LAB100

Week 08: Logical Calculations.

Your aim for this workshop is to learn how to construct logical statements in RStudio. Upon completing this workshop you should be able to:

- Construct logical statements
- Perform functions on logical statements
- Use logical statements to manipulate vectors
- Construct simple simple if/else statements

Remember that you need to change the working directory and open a new R script at the start of each workshop.

1 Constructing Logical Statements

A logical statement compares two values and assesses whether the stated logical relationship is satisfied. The output will either be `TRUE` of `FALSE`. Below is a table contains a list the logical commands.

Command	**Meaning**
`==`	Equality ($=$)
`!=`	Not equal ($\neq$)
`<`	Less than ($<$)
`<=`	Less than or equal to ($\leq$)
`>`	Greater than ($>$)
`>=`	Greater than or equal to ($\geq$)

Type the following logical statements into your editor. Can you predict the validity of each statement before executing each command?

```
3 == 3
5.5 > 10
c(1,2,3,4,5) <= 3
c(1,2,3) != c(3,2,1)
```

These logical statements are performed on an element-wise basis, i.e. performs each comparison individually. Therefore the last example is equivalent to:

```
c(1!=3, 2!=2, 3!=1)
```

Alternatively, it is sometimes more useful to compare the output of two functions. To do this we type the each function that we wish to compare on the same command line, separated with the logical command of interest. Type the following examples into your editor and try to predict the validity of each statement before executing.

```
x <- 12
(x/6 + 1) == (x/3 - 1)
(x - 9) <= ( sin(pi/x) )
```

It is not essential to have parentheses around each function in these logical statements, but they help with reading your code and checking for any mistakes.

2 Combining Logical Statements

'Do I have 6 apples AND at least 4 pears?' This statement contains a comparison of two logical statements; namely `apples == 6` and `pears >= 4` with the AND Boolean logical operator. Therefore, if both logical statements need to be true for the whole statement is true, otherwise the statement is false. The three main Boolean operators are 'conjunction' (also known as AND, `&`), 'disjunction' (also known as OR, `|`) and 'negation' (also known as NOT, `!`). Below is a truth table to illustrate how these operators work on logical statements.

Statement 1	Statement 2	AND	OR	NOT
`a`	`b`	`(a) & (b)`	`(a) | (b)`	`!(a)`
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

To illustrate, consider the apples and pears statement above. Type the the following command into your editor and execute.

```
apples <- 6
pears <- 3
(apples == 6) & (pears >= 4)
```

Try different numbers for apples and pears to determine what values will give you a truthful statement.

3 Subsetting

Recall from the previous workshop of how to construct vectors, either by using the `c`ombine, `seq`uence, `rep`eat or colon (`:`) commands. Extracting parts of a vector is called subsetting. RStudio allows us to do this by using square brackets (`[ ]`) which directly follows the object name of the vector that we wish to subset. Only a single integer, a vector of integers or a vector of logical constants can be entered into the subsetting brackets. Type the following into your editor and execute.

```
x <- seq(-5, 5, by=1)
x[ 2 ]
x[ 2:10 ]
x[ -6 ]
```

The first line constructs a vector of integers between -5 and 5. The integer `2` in the next line indicates that we wish to extract the second element of the vector `x`, hence returning the value `-4`. The third command line contains the vector of integers from `2` to `10` within the subsetting brackets. This extracts the information contained in `x` from the second to the tenth elements, returning the sequence in integers between -4 and 4. The minus sign within the square brackets means that the indicated element(s) are to be removed. Therefore, the fourth line of code requests that the vector `x` should be returned without the sixth element. Never put a number into the square brackets that is larger than the length of the object vector; for example, the command `x[12]` will return `NA`, which means Not Available.

Subsetting with a logical vector is more intuitive as each element of the object vector will only be returned whenever the respective element in the logical vector is `TRUE`. Type the following into your editor and try to predict what RStudio will return before executing.

```
logic <- c( rep(FALSE,6), rep(TRUE,5) )
x[ logic ]
```

IMPORTANT: When using logical constants to subset a vector, ensure that the object and logical vectors are the same length. `length(x) == length(logic)`.

