Homework-3

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.

In this github repo, you will create a data folder. The data for this homework can be found on Moodle in Week 3.

For this assignment, start by librarying the `tidyverse` and `palmerpenguins` packages. We will need them for this assignment.

```r
library(tidyverse)
library(palmerpenguins)
```

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

Commit and push your changes (at a minimum) after each task you complete.

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 the tidyverse 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).

Use tidyverse manipulations for all problems below to obtain full credit

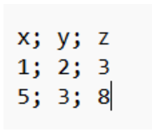
Task 1

The data for this task is called `data.txt` and `data2.txt`. Download these and put them in your data folder before answering the questions below.

We can use `read_csv` functions to read in data. CSV is a comma-separated file i.e. any **text file** that uses commas as a delimiter to separate the record values for each field. Therefore, to load data from a text file we can use the read_csv() method (or versions of it), even if the file itself does not have a .csv extension.

In the following question, we are going to read in `txt` data. Part a has us working with the `data.txt` file. Part b has you working with the `data2.txt` file.

a. Our first goal is to read in data that looks like this…

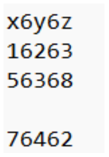


```
x; y; z
1; 2; 3
5; 3; 8
```

Pull up the help file for ?read_csv(). In 1-2 sentences, explain why we can not use specifically the read_csv() to read in these data.

Next, use a function found in that help file to read in the data. Name the object data and display the output. The output should be a 2 x 3 tibble, with the column names being x, y, and z represented as doubles.

b. Your fellow co-worker presents you the following data set. Their favorite number is 6, and have used the number 6 as a separator for the values in the data set. For example, the first value of x is 1, and not 16.



```
x6y6z
16263
56368

76462
```

Annoying as this may be, you have the tools to read in the data correctly. In the help file for `read_csv()`, use the appropriate function to read in the data set. In that function, read in x to be a factor, y to be a double, and z to be a character.

Hint: `col_types = "dc"`would change the data type of the first column to `d` representation, and the data type of the second column to be `c` representation. In the help file for `read_csv`, read the segment on column types to answer this question.

Task 2

The Portland Trailblazers are a National Basketball Association (NBA) sports team. These data reflect the points scored by 9 Portland Trailblazers players across the first 10 games of the 2021-2022 NBA season.

We are going to use these data to show off our data tidying skills. The data we will be using for this task is called `trailblazer`, and can be found on Moodle.

a. Take a `glimpse` of the `trailblazer` data set to show that you have read in the data correctly.

b. Pivot the data so that you have columns for Player, Game, Location, Points. Display the first five rows of your data set. Save your new data set as trailblazer_longer. Your data set should contain 90 rows and 4 columns.

c. Which players scored more, on average, when playing at home versus away? Answer this question using a single pipeline where you

– use pivot_wider to reshape the trailblazer_longer data frame such that you have a 90 x 4 tibble with columns Player, Game, Home, Away,

– and then create two new summary statistics (mean_home and mean_away) that represent the mean points scored for each player at home and each player away,

– and then create a third variable that represents the difference in points scored between points scored at home and points scored away. Arrange these values in descending order.

In 1 sentence, state which players scored, on average, more points at home through the first 10 games of the season than away.

Task 3

For the next tasks, we are going to use the `penguins` data set in the `palmerpenguins` package. Run `?penguins` in your Console if you would like to learn more about these data.

a. The `penguins` data set is in long format. Suppose that one of your collegues suggests that we pivot the data to be wider in order to create a data display table! They present you with the following code:

```
penguins |>
  select(species, island, bill_length_mm) |>
  pivot_wider(
    names_from = island, values_from = bill_length_mm
  )
```

```
Warning: Values from `bill_length_mm` are not uniquely identified; output will contain
list-cols.
* Use `values_fn = list` to suppress this warning.
* Use `values_fn = {summary_fun}` to summarise duplicates.
* Use the following dplyr code to identify duplicates.
  {data} |>
  dplyr::summarise(n = dplyr::n(), .by = c(species, island)) |>
  dplyr::filter(n > 1L)
```

```
# A tibble: 3 x 4
  species   Torgersen Biscoe      Dream
  <fct>     <list>    <list>      <list>
1 Adelie    <dbl [52]> <dbl [44]> <dbl [56]>
```

```
2 Gentoo    <NULL>     <dbl [124]> <NULL>
3 Chinstrap <NULL>     <NULL>      <dbl [68]>
```

What's going on here? Explain what….

– <NULL>

– `<dbl [52]>`

– and <list> means.

If you would like additional resources, check out: https://r4ds.hadley.nz/rectangling.html#list-columns!

b. Below is the table that your collegue was trying to create. Using the penguins data set, create this table!

```
`summarise()` has grouped output by 'species'. You can override using the
`.groups` argument.
```

```
# A tibble: 3 x 4
# Groups:   species [3]
  species   Biscoe Dream Torgersen
  <fct>      <dbl> <dbl>     <dbl>
1 Adelie        44    56        52
2 Chinstrap      0    68         0
3 Gentoo       124     0         0
```

Task 4

There are two NA values for bill length in the penguins data set. It just so happens that I know what the missing values are! The missing values for the Adelie species is 26, and the missing value for the Gentoo species is 30. In a single pipeline, replace the NA values in the bill length column with their appropriate value, and print out the first 10 rows of the tibble arranged in accending order to show that you have correctly filled in the NA values. Note: you do not need to fill in the NA values for any other column besides bill length.

If you would like more information on `case_when` check out the link here