STA035B Homework 1, due: 1/25, 9pm

Spencer Frei

Instructions

Upload a PDF file, named with your UC Davis email ID and homework number (e.g., sfrei_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
)
```

Return a tibble that is `scores` but with a new column, `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 may find the function `replace_na()` in `dplyr` useful.

```
scores %>%
  mutate(
    midterm1 =  replace_na(midterm1, 0),
    midterm2 =  replace_na(midterm2, 0),
    final_exam = replace_na(final_exam, 0),
    midterm_score = pmax(midterm1, midterm2),
    final_grade = 0.4*midterm_score + 0.6*final_exam
  )
```

```
# A tibble: 5 x 6
     id midterm1 midterm2 final_exam midterm_score final_grade
  <dbl>    <dbl>    <dbl>      <dbl>         <dbl>       <dbl>
1     1       80       90         85            90        87
```

2	2	0	100	90	100	94
3	3	75	95	60	95	74
4	4	95	0	60	95	74
5	5	95	98	0	98	39.2

Problem 2

Consider the following dataset:

```
reviews <- tribble(
  ~id, ~reviewtext,
  1, "I had a great experience, the product was as described.",
  2, "Good, but not great. There were some issues.  Awfully crowded.",
  3, "The service was excellent and the staff was very helpful.",
  4, "I had an awful time.",
  5, "Excellent, excellent!"
)
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) Provide code which modifies `reviews` to have two new columns, `excellent` and `awful`.

- `excellent` is TRUE if the phrase `excellent` appears (with any lowercase/uppercase combos) and `FALSE` otherwise
- `excellent` is TRUE if the phrase `excellent` appears (with any lowercase/uppercase combos) and `FALSE` otherwise

```
reviews %>%
  mutate(excellent = str_count(str_to_lower(reviewtext), "excellent") > 0,
         awful = str_count(str_to_lower(reviewtext), "awful") > 0)
```

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

(b) Provide code which calculates the percent of reviews where `awful` appears and the percent of reviews where `excellent` appears (any lowercase/uppercase combos), using `summarize`.

```
reviews %>%
  mutate(excellent = str_count(str_to_lower(reviewtext), "excellent") > 0,
         awful = str_count(str_to_lower(reviewtext), "awful") > 0) %>%
  summarize(percent_excellent = mean(excellent),
            percent_awful = mean(awful))
```

```
# A tibble: 1 x 2
  percent_excellent percent_awful
              <dbl>         <dbl>
1               0.4           0.4
```

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 <- tribble(
  ~customer, ~feedback,                                          ~day,
  1,          "Loved the service! [Rating:5]",                     1,
  2,          "Unsatisfied with the product quality. [Rating:2]",  1,
  3,          "Average experience. [Rating:3]",                    1,
  4,          "Great product, but took too long. [Rating:4]", 2,
  5,          "Not what I expected. [Rating:1]",                   2,
  6,          "Not what I expected. [Rating:x]",                   2
)
```

Provide code which returns 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.

```
(feedback_parsed <- feedback_data %>%
  separate_wider_delim(
    cols = feedback,
    names = c("feedback_text", "rating_text"),
    delim = "[") %>%
  mutate(rating = parse_number(rating_text)) %>%
  select(-rating_text) )
```

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

(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 quali... 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
```

Provide code which computes the average rating per day in `feedback_parsed`, ignoring all rows with missing data.

```r
feedback_parsed %>%
  group_by(day) %>%
  summarize(average_rating = mean(rating, na.rm=TRUE))
```

```
# A tibble: 2 x 2
    day average_rating
  <dbl>          <dbl>
1     1           3.33
2     2           2.5
```

Problem 4

Consider the following dataset:

```
health_data <- tribble(
  ~PatientID, ~Weight_2019, ~Weight_2020, ~Height_2019, ~Height_2020,
  1, 70, 72, 170, 171,
  2, 65, 68, 165, 166,
  3, 80, 82, 180, 181
)
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: Inspect the arguments of `pivot_longer()` using `?pivot_longer()`. Read the documentation for the `names_to`, `names_sep` arguments.

```
(long_health_data <- health_data %>%
  pivot_longer(
    cols = c(Weight_2019, Weight_2020, Height_2019, Height_2020),
    names_to = c(".value", "year"),
    names_sep = "_"
    ) %>%
  mutate(year = as.numeric(year)) )
```

```
# A tibble: 6 x 4
  PatientID  year Weight Height
      <dbl> <dbl>  <dbl>  <dbl>
1         1  2019     70    170
2         1  2020     72    171
3         2  2019     65    165
4         2  2020     68    166
5         3  2019     80    180
6         3  2020     82    181
```

(b) Provide code which transforms `long_health_data` into the tibble `wide_health_data` which is back in the wide format, with columns "Weight_2019", "Weight_2020", "Height_2019", "Height_2020". You can check your calculation was correct by checking `all.equal(wide_health_data, health_data)`

```
(wide_health_data <- long_health_data %>%
   pivot_wider(
     id_cols = PatientID,
     names_from = year,
     names_sep = "_",
     values_from = c(Weight, Height)
   ) )
```

```
# 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
```

```
all.equal(wide_health_data, health_data)
```

```
[1] TRUE
```