Data Analysis for Biologists

Professor Biplab Bose
Department of Biosciences and Bioengineering
Mehta Family School of Data Sciences and Artificial Intelligence
Indian Institute of Technology Guwahati

Lecture 15

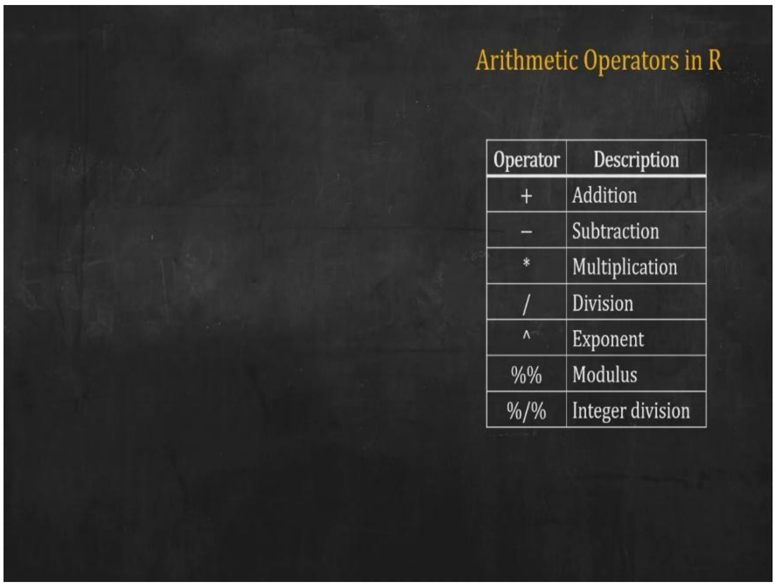
Algebraic and Logical Operations in R

Hello everyone. Welcome back. Suppose, I want to add two numbers, 2 and 3. Obviously I will get 5. So, what I am doing here I am adding two numbers and we say addition is a mathematical operation. Similarly, subtraction is a mathematical operation. Division, multiplication, all these are mathematical operations.

There are many other mathematical operations in different categories. And when you write a program in R or any other programming language to analyze data or perform some computing, what you are doing? You are using these mathematical operations to perform some job.

So, in this lecture, we will discuss about some basic operators in R which are recurrently used regularly will be used by you while performing data analysis. We have three types of operators for discussion today. One is arithmetic operators, those are the plus minus and all those thing. Then we have a relational set of operators and we will have logic operators.

(Refer Slide Time: 01:33)



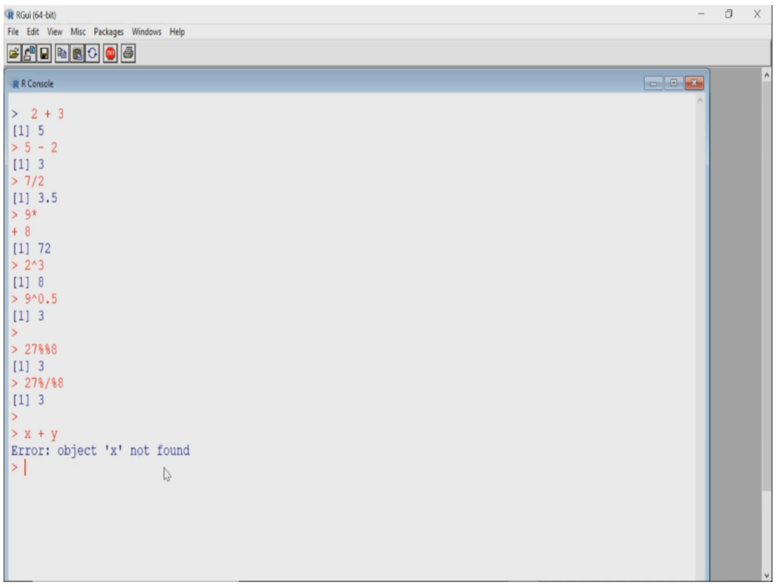
Arithmetic Operators in R

Operator	Description
+	Addition
–	Subtraction
*	Multiplication
/	Division
^	Exponent
%%	Modulus
%/%	Integer division

So, here I have shown you on the screen arithmetic operators. You have addition; subtraction, you know them, those symbols are in your keyboard. The star sign (*) that is in your keyboard is actually the multiplication. The slash(/) is for division and this caret(^) is for exponentiation. Suppose, you have to write something to the power something, you will use that.

