Chapter 5: Control Flow

Emeka Mbazor

4/2/2020

5.2.4 Exercises

1. What type of vector does each of the following calls to `ifelse()` return?

```
ifelse(TRUE, 1, "no")
```

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

```
ifelse(FALSE, 1, "no")
```

```
## [1] "no"
```

```
ifelse(NA, 1, "no")
```

```
## [1] NA
```

Read the documentation and write down the rules in your own words.

`ifelse()` evaluates the logical expression that is the first parameter. If the logical expression evaluates to `TRUE`, the function returns the second argument. If the logical expression evaluates to `FALSE`, the function returns the third argument. If the logical expression is a missing value, the function returns a missing value.

2. Why does the following code work?

```
x <- 1:10
str(length(x))
```

```
##  int 10
```

```
if (length(x)) "not empty" else "empty"
```

```
## [1] "not empty"
```

```
x <- numeric()
str(x)
```

```
##  num(0)
```

```
if (length(x)) "not empty" else "empty"
```

```
## [1] "empty"
```

The code works because `if()` accepts integers and treats nonzero integers as `TRUE` and the rest as `FALSE`.

5.3.3 Exercises

1. Why does this code succeed without errors or warnings?

```
x <- numeric()
out <- vector("list", length(x))
for (i in 1:length(x)) {
  out[i] <- x[i] ^ 2
}
out
```

```
## [[1]]
## [1] NA
```

This code succeeds without errors because `x` is a numeric vector of length 0 and `out` is a list of length 0. The `for` loop has two iterations since `i` counts down from 1 to 0. `x[1]` is `NA` because `NA ^ 2` results in `NA`. The second iteration doesn't do anything. This results with `out` being a list of length 1 with the only value being a missing one.

2. When the following code is evaluated, what can you say about the vector being iterated?

```
xs <- c(1, 2, 3)
for(x in xs) {
  xs <- c(xs, x * 2)
}
xs
```

```
## [1] 1 2 3 2 4 6
```

The vector being iterated is a seperate copy from the vector being altered within the `for` loop.

3. What does the following code tell you about when the index is updated?

```
for (i in 1:3) {
  i <- i * 2
  print(i)
}
```

```
## [1] 2
## [1] 4
## [1] 6
```

The `for` loop is updated only when it is initialized.