Introduction to data

William Jasmine

Some define statistics as the field that focuses on turning information into knowledge. The first step in that process is to summarize and describe the raw information – the data. In this lab we explore flights, specifically a random sample of 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. Since this is a large data set, along the way you'll also learn the indispensable skills of data processing and subsetting.

Getting started

Load packages

In this lab, we will explore and visualize the data using the **tidyverse** suite of packages. The data can be found in the companion package for OpenIntro labs, **openintro**.

Let's load the packages.

```
library(tidyverse)
library(openintro)
```

The data

The Bureau of Transportation Statistics (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 transportation data available, such as the flights data we will be working with in this lab.

First, we'll view the `nycflights` data frame. Type the following in your console to load the data:

```
data(nycflights)
```

The data set `nycflights` that shows up in your workspace is a *data matrix*, with each row representing an *observation* and each column representing a *variable*. R calls this data format a **data frame**, which is a term that will be used throughout the labs. For this data set, each *observation* is a single flight.

To view the names of the variables, type the command

```
names(nycflights)
```

```
##  [1] "year"      "month"     "day"       "dep_time"  "dep_delay" "arr_time"
##  [7] "arr_delay" "carrier"   "tailnum"   "flight"    "origin"    "dest"
## [13] "air_time"  "distance"  "hour"      "minute"
```

This returns the names of the variables in this data frame. The **codebook** (description of the variables) can be accessed by pulling up the help file:

```
?nycflights
```

One of the variables refers to the carrier (i.e. airline) of the flight, which is coded according to the following system.

- `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.

Remember that you can use `glimpse` to take a quick peek at your data to understand its contents better.

```
glimpse(nycflights)
```

```
## Rows: 32,735
## Columns: 16
## $ year      <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, ~
## $ month     <int> 6, 5, 12, 5, 7, 1, 12, 8, 9, 4, 6, 11, 4, 3, 10, 1, 2, 8, 10~
## $ day       <int> 30, 7, 8, 14, 21, 1, 9, 13, 26, 30, 17, 22, 26, 25, 21, 23, ~
## $ dep_time  <int> 940, 1657, 859, 1841, 1102, 1817, 1259, 1920, 725, 1323, 940~
## $ dep_delay <dbl> 15, -3, -1, -4, -3, -3, 14, 85, -10, 62, 5, 5, -2, 115, -4, ~
## $ arr_time  <int> 1216, 2104, 1238, 2122, 1230, 2008, 1617, 2032, 1027, 1549, ~
## $ arr_delay <dbl> -4, 10, 11, -34, -8, 3, 22, 71, -8, 60, -4, -2, 22, 91, -6, ~
## $ carrier   <chr> "VX", "DL", "DL", "DL", "9E", "AA", "WN", "B6", "AA", "EV", ~
## $ tailnum   <chr> "N626VA", "N3760C", "N712TW", "N914DL", "N823AY", "N3AXAA", ~
## $ flight    <int> 407, 329, 422, 2391, 3652, 353, 1428, 1407, 2279, 4162, 20, ~
## $ origin    <chr> "JFK", "JFK", "JFK", "JFK", "LGA", "LGA", "EWR", "JFK", "LGA~
## $ dest      <chr> "LAX", "SJU", "LAX", "TPA", "ORF", "ORD", "HOU", "IAD", "MIA~
## $ air_time  <dbl> 313, 216, 376, 135, 50, 138, 240, 48, 148, 110, 50, 161, 87,~
## $ distance  <dbl> 2475, 1598, 2475, 1005, 296, 733, 1411, 228, 1096, 820, 264,~
## $ hour      <dbl> 9, 16, 8, 18, 11, 18, 12, 19, 7, 13, 9, 13, 8, 20, 12, 20, 6~
## $ minute    <dbl> 40, 57, 59, 41, 2, 17, 59, 20, 25, 23, 40, 20, 9, 54, 17, 24~
```