And these two sequential percentage(%) symbol, if you use it will be considered as a modulus and if you use one percentage(%) symbol after along with one slash(/) and percentage symbol(%) it will be integer division. We will show you the example of that. Let me go to R. I will use native R and use these arithmetic operators.

(Refer Slide Time: 02:22)



```
>  2 + 3
[1] 5
> 5 - 2
[1] 3
> 7/2
[1] 3.5
> 9*
+ 8
[1] 72
> 2^3
[1] 8
> 9^0.5
[1] 3
>
> 27%%8
[1] 3
> 27%/%8
[1] 3
>
> x + y
Error: object 'x' not found
> |
```

So, I have my R console here. So, let me use it just like a calculator. So, suppose, I want to add (2 + 3), I should it should report me 5. Yeah. It does that. Suppose, we write (5 – 2), it should give me 3. I want to divide (7/2), I should get 3.5 and suppose, I want to multiply 9 with 8, I missed 8. So, it is asking me to enter the number I entered it gave me 72.

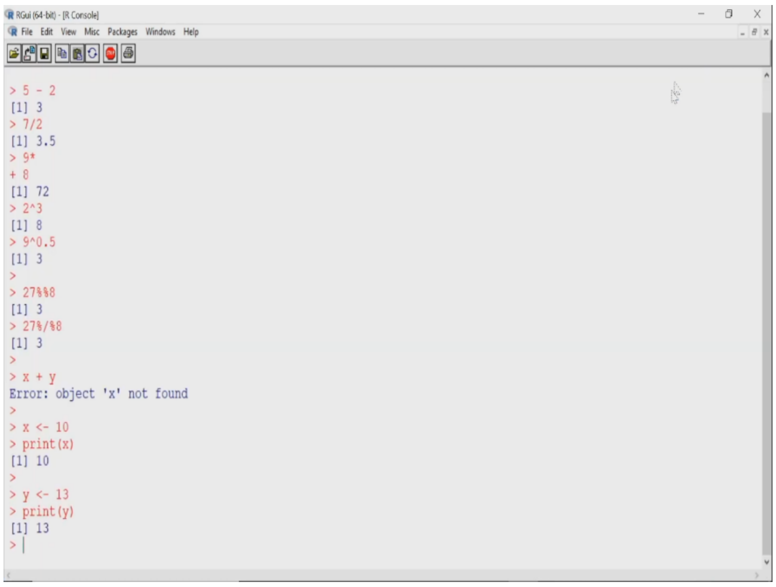
Now, let me use the exponentiation operator, the caret(^) one. So, if I want to calculate the (2^3). It will be like this. It should give me 8. Or suppose, I want to calculate the square root of 9. So, (9^0.5). Because if you remember a square root is 1/2. So, I am writing 0.5, it should give me 3. Let me use the modulus operator. It will give me the remainder of a division. So, suppose, if I divide 27 and I want to divide it with suppose, 8. So, what should be the remainder? It is 3 because 8 3s are 24 plus 3 is 27.

Now, if I want to take the only the integer part of the result of division I can use integer division symbol. So, again I will use the same thing 27, a percentage sign slash percentage

(%/%) and 8. So, I get 3 because 8 3's are 24. Now, what I have done here I have used these arithmetic operators just like a, and you use this R console as a calculator. But, you must have seen that when we are doing, discussing algorithms of data analysis, we are using algebra. In algebra, we sum x with y two symbols. So, let me try, whether it works here or not.

So, x plus y, enter. R is telling me, there is an error. The error it is saying is, object x not found. The problem with R is that it cannot understand this type of symbolic thing directly. So, it you have to assign some numerical value to each of this symbolic variable. x here is a symbolic variable; y is a symbolic variable. So, I have to assign some numerical value to this individual variable x and y. So, how can I assign it? Assigning is very simple.

