if-else conditions

Luna L Sanchez Reyes

2023-03-23

Logical and conditional statements: Review

These are pieces of code that return the `TRUE` or `FALSE` values, that is, a logical value.

The common operators of logical statements are:

- equality `==`
- inequality `!=`
- greater than `>`
- less than `<`
- greater or equal than `>=`
- less or equal than `<=`

The conditional statements allow to test several logical conditions at a time. The condition operators (or symbols) are:

- AND `&` (inside dplyr functions we can also represent AND using a `,`)
- OR `|`

We also have logical functions that test if something is `TRUE` or `FALSE`, for example:

- `is.na()` is a function that tests if a value is an `NA`
- This function is part of a whole family of functions, they all start with `is.`:
- `is.vector()`
- `is.data.frame()`
- `is.factor()`

For next class: how to get all functions from a family (method).

- `which()` : takes logical vectors, it will give you the numerical index (position) of all values that are `TRUE`

```
letters == "r"
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
## [25] FALSE FALSE
```

```
which(letters == "r")
```

```
## [1] 18
```

```
letters[18]
```

```
## [1] "r"
```

Exercise 6

Create the following variables:

```
w <- 10.2
x <- 1.3
y <- 2.8
z <- 17.5
colors <- c("red", "blue", "green")
masses <- c(45.2, 36.1, 27.8, 81.6, 42.4)
dna1 <- "attattaggaccaca"
dna2 <- "attattaggaacaca"
```

Use them to print whether or not the following statements are TRUE or FALSE.

a) w is greater than 10
b) "green" is in colors
c) x is greater than y
d) Each value in masses is greater than 40.
e) 2 * x + 0.2 is equal to y
f) dna1 is the same as dna2
g) dna1 is not the same as dna2
h) w is greater than x, or y is greater than z
i) x times w is between 13.2 and 13.5

```
x * w < 13.5
```

```
## [1] TRUE
```

```
x * w > 13.2
```

```
## [1] TRUE
```

```
13.2 < x * w < 13.5
```

This is how we would do it on paper, but in R we ca only compare things in pairs. For this we use the conditional statements:

```
x *w < 13.5 & x * w > 13.2
```

```
## [1] TRUE
```

j) Each mass in masses is between 30 and 50.

How to make simple choices with `if ()`

The general structure of an if statement:

```
if (condition is TRUE) {
  Run all lines
  of code in
  this block
  of code
}
```

If the condition is not `TRUE`, then nothing happens.

Exercise 7

Complete the following if statement so that if `age_class` is equal to `"sapling"` it sets `y <- 10`:

```
age_class = "sapling"
if (){

}
```

```
 y
```

Create the variable `age_class`:

```r
age_class = "sapling"
```

How do you test if `age_class` is equal to "sapling"?

```r
age_class == "sapling"
```

```
## [1] TRUE
```

Now, complete the `if` statement:

```r
if (age_class == "sapling") {
  y <- 10
}
y
```

```
## [1] 10
```

Remember! Inside the parentheses you have to write a logical or conditional statement. If you forget the double equal sign, R will think you are trying to create a variable, and will throw an error:

```r
if (age_class = "sapling") {
  y <- 10
}
y
```

```
## Error: <text>:1:15: unexpected '='
## 1: if (age_class =
##                   ^
```

Case when we have two options: `if else` structure

The general form of this structure:

```
if (condition) {
  code that runs if condition IS met
} else {
 code that runs if condition is NOT met
}
```

Exercise 8:

Copy the following code and complete the if statement so that if `age_class` is equal to `"sapling"` it sets `y <- 10` and if `age_class` is equal to `"seedling"` it sets `y <- 5`.

We can solve this in a couple different ways.

First test if the variable is equal to "sapling", then cover all other conditions within the `else` block:

```r
 age_class = "seedling"
 if (age_class == "sapling"){
  y <- 10
 } else {
   print(age_class == "seedling")
   y <- 5
 }
```

```
## [1] TRUE
```

```
y
```

```
## [1] 5
```

Or, test first if `age_class` is equal to “seedling”, then cover anything else:

```r
if (age_class == "seedling") {
  y <- 5
} else {
  y <- 10
}
```

Handle more than 2 choices (3 choices or more)

In this case we are using the `else if` structure:

```
if (condition1) {
 first block code that is execites if condition 1 is met
} else if (condition2) {
 second block code that executes if condition2 is met
} else if (condition3) {
 more code
} else {
  this will cover all the conditions that are not specified before
}
```

- You do not have to end up with and `else` block.
- `else if` blocks are more intentional with the conditions.
- A simple `else` will run in all other cases no matter what.

Exercise 9

Complete the if statement so that if `age_class` is equal to “sapling” it sets y <- 10 and if `age_class` is equal to “seedling” it sets y <- 5 and if `age_class` is something else then it sets the value of y <- 0.

Start with `age_class = "adult"`.

```r
 age_class = "adult"
 if (age_class == "sapling"){
  y <- 10
 } else if (age_class == "seedling") {
   y <- 5
 } else {
   y <- 0
}
y
```

```
## [1] 0
```