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

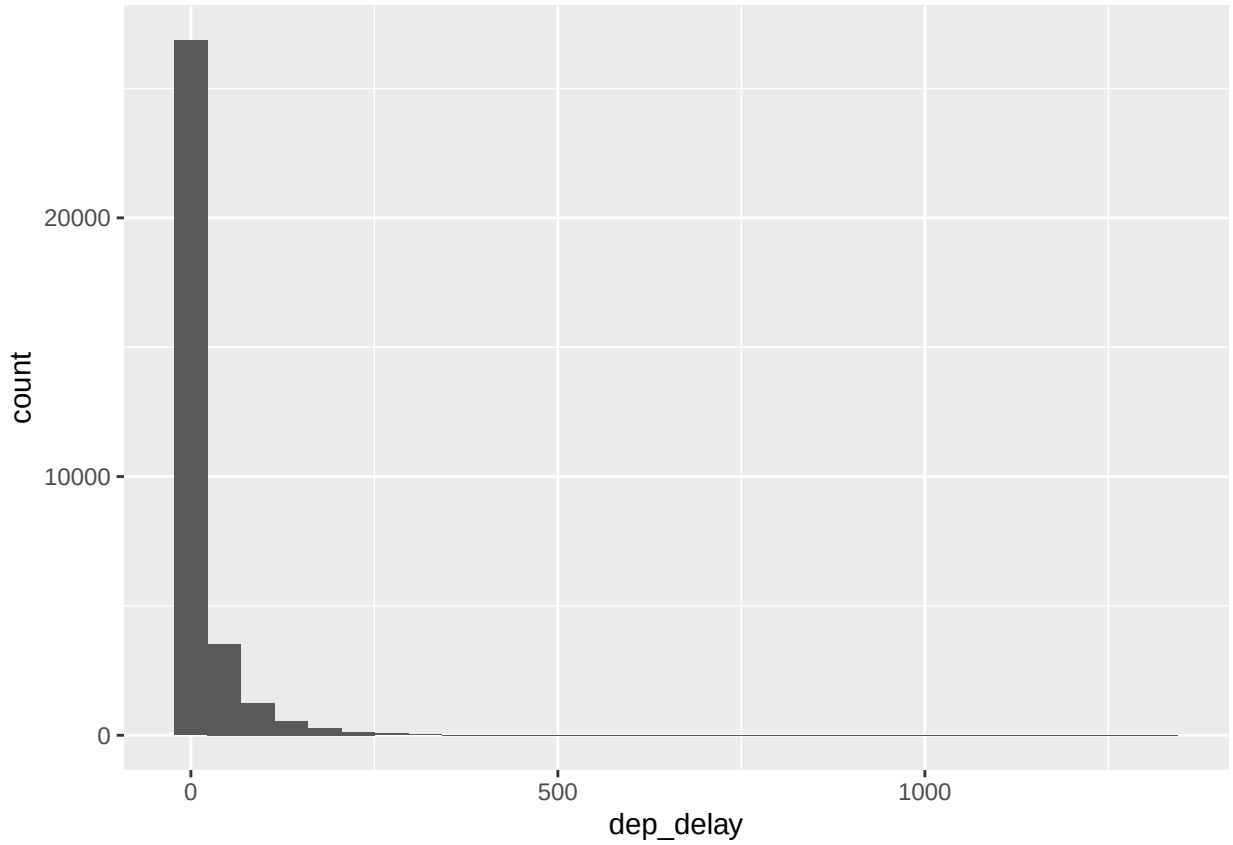
- How delayed were flights that were headed to Los Angeles?
- How do departure delays vary by month?
- Which of the three major NYC airports has the best on time percentage for departing flights?

