STAT 231: Problem Set 2A

Majd Rouhana

due by 5 PM on Monday, September 7

In order to most effectively digest the textbook chapter readings – and the new R commands each presents – series A homework assignments are designed to encourage you to read the textbook chapters actively and in line with the textbook's Prop Tip of page 33:

"**Pro Tip**: If you want to learn how to use a particular command, we highly recommend running the example code on your own"

A more thorough reading and light practice of the textbook chapter prior to class allows us to dive quicker and deeper into the topics and commands during class. Furthermore, learning a programming lanugage is like learning any other language – practice, practice, practice is the key to fluency. By having two assignments each week, I hope to encourage practice throughout the week. A little coding each day will take you a long way!

Series A assignments are intended to be completed individually. While most of our work in this class will be collaborative, it is important each individual completes the active readings. The problems should be straightforward based on the textbook readings, but if you have any questions, feel free to ask me!

Steps to proceed:

1. In RStudio, go to File > Open Project, navigate to the folder with the course-content repo, select the course-content project (course-content.Rproj), and click "Open"
2. Pull the course-content repo (e.g. using the blue-ish down arrow in the Git tab in upper right window)
3. Copy ps2A.Rmd from the course repo to your repo (see page 6 of the GitHub Classroom Guide for Stat231 if needed)
4. Close the course-content repo project in RStudio
5. Open YOUR repo project in RStudio
6. In the ps2A.Rmd file in YOUR repo, replace "YOUR NAME HERE" with your name
7. Add in your responses, committing and pushing to YOUR repo in appropriate places along the way
8. Run "Knit PDF"
9. Upload the pdf to Gradescope. Don't forget to select which of your pages are associated with each problem. *You will not get credit for work on unassigned pages (e.g., if you only selected the first page but your solution spans two pages, you would lose points for any part on the second page that the grader can't see).*

1. NYC Flights

a.

In Section 4.3.1, the `flights` and `carrier` tables within the `nycflights13` package are joined together. Recreate the `flightsJoined` dataset from page 80. Hint: make sure you've loaded the `nycflights13` package before referring to the data tables (see code on page 79).

```
library(nycflights13)
flightsJoined <- flights %>%
  inner_join(airlines, by = c("carrier" = "carrier"))
glimpse(flightsJoined)
```

```
## Rows: 336,776
## Columns: 20
## $ year           <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013...
## $ month          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ day            <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ dep_time       <int> 517, 533, 542, 544, 554, 554, 555, 557, 557, 558, 55...
## $ sched_dep_time <int> 515, 529, 540, 545, 600, 558, 600, 600, 600, 600, 60...
## $ dep_delay      <dbl> 2, 4, 2, -1, -6, -4, -5, -3, -3, -2, -2, -2, -2, -2,...
## $ arr_time       <int> 830, 850, 923, 1004, 812, 740, 913, 709, 838, 753, 8...
## $ sched_arr_time <int> 819, 830, 850, 1022, 837, 728, 854, 723, 846, 745, 8...
## $ arr_delay      <dbl> 11, 20, 33, -18, -25, 12, 19, -14, -8, 8, -2, -3, 7,...
## $ carrier        <chr> "UA", "UA", "AA", "B6", "DL", "UA", "B6", "EV", "B6"...
## $ flight         <int> 1545, 1714, 1141, 725, 461, 1696, 507, 5708, 79, 301...
## $ tailnum        <chr> "N14228", "N24211", "N619AA", "N804JB", "N668DN", "N...
## $ origin         <chr> "EWR", "LGA", "JFK", "JFK", "LGA", "EWR", "EWR", "LG...
## $ dest           <chr> "IAH", "IAH", "MIA", "BQN", "ATL", "ORD", "FLL", "IA...
## $ air_time       <dbl> 227, 227, 160, 183, 116, 150, 158, 53, 140, 138, 149...
## $ distance       <dbl> 1400, 1416, 1089, 1576, 762, 719, 1065, 229, 944, 73...
## $ hour           <dbl> 5, 5, 5, 5, 6, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 5, 6, 6...
## $ minute         <dbl> 15, 29, 40, 45, 0, 58, 0, 0, 0, 0, 0, 0, 0, 0, 0, 59...
## $ time_hour      <dttm> 2013-01-01 05:00:00, 2013-01-01 05:00:00, 2013-01-0...
## $ name           <chr> "United Air Lines Inc.", "United Air Lines Inc.", "A...
```

b.

Now, create a new dataset `flightsJoined2` that:

- creates a new variable, `distance_km`, which is distance in kilometers (note that 1 mile is about 1.6 kilometers)
- keeps only the variables: `name`, `flight`, `arr_delay`, and `distance_km`
- keeps only observations where distance is less than 500 kilometers

Hint: see examples in Section 4.1 for subsetting datasets and creating new variables.

```
flightsJoined2 <- flightsJoined %>%
  mutate(distance_km = distance * 1.6) %>%
  select(name, flight, arr_delay, distance_km) %>%
  filter(distance_km < 500)
glimpse(flightsJoined2)
```

```
## Rows: 54,921
## Columns: 4
## $ name        <chr> "ExpressJet Airlines Inc.", "JetBlue Airways", "Southwe...
```

```
## $ flight      <int> 5708, 1806, 4646, 4144, 1002, 102, 20, 44, 1172, 1838, ...
## $ arr_delay   <dbl> -14, -4, -19, 12, -10, 5, -1, 4, -19, -22, -14, -13, 85...
## $ distance_km <dbl> 366.4, 299.2, 296.0, 339.2, 299.2, 481.6, 422.4, 334.4,...
```

c.

