Don't Repeat Yourself - Removing repetition in your coding in R

Learning Objectives

After completing this tutorial, you will be able to:

- Be able to define the DRY principle.
- Describe how functions can make your code easier to read
- Apply the skill of looking at code and identifying tasks that could be replaced by functions.

What you need

You will need a computer with internet access to complete this lesson.

Efficient coding 101

If you remember from the first week of class, we discussed the concept of reproducibility. One component of reproducibility that we emphasized was making your code easier for your colleagues and your future self to work with. This is important, but there is an even more selfish reasons to consider writing efficient code.

Efficient coding will make your life easier too.

> Reproducibility is actually all about being as lazy as possible! – Hadley Wickham (via Twitter, 2015-05-03)

Don't Repeat Yourself - DRY

DRY (Don't Repeat Yourself) is a principle of software development. The focus of DRY is to avoid repetition of information.

Why?

When you write code that performs the same tasks over and over again, this means, that any edit that you make to even one of those tasks, needs to be made to every single instance of that task! Editing every instance of a task is a lot of work.

We can use functions in our code to replace tasks that we are performing over and over. Source: Francois Michonneau

Instead, we can create functions that perform those tasks, using sets of arguments to specify how the task is performed.

The benefits of functions

- **Modularity:** If you write function for specific individual tasks, you can use them over and over. A function that you write for one script can even be reused in other scripts!
- **Fewer environment 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 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.
- **Testing:** We don't discuss this in our class this week.

Modularity

By breaking down your analysis into functions, you end up with blocks of code with processing tasks that are explicit and clear.

Well crafted functions also remove repetition of the same sets of steps over and over. You can even re-use a function you create for other projects.

Fewer variables

When you code line by line, you end up creating numerous intermediate variables that you don't need to use again. These variables

1. accumulate in your environment and
2. consume memory!

When you run a function, intermediate varibales created are temporary. The result - is fewer variables in your environment. This means fewer variables that your computer has to store and fewer variable names that you have to worry about repeating / or conflicting.

Functions allow you to focus on the inputs and the outputs of your workflow rather than the intermediate steps.

Better documentation

Ideally, your code should be written so that it's easy to understand. However, what might seem clear to you now might be clear as mud 6 months from now or even 3 weeks from now (remember we discussed your future self in week 1 of this class).

Well written functions are documented with inputs and outputs clearly defined. Well written functions also use names that help you better understand what task the function performs. This makes your code easier to read for both you, your future self and your colleagues.

Easier to maintain / edit

If all your code is written line by line, with repeated code in multiple parts of your document, it can be challenging to maintain.

Imagine having to fix one element of a line of code that is repeated many times. You will have to find and replace that code to implement the fix in EVERY INSTANCE it occurs in your code. This makes your code difficult to maintain.

Do you also duplicate your comments where you duplicate parts of your scripts? How do you keep the duplicated comments in sync?

Re-organizing your code using functions (or organizing your code using functions from the beginning) allows you to explicitly document the tasks that your code performs.

Testing

While we won't cover this in our class this week, functions are also useful for testing. As your code gets longer, it is more prone to mistakes. For example, if your analysis relies on data that gets updated often, you may want to make sure that all the columns in your spreadsheet are present before performing an analysis. Or that the new data are not formatted in a different way.

Changes in data structure and format could cause your code to not run. Or in the worse case scenario, your code may run but return the wrong values!

If all your code is made up of functions, that have built in tests to ensure that they run as expected, then you can control the input and test for the output. It is something that would be difficult to do if all of your code is written, line by line with repeated steps.

Summary

In short, it is a good idea to learn how to

1. Think about your code in a modular way rather than writing it line by line
2. Write functions for parts of your code which include repeated steps
3. Document your functions clearly, specifying the structure of the inputs and outputs.

   input –> function does something –> output