(Refer Slide Time: 05:03)

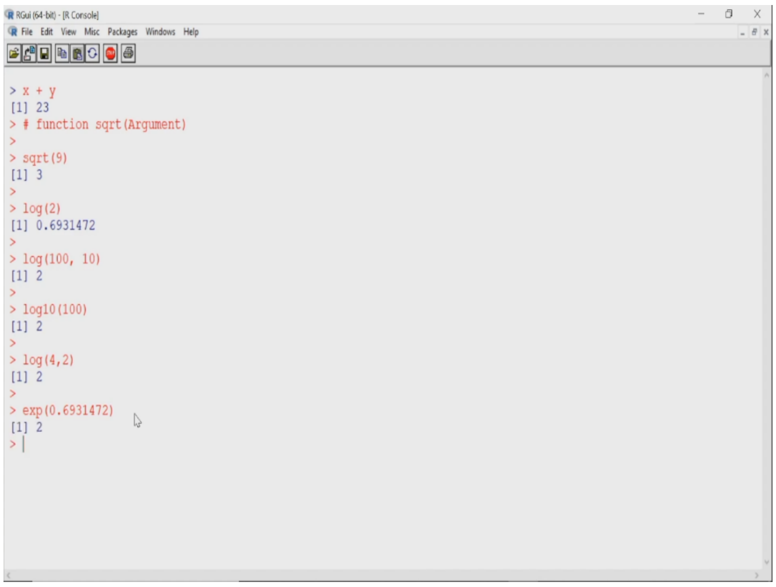

RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

```
> 5 - 2
[1] 3
> 7/2
[1] 3.5
> 9*
+ 8
[1] 72
> 2^3
[1] 8
> 9^0.5
[1] 3
>
> 27%%8
[1] 3
> 27%/%8
[1] 3
>
> x + y
Error: object 'x' not found
>
> x <- 10
> print(x)
[1] 10
>
> y <- 13
> print(y)
[1] 13
> |
```

Suppose, I want to assign 10, number 10 to the x. So, I will write the variable name x and then I will use this less than symbol then dash. So, it will be an arrow (←). The arrowhead is pointed towards the variable and on the tail side of the arrow here, I will write the value that I want to assign. So, I have assigned 10 to x.

Now, if I print x. You can see, R is telling the value of x is 10. Similarly, suppose, I assign y, again I write the arrow using the less than symbol and dash and now, I assign 13 to y. So, y will become 13. Now, if I print y, I should get 13. Let me. So, now, R know that x is actually 10 and y is equal to 13. Let me clean the console.

(Refer Slide Time: 06:09)


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

```
> x + y
[1] 23
> # function sqrt(Argument)
>
> sqrt(9)
[1] 3
>
> log(2)
[1] 0.6931472
>
> log(100, 10)
[1] 2
>
> log10(100)
[1] 2
>
> log(4,2)
[1] 2
>
> exp(0.6931472)
[1] 2
> |
```

So, now, if I sum x and y. So, I want to sum x with y. Now, R can work. So, R knows now, that x is equal to 10, y is equal to 13. So, summation of x and y, it is doing arithmetic operation of addition because I have used the addition symbol there. So, it has added and given me 23. So, usually when you are writing a code in R, you have to assign variables with certain numerical values. Some arithmetic operations like exponentiation to get the square root can be performed using inbuilt function in R.

What is the function? Let us explain it in a very simple way. In mathematically a function is something which takes some numerical values and throws out something, is not it. So, if you say, y = x + 3 that means x + 3 is the function of x and now, if you put x = 2, my function output will be 2 + 3, 5. So, these a function will take something as an input and it will spit out something as output.

Similarly, in computer languages in almost all languages, we will have some functions; either some inbuilt function or you can create functions. The advantage of these functions is that you can give them some input and it will throw out, do some calculation and throw out some output.

So, similarly in R we have a function, function called sqrt and then you put a round bracket. This sqrt is short of square root. So, sqrt function will give me square root of a number and I have to put the number as an argument. We will write it like this. We will put the argument inside the round bracket. So, argument must be some value or some variable which has some numerical value.

