Econ 294 Assignment 4

Curtis Kephart

Winter 2016

This assignment is designed to get you to explore:

- `dplyr`'s single-table verbs (`select`, `filter`, `arrange`, `mutate`, `summarize`, plus helper functions), two-table verbs (`left_join`, `inner_join` , `right_join`, `full_join` , `semi_join`, or `anti_join`), and the `group_by` split-apply-combine operator. Also `ungroup()` may come in handy .
- `tidyr`'s `gather`, `spread`, `separate` and `unite` tools.
- and the `magrettr` pipe `%>%` operator.

Please refer to the examples from the dplyr single table verbs vignette, two-table verbs vignette, and intro tidyr vignette for help (the assignment borrows heavily from these, and other work from Hadley Wickham.)

Due by Feb 19th 2016 (after the next lecture). Turn in your `.R` script by pushing it to your public github repo and emailing the URL to your instructor (at `curtisk+econ294_04@ucsc.edu`)

0. Use a `print` call to report your first name, last name, and student ID number.

1. In the instructors github repo's data directory, load `flights.csv`, `planes.csv`, `weather.csv`, and `airports.csv`.

   In your `read.csv`, be sure to load all strings as string, not factors. (Note the `stringsAsFactors` argument defaults to `TRUE`, check out `?read.csv` arguments list for help.)

   The df `flights` logs all flights *leaving* Houston in 2011. With `weather` logging hourly Houston weather data, `planes` plane metadata, and `airports` airport metaldata

2. Convert any `date` column from type char to type date using the `as.Date()` function.

3. Extract all flights that match these critera:

- Create `flights.2a`, all flights that went to the city of San Francisco or Oakland CA. Print the number of observation you find.
- Create `flights.2b`, all flights delayed by an hour or more. Print the number of observations.
- Create `flights.2c`, all flights in which the arrival delay was more than twice as much as the departure delay. Print the number of observations for this question.

4. Using `select()`'s helper functions, come up with different three ways to select the delay varaibles from `flights` (see `?dplyr::select` for details.)

5. Working with `arrange()`

   5a use arrange to find and print the top five most (departure) delayed flights.

   5b use arrange to find and print the top five flights that caught up the most (in minutes) during the flight. (Arrange allows you to use compound expressions, but feel free to use `mutate`.)

6. Working with `mutate()`.

Change the `flights` data frame, by adding a number of new columns:

- Where the existing `time` variable is travel time in minutes, find `speed` in mph.
- Create `delta`, the amount of time made-up or lost in the flight (e.g. a `delta` of 60 means the flight made up 60 minutes between departure and arrival).

(Hint, use `View()` to inspect your new columns)

6a. Print the top five flights by speed.

6b. Print the top five flights that made up the most time in flight (this should match 5b)

6c. Print the top flights that **lost** the most time in flight.

7. Working with `group_by` and `summarize`. Recall `?dplyr::summarize` maps a data frame to a single row of summary statstics you have set-up. The `group_by` function allows you to find a set of statistics within each group you define (and `ungroup()` undos `group_by()`).

- Grouping by carrier. Create `flights.7a` with the following summary statistics grouped by `carrier`: The number of cancelled flights by each carrier, total flights (`n()`), the percent of canceled flights relative to total flights, and from the `delta` variable (created above) find the `min`, first quartile, `median`, `mean`, third quartile, 90th quantile, and `max`. (`quantile` can find quartiles too - don't forget `na.rm = T` when referring to delta)
  - Print the summary table sorted with the carriers with the worst relative cancelled percent on top (arrange helper function `desc()`).
- Grouping by Day and Hour. Use `print` to explain to me what the following code does. Rewrite it using the `magrettr`'s `%>%` operator (tip for printing code blocks, print multiple lines with `cat()` instead of `print`).

```
day_delay <- dplyr::filter(
    summarize(
        group_by(
            dplyr::filter(
                flights,
                !is.na(dep_delay)
            ),
            date
        ),
        delay = mean(dep_delay),
      n = n()
    ),
    n > 10
)
```

8. Add a new column to `day_delay` (created by the codeblock above) with the difference between today and yesterday's average delay. (tip, check out `?dplyr::lag`). Print the top five days that had the biggest increase in average `dep_delay` from one day to the next.
9. Two table verbs. Merging options.

   Create a new table called `dest_delay` that summarizes the destination-level average `arr_delay` and the number of flights that flew into it (`group_by`, `summarize` and `n()` may come handy).

From the `airports` dataset, select the iata, airport, city, state, lat and long. In the process, rename `iata` to `dest` and `airport` to `name`.

Create df.9a. Use `left_join` to join `dest_delay` (as first table) with `airports`, by destination. Print a table with the top five cities wtih the highest average arrival delays.

Create df.9b. Do the same join via an `inner_join`. Do the number of observations via the `left_join` match those of the `inner_join`?

Create df.9c. Now join `dest_delay` (as first table) and `airports` via `right_join`. How many observations are in this new table? Do any `NA`s appear in the average `arr_delay`? if so, why?

Create df.9d. Now join `dest_delay` (as first table) and `airports` via `full_join`. Again, how many observations are in this new table? Do any `NA`s appear in the average `arr_delay` column (pulled from `dest_delay`)? if so, why?

10. Similar to `day_delay` above, create `hourly_delay`, with depature delays (`dep_delays`) at the year-month-day-hourly level. Merge `hourly_delay` with the `weather` data.frame. Using the newly merged `conditions` variable. Print a table summarizing which weather conditions are associated with the biggest delays.

11. Tidy(r) Data

11a. Starting with `df` below, use `tidyr` and `dplyr` tools to reach the following table (`gather`).

```
df <- data.frame(treatment = c("a", "b"), subject1 = c(3, 4), subject2 = c(5, 6))
df
```

```
##   treatment subject1 subject2
## 1         a        3        5
## 2         b        4        6
```

use dplyr and tidyr to map to:

```
##   subject treatment value
## 1       1         a     3
## 2       1         b     4
## 3       2         a     5
## 4       2         b     6
```

(One method uses gather, mutate, select and arrange)

11b. Starting with `df` below, use tidyr and dplyr tools to reach the following table (`spread`).

```
df <- data.frame(
  subject = c(1,1,2,2),
  treatment = c("a","b","a","b"),
  value = c(3,4,5,6)
)
```

use dplyr and tidyr to map to:

```
##   treatment subject1 subject2
## 1         a        3        5
## 2         b        4        6
```

(one method to do this spread, and rename with grave marks)

11c. Starting with `df` below, use tidyr and dplyr tools to reach the following table (`separate` and `unite`).

```
df <- data.frame(
  subject = c(1,2,3,4),
  demo = c("f_15_CA","f_50_NY","m_45_HI","m_18_DC"),
  value = c(3,4,5,6)
)
```

use dplyr and tidyr to map to

```
##   subject sex age state value
## 1       1   f  15    CA     3
## 2       2   f  50    NY     4
## 3       3   m  45    HI     5
## 4       4   m  18    DC     6
```

11d. Starting with `df` below, use tidyr and dplyr tools to reach the following table (`separate` and `unite`).

```
df <- data.frame(
  subject = c(1,2,3,4),
  sex = c("f","f","m",NA),
  age = c(11,55,65,NA),
  city = c("DC","NY","WA",NA),
  value = c(3,4,5,6)
)
```

use dplyr and tidyr to map to

```
##   subject    demo value
## 1       1 f.11.DC     3
## 2       2 f.55.NY     4
## 3       3 m.65.WA     5
## 4       4    <NA>     6
```

(tip, one method uses `unite` and `replace`)