Basic Computing 2 activity — code snippets and creating functions*

Peter Carbonetto *University of Chicago*

In this activity, will examine short code examples ('code snippets') that current and former BSD graduate students have used in their research. Our primary aim is to rework these code snippets as functions, and understand the benefits of doing so.

Instructions

Locate the "basic_computing_2" folder on your computer. Within that folder, you should see a subfolder, "snippets". This folder contains working examples of code that we will study closely, both separately in our teams, and together as a group.

Each of these code examples is also a *script*—that is, the code can be run in a clean R environment and should produce a useful result (in some cases an R package might need to be installed first). For example, try running `source("popmean.R")` in a clean workspace (make sure your working directory is set to the "snippets" directory).

We will begin this exercise by working through these examples with our teams. Feel free to split into smaller groups if that works better. Afterward we will share our improvements and discuss the code together as a class.

Aims

Get acquainted with the code. To understand what the code does, two useful strategies are: (1) try running each line of code and inspecting the results; (2) use R's documentation (*i.e.,* the "help" function) to learn about functions used. Add comments to the code as needed. Here is an example of comments added to the `popmean.R` snippet to aid understanding:

```
library(dplyr)
data(iris)

# Set the "Species" column to be the grouping variable.
iris_grouped <- group_by(iris, .data[["Species"]])

# Compute the mean petal length, separately for each group (i.e., species).
iris_summarized <- summarize(iris_grouped,
                             mean_value = mean(.data[["Petal.Length"]]))
```

*This document is included as part of the Basic Computing 2—Introduction to R tutorial packet for the BSD qBio Bootcamp, University of Chicago, 2022. **Current version**: August 11, 2022; **Corresponding author**: pcarbo@uchicago.edu. Thanks to Katie Aracena, Arjun Biddanda, John Blischak, Jennifer Blanc, Maryn Carlson, Grace Hansen and Suraj (Neil) Sheth for providing code examples.

```
# The summarize function outputs a "tibble", but I prefer a basic data frame.
iris_summarized <- as.data.frame(iris_summarized)
```

Separate specific concerns from general ones. Our second aim is to separate out parts of the code that are specific to this example in order to better isolate the bits that could be more generally useful. Two important strategies here are: (1) introducing new variables to keep track of the specific concerns, and (2) making variable names more general. Let's illustrate these ideas in the "popmean" example:

```
library(dplyr)
data(iris)
dat       <- iris           # The data frame to analyze.
group_col <- "Species"      # Column used for grouping.
mean_col  <- "Petal.Length" # Take means of this column.
grouped_data <- group_by(iris, .data[[group_col]])
summarized_data <- summarize(grouped_data,
                             mean_value = mean(.data[[mean_col]]))
summarized_data <- as.data.frame(summarized_data)
```

Notice that last four lines of code are not specific to the data set we are analyzing.

At first glance, this code seems more complicated than what we had before. What have we gained by doing this? Write down some motivations here.

Package the general code into a function to that can be easily reused. It is also helpful to accompany the function with comments giving a human-readable summarizy of what the function does (or what it is expected to do), and in particular what the inputs and outputs are. In the "popmean" example, our function might look like this:

```
# Compute population-specific means: "dat" is a data frame,
# "group_col" is the column giving the population labels, and
# "mean_col" is the column we will summarize by taking means.
calculate_pop_means <- function (dat, group_col, mean_col) {
  grouped_data <- group_by(dat, .data[[group_col]])
  summarized_data <- summarize(grouped_data,
                               mean_value = mean(.data[[mean_col]]))
  summarized_data <- as.data.frame(summarized_data)
  return(summarized_data)
}
```

Now we can apply this function to our data:

```
library(dplyr)
data(iris)
iris_summarized <- calculate_pop_means(iris, "Species", "Petal.Length")
```

Notice that not all of the results are outputted in `calculate_pop_means`. (In your code, deciding which outputs are worth keeping is your decision to make.)

How does packaging our code as a function help us? Write down some of the most important benefits here.