So, for example, if I want to calculate the square root of 9 rather than using the character symbol to calculate square root, what I can do, I can write sqrt square root function and then I put the value 9, sqrt(9). I want to calculate the square root of 9, so that is the argument and R should be able to calculate that and it gives me 3. So, square root, sqrt, square root is an inbuilt function to easily calculate the square root of a number.

Similarly, suppose, you want to calculate log of a number, very frequently you may have to calculate. So, you have an inbuilt function log and it by default calculates the natural log, ln. So, let me we know that natural log value for 2. So, let me calculate that. So, again you can notice here log is the name of the function and then in the round bracket I am putting the argument, log(2). So, it should be able to calculate and it has given 0.6931472 and you must be knowing this value.

Now, you want to calculate log with a specific base. That is what you frequent may be doing. You do not want natural log. So, in that case this, same function can be used, but I have to add another argument inside that round bracket. How should I do that? So, suppose, I want to calculate the log of 100 base 10.

So, I will write log the name of the function, then in the round bracket I will put 100. This is the first argument because I want to calculate the log of that and I will put the base, the base is 10, log(100,10). So, I am asking R to calculate log of 100 base 10 and it is 2. Because you know 10 to the power 2 is 100. So, log of 100 base 10 should be equal to 2.

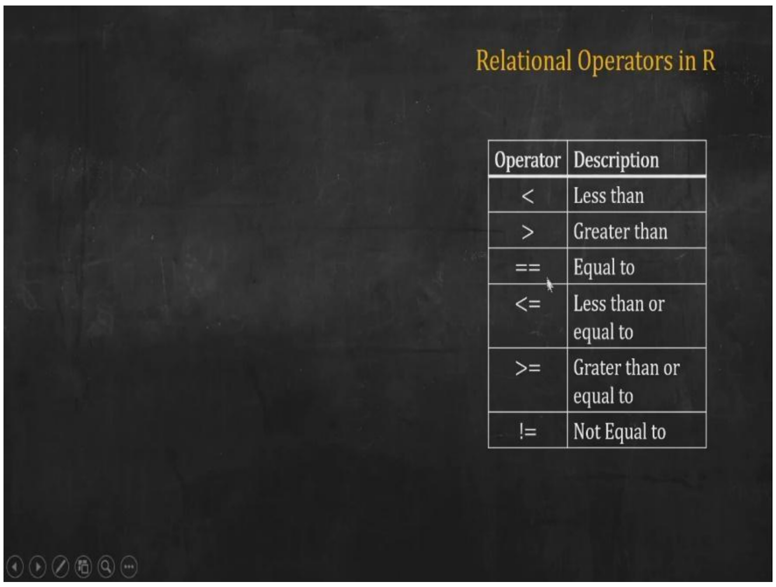
Now, the people who developed R has made our life easy. They have created a function specifically for calculating log with base 10. So, in that case you do not have to specify the base also. So, that is called log 10. So, a log 10 is a function which by default will consider base as 10. So, let me take a 100 as argument that means I am asking R to calculate the log of 100 base 10. It should give me 2. Similarly, I have log of suppose, I want to calculate log of 4 base 2, it should give me 2.

Now, as it has function of log you can immediately expect that there must be a function for e to the power, exponentiation with respect to e. So, it is there. So, it is called exp function. And I will put the argument within the round bracket. So, suppose, I put a exponentiation of that 0.6931472. So, I want to calculate e to the power 0.6931472 and it should give me 2. Because natural log of 2 is that particular value. So, I am just reversing it.

So, in this way there are lots and lots of inbuilt functions. I am just showing those which are useful for our arithmetic operations. And all the data analysis that we will perform we will actually use inbuilt functions of R. Now, till now, I have discussed only arithmetic operators.

Now, I will enter into some type of a particular type of operator called relational operator. Sometime you may have to compare like this that, ok, if this number x is bigger than y or x is less than y. So, you want to find a relation, which is bigger, which is smaller, which are equal to each other, this type of relation you want to find between variables. So, for that we have to use relational operators. Let us first check what are the different types of relational operators are there in R.

