Homework 2

For this homework you will create a github repo, clone the repo to your computer as an R project, create a `.qmd` file, and practice working using a proper github workflow.You'll submit a pdf to Gradescope.

Your submission should include both the code and corresponding output/text that answers the question.

If you were unable to get RStudio and github connected, you may use the Posit Workbench interface located on Moodle. Then, please set up a meeting with Dr. Meyer or Shuvrarghya to get that figured out!

Note: There is a 24 hour late window, in which 10% will be deducted. We understand that you are busy/life happens. Please take advantage of this window if needed.

Step 1

- Head to github and create a new repo.
  - Be sure to make the repo puble and **do not** choose a `.gitignore`

Step 2

- Create a new R project from version control (as we did in the notes/videos) that clones this repository locally.
  - Recall you can click on the green button on the github.com repo website to copy the repo link.
  - A `.gitignore` file may be created in this process. That isn't a worry!

Step 3

- Create a new `.qmd` document that outputs to PDF. You can give this a title about programming in Base R. Save the file in the main repo folder.
- In this document, answer the questions below. **Use BaseR manipulations for all problems below to obtain full credit.**

Formatting your `.qmd` file

Outside of updating your YAML, please follow the instructions below for proper formatting.

Please recreate the Task section headers in your .qmd by using two `#`, followed by the header text (ex. Tast 1: Basic Vector practice).

For each question within the task, please put three `#`, followed by the question number (ex. Question 1).

Task 1: Basic Vector practice

Suppose we have data from a medical experiment on blood pressure. We have the following pre-treatment values for subjects 1 through 20:

- 130, 128, 116, 124, 133, 134, 118, 126, 114, 127, 141, 138, 128, 140, 137, 131, 120, 128, 139, 135

after treatment, the subjects were measured again (subjects 1 through 20 match)

- 114, 98, 113, 99, 107, 116, 113, 111, 119, 117, 101, 119, 130, 122, 106, 106, 124, 102, 117, 113

1. Create two vectors named `pre` and `post`. One vector corresponding to the pre measurements and one to the post measurements.
2. Assign `names` to the vector elements using the `paste()` function. Note that `names()` can be overwritten by a character vector. To quickly create the names, try running the code

```
paste("Subject", 1:20, sep = "_")
```

```
##  [1] "Subject_1"  "Subject_2"  "Subject_3"  "Subject_4"  "Subject_5"
##  [6] "Subject_6"  "Subject_7"  "Subject_8"  "Subject_9"  "Subject_10"
## [11] "Subject_11" "Subject_12" "Subject_13" "Subject_14" "Subject_15"
## [16] "Subject_16" "Subject_17" "Subject_18" "Subject_19" "Subject_20"
```

   Create the same names for each vector's elements.

Hint: Save the above code as a R object, and use this object to assign names to pre and post.

3. Calculate the change in blood pressure for each patient by subtracting post-treatment measurements from pre-treatment measurements. Recall that `R` does math elementwise! Save this calculation as a new object in R (also a vector) called `diff_op`. Include `diff_op` in your code chunk to print the results.
4. Calculate the average decrease in blood pressure across all patients.
5. Determine which patients experienced a decrease in blood pressure after treatment (a positive change). Use the `which()` function to just return the indices (and names) associated with this type of change.
6. Subset the vector of differences to only return those that have a positive change.
7. Calculate the average decrease in blood pressure for those where the blood pressure decreased (positive change).

Task 2: Basic Data Frame practice

Continue the previous example.

1. Create a data frame object with four columns corresponding to your data above: `patient`, `pre_bp`, `post_bp`, and `diff_bp`. Hint: You can use your R object in question 2 from task 1 as patient.
2. Return only rows where the `diff_bp` column is negative. (Use `[` or learn about the `subset()` function if you'd like. If you use `[`, don't reference the original vector from the first part, access the column of the data frame when looking at making a comparison.)

3. Add a new column to the data frame corresponding to `TRUE` if the `post_bp` is less than 120. Recall you can use `$` to access a column. If you reference a column that doesn't exist, and save a vector (of appropriate length in it), that vector becomes a column of your data frame! Similar to the previous question, don't reference the original vector from the first part, access the column of the data frame when looking at making a comparison.
4. Finally, print the data frame out nicely in your final document by modifying the code below appropriately.

```
knitr::kable(bp_df)
```

Task 3: List practice

Continue the previous example. Suppose we now also have data from another experiment where the 'treatment' was actually a placebo.

We have the following pre-treatment values for subjects 1 through 10 (different set of subjects):

- 138, 135, 147, 117, 152, 134, 114, 121, 131, 130

after treatment, the subjects were measured again (subjects 1 through 10 match)

- 105, 136, 123, 130, 134, 143, 135, 139, 120, 124

1. Create a new data frame with these data that is similar to the data frame from task 2 (including the new column). That is, include a patient, pre, post, diff, and normal (less than 120) column using the data above. Name this new data frame `bp_df_placebo`.
2. Now create and store a list with two elements:
    - 1st element named `treatment` and contains the first data frame you created.
    - 2nd element named `placebo` and contains the second data frame you created.

Name your list `bp_list`.

3. Access the first list element using three different types of syntax.
4. In one line, access the `placebo` data frame, `pre_bp` column.

Task 4: Control Flow Practice

Continue the previous example.

1. Suppose we want to characterize the post-treatment (or placebo) blood pressure measurement as `optimal` (≤ 120), `borderline` ($120 < bp \leq 130$), and `high` (> 130). First, create a new column in each data frame from above called `status`. You can do this via

```
your_df$status <- character(20) #or 10 depending on number of observations
```

   Note: You want to do this additional column to the data frames that are stored in your list (R doesn't do referencing to the original object). That is, you are inializing columns to be created in question 2.

2. For the non-placebo data frame (within the list), create a for loop and use if/then/else logic to create the `status` column's values.
3. Repeat for the placebo data frame (within the list).

Task 5: Function Writing

Continue the previous example. Suppose you would eventually have many datasets in the form of the two above. You want to write a function to do some things for you quickly.

1. Write a function that
    - takes in a list with two data frames in it (a `treatment` and a `placebo` data frame) as an argument. Give no default value.
    - takes in an R function (that would find a summary of a numeric column) with the default value being set to `"mean"` (notice this is a quoted string).
    - Finds the statistic of interest (as defined by the user input) for the `pre`, `post`, and `diff` columns of both data frames.
        - Use `my_fun <- get(stat)` within the function to get the function from the quoted string.
    - These six values should then be returned as a named list with meaningful names - this is a somewhat challenging part!
        - Hint: Create a vector of names that is created dynamically based on the statistic passed, create a vector with the actual statistic values, and then assign `names()` to your vector that you return. Then return that (an atomic vector with names can be returned instead of a list).
    - Finally, apply your function to you list of data frames from previous. Use it without specifying your statistic, with specifying your statistic as `"var"`, `"sd"`, `"min"`, and `"max"`.

Note: If you can not get your function to work, show your function code + how you would use your named function to calculate the requested statistics above for partial credit. Set this specific code chunk to `eval:false` so your document will render.

Turning in your assignment

Before turning in your assignment, make sure to render your document AND look through it to make sure it is formatting correctly + code does not spill off the page.

Next, make sure to push your final changes to your GitHub repo.

Now, take your final PDF, and submit it on Gradescope. Make sure that you select your pages correctly (or there will be a penalty). If you have questions on how to submit on Gradescope, please just ask!