STAT 231: Problem Set 2A

Joshua Kim

due by 5 PM on Monday, March 1

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)
library(mdsr)
library(dplyr)
library(tidyverse)

glimpse(flights)
```

```
## Rows: 336,776
## Columns: 19
## $ 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...
```

```
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) %>%
  filter(distance_km < 500) %>%
  select(name, flight, arr_delay, distance_km)

flightsJoined2
```

```
## # A tibble: 163,892 x 4
##    name                     flight arr_delay distance_km
##    <chr>                     <int>     <dbl>       <dbl>
##  1 Delta Air Lines Inc.        461       -25        476.
##  2 United Air Lines Inc.      1696        12        449.
##  3 ExpressJet Airlines Inc.   5708       -14        143.
##  4 American Airlines Inc.      301         8        458.
##  5 JetBlue Airways            1806        -4        117.
##  6 Envoy Air                  4650        12        476.
##  7 Envoy Air                  4401        16        314.
##  8 Delta Air Lines Inc.       1743        -8        475
##  9 Envoy Air                  3768        32        449.
## 10 Delta Air Lines Inc.        575        -9        466.
## # ... with 163,882 more rows
```

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`.er 4

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)`.

```
flightsJoined2 %>%
  filter(is.na(arr_delay)==FALSE) %>%
  group_by(name) %>%
  summarize(
```

```r
    N = n(),
    avg_arr_delay = mean(arr_delay),
    avg_dist_km = mean(distance_km)) %>%
  arrange(desc(avg_arr_delay))
```

```
## # A tibble: 12 x 4
##    name                            N avg_arr_delay avg_dist_km
##    <chr>                       <int>         <dbl>       <dbl>
##  1 AirTran Airways Corporation  3175        20.1          415.
##  2 ExpressJet Airlines Inc.    40554        15.8          281.
##  3 Mesa Airlines Inc.            544        15.6          235.
##  4 SkyWest Airlines Inc.          25        14.2          268.
##  5 Envoy Air                   22715        11.0          326.
##  6 Southwest Airlines Co.       6831         9.84         446.
##  7 JetBlue Airways             16546         9.26         190.
##  8 Endeavor Air Inc.           14357         7.47         260.
##  9 United Air Lines Inc.       12029         5.61         332.
## 10 Delta Air Lines Inc.        15740         5.15         404.
## 11 US Airways Inc.             17591         2.33         225.
## 12 American Airlines Inc.       7274        -0.205        391.
```

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 < 2018 & year > 1999)

babynames2
```

```
## # A tibble: 591,925 x 5
##     year sex   name          n    prop
##    <dbl> <chr> <chr>     <int>   <dbl>
##  1  2000 F     Emily     25953 0.0130
##  2  2000 F     Hannah    23080 0.0116
##  3  2000 F     Madison   19967 0.0100
##  4  2000 F     Ashley    17997 0.00902
##  5  2000 F     Sarah     17697 0.00887
##  6  2000 F     Alexis    17629 0.00884
##  7  2000 F     Samantha  17266 0.00866
##  8  2000 F     Jessica   15709 0.00787
##  9  2000 F     Elizabeth 15094 0.00757
## 10  2000 F     Taylor    15078 0.00756
## # ... with 591,915 more rows
```

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: We call this data set narrow because it is more tall and narrow with only 3 columns but 591,925 rows. It is now tidy data. Each row is a unique data point with columns recording the same sort of value. The data became narrow and long as a result.

```
BabyNarrow <- babynames2 %>%
  group_by(sex, name) %>%
    summarize(
      total = sum(n)
    )
```

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

```
BabyNarrow
```

```
## # A tibble: 73,332 x 3
## # Groups:   sex [2]
##    sex   name      total
##    <chr> <chr>     <int>
##  1 F     Aabha        35
##  2 F     Aabriella    32
##  3 F     Aada          5
##  4 F     Aaden         5
```

```
##   5 F     Aadhira      77
##   6 F     Aadhvika      9
##   7 F     Aadhya     1478
##   8 F     Aadi         16
##   9 F     Aadilynn      5
##  10 F     Aadison      11
## # ... with 73,322 more rows
```

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.)

- Note: you can use the `pivot_wider()` function instead of the `spread()` function if using the 2nd edition of the textbook (e.g., see Section 6.2.2 and 6.2.3 in the 2nd edition). I find `pivot_wider()` and `pivot_longer()` to be more intuitive than `spread()` and `gather()`.

  ANSWER: Justice, Skyler and Quinn have the largest ratio at .972, .773, and .763.

```
# this will bring up "Pivoting Introduction" vignette in your Help tab
#vignette("pivot")

BabyWide <- BabyNarrow %>%
  pivot_wider(
    names_from = sex,
    values_from = total) %>%
  filter(M > 10000 & F > 10000) %>%
  mutate(ratio = pmin(M/F, F/M)) %>%
  arrange(desc(ratio))

BabyWide
```

```
## # A tibble: 25 x 4
##    name        F     M ratio
##    <chr>   <int> <int> <dbl>
##  1 Justice 10947 11267 0.972
##  2 Skyler  17120 22154 0.773
##  3 Quinn   25022 19080 0.763
##  4 Amari   11778 15676 0.751
##  5 Casey   12109 16809 0.720
##  6 Riley   89827 59823 0.666
##  7 Peyton  61217 39261 0.641
##  8 Emerson 18592 11742 0.632
##  9 Charlie 13255 21243 0.624
## 10 Dakota  21950 35840 0.612
## # ... with 15 more rows
```

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: BabyNarrow2 has significantly less rows compared to BabyNarrow. This is because the

BabyNarrow2 dataset had filtered for names that have 10,000 for each gender while the original BabyNarrow dataset accounted for all names.

```r
BabyNarrow2 <- BabyWide %>%
  pivot_longer(M:F, names_to = "sex", values_to = "count")

BabyNarrow2
```

```
## # A tibble: 50 x 4
##    name    ratio sex   count
##    <chr>   <dbl> <chr> <int>
##  1 Justice 0.972 M     11267
##  2 Justice 0.972 F     10947
##  3 Skyler  0.773 M     22154
##  4 Skyler  0.773 F     17120
##  5 Quinn   0.763 M     19080
##  6 Quinn   0.763 F     25022
##  7 Amari   0.751 M     15676
##  8 Amari   0.751 F     11778
##  9 Casey   0.720 M     16809
## 10 Casey   0.720 F     12109
## # ... with 40 more rows
```