Analysis

Departure delays

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

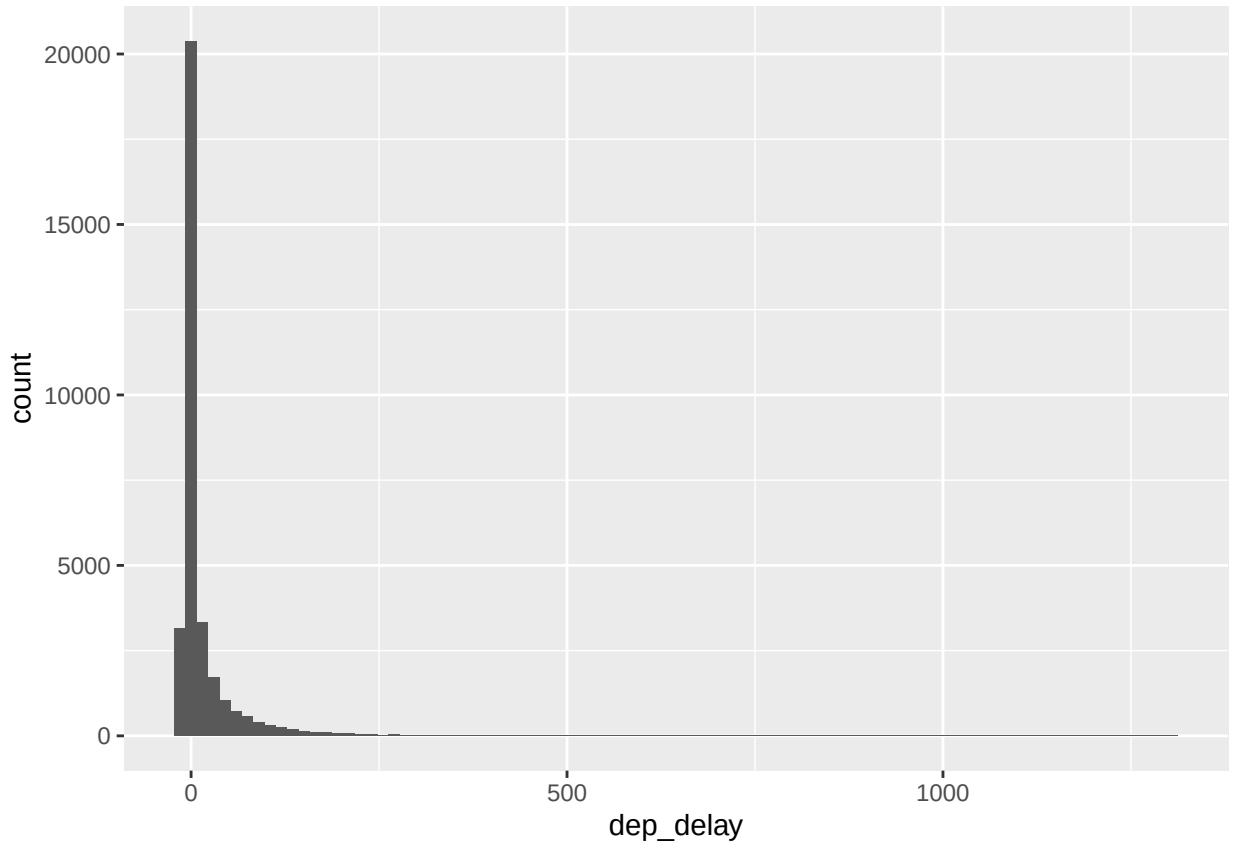
```
ggplot(data = nycflights, aes(x = dep_delay)) +
  geom_histogram()
```



