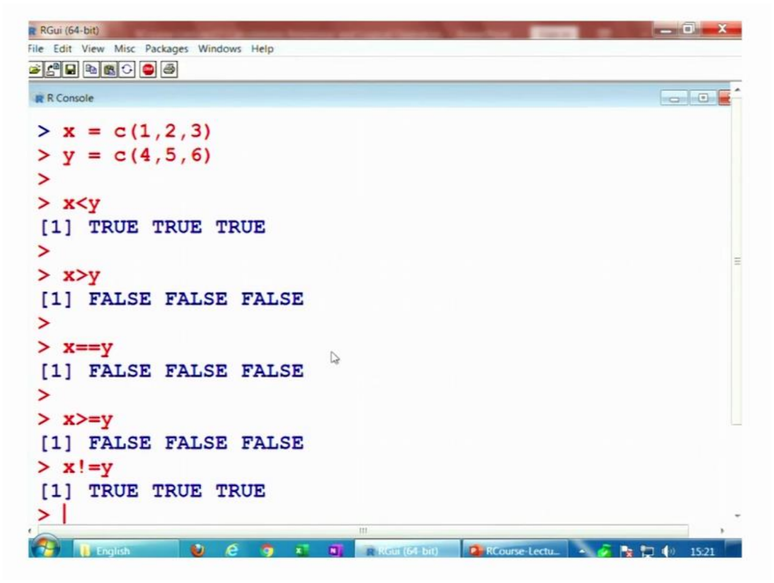
(Refer Slide Time: 25:01)



```
> 8 < 8
[1] FALSE
> 8 =< 8
Error: unexpected '<' in "8 =<"
> 8 <= 8
[1] TRUE
>
> 9>9
[1] FALSE
> 9>=9
[1] TRUE
> 
```

And similarly, if you want to do a some more operation like 8 is less than 8, FALSE, but if you try to make it here 8 is less than or equal to 8, this is again a wrong thing. Why? Because your operator should be like here this TRUE. And similarly, if you try to make it here another operation 9 is greater than 9, answer is FALSE, but if you try to take it here 9 greater than or equal to 9, then it is TRUE, right.

(Refer Slide Time: 25:40)



```
> x = c(1,2,3)
> y = c(4,5,6)
>
> x<y
[1] TRUE TRUE TRUE
>
> x>y
[1] FALSE FALSE FALSE
>
> x==y
[1] FALSE FALSE FALSE
>
> x>=y
[1] FALSE FALSE FALSE
> x!=y
[1] TRUE TRUE TRUE
> 
```

And similarly, if you try to look at here this operation where I am trying to take two data vectors. So, let me try to take here x is equal to c 1, 2, 3 and y here is c 4, 5, 6 and if you try to see here x less than y this gives you a TRUE, TRUE, TRUE, x greater than y will give you FALSE, FALSE, FALSE, x equality y FALSE, FALSE, FALSE, x greater than or equal to y this will give here FALSE, FALSE, FALSE.

And similarly if you try to take here x say what equal to y it will give you here TRUE, TRUE, TRUE, right. So, that is how you can make such calculations without any problem and you can see these are very simple operation. Now, I want to show you the outcome of these two more operation is dot TRUE and is dot FALSE. So, I try to write down here i s in lowercase alphabets and TRUE in the upper case alphabets and I want to check is 8 smaller than 6, answer is no.

So, if you try to write down here is to 8 less than 6, the answer here is FALSE. And similarly if you want to check is greater is 8 greater than 6 simply write i s and then T R U E. So, i s is in lower letter T R U E in capital letter and inside the parenthesis write down the condition 8 greater than 6 it is TRUE. So, it is the answer comes out to be here TRUE.

