Introduction to data

Your reproducible lab report: Before you get started, download the R Markdown template for this lab. Remember all of your code and answers go in this document:

```
download.file("https://dyurovsky.github.io/85309/post/rmd/lab2.Rmd",
              destfile = "lab2.Rmd")
```

You can think of statistics as a field that focuses on turning data into information. The first step in that process is to summarize and describe the raw data. In this lab we explore flights, specifically a random sample of 32735 domestic flights that departed from the three major New York City airports in 2013. We will generate simple graphical and numerical summaries of data on these flights and explore delay times. As this is a large data set, along the way you'll also learn some skills for data processing and subsetting that you'll need over the course of the quarter.

Getting started

Load packages

In this lab we will explore the data using the `dplyr` package and visualize it using the `ggplot2` package for data visualization. These are both parts of the `tidyverse` ecosystem.

Let's load the `tidyverse` package.

```
library(tidyverse)
```

The data

The Bureau of Transportation Statistics (http://www.rita.dot.gov/bts/about/) (BTS) is a statistical agency that is a part of the Research and Innovative Technology Administration (RITA). As its name implies, BTS collects and makes available transportation data, such as the flights data we will be working with in this lab.

We begin by reading the `nycflights` data from a comma-separated value (csv) file into a dataframe. Type the following in your console to load the data:

```
nycflights <- read_csv("https://dyurovsky.github.io/85309/data/lab2/nycflights.csv")
```

The data set `nycflights` that shows up in your workspace is a **tibble**, with each row representing an *observation* and each column representing a *variable*. For this data set, each *observation* is a single flight.

To view the names of the variables, type the command

```
names(nycflights)
```

The **codebook** (description of the variables) is included below.

- `year` , `month` , `day` : Date of departure
- `dep_time` , `arr_time` : Departure and arrival times, local timezone.
- `dep_delay` , `arr_delay` : Departure and arrival delays, in minutes. Negative times represent early departures/arrivals.
- `carrier` : Two letter carrier abbreviation.
    - `9E` : Endeavor Air Inc.
    - `AA` : American Airlines Inc.
    - `AS` : Alaska Airlines Inc.
    - `B6` : JetBlue Airways
    - `DL` : Delta Air Lines Inc.
    - `EV` : ExpressJet Airlines Inc.
    - `F9` : Frontier Airlines Inc.
    - `FL` : AirTran Airways Corporation
    - `HA` : Hawaiian Airlines Inc.
    - `MQ` : Envoy Air
    - `OO` : SkyWest Airlines Inc.
    - `UA` : United Air Lines Inc.
    - `US` : US Airways Inc.
    - `VX` : Virgin America
    - `WN` : Southwest Airlines Co.
    - `YV` : Mesa Airlines Inc.
- `tailnum` : Plane tail number
- `flight` : Flight number
- `origin` , `dest` : Airport codes for origin and destination. (Google can help you with what code stands for which airport.)
- `air_time` : Amount of time spent in the air, in minutes.
- `distance` : Distance flown, in miles.
- `hour` , `minute` : Time of departure broken in to hour and minutes.

A very useful function for taking a quick peek at your tibble, and viewing its dimensions and data types is `str` , which stands for **str**ucture.

```
str(nycflights)
```

The `nycflights` tibble is a massive trove of information. Let's think about some questions we might want to answer with these data:

- How delayed were flights that were headed to Pittsburgh?
- How do departure delays vary over months?
- Which of the three major NYC airports has a better on time percentage for departing flights?

Analysis

Departure delays

Let's start by examining the distribution of departure delays of all flights with a histogram.

```
ggplot(nycflights, aes(x = dep_delay)) +
  geom_histogram()
```

This function says to plot the `dep_delay` variable from the `nycflights` data frame on the x-axis. It also defines a `geom` (short for geometric object), which describes the type of plot you will produce.

Histograms are generally a very good way to see the shape of a single distribution of numerical data, but that shape can change depending on how the data is split between the different bins. You can easily define the binwidth you want to use:

```
ggplot(nycflights, aes(x = dep_delay)) +
  geom_histogram(binwidth = 15)

ggplot(nycflights, aes(x = dep_delay)) +
  geom_histogram(binwidth = 150)
```

Exercise 1 Look carefully at these three histograms. How do they compare? Are features revealed in one that are obscured in another?

If we want to focus only on departure delays of flights headed to Pittsburgh, we need to first `filter` the data for flights with that destination (`dest == "PIT"`) and then make a histogram of the departure delays of only those flights.

```
pit_flights <- nycflights %>%
  filter(dest == "PIT")

ggplot(pit_flights, aes(x = dep_delay)) +
  geom_histogram(binwidth = 15)
```

Let's decipher these two commands (OK, so it might look like four lines, but the first two physical lines of code are actually part of the same command, and so are the second two. It's common to add a break to a new line after `%>%` to help readability).