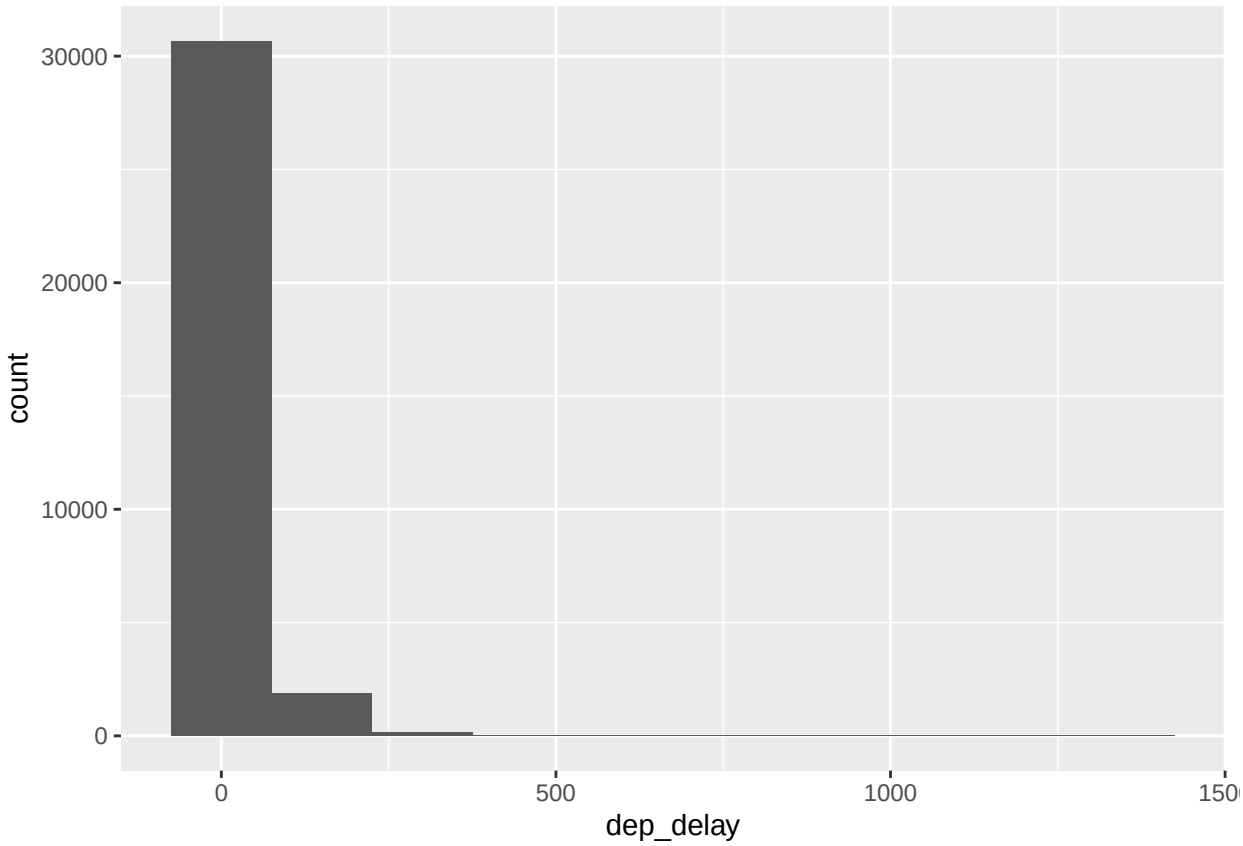
This function says to plot the `dep_delay` variable from the `nycflights` data frame on the x-axis. It also defines a `geom` (short for geometric object), which describes the type of plot you will produce.

