HW08: Wrangling II and Function Writing

Graphical Analysis of Biological Data

By the end of this assignment, you should be able to achieve the following tasks in R:

- use R notebooks and R markdown;
- insert, write, and evaluate code chucks;
- import data stored in a variety of text file formats;
- make untidy data tidy;
- mutate, summarize, and group data;
- use pipes;
- produce plots with `ggplot2`;
- use a typical workflow to wrangle and plot data;
- write custom functions, and
- confidently stage, commit, and push with Git.

These achievements belong to Learning Outcomes 2, 3, 4, 5, 6.

Click on any blue text to visit the external website.

This assignment has two parts.

Note: If you contact me for help or (better yet) open an issue in the public discussion forum, please include the code that is not working and also tell me what you have tried.

Preparation

- Open your `.Rproj` project file in RStudio.
- Create an `hw08` folder inside the same folder as your project file. You'll save your worksheet from part 1 in the `hw08` folder.
- Create a `scripts` folder inside the same folder as your project file. You'll save your R script from part 2 in the `scripts` folder.
- Remember to format your code properly, as described in the notes.

Part 1: Read chapter 5, sections 5-7.

- Download this R Notebook file. Right-click to save the file as `<lastname>_hw08.Rmd` inside the `hw08` folder. Make sure the file ends with `.Rmd` and not `.txt`.

Read the sections assigned in the worksheet. As you read the assigned sections, insert a code chunk in your notebook and recreate *every* example in that section, except for those I tell you in the notebook to skip. *Type the code!* In the next assignment, you will have to type the code from scratch so typing the code will help you remember it better. The point here is *learn* how to do this!

- **Do not answer any of the questions this week.**
- **Stage and commit regularly!** A good habit would be to stage and commit after you complete each section. You should also push each time but at least push your completed notebook to your remote repo.

Note: If you have a problem with your code that you want to ask about, commit and push your notebook, including your non-working code, to your Github repo. I can pull your notebook to run your code chunks. That helps me help you because I can see what you have tried.

Part 2: Write Two Functions

- Review the notes on how to write a function.
- Create a new R Script (not notebook) from the `File > New File` menu. Save the script as `my_functions.R` in the `scripts` folder. This will make your functions available for use with the next assignment. You do not need to load any packages in your script.

Standard error function

- Write a function called `std_err` that calculates standard error of the mean. You did review the notes, right?
- Your `std_err()` function must have two arguments. The first argument will accept the vector of numeric values for which your function calculates the standard error. The second argument should default to `na.rm = FALSE`.
- When finished, make R aware of your function. Use your cursor to highlight (select) your entire function, then press the `Run` button or press `cmd/ctrl + enter`. R will run your function code. If you have written a proper function, you will not get any feedback. If you click on the "Environment" tab of the upper right pane, you'll see your function listed, indicating R did not find any errors.

  If you get an error, then something is wrong with your code. *Read the error message closely and compare against your code to help track down the mistake.* You will have to find and fix the error in your code and run it again. Repeat until until you don't get an error.
- Test your function using a vector of the first 10 numbers of the Fibonacci sequence: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34. If your function works correctly, your result should be 3.48903. **Test your function from the console. Do not include the test code for this function or the next in your R script.**
- Add an `NA` to your fibonacci sequence and test your function again, with `na.rm = FALSE` and with `na.rm = TRUE`. Remember that the function default is `na.rm = FALSE`. You override it by passing `na.rm = TRUE` when you call the function.

Scaled mass index function

- Write a function called `scaled_mass` that calculates the *scaled mass index* (SMI). The index scales the mass of the bird to the length of the tarsus (part of the leg) so that birds of different sizes can be compared fairly.

  The equation to calculate SMI for each individual bird (i) is,

  $$\text{SMI}_i = M_i \times (L_0/L_i)^b,$$

  where L_0 is the mean tarsus length of all birds in the sample, M_i and L_i are the mass and tarsus length of each individual, and b is the slope estimate of the regression line on the log-transformed mass and log-transformed tarsus length.
- Your function should have three arguments. Use `mass` for mass, `tarsus` for tarsus length, and `slope` for the slope. Set the default for *each* argument to zero. You do this just like you set the default for `na.rm` to `FALSE` in your `std_err` function.
- Make R aware of your `scaled_mass` function as you did for the `std_err` function.

- For data, use `mass <- c(26.7, 22.8, 25.7, 26.1, 23.9)` and `tarsus <- c(18.4, 17.5, 18.4, 18.0, 18.2)`. These values are the first five values from the data set you will use in the next assignment.
- Use `slope = 1.5`. *Do not hard-code this value* into your function. Pass it as an argument. You'll see why next week.
- Your working function will return a vector of

```
[1] 26.04968 23.98257 25.07404 26.31780 23.70329
```

Things to remember for your functions

- You are writing code in a script (.R) not a notebook (.Rmd).
- Use comments `#` liberally to document your functions. Here's a suggested format. Use several lines of comments to tell the name of your function and describe briefly the purpose of your function. Use comments interspersed among your code to describe the different steps.

```
### Standard Error function
## Calculate the standard error of the mean.
## The first argument accepts a vector of numeric values
## The second argument tells whether to remove NA. Detaults to FALSE.
my_function <- function(arg1, arg2 = FALSE) {
  code
  # Comments
  More code  # Comments can also be on the same line after code.
  etc.
}
```