# Functions: Part 2

*Daniel Anderson*
*Week 7, Class 1*

# Agenda

- Review take-home midterm
- Purity (quickly)
- Function conditional
  - `if (condition) {}`
  - embedding warnings, messages, and errors

# Learning objectives

- Understand the concept of purity, and why it is often desirable
    - And be able to define a side effect
- Be able to change the behavior of a function based on the input
- Be able to embed warnings/messages/errors

# Take-home midterm review

# Purity

A function is pure if

1. Its output depends *only* on its inputs
2. It makes no changes to the state of the world

Any behavior that results in a function being impure is referred to as a *side-effect*

# Common side effect functions

- We've talked about a few... what are they?

  *A couple examples*

- `print`
- `plot`
- `write.csv`
- `read.csv`
- `Sys.time`
- `options`
- `library`
- `install.packages`

# Conditionals

# Example from lab

1. Write a function that takes two vectors of the same length and returns the total number of instances where the value is `NA` for both vectors. For example, given the following two vectors

```
c(1, NA, NA, 3, 3, 9, NA)
c(NA, 3, NA, 4, NA, NA, NA)
```

The function should return a value of `2`, because the vectors are both `NA` at the third and seventh locations. Provide at least one additional test that the function works as expected.

# How do you *start* to solve this problem?

~~Start with writing a function~~

Solve it on a test case, then generalize!

*Use the vectors to solve!*

```
a <- c(1, NA, NA, 3, 3, 9, NA)
b <- c(NA, 3, NA, 4, NA, NA, NA)
```

# One approach

```
is.na(a)
```

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

```
is.na(b)
```

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

```
is.na(a) & is.na(b)
```

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

```
sum(is.na(a) & is.na(b))
```

```
## [1] 2
```

# Generalize to function

```
both_na <- function(x, y) {
    sum(is.na(x) & is.na(y))
}
```

*What happens if not same length?*

# Test it

```
both_na(a, b)
```

```
## [1] 2
```

```
both_na(c(a, a), c(b, b))
```

```
## [1] 4
```

```
both_na(a, c(b, b)) # ???
```

```
## [1] 4
```

*What's going on here?*

# Recycling

- R will *recycle* vectors if they are divisible

```
data.frame(nums = 1:4,
           lets = c("a", "b"))
```

```
##   nums lets
## 1    1    a
## 2    2    b
## 3    3    a
## 4    4    b
```

- This will not work if they are not divisible

```
data.frame(nums = 1:3,
           lets = c("a", "b"))
```

```
## Error in data.frame(nums = 1:3, lets = c("a", "b")): arguments imply differing
```

# Unexpected results

- In the `both_na` function, recycling can lead to unexpected results, as we saw
- What should we do?
- Check that they are the same length, return an error if not

# Check lengths

- Stop the evaluation of a function and return an error message with `stop`, but only if a condition has been met.

## *Basic structure*

```
both_na <- function(x, y) {
    if(condition) {
        stop("message")
    }
    sum(is.na(x) & is.na(y))
}
```

# 04:00

Modify the code below to check that the vectors are of the same length. Return a *meaningful* error message if not. Test it out!

```
both_na <- function(x, y) {
    if(condition) {
        stop("message")
    }
    sum(is.na(x) & is.na(y))
}
```

# Attempt 1

- Did yours look something like this?

```
both_na <- function(x, y) {
    if(length(x) != length(y)) {
        stop("Vectors are of different lengths")
    }
    sum(is.na(x) & is.na(y))
}
both_na(a, b)
```

```
## [1] 2
```

```
both_na(a, c(b, b))
```

```
## Error in both_na(a, c(b, b)): Vectors are of different lengths
```

# More meaningful error message?

*What would make it more meaningful?*

State the lengths of each

```
both_na <- function(x, y) {
    if(length(x) != length(y)) {
        v_lngths <- paste0("x = ", length(x), ", y = ", length(y))
        stop("Vectors are of different lengths:", v_lngths)
    }
    sum(is.na(x) & is.na(y))
}
both_na(a, c(b, b))
```

```
## Error in both_na(a, c(b, b)): Vectors are of different lengths:x = 7, y = 14
```

# warnings

If you want to embed a warning, just swap out `stop` for `warning`

07:00

# Challenge

*This is a tricky one*

Modify your prior code to so it runs, but returns a warning, if the vectors are recyclable, and otherwise returns a meaningful error message.

Hint 1: You'll need two conditions

Hint 2: Check if a number is fractional with `%%`, which returns the remainder in a division problem. So `8 %% 2` and `8 %% 4` both return zero (because there is no remainder), while and `7 %% 2` returns 1 and `7 %% 4` returns 3.

# One approach

```
both_na <- function(x, y) {
    if(length(x) != length(y)) {
        lx <- length(x)
        ly <- length(y)

        v_lngths <- paste0("x = ", lx, ", y = ", ly)

        if(lx %% ly == 0 | ly %% lx == 0) {
            warning("Vectors were recycled (", v_lngths, ")")
        }
        else {
            stop("Vectors are of different lengths and are not recyclable:",
                 v_lngths)
        }
    }
    sum(is.na(x) & is.na(y))
}
```

# Test it

```
both_na(a, c(b, b))
```

```
## Warning in both_na(a, c(b, b)): Vectors were recycled (x = 7, y = 14)
```

```
## [1] 4
```

```
both_na(a, c(b, b)[-1])
```

```
## Error in both_na(a, c(b, b)[-1]): Vectors are of different lengths and are not
```

# Step back from the ledge

# How important is this?

- For most of the work you do? Not very
- Develop a package? Very!
- Develop functions that others use, even if not through a function? Sort of.

# Next time

- Non-standard evaluation
- Building up functions
- Thinking more about messages/warnings/erros