- Command 1: Take the `nycflights` tibble, `filter` for flights headed to PIT, and save the result as a new tibble called `pit_flights`.
    - `==` means "if it's equal to".
    - `PIT` is in quotation marks since it is a character string.
- Command 2: Basically the same `ggplot2` call from earlier for making a histogram, except that it uses the smaller tibble for flights headed to PIT instead of all flights.

Logical operators: Filtering for certain observations (e.g. flights from a particular airport) is often of interest in tibbles where we might want to examine observations with certain characteristics separately from the rest of the data. To do so we use the `filter` function and a series of **logical operators**. The most commonly used logical operators for data analysis are as follows:

- `==` means "equal to"
- `!=` means "not equal to"
- `>` or `<` means "greater than" or "less than"
- `>=` or `<=` means "greater than or equal to" or "less than or equal to"

We can also obtain numerical summaries for these flights:

```
pit_flights %>%
  summarise(mean_dd = mean(dep_delay),
            median_dd = median(dep_delay),
            n = n())
```

Note that in the `summarise` function we created a list of three different numerical summaries that we were interested in. The names of these elements are user defined, like `mean_dd`, `median_dd`, `n`, and you could customize these names as you like (just don't use spaces in your names). Calculating these summary statistics also require that you know the function calls. Note that `n()` reports the sample size.

Summary statistics: Some useful function calls for summary statistics for a single numerical variable are as follows:

- `mean`
- `median`
- `sd`
- `var`
- `IQR`
- `min`
- `max`

Note that each of these functions take a single vector as an argument, and returns a single value.

We can also filter based on multiple criteria. Suppose we are interested in flights headed to San Francisco (SFO) in February:

```
sfo_feb_flights <- nycflights %>%
  filter(dest == "SFO", month == 2)
```

Note that we can separate the conditions using commas if we want flights that are both headed to SFO **and** in February. If we are interested in either flights headed to SFO **or** in February we can use the `|` instead of the comma.

Exercise 2 Create a new tibble that includes flights headed to SFO in February, and save this tibble as `sfo_feb_flights`. How many flights meet these criteria?

Exercise 3 Describe the distribution of the **arrival** delays of these flights using a histogram and appropriate summary statistics. **Hint:** The summary statistics you use should depend on the shape of the distribution.

Another useful technique is quickly calculating summary statistics for various groups in your tibble. For example, we can modify the above command using the `group_by` function to get the same summary stats for each origin airport:

```
sfo_feb_flights %>%
  group_by(origin) %>%
  summarise(median_dd = median(dep_delay),
            iqr_dd = IQR(dep_delay),
            n_flights = n())
```

Here, we first grouped the data by `origin`, and then calculated the summary statistics.

Exercise 4 Calculate the median and interquartile range for `arr_delay`s of flights in in the `sfo_feb_flights` tibble, grouped by carrier. Which carrier has the most variable arrival delays?

Departure delays over months

Which month would you expect to have the highest average delay departing from an NYC airport?

Let's think about how we would answer this question:

- First, calculate monthly averages for departure delays. With the new language we are learning, we need to
    - `group_by` months, then
    - `summarise` mean departure delays.
- Then, we need to `arrange` these average delays in `desc` ending order

```
nycflights %>%
  group_by(month) %>%
  summarise(mean_dd = mean(dep_delay)) %>%
  arrange(desc(mean_dd))
```

Exercise 5 Suppose you really dislike departure delays, and you want to schedule your travel in a month that minimizes your potential departure delay leaving NYC. One option is to choose the month with the lowest mean departure delay. Another option is to choose the month with the lowest median departure delay. What are the pros and cons of these two choices?

We can also visualize the distributions of departure delays across months using side-by-side box plots:

```
ggplot(nycflights, aes(x = factor(month), y = dep_delay)) +
  geom_boxplot()
```

There is some new syntax here: We want departure delays on the y-axis and the months on the x-axis to produce side-by-side box plots. Side-by-side box plots require a categorical variable on the x-axis, however in the tibble `month` is stored as a numerical variable (numbers 1 - 12). Therefore we can force R to treat this variable as categorical, what R calls a **factor**, variable with `factor(month)`.

On time departure rate for NYC airports

Suppose you will be flying out of NYC and want to know which of the three major NYC airports has the best on time departure rate of departing flights. Suppose also that for you a flight that is delayed for less than 5 minutes is basically "on time". You consider any flight delayed for 5 minutes of more to be "delayed".

In order to determine which airport has the best on time departure rate, we need to

- first classify each flight as "on time" or "delayed",
- then group flights by origin airport,
- then calculate on time departure rates for each origin airport,
- and finally arrange the airports in descending order for on time departure percentage.