Lastly, using the functions introduced in Section 4.1.4, compute the number of flights (call this `N`), the average arrival delay (call this `avg_arr_delay`), and the average distance in kilometers (call this `avg_dist_km`) among these flights with distances less than 500 km (i.e. working off of `flightsJoined2`) *grouping by the carrier name*. Sort the results in descending order based on `avg_arr_delay`.

Getting NAs for `avg_arr_delay`? That happens when some observations are missing that data. Before grouping and summarizing, add a line to exclude observations with missing arrival delay information using `filter(is.na(arr_delay)==FALSE)`.

```
flightsJoined3 <- flightsJoined2 %>%
  filter(is.na(arr_delay)==FALSE) %>%
  group_by(name) %>%
    summarize(
      N = n(), avg_arr_delay = mean(arr_delay), avg_dist_km = mean(distance_km))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
glimpse(flightsJoined3)
```

```
## Rows: 11
## Columns: 4
## $ name          <chr> "American Airlines Inc.", "Delta Air Lines Inc.", "En...
## $ N             <int> 1428, 1201, 6144, 2741, 14753, 13443, 286, 1, 200, 33...
## $ avg_arr_delay <dbl> 1.8788515, -0.6427977, 6.8190104, 10.9992703, 15.5943...
## $ avg_dist_km   <dbl> 299.2000, 324.8946, 338.8938, 350.6697, 373.0808, 384...
```

2. Baby names

a.

Working with the `babynames` data table in the `babynames` package, create a dataset `babynames2` that only includes years 2000 to 2017.

```
library(babynames)
babynames2 <- babynames %>%
  filter(year >= 2000, year <= 2017)
```

b.

Following the code presented in Section 5.2.4, create a dataset called `BabyNarrow` that summarizes the total number of people with each name (born between 2000 and 2017), grouped by sex. (Hint: follow the second code chunk on page 102, but don't filter on any particular names.) Look at the dataset. Why have we called this dataset "narrow"?

> ANSWER: This dataset is called "narrow" because it is in a narrow format–it only contains two columns. Narrow datasets only include a few columns and are more flexible for including additional variables.

```
BabyNarrow <- babynames2 %>%
  group_by(sex) %>%
  summarise(total = sum(n)) %>%
  spread(key = sex, value = total, fill = 0)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
glimpse(BabyNarrow)
```

```
## Rows: 1
## Columns: 2
## $ F <dbl> 32534995
## $ M <dbl> 35084501
```

c.

Now, following the code chunk presented on page 103, put the data into a wide format (call the new dataset `BabyWide`), and only keep observations where both `M` and `F` are greater than 10,000. Compute the `ratio` (as `pmin(M/F, F/M)`) and identify the top three names with the largest ratio. (Note: these names could be different from the ones found on page 103 since we limited the dataset to years 2000-2017 and names with greater than 10,000 individuals.)

> ANSWER: The top three names with the largest ratio are Justice, Skyler and Quinn.

```
BabyWide <- babynames2 %>%
  group_by(sex, name) %>%
  summarize(total = sum(n)) %>%
  spread(key = sex, value = total, fill = 0) %>%
  filter(M > 10000, F > 10000) %>%
  mutate(ratio = pmin(M / F, F / M) ) %>%
  arrange(desc(ratio)) %>%
  head(3)
```

```
## `summarise()` regrouping output by 'sex' (override with `.groups` argument)
```

```
glimpse(BabyWide)
```

```
## Rows: 3
## Columns: 4
## $ name  <chr> "Justice", "Skyler", "Quinn"
## $ F     <dbl> 10947, 17120, 25022
## $ M     <dbl> 11267, 22154, 19080
## $ ratio <dbl> 0.9715985, 0.7727724, 0.7625290
```

d.

Lastly, use the `gather()` function (or the `pivot_longer()` function) to put the dataset back into narrow form. Call this dataset `BabyNarrow2`. Hint: see Section 5.2.3. Why are the number of observations in `BabyNarrow2` different from that in `BabyNarrow`?

> ANSWER: There are a larger number of observations because `BabyNarrow` only includes the variable `sex` and the total amount of each sex born between 2000 and 2017, while `BabyNarrow2` includes `name`, `sex` and `ratio`.

```
BabyNarrow2 <- BabyWide %>%
  gather(key = sex, value = total)
glimpse(BabyNarrow2)
```

```
## Rows: 12
## Columns: 2
## $ sex   <chr> "name", "name", "name", "F", "F", "F", "M", "M", "M", "ratio"...
## $ total <chr> "Justice", "Skyler", "Quinn", "10947", "17120", "25022", "112...
```