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). Please see HW 1 for full instructions on that process!

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!

Setting up your environment:

Continue the example from the previous homework. There is a `.rda` file (basically a file with saved R objects) available in the homework activity on Moodle. This has a version of the list you created in the last part of the previous homework.

You should download this file, put it in your working directory and read it in via the code below:

```
load("hw2_list.rda")
```

You should see `bp_list` in your environment!

Task 1: Control Flow Practice

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 not the original data frames you had in your environment. Although R uses smart ways to avoid using excess memory, it doesn't overwrite the data frame stored in your list if you modify the original object.

   ```
   bp_list$treatment$status <- character(20)
   bp_list$placebo$status <- character(10)
   ```

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).

   ```
   #many ways to do this
   #one way
   for (i in 1:nrow(bp_list$treatment)){
     if (bp_list$treatment$post_bp[i] <= 120){
   ```

```
bp_list$treatment$status[i] <- "optimal"
  } else if (bp_list$treatment$post_bp[i] <= 130){
bp_list$treatment$status[i] <- "borderline"
  } else {
bp_list$treatment$status[i] <- "high"
  }
}

#second way
counter <- 1
for (i in bp_list$placebo$post_bp){
  if (i <= 120){
bp_list$placebo$status[counter] <- "optimal"
counter <- counter + 1
  } else if (i <= 130){
bp_list$placebo$status[counter] <- "borderline"
counter <- counter + 1
  } else {
bp_list$placebo$status[counter] <- "high"
counter <- counter + 1
  }
}
```

Task 2: Function Writing

Continue the previous example. Suppose you would eventually have many data sets 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 (our standard 1D 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"`.

```
calculate_stat <- function(list_of_df, stat = "mean"){
  vals <- c("pre", "post", "diff")
  types <- c("placebo", "treatment")
  my_names <- character(6)
```

```
  counter <- 1
  for (i in types){
for(j in vals){
      my_names[counter] <- paste(i, j, stat, sep = "_")
      counter <- counter + 1
}
  }
  fun_stat <- get(stat)
  to_return <- c(fun_stat(list_of_df$placebo$pre_bp),
                 fun_stat(list_of_df$placebo$post_bp),
                 fun_stat(list_of_df$placebo$diff_bp),
                 fun_stat(list_of_df$treatment$pre_bp),
                 fun_stat(list_of_df$treatment$post_bp),
                 fun_stat(list_of_df$treatment$diff_bp))
  names(to_return) <- my_names
  return(to_return)
}
calculate_stat(bp_list)
```

```
##    placebo_pre_mean   placebo_post_mean   placebo_diff_mean  treatment_pre_mean
##              131.90              128.90                3.00              131.60
## treatment_post_mean treatment_diff_mean
##              125.95                5.65
```

```
calculate_stat(bp_list, "var")
```

```
##    placebo_pre_var   placebo_post_var   placebo_diff_var  treatment_pre_var
##          149.87778          124.98889          341.33333           75.72632
## treatment_post_var treatment_diff_var
##           78.99737          117.81842
```

```
calculate_stat(bp_list, "sd")
```

```
##    placebo_pre_sd   placebo_post_sd   placebo_diff_sd  treatment_pre_sd
##         12.242458         11.179843         18.475209          8.702087
## treatment_post_sd treatment_diff_sd
##          8.888046         10.854419
```

```
calculate_stat(bp_list, "min")
```

```
##    placebo_pre_min   placebo_post_min   placebo_diff_min  treatment_pre_min
##                114                105                -21                115
## treatment_post_min treatment_diff_min
##                114                -24
```

```
calculate_stat(bp_list, "max")
```

```
##    placebo_pre_max   placebo_post_max   placebo_diff_max  treatment_pre_max
##                152                143                 33                151
## treatment_post_max treatment_diff_max
##                146                 21
```

- 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.).
- Render things appropriately to create your website. **Copy the link to that nicely rendered site and that is what you'll turn in for this assignment!**