Homework 3

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).

If you were unable to get RStudio and github connected, try to set up a meeting with Dr. Post or Gabby to get that figured out! For now, it is ok to use the web interface but you want to move past that method quickly!

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. **Use tidyverse functions and manipulations for most aspects of this homework.**

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. If your working directory is `myfolder/homework/`, what *relative* path would you specify to get the file located at `myfolder/MyData.csv`?
2. What are the major benefits of using R projects?
3. What is git and what is github?
4. What are the two main differences between a `tibble` and a `data.frame`?
5. Rewrite the following nested function call using baseR's chaining operator:

```
arrange(filter(select(as_tibble(iris), starts_with("Petal"), Species), Petal.Length <
    1.55), Species)
```

6. What is meant by long format data and wide format data? Which do we generally prefer for statistical analysis?

Under each question, answer the question. Use markdown to put your answer in a 'block quote' form (`> text to answer question`)

Task 2 Reading Delimited Data

Note: Use chaining where possible!

The data sets we'll use for this part comes from the UCI machine learning repository.

Glass data

The first data set is called `glass.data`. You'll need to open the raw data set to determine the type of delimiter. The data is available at: https://www4.stat.ncsu.edu/~online/datasets/glass.data.

- The description of the data (not super useful!):

  Vina conducted a comparison test of her rule-based system, BEAGLE, the nearest-neighbor algorithm, and discriminant analysis. BEAGLE isa product available through VRS Consulting, Inc.; 4676 Admiralty Way, Suite 206; Marina Del Ray, CA 90292 (213) 827-7890 and FAX: -3189. In determining whether the glass was a type of 'float' glass or not, the following results were obtained (# incorrect answers): Type of Sample Beagle NN DA Windows that were float processed (87) 10 12 21 Windows that were not: (76) 19 16 22 The study of classification of types of glass was motivated by criminological investigation. At the scene of the crime, the glass left can be used as evidence…if it is correctly identified!

- The variables and their descriptions:

Variable	Description
Id	Number 1-214
RI	Refractive index
Na	Sodium (unit measurement: weight percent in corresponding oxide, as are attributes 4-10)
Mg	Magnesium
Al	Aluminum
Si	Silicon
K	Potassium
Ca	Calcium
Ba	Barium
Fe	Iron

With the last variable being `Type of Glass` with values of – 1 building_windows_float_processed, – 2 building_windows_non_float_processed, – 3 vehicle_windows_float_processed, – 4 vehicle_windows_non_float_processed (none in this database), – 5 containers, – 6 tableware, – 7 headlamps.

1. Read this data into R directly from the URL using functions from the tidyverse. Notice that the data doesn't include column names - add those (in a manner of your choosing). Print out the tibble (just call the object name).
2. Start a chain that would overwrite the `Type_of_glass` variable using `mutate()`. Create a character string version (that is, replace 1 with "building_windows_float_processed", 2 with "building_win…", etc.) instead (see the variable descriptions above to give meaningful values). (If you are familiar with `factors`, feel free to use that instead of a character string variable - otherwise, think `if/then/else` via `ifelse()`.)
3. Continue your chain and keep only observations where the `Fe` variable is less than 0.2 and the Type of Glass is either "tableware" or "headlamp".

Yeast data

The second data set is called `yeast.data`. You'll need to open the raw data set to determine the type of delimiter. The data is available at: https://www4.stat.ncsu.edu/~online/datasets/yeast.data.

- The description of the data (not super useful!):

  The references below describe a predecessor to this dataset and its development. They also give results (not cross-validated) for classification by a rule-based expert system with that version of the dataset. Reference: 'Expert Sytem for Predicting Protein Localization Sites in Gram-Negative Bacteria', Kenta Nakai & Minoru Kanehisa, PROTEINS: Structure, Function, and Genetics 11:95-110, 1991.

- The variables and their descriptions:

Variable	Description
seq_name	Accession number for the SWISS-PROT database
mcg	McGeoch's method for signal sequence recognition.
gvh	von Heijne's method for signal sequence recognition.
alm	Score of the ALOM membrane spanning region prediction program.
mit	Score of discriminant analysis of the amino acid content of the N-terminal region (20 residues long) of mitochondrial andnon-mitochondrial proteins.
erl	Presence of 'HDEL' substring (thought to act as a signal for retention in the endoplasmic reticulum lumen). Binary attribute.
pox	Peroxisomal targeting signal in the C-terminus.
vac	Score of discriminant analysis of the amino acid content of vacuolar and extracellular proteins.
nuc	Score of discriminant analysis of nuclear localization signals of nuclear and non-nuclear proteins.
class	Localization site

1. Read this data into R directly from the URL using functions from the tidyverse. Notice that the data doesn't include column names - add those (in a manner of your choosing). Print out the tibble (just call the object name).

