Introduction to R Software
Swayam Prabha

Lecture 14

Logical Operators: More Operations

Shalabh

Department of Mathematics and Statistics

Indian Institute of Technology Kanpur

Slides can be downloaded from

http://home.iitk.ac.in/~shalab/sp

Logical Operators and Comparisons

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

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

Logical Operators and Comparisons

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

Operator	Executions
`xor()`	either... or (exclusive)
`isTRUE(x)`	test if **x** is TRUE
`TRUE`	true
`FALSE`	false

Examples:

```
> 8 > 6

[1] TRUE


> 9 < 4

[1] FALSE
```
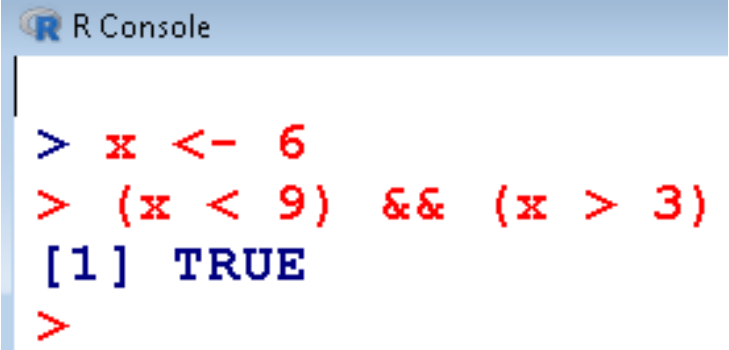
Is 7 less than 5?

```
> isTRUE(7<5)

[1] FALSE
```

Is 4 greater than 3?

```
> isTRUE(4>3)

[1] TRUE
```

Examples:

```
> x <- 6

> (x < 9) && (x > 3)     # && means AND

[1] TRUE
```

Examples:

```
> x <- 6
```

Is **x** less than 9 or **x** is greater than 4 ?
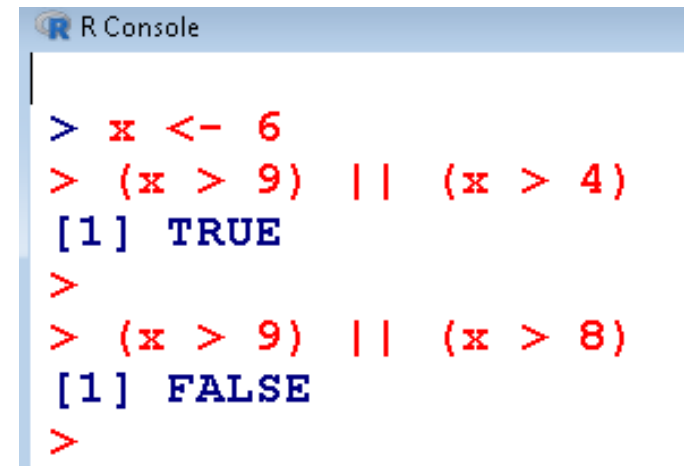
```
> (x < 9) || (x > 4)    # || means OR

[1] TRUE
```

Is **x** greater than 9 or **x** is greater than 8 ?

```
> (x > 9) || (x > 8)

[1] FALSE
```

R Console

```
> x <- 6
> (x > 9) || (x > 4)
[1] TRUE
>
> (x > 9) || (x > 8)
[1] FALSE
>
```

Examples:

```
> x = 20
> y = 30
```

Is **x** equal to 20 and is **y** equal to 30?

```
> (x == 20) & (y == 30) # == means exactly

[1] TRUE                                equal to
```

Is **x** equal to 20 and is **y** equal to 2?

```
> (x == 20) & (y == 2)

[1] FALSE
```

Examples:

```
> x = 20
> y = 30
```

Is **x** equal to 1 and is **y** equal to 20?

```
> (x == 1) & (y == 20)        # == means exactly

[1] FALSE                              equal to
```

Is **x** equal to 1 and is **y** equal to 2?

```
> (x == 1) & (y == 2)

[1] FALSE
```

R Console

```
> (x == 1) & (y == 20)
[1] FALSE
>
> (x == 1) & (y == 2)
[1] FALSE
```

Examples:

```
> x = 1:5                 # Generates x=1,2,3,4,5


> (x > 2) & (x < 4)    # Checks whether the
                          values are greater
                          than 2 and less than 2

[1] FALSE FALSE  TRUE FALSE FALSE
```

Examples:

```
> x = 1:5                # Generates x=1,2,3,4,5


> x[(x > 3) & (x < 5)]   # Finds which values
 [1] 4                       are greater than 2 and
                             smaller than 5
```

Examples:

```
R Console

> x = 1:5
> (x > 2) & (x < 4)
[1] FALSE FALSE  TRUE FALSE FALSE
>
> x[(x > 3) & (x < 5)]
[1] 4
>
```

Examples:

```
> x = 1:5                    # Generates x=1,2,3,4,5

> (x > 2) | (x < 4)   # Checks whether the
                         values are greater
                         than 2 or less than 4

[1] TRUE TRUE TRUE TRUE TRUE
```

Examples:

```
> x = 1:5                 # Generates x=1,2,3,4,5


> x[(x > 2) | (x < 4)] # Finds which values
                          are greater than 2 or
                          smaller than 4
[1] 1 2 3 4 5
```

