Homework 6

For this homework you will create a github repo, set up github pages, clone the repo to your computer as an R project, create a `.qmd` file, and push those changes back to github to create a webpage! You'll submit the link to your github pages site (the one that looks like a nice website).

The steps for setting things up exist in the first two homework assignments and are not repeated here.

- Create a new `.qmd` document that outputs to HTML. You can give this a title of your choosing. Save the file in the main repo folder.
- In this document, answer the questions below.

Task 1: Conceptual Questions

On the exam, you'll be asked to explain some topics. How about some practice?! Create a markdown list with the following questions:

1. What is the purpose of the `lapply()` function? What is the equivalent `purrr` function?
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 use `lapply()` to run the code `cor(numeric_matrix, method = "kendall")` on each element of the list. Write code to do this below! (I'm really trying to ask you how you specify `method = "kendall"` when calling `lapply()`)
3. What are two advantages of using `purrr` functions instead of the `BaseR` apply family?
4. What is a side-effect function?
5. Why can you name a variable `sd` in a function and not cause any issues with the `sd` function?

Task 2 - Writing R Functions

1. When we start doing machine learning later in the course, a common metric used to evaluate predictions is called Root Mean Square Error (RMSE).
   For a given set of responses, $y_1, ..., y_n$ (variable of interest that we want to predict) and a set of corresponding predictions for those observations, $\hat{y}_1, ..., \hat{y}_n$ the RMSE is defined as

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

   Write a basic function (call it `getRMSE()`) that takes in a *vector* of responses and a *vector* of predictions and outputs the RMSE.

- If a value is missing for the vector of responses (i.e. an `NA` is present), allow for additional arguments to the `mean()` function (elipses) that removes the `NA` values in the computation.

2. Run the following code to create some response values and predictions.

```
set.seed(10)
n <- 100
x <- runif(n)
resp <- 3 + 10 * x + rnorm(n)
pred <- predict(lm(resp ~ x), data.frame(x))
```

- Test your RMSE function using this data.
- Repeat after replacing two of the response values with missing values (`NA_real_`).
  - Test your RMSE function with and without specifying the behavior to deal with missing values.

3. Another common metric for evaluating predictions is mean absolute deviation given by

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Write a function called `getMAE()` that follows the specifications of the `getRMSE()` function.

4. Run the following code to create some response values and predictions.

```
set.seed(10)
n <- 100
x <- runif(n)
resp <- 3 + 10 * x + rnorm(n)
pred <- predict(lm(resp ~ x), data.frame(x))
```

- Test your MAE function using this data.
- Repeat after replacing two of the response values with missing values (`NA_real_`).
  - Test your MAE function with and without specifying the behavior to deal with missing values.

5. Let's create a **wrapper** function that can be used to get either or both metrics returned with a single function call. Do not rewrite your above two functions, call them inside the wrapper function (we would call the `getRMSE()` and `getMAE()` functions **helper** functions). When returning your values, give them appropriate names.

- The function should check that two numeric (atomic) vectors have been passed (consider `is.vector()`, `is.atomic()`, and `is.numeric()`). If not, a message should print and the function should exit.
- The function should return both metrics by default and include names. The behavior should be able to be changed using a character string of metrics to find.

6. Run the following code to create some response values and predictions.

```
set.seed(10)
n <- 100
x <- runif(n)
resp <- 3 + 10 * x + rnorm(n)
pred <- predict(lm(resp ~ x), data.frame(x))
```

- Test your new function using this data. Call it once asking for each metric individually and once specifying both metrics
- Repeat with replacing two of the response values with missing values (`NA_real_`).
- Finally, test your function by passing it incorrect data (i.e. a data frame or something else instead of vectors)

Task 3 - Querying an API and a Tidy-Style Function

For this section, you'll connect to the news API here: newsapi.org. You'll need to go to register for a key at that web site!

1. Use `GET()` from the `httr` package to return information about a topic that you are interested in that has been in the news lately (store the result as an R object). Note: We can only look 30 days into the past with a free account.
2. Parse what is returned and find your way to the data frame that has the actual article information in it (check `content`). Use the `pluck()` function from `purrr` to grab the `articles` element. Note the first column should be a list column!
3. Now write a quick function that allows the user to easily query this API. The inputs to the function should be the title/subject to search for (string), a time period to search from (string - you'll search from that time until the present), and an API key.

Use your function twice to grab some data (save each as an object)!

4. With one of your objects, summarize the `name` of the `source` for each article. That is, find a one-way contingency table for this information.
5. For each of your returned data objects, turn the `publishedAt` column into a date column using the `lubridate` package (see the PARSE DATE-TIMES section of the cheat sheet!). Then sort the two data frames, each by their new parsed date published column. Finally, create a new variable called `pub_diff` that is the difference in time between the articles' published dates (use `lag()` with `mutate()`). Save the modifications as new data frames.
6. With each of your resulting two data objects (each a data frame, which is a special case of a list) do the following actions:

- Choose one of your data frames. Subset the data frame to only return the date version of `publishedAt` and the `pub_diff` variables. Then use one call to the `map()` function to return the mean, standard deviation, and median of these columns. You should use a custom anonymous function using 'shorthand' notation (`\(x) ...`). Note that the `pub_diff` variable includes an `NA` so you'll need to set `na.rm = TRUE` in the calls to `mean(0`, `sd()`, and `median()`.

You're done. Way to go! **Copy the link to your nicely rendered site and turn that in for this assignment!**