2. Start a chain that removes the `seq_name` and `nuc` columns.

3. Continue your chain to add columns corresponding to the mean and median of each numeric variable (`mcg`, `gvh`, `alm`, `mit`, `erl`, `pox`, and `vac`) at each `class` grouping (see the `across()` function as we did in the `dplyr` video!).

Task 2: Combining Excel and Delimited Data

The data set we'll use for this part comes from the UCI machine learning repository. There are two data sets that are 'related to red and white variants of the Portuguese "Vinho Verde" wine.' There are physicochemical variables and a quality score, as rated by experts.

Input variables (based on physicochemical tests):

- fixed acidity
- volatile acidity
- citric acid
- residual sugar
- chlorides
- free sulfur dioxide
- total sulfur dioxide
- density
- pH

- sulphates
- alcohol

Output variable (based on sensory data):

- quality (score between 0 and 10)

1. There is an excel version of the white wine data set available at https://www4.stat.ncsu.edu/~online/datasets/white-wine.xlsx.

    - Download this file
    - Place it in a folder you know (such as your working directory for your project)
    - Import the data from the first sheet using the `readxl` package
    - Print out the tibble (just call the object name)

2. When you print the data set out to the console, you may notice that some of the variable names are surrounded by backticks. This is because they are non-standard (they include a space in them). We can rename them in a number of ways. We'll do it by reading in the variable names from the 2nd sheet of the same file.

    - Read in the data from the 2nd sheet. This should return a data frame with one column containing alternative versions of the variable names.
    - Grab that column and overwrite the current column names (`colnames()`) of your white wine `tibble`.

3. Lastly, add a column to this data set to indicate the wines are white. That is, add a column that has values of 'white' for every observation.

4. There is a semi-colon delimited version of the red wine data set available at https://www4.stat.ncsu.edu/~online/datasets/red-wine.csv.

    - Read this in using the `readr` package. Be careful that the columns are read in as the correct type!
    - You should replace the variable names as done above
    - You should append a column denoting the `type` as "red"

5. Combine these two data sets into one data set. They both have the exact same columns so this is an easy append task!

    - Use the `dplyr::bind_rows()` function (see the help) to create one `tibble` containing all of the wine data.

6. Start a chain onr your new combined data object to filter the data to only look at high-quality wines (`quality` > 6.5) and wines that have a reasonable alcohol value (`alcohol` < 132).

7. Continue your chain to now sort the data from highest `quality` to lowest.

8. Continue your chain to select only the variables that contain `acid`, the `alcohol` variable, the `type` variable, and the `quality` variable.

9. Continue your chain to add the mean and standard deviation of the `alcohol` variable to the data set for each setting of the `quality` variable.

Task 3: Database Practice

Download the `Lahman.db` file associated with the *Connecting to Databases* notes/video.

1. Connect to the database and then look at all of the tables in the database.
2. Use the `tbl()` function and `dplyr` to **return** all of the data from the `Teams` table for the year 2015.
3. Repeat the above by using the `sql()` function within `tbl()` (here you have to write actual SQL!).

Either use `dplyr` or write SQL queries for the rest of these questions!

4. Return all of the players in the hall of fame, the year they were voted into the hall of fame, and their category (only those three variables!). See the `HallOfFame` table, the inducted variable is important here.
5. Combine the table from the previous question with the `People` table in order to have the `nameFirst` and `nameLast` names added to what was returned above.
6. Return only the `playerID` (manager ID in this case), `G`, `W`, and `L` columns from the `Managers` table. Use chaining to then:

   - Determine the overall win/loss records (sum of wins and sum of losses) for each of these hall of fame managers.
     - We haven't quite covered this so I'll help out. We want to use `group_by()` with `summarize()` rather than `mutate()`. Something like this:

```
tbl(con, "Managers") |>
 select(...) |>
 group_by(playerID) |>
 summarize(G_managed = sum(G, na.rm = TRUE),
        Total_W = sum(W, na.rm = TRUE),
        Total_L = sum(L, na.rm = TRUE))
```

   - Create a new variable that is the career win/loss percentage (`Total_W/G_managed`). (Note: I did this after `collect()` otherwise the column type has to be set…)
   - Sort the resulting data by the win/loss percentage variable (from largest to smallest).

7. Use the results from the previous two questions to answer this one! Return information for only the people that managed a team and were inducted into the hall of fame (regardless of their `category` in the hall of fame - you want to return all people in the hall of fame that ever managed at all). (Just return all of the variables from both of the above two questions - 9 total columns)

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