Instead of manually constructing a vector of logical constants, we can construct a logical vector using the logical commands and operators previously discussed in this workshop. The previous example retuned the elements of `x` such that $x \geq 1$. Using logical commands, an alternative way of performing this subsetting can be done by:

```
condition <- x>=1
x[condition]
```

To assess how many components of the vector `x` have $x \geq 1$ we can either perform

```
sum(condition)
```

which gives the number that are TRUE because they are taken to have value 1 whereas the FALSE statements are given value 0. Alternatively we can use

```
length(x[condition])
```

because the function `length` tells us the number of elements (or length) of a vector and the subsetting command `[condition]` has ensured that only the elements for which the condition is true have been retained.

4 if/else Statements

An if/else statement executes a particular statement when a specified condition is satisfied or a second statement otherwise. In mathematical notation, these statements are useful when we have a piecewise function of the form:

$$f(x) = \begin{cases} \text{STATEMENT 1} & \text{if CONDITON} \\ \text{STATEMENT 2} & \text{otherwise} \end{cases}$$

The general command structure for an if/else statement is:

```
{
  if( <CONDITION> ){
    <STATEMENT1>
  }
  else{
    <STATEMENT2>
  }
}
```

The curly brackets (`{ }`) are used around a collection of statements such that RStudio reads them in one go. For this if/else statement, RStudio first examines the condition `<CONDITION>` and if it is determined to be true, `<STATEMENT1>` is executed. However if the condition is proven to be false, `<STATEMENT2>` is executed instead. The outer most curly brackets ensure that RSudio reads the `if` and `else` statements together.

Consider the following piecewise function that calculates the absolute value:

$$|x| = \begin{cases} x & \text{if } x \geq 0 \\ -x & \text{otherwise} \end{cases}$$

An example of an if/else statement that calculates $|-3|$ is as follows:

```
x <- -3
{
  if( x>=0 ){
    modx <- x
  }
  else{
    modx <- -x
  }
}
modx
```

Type this into your editor and run it. Check that you get the correct result by using the `abs()` command. Try different values for `x` to make sure that the if/else statement does what you expect.

Quiz 1: Command matching

Which of the following commands evaluates the logical statement

$$\arctan(x) \leq \log(\frac{3}{y}) < \sinh(z)$$

for scalars x, y, z assigned as objects `x`,`y` and `z` respectively?

(**A**) `(atan(x) <= log(3/y) | sinh(z) > log(3/y))`

(**B**) `(sinh(z) < log(3/y) & atan(x) < log(3/y))`

(**C**) `(log(3/y) < sinh(z) & 1/tan(x) <= log(3/y))`

(**D**) `(atan(x) < sinh(z) & atan(x) < log(3/y))`

(**E**) `(sinh(z) > log(3/y) & atan(x) <= log(3/y))`

Quiz 2: Positive elements in a sequence

Let

$$(a_1, a_2, \ldots, a_{100})$$

be a sequence with kth element

$$a_k = \sin(\frac{3k\pi}{44})$$

Find the number of elements of the sequence for which $a_k > 0$

Quiz 3: Elements within a range

For the same sequence,

$$(a_1, a_2, \ldots, a_{100})$$

find the number of elements of the sequence for which

$$-\frac{1}{2} < a_k \leq -\frac{1}{4}$$

Quiz 4: Comparison of sequences

A second sequence

$$(b_1, b_2, \ldots, b_{100})$$

has kth element

$$b_k = \cos(\frac{5k\pi}{13})$$

How many of the 100 pairs of elements satisfy the condition

$$|a_k| > \frac{1}{4} \quad \text{OR} \quad a_k < b_k$$

Quiz 5: If Statements

Let x be a scalar real number within the interval $(0, 50]$ assigned to object `x` If the following **R** code is peformed

```
if (x > 25) {
  y <- sum( x < 1:50 )
  }else {
  y <- sum( x > 0:49 )
}
```

then the object `y` represents

$$y \quad = \quad \begin{cases} \text{A} & \text{if} \quad x > 25 \\ \text{B} & \text{if} \quad x \leq 25 \end{cases}$$

for some statements **A** and **B**.

Match up **A** and **B** from the following list

i $\lfloor x \rfloor$

ii $\lceil x \rceil$

iii $\lfloor 50 - x \rfloor$

iv $\lceil 50 - x \rceil$

v $\lfloor 25 - x \rfloor$

where $\lfloor x \rfloor$ denotes rounding down to the nearest integer and $\lceil x \rceil$ denotes rounding up. In **R** these operations can be performed directly using `floor(x)` and `ceiling(x)`.
