WRITING FUNCTIONS

Jeff Goldsmith, PhD
Department of Biostatistics

When to write functions

"If you use the same code more than twice, write a function"

- everyone, basically

Why to write functions

- Makes your code easier to read
- Makes your code easier to change or fix
- Helps prevent mistakes, especially from copy-and-paste

What is a function?

- Like in math, a function takes inputs, does something, and returns a result
- In both, the goal is to abstract some process
  - $4 = 2^2$ is a specific calculation
  - $y = x^2$ is a function
  - `sum_x = x[1] + x[2] + x[3]` uses a specific computation
  - `sum_x = sum(x)` uses a function

- For computations or operations you define and need to repeat, write a function for arbitrary inputs to produce the corresponding outputs

Parts of a function

- Every function consists of
  - Arguments (inputs)
  - Body (code that does stuff)
  - Return objects (what the function produces)

- Each of these can be simple or complex

Arguments

- What goes in to your function
- These get used by the code in the body
  - e.g. `x` in `mean(x)`

- Can take default values, which define a function's input until a user overwrites them
  - e.g. `na.rm = FALSE` in `mean(x)`

- Names matter; use reasonable things
- Some common names can (and should) be used
  - `x`, `y`, `z` for vectors
  - `df`, `data` for data frames
  - `n` for number of rows / sample size

Body

- Do what you want to do with your code

- A common structure is
  - Data / input checks using conditional execution
  - Perform operations
  - Format output

Return

- Implicit (last value produced) or explicit (using `return()`)
- Single value (e.g. a p-value) or a collection (estimate, SE, statistic, p-value)
- Named (named vector, list, df) or un-named (value, vector)

Scoping

- Don’t need to understand in huge detail
- Will help prevent / identify errors

- Scoping is how R looks for variables
  - The “global environment” is the interactive workspace where you spend the vast majority of your time
  - Each time you call a function, a new environment is created to host it’s execution
  - If the function use a variable that isn’t defined in that environment, it will go looking in the global environment

- You usually don’t want your functions using stuff in your global environment

Conditional execution

- Sometimes you only want to do something if a condition is met
  - e.g. produce one output for numeric variables and a different one for factors
- This kind of execution is called conditional

- Follows basic logic rules:
  - `if (condition_1) { thing_1 }`
  - `else if (condition_2) { thing_2 }`
  - `else { thing_3 }`

- Proper formatting helps a lot

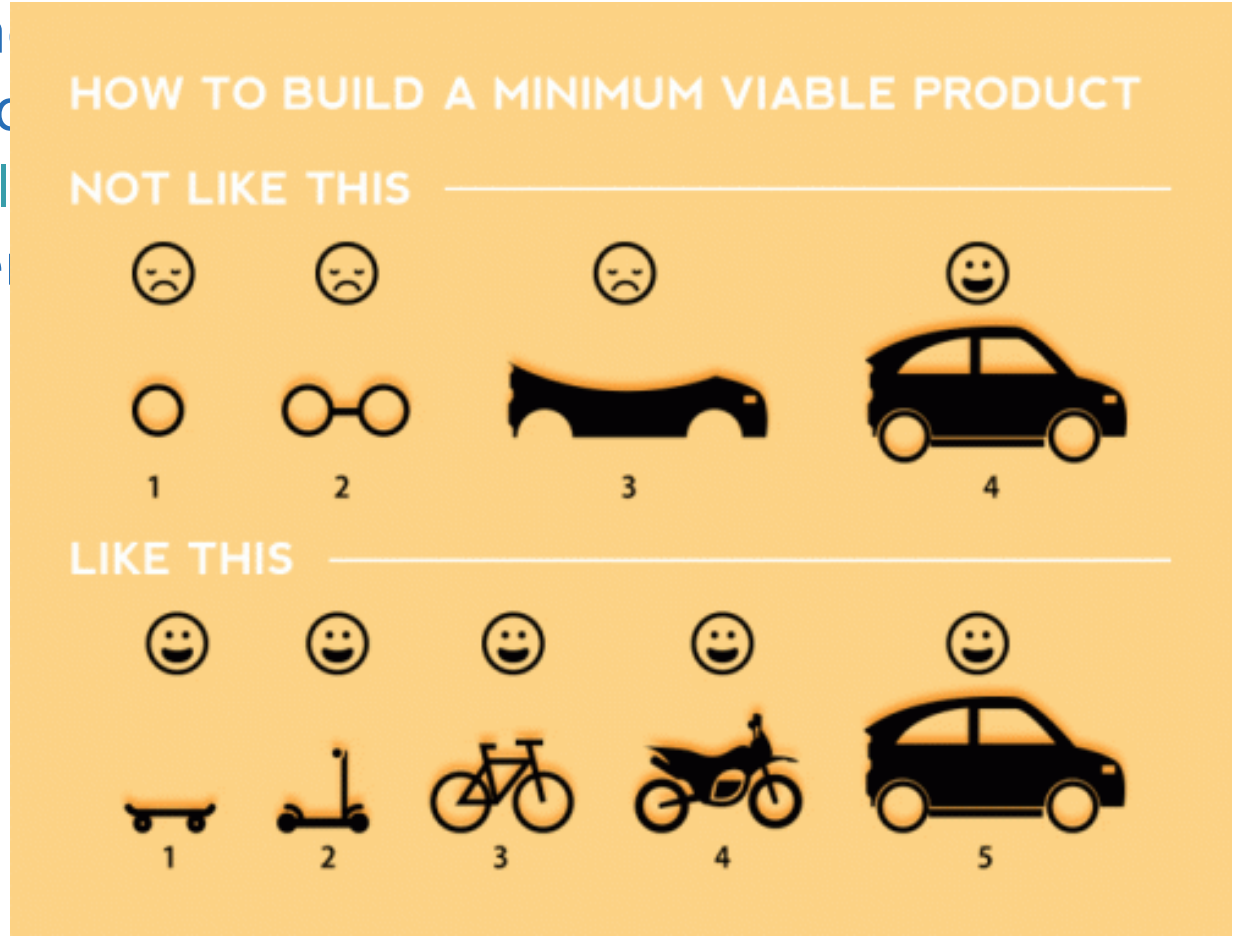
How to write functions

- Start small – with a working example, if possible
- Write small functions that do one thing well and interact easily
  - Avoid unneeded complexity
- Clarity is better than cleverness

Adapted from ”Basics of UNIX Philosophy”

How to write functions

- Start small – with a workin
- Write small functions that
  - Avoid unneeded compl
- Clarity is better than cleve



Attributed to Spotify Dev Team

Adapted from "Basics of UNIX Philosophy"