if.else

2023-04-01

##Logical and conditional statements

These are the 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 then `>=` - 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 function we can 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 os anNA-This function i part of a whole family of functions, they all start withis.`: 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 a numerical index(position) of all values that are TRUE

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

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

```
letters[18]
```

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

```
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"
```

```
w > 10
```

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

```
colors = "green"
x > y
```

```
## [1] FALSE
```

```
masses > 40
```

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

```
(2 * x + 0.2) == y
```

```
## [1] FALSE
```

```
dna1 = dna2
dna1 != dna2
```

```
## [1] FALSE
```

```
w > x
```

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

```
x * w < 13.5
```

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

```
x * w > 13.2
```

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

```
#13.2 < x * w < 13.5 This is how we do it in paper
#but in R we have to compare things in pairs:
#for this we use the conditional statements
x * w < 13.5 & x * w > 13.2
```

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

```
masses < 30 & 50 > 30 & 50
```

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

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.

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

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

Case when we have two options: if-else structure

The general form of this structure:

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

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

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

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

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

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

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

Handle more than 2 choices

In this case we are using the elseif structure:

```
if(condition1){
first block code that is executes 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 are more intentionl with the conditions.
A simple ‘else‘ will run
}
}
}
```

Handling more than 3 choices

```
if((age_class == "sapling")) {
 y <- 10
} else if(age_class == "seedling") {
 y <-5
} else if((age_class == "adult")) {
 y <- 20
}
```

#Values of y by age class