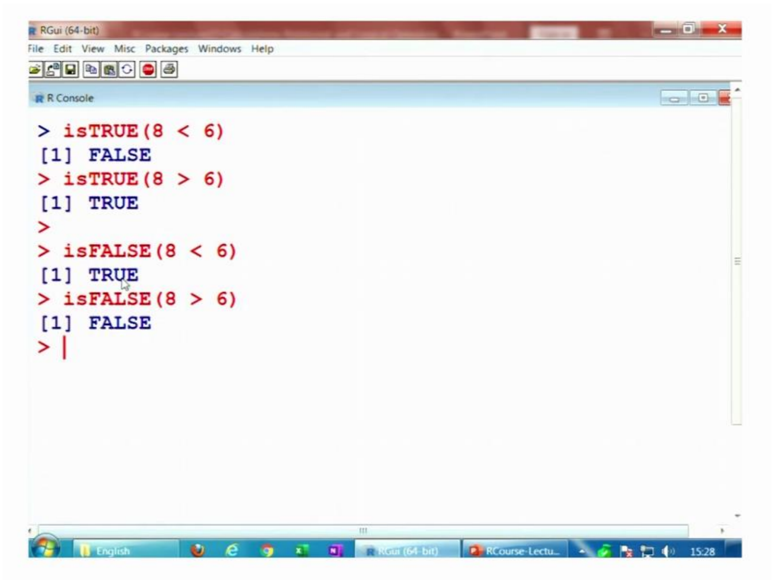
Now, after this I would like to show you one more application of the statement is FALSE. So, suppose I want to check is 5 less than 8. So, I will write down here i s in lowercase alphabets and then FALSE, F A L S E in upper case alphabets and then inside the parenthesis I write here 5 smaller than 8 and the answer comes out to be here FALSE. Do not you think that this is confusing. So, let us try to understand what it is trying to do.

So, the condition here what it want to check is 5 less than 8, the answer here is yes. Yes means TRUE. And what are you asking? You are asking is FALSE. So, is this FALSE and the answer that is why comes out to be here FALSE, that you are asking that it is FALSE, but you are FALSE; that means, it is 5 is smaller than 8 is correct. So, this is how you have to understand what it is trying to do.

Similarly, if you try to use this condition here isFALSE 5 greater than 8, now once again if you try to see what is happening? 5 is greater than 8, is this correct? No, this is not TRUE, but answer is here TRUE. What does this mean? This 5 is not greater than 8. So, this mean this is FALSE, but your question is this FALSE? Yes, this is FALSE and so, the answer comes out to be here TRUE.

That is was the reason that it is coming out to be here TRUE. So, do not get confused and try to means understand these operations logically. And you know that unless and until you understand this basic operation and you do not understand this basic concept you cannot do it.

(Refer Slide Time: 29:47)



```
> isTRUE(8 < 6)
[1] FALSE
> isTRUE(8 > 6)
[1] TRUE
>
> isFALSE(8 < 6)
[1] TRUE
> isFALSE(8 > 6)
[1] FALSE
> |
```

So, let me try to show you these operations on the R console. So, if you try to see here is TRUE 8 smaller than 6 is FALSE and is 8 greater than 6 is TRUE. And if I try to do the same thing over here this that I try to replace here by here FALSE you can see the change very clearly isFALSE this is here TRUE. And if you try to see here isFALSE 8 greater than 6, answer comes out to be here FALSE. So, that is how the things work with this logical operators and relational operators in the R software.

So, now I have given you a decent introduction on the use of logical operators and relational operators. Now, it is your turn, because as I said these things are very useful particularly when you are trying to deal with databases and you want to create different types of questions, different types of arrays. You can write is x greater than 3 and y is greater than 4 and z is less than 8. And one more important thing. In this examples, I have taken here only two variables. And at a time I have taken only here one operation either like as & or the or operator.

But, if you want to have more than one statements and you want to test more than two conditions at a time or more than one conditions at a time. You can also do it. Means I can write here a statement x greater than 2 and y less than 3 or z greater than 5, but then you have to understand that what will be the outcome and why do you want to do it and you should be able to understand the logic of the outcome and this is going to be really helpful.

So, my request to you all is that you try to create your own examples. Try to take very simple values, try to first understand them that if you try to operate the operation like and or etcetera. Then what will happen? And then try to see, are you getting the same outcome that you expected? This will give you more confidence to practice it and I will see you in the next lecture till then goodbye.