# Prep2

Sebastian Montesinos

Due by midnight, Monday Feb. 21

Reminder: Prep assignments are to be completed individually. Upload a final copy of the .Rmd and renamed .pdf to your private repo, and submit the renamed pdf to Gradescope.

## Reading

The associated reading for the week is Chapter 4, Chapter 5, Chapter 6 (skip 6.4) and Sections 8.3 and 8.4. This reading explores major functions in wrangling data, including reshaping data.

Remember, I recommend you code along with the book examples. You can try out the code yourself - just be sure to load the mdsr package and any other packages referenced. You can get the code in R script files (basically, files of just R code, not like a .Rmd) from the book website.

# 1 - Some basics

Many different data wrangling commands are covered in these chapters. Identify the command you'd use for each of the operations below.

part a - Add the average of 3 variables to the data set as a new variable.

Solution: You would use mutate().

part b - Keep only 4 columns of a data frame in a new data set.

Solution: You would use select().

part c - Choose observations that match a particular category of a categorical variable to keep in a new data set.

Solution: You would use filter().

part d - Combine two data sets based on common variables where all rows from the first are returned, along with any matches in the second.

Solution: You would use left.join().

## 2 - NYC Flights

In Section 5.1, the `flights` and `airlines` tables within the `nycflights13` package are joined together.

> part a - Recreate the `flights_joined` dataset from Section 5.1, being sure to *glimpse* the data in the Console (or via the code chunk) to verify the join worked.

Solution:

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

```
## Rows: 336,776
## Columns: 20
## $ year           <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2~
## $ 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, 558, ~
## $ sched_dep_time <int> 515, 529, 540, 545, 600, 558, 600, 600, 600, 600, 600, ~
## $ dep_delay      <dbl> 2, 4, 2, -1, -6, -4, -5, -3, -3, -2, -2, -2, -2, -2, -1~
## $ arr_time       <int> 830, 850, 923, 1004, 812, 740, 913, 709, 838, 753, 849,~
## $ sched_arr_time <int> 819, 830, 850, 1022, 837, 728, 854, 723, 846, 745, 851,~
## $ arr_delay      <dbl> 11, 20, 33, -18, -25, 12, 19, -14, -8, 8, -2, -3, 7, -1~
## $ carrier        <chr> "UA", "UA", "AA", "B6", "DL", "UA", "B6", "EV", "B6", "~
## $ flight         <int> 1545, 1714, 1141, 725, 461, 1696, 507, 5708, 79, 301, 4~
## $ tailnum        <chr> "N14228", "N24211", "N619AA", "N804JB", "N668DN", "N394~
## $ origin         <chr> "EWR", "LGA", "JFK", "JFK", "LGA", "EWR", "EWR", "LGA",~
## $ dest           <chr> "IAH", "IAH", "MIA", "BQN", "ATL", "ORD", "FLL", "IAD",~
## $ air_time       <dbl> 227, 227, 160, 183, 116, 150, 158, 53, 140, 138, 149, 1~
## $ distance       <dbl> 1400, 1416, 1089, 1576, 762, 719, 1065, 229, 944, 733, ~
## $ hour           <dbl> 5, 5, 5, 5, 6, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 5, 6, 6, 6~
## $ minute         <dbl> 15, 29, 40, 45, 0, 58, 0, 0, 0, 0, 0, 0, 0, 0, 0, 59, 0~
## $ time_hour      <dttm> 2013-01-01 05:00:00, 2013-01-01 05:00:00, 2013-01-01 0~
## $ name           <chr> "United Air Lines Inc.", "United Air Lines Inc.", "Amer~
```

> part b - Now, starting from `flights_joined`, create a new dataset `flights_short` that does the following:

- 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` and
- keeps only observations where the distance is less than 480 kilometers (300 miles).

Solution:

```
flights_short <- flights_joined %>%
  mutate(distance_km = distance*1.6)
flights_short <- flights_short %>%
  filter(distance_km < 480) %>%
  select(name, flight, arr_delay, distance_km)
glimpse(flights_short)
```

```
## Rows: 51,287
## Columns: 4
## $ name        <chr> "ExpressJet Airlines Inc.", "JetBlue Airways", "Southwest ~
## $ flight      <int> 5708, 1806, 4646, 4144, 1002, 20, 44, 1172, 1838, 27, 4418~
## $ arr_delay   <dbl> -14, -4, -19, 12, -10, -1, 4, -19, -22, -14, -13, 851, -7,~
## $ distance_km <dbl> 366.4, 299.2, 296.0, 339.2, 299.2, 422.4, 334.4, 320.0, 29~
```

part c - 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 480 km (i.e. working off of `flights_short`), grouping by the carrier name. Sort the results in descending order based on `avg_arr_delay`. Save the results in a tibble object called `delay_summary`, and display the table.

