Homework 2

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!

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 HTML. 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.**

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. 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.

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).

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`

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 this data in it that is similar to the data frame from task 2 (including the new column).
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.
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).
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!
        - I'm going to let you consider what to do but you might 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. 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"`.

- You can render the document to check things are looking good. Make sure that all code chunks show (and are evaluated). **Use headings to separate the sections.** Write text before each code chunk explaining what you are trying to do. Use markdown where appropriate (to create lists, bold things, etc.).

Step 4

(The subsequent steps are the same steps from homework 1 - more detail is given there.)

In your repo folder (locally), create a file called `_quarto.yml`. Open this file (perhaps in RStudio or a text editor) and place the following in the file (spacing is important!):

```
project:
  type: website
  output-dir: docs
```

Step 5

Now create a file called `.nojekyll` in your project repo. This file doesn't need to have anything in it! You just need that file there (it may be a hidden file after you create it. Github should still track it.)

Step 6

Open the terminal in RStudio and run the following code:

```
quarto render
```

Step 7

Push all changes up to your repo! You can do this via menus or the command line (or via the github web interface).

Step 8

Head to your github repo page. Go to settings, choose pages, and under "Branch" choose 'main' and change the folder to /docs. Then hit save!

Step 9

Wait about 2 minutes... Head back to your main github repo page. You'll now see a 'Deployments' section on the bottom right.

Click on that. Hopefully, after a minute you see a green check and something that says your site is ready!

Click on that and you should see a nicely rendered website! **Copy the link to that site and that is what you'll turn in for this assignment!**