(Refer Slide Time: 12:19)



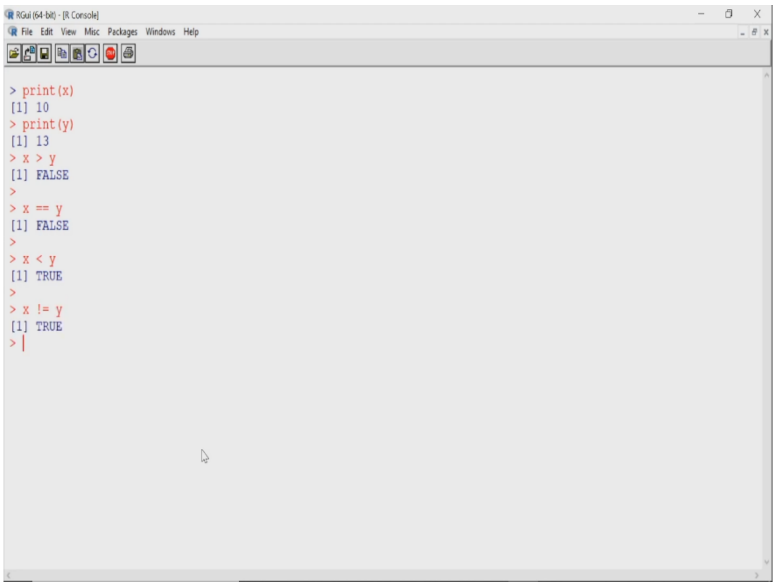
Relational Operators in R

Operator	Description
<	Less than
>	Greater than
==	Equal to
<=	Less than or equal to
>=	Grater than or equal to
!=	Not Equal to

So, here is the list of relational operators in R. You can use a less than(<) to compare between two numbers, whether check whether one is lesser than the other one or not. Then you have the greater than symbol(>) that is for the greater than operator. Then when you want to compare whether two numbers are equal or not, you simply do not use one equal to sign, you have to use two consecutive equal to sign(==) as we have shown here.

So, to check equal to, we have to use two consecutive equal to sign and that will tell R that you want to compare two values and check whether they are equal or not. You can also use one less than symbol followed by equal to sign(<=) to represent less than equal to. You can use greater than equal to by using the greater than symbol and then equal to symbol(>=). And then sometime you can check whether two numbers are not equal. So, in that case you give the exclamatory sign followed by this equal to sign(!=). So, these are the, these six are the relational operator in R. So, let us try to use those in our R console.

(Refer Slide Time: 13:31)


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

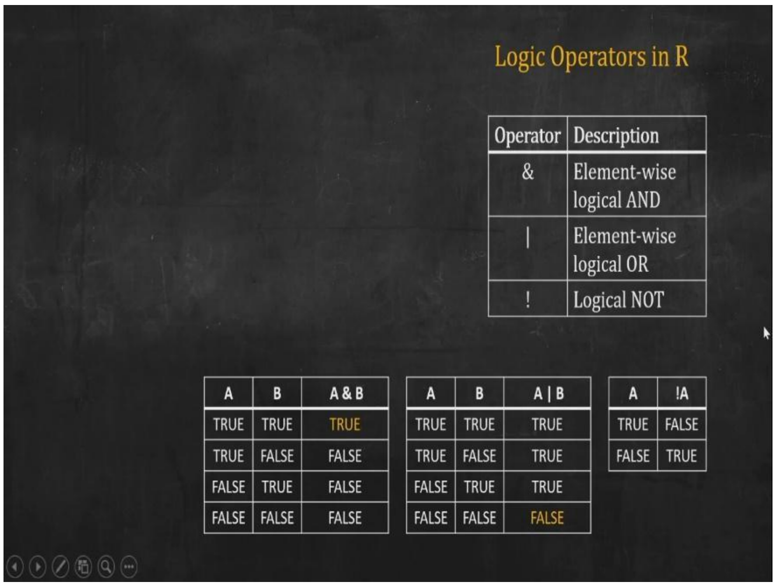
```
> print(x)
[1] 10
> print(y)
[1] 13
> x > y
[1] FALSE
>
> x == y
[1] FALSE
>
> x < y
[1] TRUE
>
> x != y
[1] TRUE
> |
```

So, let me print the value of x and y first. x is 10, we have assigned that earlier and y is 13. I want to check whether x is bigger than y or not. So, what I will write x > y and if you now press enter. R is telling it is false. That is true, because x is 10, it cannot be bigger than 13 and y is equal to 13.