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




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



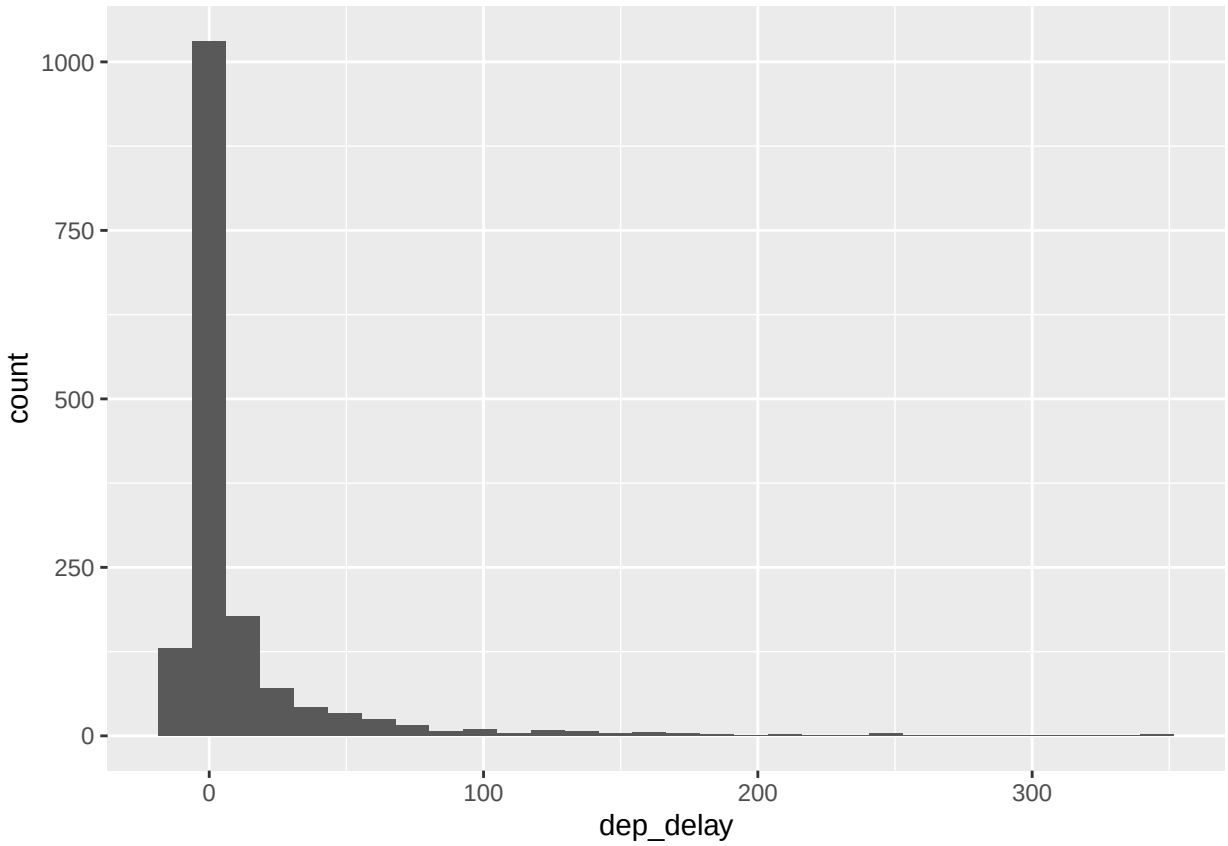
1. Look carefully at these three histograms. How do they compare? Are features revealed in one that are obscured in another?

WJ Response:

Looking at the three histograms, it is clear that the second one reveals a rise and subsequent fall in the distribution when looking at the bins from left to right. The other two histograms do not show this initial rise due to the fact that their bin sizes are larger, obfuscating the some of the information that is clear in the second one.

If you want to visualize only on delays of flights headed to Los Angeles, you need to first `filter` the data for flights with that destination (`dest == "LAX"`) and then make a histogram of the departure delays of only those flights.

```
lax_flights <- nycflights %>%
  filter(dest == "LAX")
ggplot(data = lax_flights, aes(x = dep_delay)) +
  geom_histogram()
```



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. It's common to add a break to a new line after `%>%` to help readability).

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

Logical operators: Filtering for certain observations (e.g. flights from a particular airport) is often of interest in data frames where we might want to examine observations with certain characteristics separately from the rest of the data. To do so, you can 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"

You can also obtain numerical summaries for these flights:

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

```
## # A tibble: 1 x 3
##   mean_dd median_dd     n
```

```
##     <dbl>     <dbl> <int>
## 1    9.78        -1  1583
```

Note that in the `summarise` function you created a list of three different numerical summaries that you were interested in. The names of these elements are user defined, like `mean_dd`, `median_dd`, `n`, and you can customize these names as you like (just don't use spaces in your names). Calculating these summary statistics also requires 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 takes a single vector as an argument and returns a single value.

