Homework 4: Writing Functions, Querying an API, and Tidy-Style Functions

Mike Maccia

```
#|message=FALSE
#|warning=FALSE

# Load in necessary packages
suppressPackageStartupMessages(library(tidyverse))
suppressPackageStartupMessages(library(knitr))
suppressPackageStartupMessages(library(kableExtra))
```

Task 1: Conceptual Questions

1. What is the purpose of the `lapply()` function? What is the equivalent `purrr` function?

```
t1_q1 <- "The `lapply()` function is used to apply functions to lists. For example,
if we used the function `mean()` within the `lapply()`, we could apply that function
to each list element. The equivalent function in the `purrr` package is `map()`,
which can be applied to 1 list. There is also `map2()` which can apply functions
to 2 lists and then `pmap()` can apply the function to any number of lists"
```

2. Suppose we have a list called `my_list`. Each element of the list is a numeric data frame (all columns are numeric). We want to use `lapply()` to run the code `cor(numeric_matrix, method = "kendall")` on each element of the list. Write code to do this below!

```r
#make a numeric list

df_1 <- data.frame(a = 1:5, b = 11:15)
df_2 <- data.frame (c = rnorm(5), d = rnorm(5))
df_3 <- data.frame(e = rnorm(4), f = c(2, 5, 22, 23))

my_list <- list(df_1, df_2, df_3)

t1_q2 <- lapply(my_list, FUN = cor, method = "kendall")
```

3. What are two advantages of using `purrr` functions instead of the `BaseR` apply family?

```r
t1_q3 <-"One advantage of `purrr` over the apply family is that it provides a
`tidyverse` alternative to the `apply()` family, allowing for consistent syntax
and the output is more predictable. Another advantage is that `purrr` allows you
shorthand to make anonymous functions."
```

4. What is a side-effect function?

```r
t1_q4 <- "Side-effect functions are functions that produce something but do not
change the data. Some examples of side-effect functions include `print()`,
`write_csv()`, or `plot()`"
```

5. Why can you name a variable `sd` in a function and not cause any issues with the `sd` function?

```r
t1_q5 <- "You can name a variable `sd` within a function and not cause any issues
with the `sd` function due to Lexical Scoping. When a function is called, the
environment that the function is performed in is temporary. Once the function is
complete, that variable created within the function is not saved. It only works
within the local function and not the `globalenv()`."
```

Creating a list with the above question answers

```r
task1_answer_list <- list(t1_q1, t1_q2, t1_q3, t1_q4, t1_q5)
```