# Creating your own functions

## Learning the basics of R - Part 3

Ernest Guevarra

12 November 2024

# Outline

- Why write functions
- When to write functions
- How to write functions
- Practical session

# Why write functions

- Allow automation of common tasks in a more powerful and general way than *copy-and-pasting*;
    - You can give a function an evocative name that makes your code easier to understand;
    - As requirements change, you only need to update code in one place, instead of many.
    - You eliminate the chance of making incidental mistakes when you copy and paste (i.e. updating a variable name in one place, but not in another).
- Fewer global variables: When you run a function, the intermediate variables that it creates are not stored in your global environment. This saves memory and keeps your global environment cleaner.
- Better documentation: Well documented functions help the user understand the steps of your processing.
- Easier to maintain / edit: When you create a function for a repeated task, it is easy to edit that one function. Then every location in your code where that same task is performed is automatically updated.

# When to write functions

You should consider writing a function whenever you've copied and pasted a block of code more than twice (i.e. you now have three copies of the same code).

For example, take a look at this code.

What does it do?

```
df <- data.frame(
  a = c(1, 2, 1, 1, 1, 2, 1, 1, 2, 2),
  b = c(2, 2, 2, 1, 1, 1, 1, 2, 1, 2),
  c = c(1, 2, 1, 1, 2, 1, 2, 1, 2, 2)
)

df$a <- ifelse(df$a == 2, 0, df$a)
df$b <- ifelse(df$b == 2, 0, df$b)
df$c <- ifelse(df$c == 2, 0, df$c)
```

# When to write functions

Original data:

```
##    a b c
## 1  1 2 1
## 2  2 2 2
## 3  1 2 1
## 4  1 1 1
## 5  1 1 2
## 6  2 1 1
## 7  1 1 2
## 8  1 2 1
## 9  2 1 2
## 10 2 2 2
```

Recoded data:

```
##    a b c
## 1  1 0 1
## 2  0 0 0
## 3  1 0 1
## 4  1 1 1
## 5  1 1 0
## 6  0 1 1
## 7  1 1 0
## 8  1 0 1
## 9  0 1 0
## 10 0 0 0
```

This is a good example of when writing a function will be useful/beneficial.

# How to write functions

We can create a function called `recode_values()`:

```
recode_values <- function(x) {
  ifelse(x == 2, 0, x)
}
```

And apply it to the same data as follows:

```
df$a <- recode_values(df$a)
df$b <- recode_values(df$b)
df$c <- recode_values(df$c)
```

We get:

```
##    a b c
## 1  1 0 1
## 2  0 0 0
## 3  1 0 1
## 4  1 1 1
## 5  1 1 0
## 6  0 1 1
## 7  1 1 0
## 8  1 0 1
## 9  0 1 0
## 10 0 0 0
```

# How to write functions

- You need to pick a name for the function. In the example I used `recode_values` because this function recodes the values based on a specified rule (i.e., value of 2 is converted to 0).
- You list the inputs, or **arguments**, to the function inside `function`. Here we have just one argument. If we had more the call would look like `function(x, y, z)`.
- You place the code you have developed in body of the function, a `{` block that immediately follows `function(...)`.

# Questions?

# Practical session

We'll work through *Exercise 2 - Manipulating objects and creating new functions* in Practical R for Epidemiologists (https://practical-r.org/exercise2.html) as a GitHub Classroom assignment

# Thank you!

Slides can be viewed at https://oxford-ihtm.io/open-reproducible-science/session4.html

PDF version of slides can be downloaded at https://oxford-ihtm.io/open-reproducible-science/pdf/session4-r-basics-part3.pdf

R scripts for slides available here