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

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

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

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

WJ Response:

The following R chunk uses `nrow()` function to count the number of rows in the `sfo_feb_flights` dataframe created above.

```
nrow(sfo_feb_flights)
```

```
## [1] 68
```

The above output reveals that there are 68 flights that departed for the SFO airport from NYC in February.

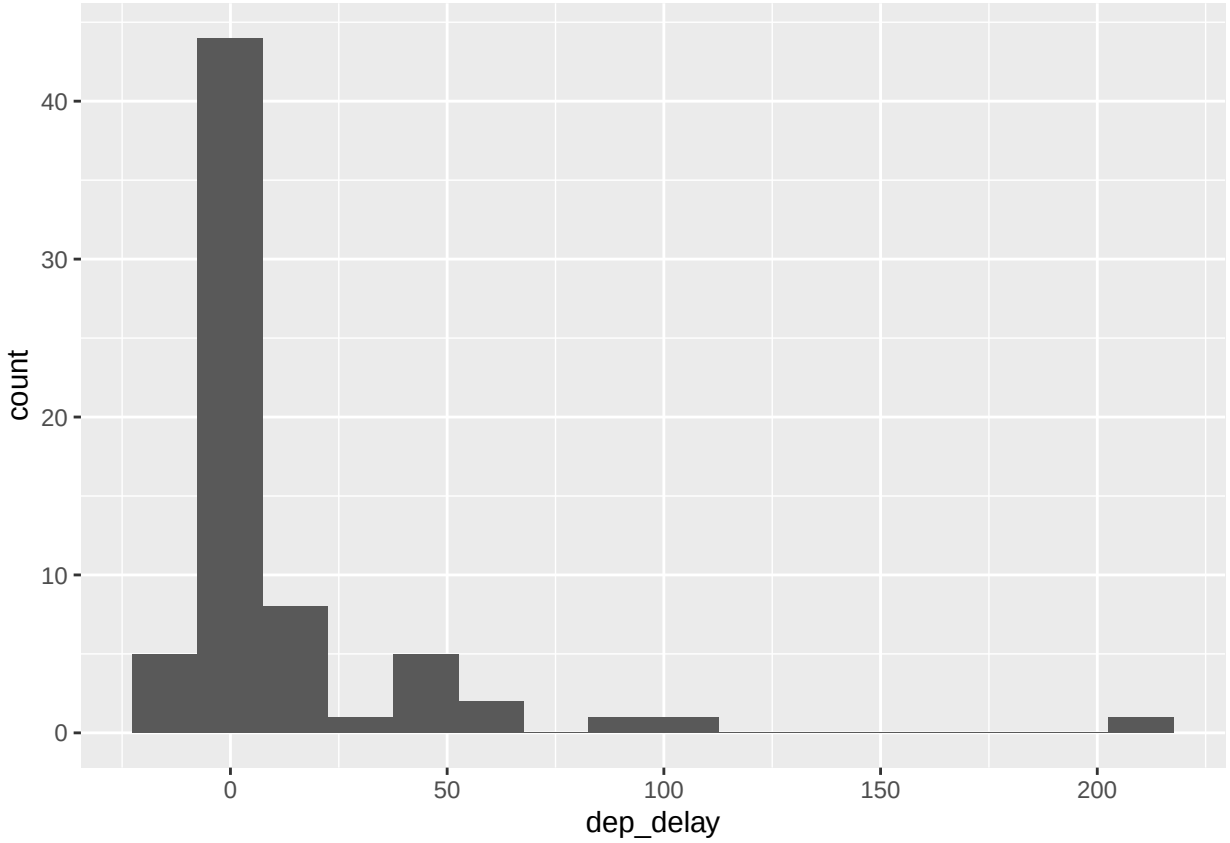
---

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.

WJ Response:

First, the R chunk below provides a histogram to reveal the shape of the distribution of *dep_delay* values in the `sfo_feb_flights` dataframe.