Examples:
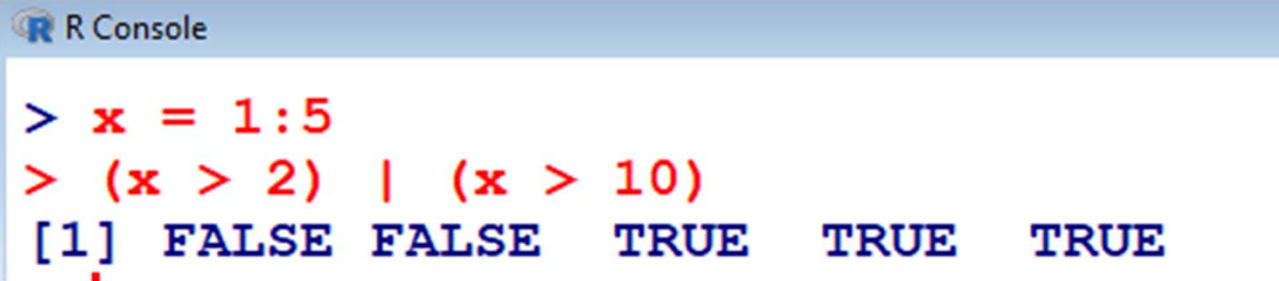
```
> x = 1:5
> (x > 2) | (x < 4)
[1]  TRUE  TRUE  TRUE  TRUE  TRUE
>
> x[(x > 2) | (x < 4)]
[1]  1 2 3 4 5
>
```

Examples:

```
> x = 1:5                   # Generates x=1,2,3,4,5

> (x > 2) | (x > 10)        # Checks whether the
                              values are greater than
                              2 or greater than 10

[1] FALSE FALSE  TRUE   TRUE   TRUE
```

Examples:

```
> x = 1:5                    # Generates x = 1,2,3,4,5


> x[(x > 2) | (x > 10)]  # Finds which values
                              are greater than 2 or
                              smaller than 10

[1] 3 4 5
```

Examples:

```
R Console

> x = 1:5
> (x > 2) | (x > 10)
[1] FALSE FALSE  TRUE   TRUE   TRUE
>
> x[(x > 2) | (x > 10)]
[1] 3 4 5
>
```

Logical Operators and Comparisons

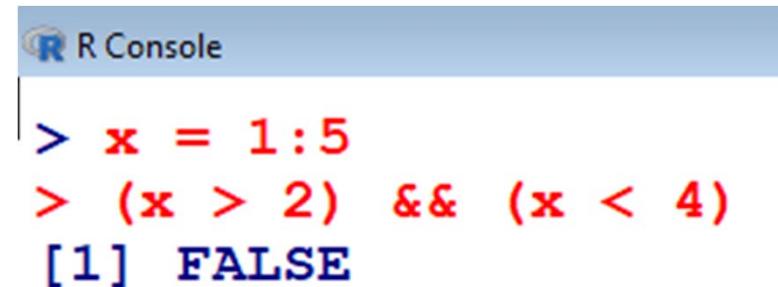| Operator	Executions
&, &&	and
|, ||	or

- The <u>shorter form</u> performs element-wise comparisons in almost the same way as arithmetic operators.

- The <u>longer form</u> evaluates left to right examining only the first element of each vector. Evaluation proceeds only until the result is determined.

Example of "The longer form evaluates left to right examining only the first element of each vector"

```
> x = 1:5          # Generates x = 1,2,3,4,5


> (x > 2) && (x < 4)

[1] FALSE
```
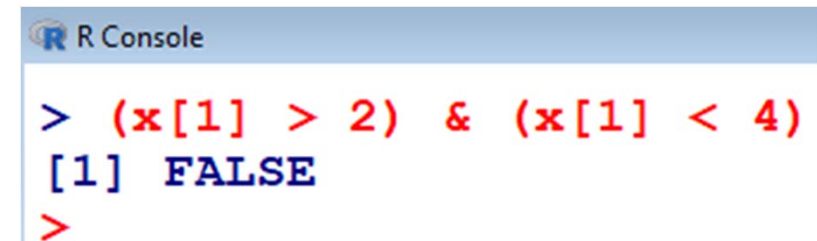

is equivalent to:

```
> (x[1] > 2) & (x[1] < 4)

   [1] FALSE
```


Note that `x[1]` is only the first element in `x`
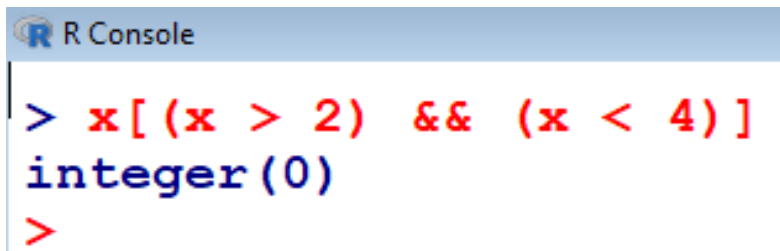
Example of "The longer form evaluates left to right examining only the first element of each vector"

(Contd...)

```
> x[(x > 2) && (x < 4)]   # Finds which values
 integer(0)                  are greater than 2 and
                             smaller than 4
```



```
> x[(x > 2) && (x < 4)]
integer(0)
>
```
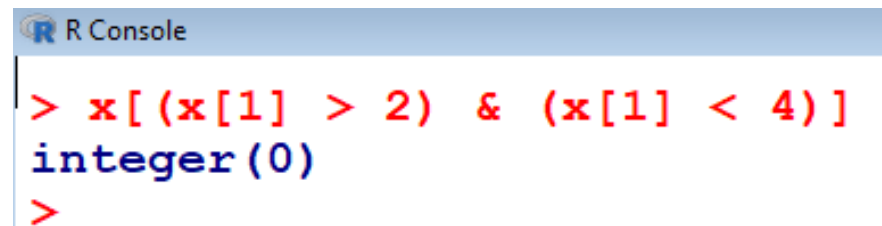
This statement is equivalent to

```
> x[(x[1] > 2) & (x[1] < 4)]

integer(0)
```



```
> x[(x[1] > 2) & (x[1] < 4)]
integer(0)
>
```