Let's start with classifying each flight as "on time" or "delayed" by creating a new variable with the `mutate` function.

```
nycflights <- nycflights %>%
  mutate(dep_type = if_else(dep_delay < 5, "on time", "delayed"))
```

The first argument in the `mutate` function is the name of the new variable we want to create, in this case `dep_type`. Then if `dep_delay < 5` we classify the flight as `"on time"` and `"delayed"` if not, i.e. if the flight is delayed for 5 or more minutes.

Note that we are also overwriting the `nycflights` tibble with the new version of this tibble that includes the new `dep_type` variable.

We can handle all of the remaining steps in one code chunk:

```
nycflights %>%
  group_by(origin) %>%
  summarise(ot_dep_rate = sum(dep_type == "on time") / n()) %>%
  arrange(desc(ot_dep_rate))
```

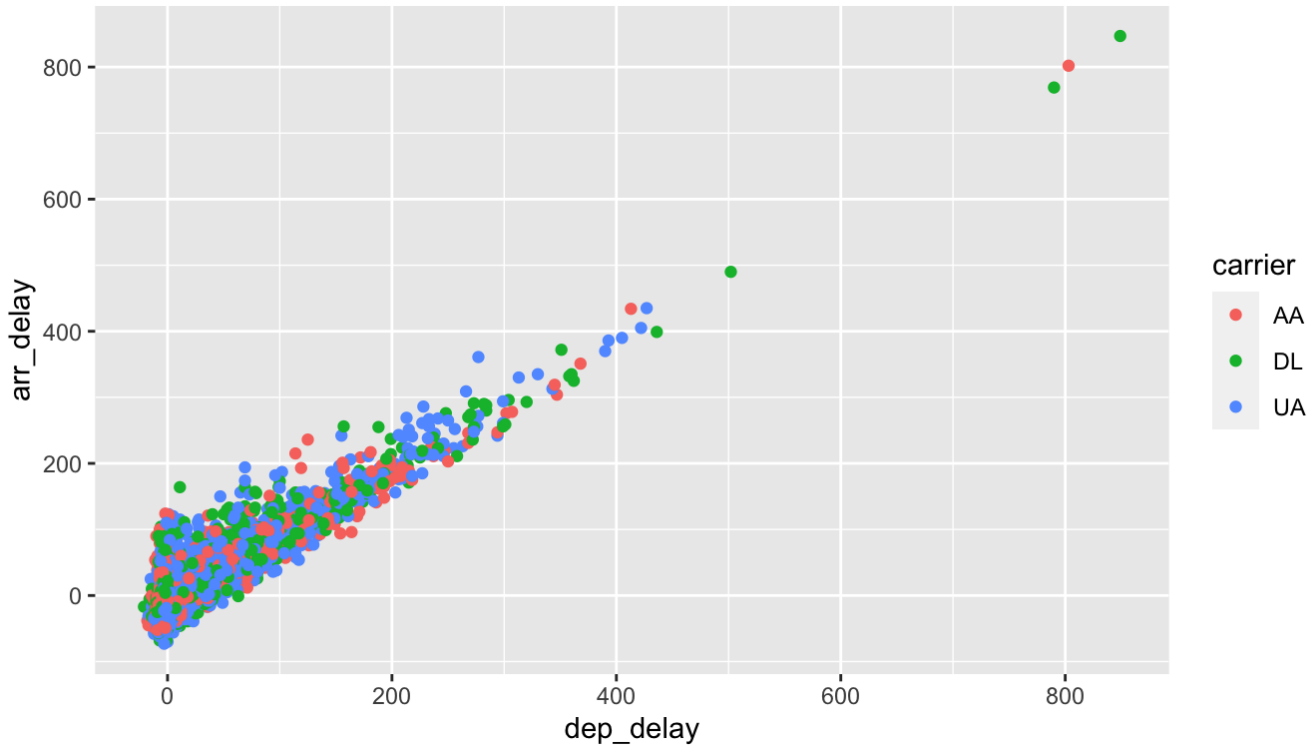
Exercise 6 If you were selecting an airport simply based on on time departure percentage, which NYC airport would you choose to fly out of?

More Practice

Exercise 7 Mutate the tibble so that it includes a new variable that contains the average speed, `avg_speed` traveled by the plane for each flight (in mph). **Hint:** Average speed can be calculated as distance divided by number of hours of travel, and note that `air_time` is given in minutes.

Exercise 8 Make a scatterplot of `avg_speed` vs. `distance`. Describe the relationship between average speed and distance. **Hint:** Use `geom_point()`.

Exercise 9 Replicate the following plot. **Hint:** The tibble plotted only contains flights from American Airlines, Delta Airlines, and United Airlines, and the points are `color`ed by `carrier`. Once you replicate the plot, determine (roughly) what the cutoff point is for departure delays where you can still expect to get to your destination on time.


800
600
400
200
0
arr_delay
0
200
400
600
800
dep_delay
carrier
AA
DL
UA