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



The histogram above reveals that this distribution is skewed to right, and contains at least one extreme value (where $dep\_delay > 200$) minutes. Due to these factors, the median and interquartile range (IQR) values will be most useful descriptive statistics in summarizing this distribution (compared to the mean and the standard deviation). These stats are calculated in the R chunk below:

```
sfo_feb_flights %>%
  summarise(median = median(dep_delay),
            IQR = IQR(dep_delay),
            avg = mean(dep_delay),
            std_dev = sd(dep_delay))
```

```
## # A tibble: 1 x 4
##   median   IQR   avg std_dev
##    <dbl> <dbl> <dbl>   <dbl>
## 1     -2    14  10.5    33.3
```

Looking at the above statistics, it is clear that the average and standard deviation paint a pretty bad picture when it comes to flight departures: the average flight is more than 10 minutes late, with a standard deviation greater than 30 minutes. However, as mentioned above, this picture is skewed due to the extreme values visible in the histogram and when looking at the median and IQR we get a more accurate view of what we might expect when taking a flight: the median departure time is actually 2 minutes earlier than the scheduled time, with an IQR of only 14 minutes.

Another useful technique is quickly calculating summary statistics for various groups in your data frame. 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())
```

```
## # A tibble: 2 x 4
##   origin median_dd iqr_dd n_flights
##   <chr>      <dbl>  <dbl>     <int>
## 1 EWR          0.5   5.75         8
## 2 JFK         -2.5  15.2         60
```

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

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

WJ Response:

The R chunk below shows the median and IQR by carrier from the `sfo_feb_flights` dataframe, and arranges them in descending order by their IQR:

```
sfo_feb_flights %>%
  group_by(carrier) %>%
    summarise(median = median(arr_delay),
              IQR = IQR(arr_delay)) %>%
      arrange(desc(IQR))
```

```
## # A tibble: 5 x 3
##   carrier median   IQR
##   <chr>    <dbl> <dbl>
## 1 DL       -15    22
## 2 UA       -10    22
## 3 VX       -22.5  21.2
## 4 AA         5    17.5
## 5 B6       -10.5  12.2
```

According to the output printed above, we can see that the DL (Delta Airlines Inc.) and UA (United Airlines Inc.) carriers have the largest IQR values (22 minutes) when evaluating the *arr_delay* field. As such, these are the carriers with the largest "spread" or variability in arrival delays.

Departure delays by month

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

Let's think about how you could answer this question:

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

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

```
## # A tibble: 12 x 2
##    month mean_dd
##    <int>   <dbl>
##  1     7   20.8
##  2     6   20.4
##  3    12   17.4
##  4     4   14.6
##  5     3   13.5
##  6     5   13.3
##  7     8   12.6
##  8     2   10.7
##  9     1   10.2
## 10     9    6.87
## 11    11    6.10
## 12    10    5.88
```

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?

WJ Response:

The pros and cons of using median or mean to evaluate which month has the worst departure delays goes back to the previously mentioned point regarding the shape of the distribution of departure times for each month. For example, say one month has ten flights and that nine of them have departure delays of 5 minutes while one flight was delayed for 4 hours (240 minutes). The average departure delay in this case is 28.5 minutes, while the median is 5 minutes. Should you expect to wait 28.5 minutes before your flight takes off? Probably not, the median in this case is a more accurate representation of what should be expected as it was not skewed by the one flight with an extreme departure delay value. However, if the departure delays do show to be a reasonably symmetrical distribution month over month this argument is not longer valid: the mean will provide a reasonable estimation of how long one might expect to wait for a flight, now that is no longer heavily influenced by any outlier values.

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. Also supposed 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, you can

- 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 = ifelse(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` data frame with the new version of this data frame 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))
```

```
## # A tibble: 3 x 2
##   origin ot_dep_rate
##   <chr>        <dbl>
## 1 LGA          0.728
## 2 JFK          0.694
## 3 EWR          0.637
```

