conditional_logic

September 27, 2020

Conditional Logic in R

1 1. The `if` Statement

- The `if` statement lets us execute code given that a condition is true.
- Similar to many other languages and tools (e.g., Excel)

```
In [1]: if (5 > 4) {
            print("yep!!")
            x <- 555
            print(x)
        }
```

```
[1] "yep!!"
[1] 555
```

```
In [2]: x <- c("foo", "bar", "baz")

        if (length(x) == 3) {
            print("Yes, this is a vector of length 3! Hooray!!")
        }
```

```
[1] "Yes, this is a vector of length 3! Hooray!!"
```

1.1 1.1 When `if` "fails"

```
In [3]: if ("potato" == "fries") {
            print("you will never see this print")      # this never gets executed
        }
```

2 2. The `else` Statment

- `else` gives us a way to execute code when `if` block doesn't get executed

```
In [4]: if ("potato" == "fries") {
            print("you will never see this print")      # this never gets executed
        } else {
            print("both are delicious")                 # this DOES get executec
            v <- rnorm(10)
            print(v)
        }
```

```
[1] "both are delicious"
 [1] -0.85961697 -0.75159053  0.42964662 -0.06570654  0.60003514 -1.07040394
 [7] -0.73066550  0.50321874  0.42242876  0.75073097
```

2.1 2.1 Combining `else` and `if`

- We can use `else` and `if` together in sequence

```
In [5]: coin_value <- 5

        if (coin_value == 25) {
            print("washington")
        } else if (coin_value == 10) {
            print("fdr")
        } else if (coin_value == 5) {
            print("jefferson")
        } else {
            print("lincoln")
        }
```

```
[1] "jefferson"
```

3 3. Using `if` and `else` in Functions

- Like most other language features, `if` and `else` can be used in fuctions

```
In [6]: is_even <- function(n) {

            if (n %% 2 == 0) {       # Use `%%` to check if n/2 has remainder equal to 0
                res <- TRUE
            } else {
                res <- FALSE
            }
            return(res)
        }
```

```
In [7]: is_even(12341)                # use our newly defined function
```

FALSE

3.1 3.1 Nesting `if` Statements

- We can also nest our `if` statments in other `if` statements

```
In [8]: is_even("potato")           # is the string "potato" even?
```

```
        Error in n%%2: non-numeric argument to binary operator
    Traceback:


        1. is_even("potato")
```

3.1.1 3.1.1 Solve "Potato Problem"

- Recal our original `is_even()` function

```
In [ ]: is_even <- function(n) {

            if (is.numeric(n)) {
                if (n %% 2 == 0) {       # Use `%%` to check if n/2 has remainder equal to 0
                    res <- TRUE
                } else {
                    res <- FALSE
                }
            } else {
                res <- "Are you sure n is numeric?"
            }
            return(res)
        }
```

```
In [ ]: is_even("potato soup")
```