Then suppose, I want to check whether x is equal to y or not. So, what I will write, I will write x, then I will put two equal to sign not one, mind it, you have to use two consecutive equal to sign, then only R will understand that you are using a relational operator. So, x == y and then if you press enter, it says false. That is true because x is not equal to y. x is 10, y is 13. Similarly, if I say, x < y. Now, it says true. That is true because 10 is smaller than 13. Or if I say x != y, that means I am saying x not equal to y.

You guess it what we will get? We should get true. So, in this way when you are writing algorithm or code in R or performing some analysis using R, we may have to use this relational operator to find relation between different variables. This brings us to our third type operators that will learn to in today's lecture. What is those? That is called logic operator. Let me show you what are the logic operators and how they are what is the meaning of those and then subsequently I will show you how to use them.

(Refer Slide Time: 15:27)



Logic Operators in R

Operator	Description
&	Element-wise logical AND
|	Element-wise logical OR
!	Logical NOT

A	B	A&B
TRUE	TRUE	TRUE
TRUE	FALSE	FALSE
FALSE	TRUE	FALSE
FALSE	FALSE	FALSE

A	B	A|B
TRUE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	TRUE	TRUE
FALSE	FALSE	FALSE

A	!A
TRUE	FALSE
FALSE	TRUE

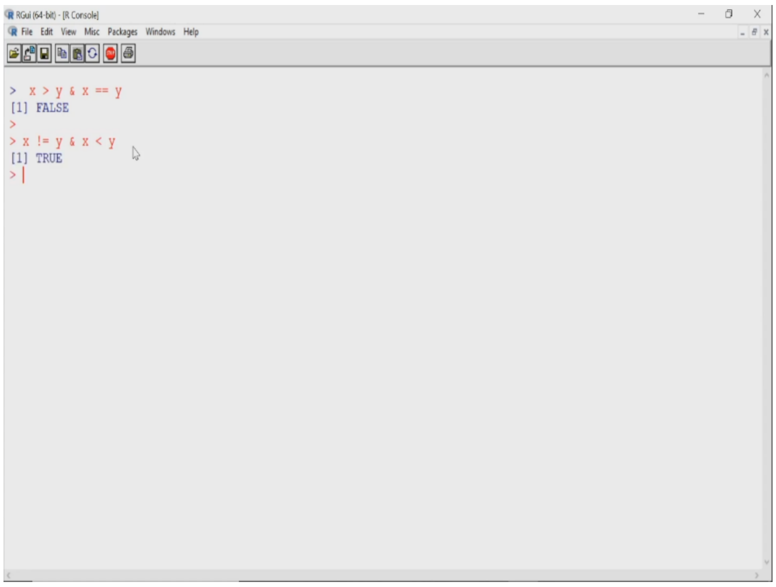
Here I have listed three logic operators in R. This '&' symbol that you use very frequently in short is actually called elementwise logical AND. I will explain what do I mean by that. And then this '|' symbol, this vertical line that you have in your keyboard is used for elementwise logical OR. And this exclamatory sign(!) means a logical NOT. So, I have three operation; AND OR and NOT.

So, what do I mean by AND. Suppose, I have two statement A and B. I have two statements A and B. And when A is true and B is also true, this is a situation and then if I write A & B, the output should be TRUE. So, this table that I have shown here is called truth table. Now, as per this truth table, if suppose, I have considered A is true, the statement of called A is true whereas the statement B is false then if I say A & B then my output should be FALSE, whereas a FALSE and TRUE gives me FALSE and a FALSE and FALSE give me also FALSE.

The next table that I have shown here is the truth table for OR. A is one statement the first column, B is the second statement, third is A OR B. You can see this pipe(|) symbol is there which is logical OR in R. So, when both A and B are FALSE, in the last row A OR B will give me output of FALSE because either of this is FALSE, whereas when A and B both are TRUE then A OR B should give me TRUE. A TRUE, B FALSE, when I combine them using OR, I should get TRUE because either of them is TRUE. And again, when I say A is FALSE or B is TRUE then my outcome of this OR operation is TRUE.