Solution:

```
flights_short <- drop_na(flights_short, arr_delay)
delay_summary <- flights_short %>%
  group_by(name) %>%
  summarize(
    N = n(),
    avg_arr_delay = (mean(arr_delay)),
    avg_dist_km = mean(distance_km)
  ) %>%
  arrange(desc(avg_arr_delay))
delay_summary
```

```
## # A tibble: 11 x 4
##    name                        N avg_arr_delay avg_dist_km
##    <chr>                   <int>         <dbl>       <dbl>
##  1 Mesa Airlines Inc.        286        18.0          360.
##  2 ExpressJet Airlines Inc. 14707        15.6          373.
##  3 Envoy Air                 2741        11.0          351.
##  4 JetBlue Airways          10670         8.36         360.
##  5 Endeavor Air Inc.         5405         6.84         319.
##  6 Southwest Airlines Co.     200         4.92         272.
##  7 United Air Lines Inc.     3307         4.09         320.
##  8 SkyWest Airlines Inc.        1         3            366.
##  9 US Airways Inc.           9093         2.22         308.
## 10 American Airlines Inc.    1428         1.88         299.
## 11 Delta Air Lines Inc.      1201        -0.643        325.
```

part d - Rename the four columns in the `delay_summary` data table to `Airline`, `"Total flights under 480 km"`, `"Average arrival delay (mins)"` and `"Average distance (km)"`, respectively, then use `kable(booktabs = TRUE, digits = 0)` to make the final table output in the pdf close to publication quality.

Solution:

```
delay_summary <- delay_summary %>%
  rename("airline" = name,
         "Total Flights under 480 km" = N,
         "Average arrival delay (mins)" = avg_arr_delay,
         "Average distance (km)" = avg_dist_km)
kable(delay_summary, booktabs = TRUE, digits = 0)
```

| airline | Total Flights under 480 km | Average arrival delay (mins) | Average distance (km) |
|---|---|---|---|
| Mesa Airlines Inc. | 286 | 18 | 360 |
| ExpressJet Airlines Inc. | 14707 | 16 | 373 |
| Envoy Air | 2741 | 11 | 351 |
| JetBlue Airways | 10670 | 8 | 360 |
| Endeavor Air Inc. | 5405 | 7 | 319 |
| Southwest Airlines Co. | 200 | 5 | 272 |
| United Air Lines Inc. | 3307 | 4 | 320 |
| SkyWest Airlines Inc. | 1 | 3 | 366 |
| US Airways Inc. | 9093 | 2 | 308 |
| American Airlines Inc. | 1428 | 2 | 299 |
| Delta Air Lines Inc. | 1201 | -1 | 325 |

# 3 - Baby names - Variant of 6.2.5 example

part a - Working with the `babynames` data in the **babynames** package, create a dataset `recent_names` that only includes years 2003 to 2017 (giving us the most recent 15 years of data).

Solution:

```
recent_names <- babynames %>%
  filter(year > 2002 & year < 2018)
```

part b - Following the code presented in Section 6.2.5, create a dataset called `recentnames_summary` that summarizes the total number of people in recent history (years 2003 to 2017) with each name, grouped by sex.

Solution:

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

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

```
recentnames_summary
```

```
## # A tibble: 69,431 x 3
## # Groups:   name [63,515]
##    name      sex   total
##    <chr>     <chr> <int>
##  1 Aaban     M       107
##  2 Aabha     F        35
##  3 Aabid     M        10
##  4 Aabir     M         5
##  5 Aabriella F        32
##  6 Aada      F         5
##  7 Aadam     M       185
##  8 Aadan     M       130
##  9 Aadarsh   M       177
## 10 Aaden     F         5
## # ... with 69,421 more rows
```

part c - Now, following the fourth and fifth code chunks presented in Section 6.2.5, reshape or *pivot* the summary data from *long* format to *wide* format. Only keep observations where more than 8,000 babies have been named in each sex (`M` and `F`), and find the smaller of the two ratios `M / F` and `F / M` to identify the top three sex-balanced names (and only the top three!). Save the wide data as `recentnames_balanced_wide`. Display the table.

Solution:

```
recentnames_balancedwide <- recentnames_summary %>%
  pivot_wider(names_from = sex, values_from = total,
              values_fill = 0) %>%
  filter(M > 8000 & F > 8000) %>%
  mutate(ratio = pmin(M / F, F / M)) %>%
  arrange(desc(ratio)) %>%
  head(3)

head(recentnames_balancedwide)
```

```
## # A tibble: 3 x 4
## # Groups:   name [3]
##   name        M     F ratio
##   <chr>   <int> <int> <dbl>
## 1 Justice  9062  9121 0.994
## 2 Dakota  23823 18971 0.796
## 3 Skyler  17617 13529 0.768
```

part d - Finally, use `pivot_longer()` to put the dataset back into *long* form. Call this dataset `recentnames_balanced` and display the table.

Solution:

```
recentnames_balanced <- recentnames_balancedwide %>%
  pivot_longer(-name, names_to = "sex", values_to = "total")

recentnames_balanced <- filter(recentnames_balanced, sex == "F" | sex == "M")

recentnames_balanced
```

```
## # A tibble: 6 x 3
## # Groups:   name [3]
##   name    sex   total
##   <chr>   <chr> <dbl>
## 1 Justice M      9062
## 2 Justice F      9121
## 3 Dakota  M     23823
## 4 Dakota  F     18971
## 5 Skyler  M     17617
## 6 Skyler  F     13529
```

# 4 - Ethics

Each subsection of Section 8.4 discusses an ethical scenario and ends with one or more questions. Consider the subsection 8.4.6 on "Reproducible spreadsheet analysis".

Write two or three sentences reflecting on how using RMarkdown would help avoid some of the issues described in this scenario, or at least make them easier to spot.

Solution: RMarkdown allows the data manipulation and coding someone does to be easily reproducible by other statisticians. Therefore, RMarkdown would have allowed the errors to have been much more transparent and viewable to others.