Module 2, Assignment 2

Ellen Bledsoe

2022-10-04

Assignment Details

Purpose

The goal of this assignment is to assess your ability to write custom functions and iterate code using a `for loop`.

Task

Write R code which produces the correct answers and correctly interpret the plots produced.

Criteria for Success

- Code is within the provided code chunks
- Code is commented with brief descriptions of what the code does
- Code chunks run without errors
- Code produces the correct result
  - Code that produces the correct answer will receive full credit
  - Code attempts with logical direction will receive partial credit
- Written answers address the questions in sufficient detail

Due Date

October 11 at midnight MST

Assignment Questions

For this assignment, we will continue using the `penguins` data set from the `palmerpenguin` data package.

You should reference the 2_Functions and 3_SickFish lessons to complete this assignment.

1. Load both the `tidyverse` and `palmerpenguin` packages into our work space. (2 points)

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.8     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(palmerpenguins)
```

Optional If you want the `penguins` data frame to show up in the environment, run the following code chunk.

```
penguins <- penguins
```

Right now, the bill length, bill depth, and flipper length measurements are in millimeters (we know this from the column names). A colleague has asked for the data in centimeters.

---

2. First, let's write a custom function that will convert any value we give it from millimeters to centimeters. (3 points)

   Add in all the necessary information, including:

   - argument(s)
   - code to convert from mm to cm (hint: divide by 10)
   - what value the function should return

```
mm_to_cm <- function(mm){
  cm <- mm / 10
  return(cm)
}
```

Don't forget to run the code chunk above to add your function to the environment!

3. Let's test out our function to make sure it works. First, test the function with the number 42. Then test it with the vector called `vec` that is already in the code. (2 points)

```
# run the line of code below to create a vector called `vec`
vec <- c(66, 39, 40, 2, 439, 0, 54)

mm_to_cm(42)
```

```
## [1] 4.2
```

```
mm_to_cm(vec)
```

```
## [1]  6.6  3.9  4.0  0.2 43.9  0.0  5.4
```

4. Now that we have our new function working, we can iterate! Let's use it to create 3 new columns: `bill_length_cm`, `bill_depth_cm`, and `flipper_length_cm`. (3 points)

   Use the `mutate` function to do this, and call this new data frame `penguins_cm`.

   Note: you can either put multiple arguments into the `mutate` function (as we've done with `filter` and `summarize`) or you can overwrite `penguins_cm` multiple times, but I want all three new columns in the new `penguins_cm` data frame.

```
penguins_cm <- penguins %>%
  mutate(bill_length_cm = mm_to_cm(bill_length_mm),
         bill_depth_cm = mm_to_cm(bill_depth_mm),
         flipper_length_cm = mm_to_cm(flipper_length_mm))
```

Our colleague has now also asked us for the penguin body mass data to be in kilograms, not grams. (If he learned how to code, he could easily convert this data himself!)

5. First, we want to write a function that will convert grams into kilograms. (3 points)

   Add in all the necessary information, including:

   - argument(s)
   - code to convert from g to kg (hint: there are 1000 g in 1 kg)
   - what value the function should return

```
g_to_kg <- function(g){
  kg <- g / 1000
  return(kg)
}
```

6. Again, let's check that our function works, also using 42 and `vec`. (1 points)

```
g_to_kg(42)
```

```
## [1] 0.042
```

```
g_to_kg(vec)
```

```
## [1] 0.066 0.039 0.040 0.002 0.439 0.000 0.054
```

7. This time, instead of using `mutate`, let's write a `for loop` to create a new column in the `penguins_cm` data frame. (4 points)

   First, we need to create a new column called `body_mass_kg`. Fill in the missing code (____) with the appropriate code. Remember to use the `$` operator to indicate the column!

```r
penguins_cm$body_mass_kg <- NA

for (i in 1:nrow(penguins_cm)){
  penguins_cm$body_mass_kg[i] <- g_to_kg(penguins_cm$body_mass_g[i])
}
```

8. Finally, let's create a data frame with only the columns our colleague needs–no columns with millimeters or grams. Call this data frame `penguins2`. (2 points)

```r
penguins2 <- penguins_cm %>%
  select(-bill_length_mm, -bill_depth_mm, -flipper_length_mm, -body_mass_g)
```