The third table that I have shown here is for NOT. You have already seen the use of this logical NOT when you say not equal to, in that case, you have this exclamatory sign followed by equal to. So, by this exclamatory(!) sign, you are saying NOT and then you have given equal to sign that means the machine understand this is not equal to. So, in this way, you can convert false to true, true to false something like that. So, let us do that, test that in R. How we do that?

(Refer Slide Time: 18:00)


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

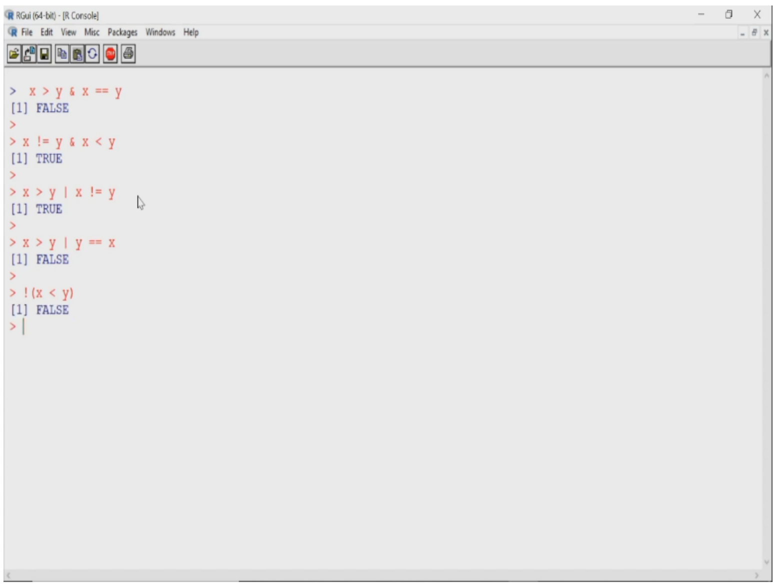
```
> x > y & x == y
[1] FALSE
>
> x != y & x < y
[1] TRUE
> 
```

Suppose, I write a statement x > y. I know x is smaller than y, because x is 10 y is 13. So, in this case x > y should give me FALSE. And now, I combine that with using a AND operation saying x == y. So, let us see what I have written here. I have first written x > y. So, this is my first relational operation and then I have given a AND symbol and then I have written another relation operation I am doing here, x == y.

So, the first relational operation x > y, this should be FALSE, similarly after AND I have also a FALSE statement. So, I have two FALSE statement connected by a AND. And what should I get? I get FALSE because FALSE and FALSE should give me FALSE.

Let us check this another thing. x != y & x < y. So, now, I have two statement. The first statement is x != y. x is 10, y is 13, obviously they are not equal to. So, this first statement is logically TRUE then the next statement connected by a AND symbol is x < y, that is also TRUE. So, what I have here? TRUE and TRUE. So, what should be the output? Output should be TRUE. So, this is how you use a AND symbol, AND operator, logical AND operator.

(Refer Slide Time: 19:42)


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

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

Now, let me use the OR operator. So, again I will use (x > y | x != y). So, the first statement before OR is x > y, it is FALSE because x is 10, y is 13. After the OR, I have x != y, so that is TRUE. So, I have FALSE or TRUE. So, what should I get? I should get TRUE because a FALSE statement OR a TRUE statement should give me result as TRUE. That is what our truth table said.

Let us try again for another one. So, suppose, in this case we write (x > y | y == x). So, both the statement, x > y and y = x, are both are FALSE and I have connected them using a OR operator. So, FALSE or FALSE, it should give me FALSE. That is simple to use this operator. You can also try to use NOT operation. For example, if I say !(x < y). Let us see what happens. So, I have put a statement inside the first bracket, the x < y, so that is true. So, if I put a NOT before that, it should become FALSE. That is what it has done, it has reported FALSE.

So, this brings us to the end of this lecture. What we have learned in this lecture? We have learned about three common operators, three types of common operators in R; those are arithmetic operator, relational operator and logic operator. Most of the algorithm, most of the code, that you will use in R, we will use all these operators and perform the computation. So, thank you for joining me today. Happy learning. See you in the next video.