STA 35B: Homework 1

Instructor: Akira Horiguchi

Student name: ABCDE FGHIJ; Student ID: 123456789

Due date: Apr 14, 2025 at 9 PM (PT)

Instructions

Upload a PDF file, named with your UC Davis email ID and homework number (e.g., ahoriguchi_hw1.pdf), to Gradescope (accessible through Canvas). You will give the commands to answer each question in its own code block, which will also produce output that will be automatically embedded in the output file. All code used to answer the question must be supplied, as well as written statements where appropriate.

All code used to produce your results must be shown in your PDF file (e.g., do not use `echo = FALSE` or `include = FALSE` as options anywhere). Rmd files do not need to be submitted, but may be requested by the TA and must be available when the assignment is submitted.

Students may choose to collaborate with each other on the homework, but must clearly indicate with whom they collaborated.

Problem 1

Consider the scores of students on exams:

```
scores <- tribble(
  ~id, ~midterm1, ~midterm2, ~final_exam,
  1, 80, 90, 85,
  2, NA, 100, 90,
  3, 75, 95, 60,
  4, 95, NA, 60,
  5, 95, 98, NA
)
```

(a) Use the `filter()` function from `dplyr` to see which students did better on their final exam than on their midterm 1.

```
# put your code here
```

(b) Use the `select()` function from `dplyr` to select all columns that contain the character string "id".

```
# put your code here
```

(c) Use the `summarize()` function to find the average score for midterm 1 and the average score for midterm 2. (All averages should omit any `NA` values.)

```
# put your code here
```

(d) We now have some extra information! Students with `id` 1, 2, and 5 are in the `A` section of the class, while students with `id` 3 and 4 are in the `B` section. To the tibble `scores`, add a new column called `section` to the tibble `scores`, which contains the section for each student. Then, for each section, use the `summarize()` function to find the average score for midterm 1 and the average score for midterm 2. (All averages should omit any `NA` values.)

```
# put your code here
```

(e) To the tibble `scores`, add a new column called `final_grade` that is calculated by the following:

- midterm contribution is 40%, and consists of the largest of the two midterm scores, with missing data treated as zeros
- final exam counting for 60%, with missing data treated as zero

Hint: you might find the function `replace_na()` in `dplyr` useful. You might also find the function `pmax()` useful.

```
# put your code here
```

Problem 2

Consider the following dataset:

```
reviews
```

```
# A tibble: 5 x 2
     id reviewtext
  <dbl> <chr>
1     1 I had a great experience, the product was as described.
2     2 Good, but not great. There were some issues.  Awfully crowded.
3     3 The service was excellent and the staff was very helpful.
4     4 I had an awful time.
5     5 Excellent, excellent!
```

(a) Use the function `filter()` from `dplyr` to return a tibble with only the rows where the sequence of characters "great" appears in `reviewtext` (with any lowercase/uppercase combos). *Hint*: you might find the functions `str_to_lower()` and `str_detect()` in `stringr` useful.

```
# put your code here
```

(b) Provide code which modifies `reviews` to have two new columns, `excellent` and `awful`, where:

- `excellent` is `TRUE` if the sequence of characters "excellent" appears in `reviewtext` (with any lowercase/uppercase combos) and `FALSE` otherwise
- `awful` is `TRUE` if the sequence of characters "awful" appears in `reviewtext` (with any lowercase/uppercase combos) and `FALSE` otherwise
- *Hint*: you might find the functions `str_to_lower()` and `str_detect()` in `stringr` useful.

```
# put your code here
```

(c) Use the `summarize()` function to calculate the percent of reviews where "awful" appears and the percent of reviews where "excellent" appears (any lowercase/uppercase combos).

```
# put your code here
```

Problem 3

Consider the following dataset with three variables,

- `customer`, an integer identifying the customer.
- `feedback`, a character string which has the input from a textbox plus `[Rating:#]` where `#` is expected to be a number between 1 and 5
- `day`: day on which feedback is given.

```
feedback_data
```

```
# A tibble: 6 x 3
  customer feedback                                          day
     <dbl> <chr>                                           <dbl>
1        1 Loved the service! [Rating:5]                       1
2        2 Unsatisfied with the product quality. [Rating:2]    1
3        3 Average experience. [Rating:3]                      1
4        4 Great product, but took too long. [Rating:4]        2
5        5 Not what I expected. [Rating:1]                     2
6        6 Not what I expected. [Rating:x]                     2
```

(a) Return a tibble, named `feedback_parsed`, with four columns: `customer`, `day`, `feedback_text`, `rating`.

- `customer` and `day` are as in the original tibble
- `feedback_text` has all of the text of `feedback` which appears before `[Rating:#]`
- `rating` is an integer if the `#` inside `[Rating:#]` is an integer, otherwise returns `NA`.

```
# put your code here
```

(b) The computation from part (a) should result in a tibble which looks like the following:

```
## A tibble: 6 × 5
#  customer feedback_text                       feedback   day rating
#     <dbl> <chr>                               <chr>    <dbl>  <dbl>
#1        1 "Loved the service! "               Loved t...     1      5
#2        2 "Unsatisfied with the product q...  Unsatis...     1      2
#3        3 "Average experience. "              Average...     1      3
#4        4 "Great product, but took too long. " Great p...     2      4
#5        5 "Not what I expected. "             Not wha...     2      1
#6        6 "Not what I expected. "             Not wha...     2     NA
```

Compute the average rating per day in `feedback_parsed`, ignoring all rows with missing data.

```
# put your code here
```

Problem 4

Consider the following dataset:

```
health_data
```

```
# A tibble: 3 x 5
  PatientID Weight_2019 Weight_2020 Height_2019 Height_2020
      <dbl>       <dbl>       <dbl>       <dbl>       <dbl>
1         1          70          72         170         171
2         2          65          68         165         166
3         3          80          82         180         181
```

(a) Transform this tibble into a tibble `long_health_data` so that there are four columns:

- PatientID, numeric type
- Weight, numeric type
- Height, numeric type
- Year, numeric type

Hint: Use `?pivot_longer` to read the documentation for the `names_to` and `names_sep` arguments of `pivot_longer()`.

```
# put your code here
```

(b) Transform the tibble `long_health_data` from part (a) into the tibble `wide_health_data` which is back in the wide format, with columns "Weight_2019", "Weight_2020", "Height_2019", "Height_2020". Check that your calculation was correct by checking `all.equal(wide_health_data, health_data)`.

```
# put your code here
```