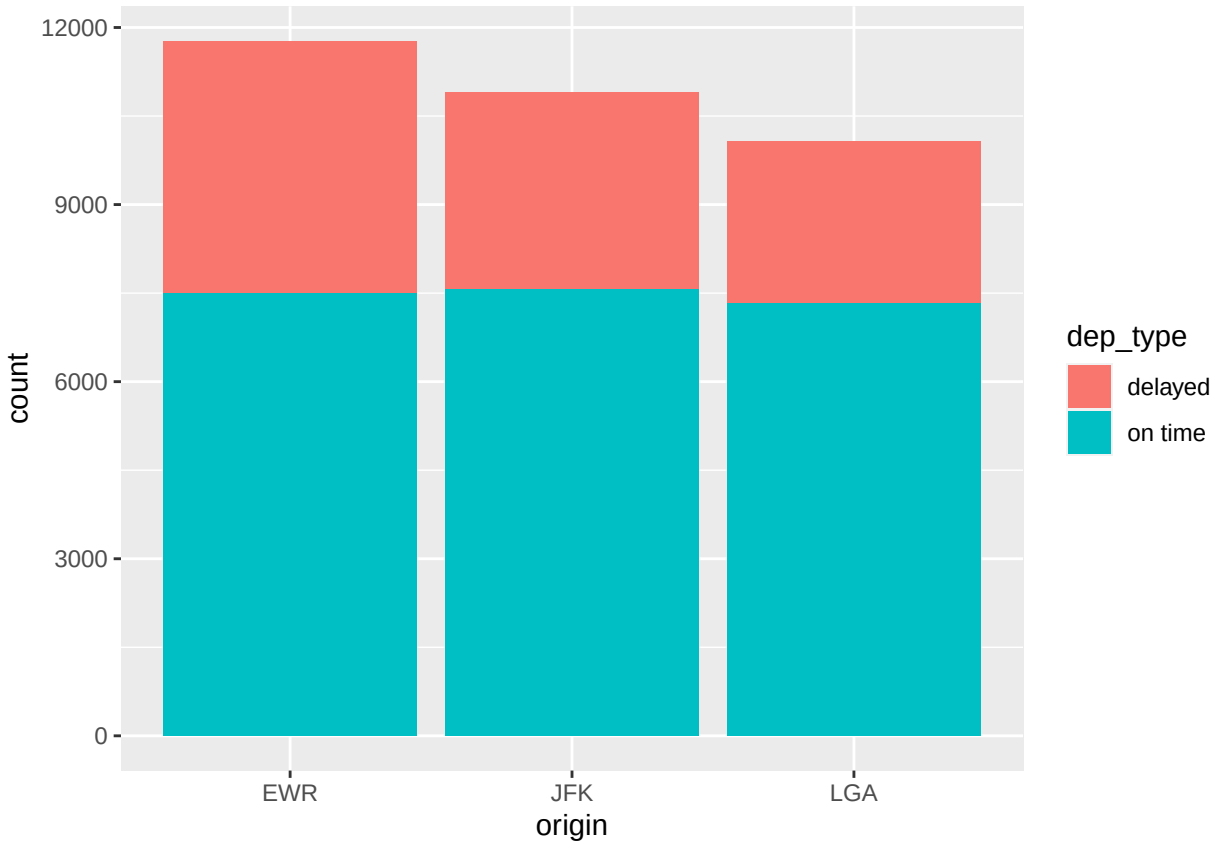
6. If you were selecting an airport simply based on on time departure percentage, which NYC airport would you choose to fly out of?

WJ Response:

Based on the output of the code chunk above, the best NYC airport to depart from based on on time departure percentage is LGA (LaGuardia Airport).

You can also visualize the distribution of on on time departure rate across the three airports using a segmented bar plot.

```
ggplot(data = nycflights, aes(x = origin, fill = dep_type)) +
  geom_bar()
```



More Practice

7. Mutate the data frame 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.

WJ Response:

The code below adds the `avg_speed` column to the `nyc_flights` dataframe by dividing the flight distance by its air time (convered to hours from minutues).

```
nycflights <- nycflights %>%
  mutate(avg_speed = distance / (air_time / 60))
```

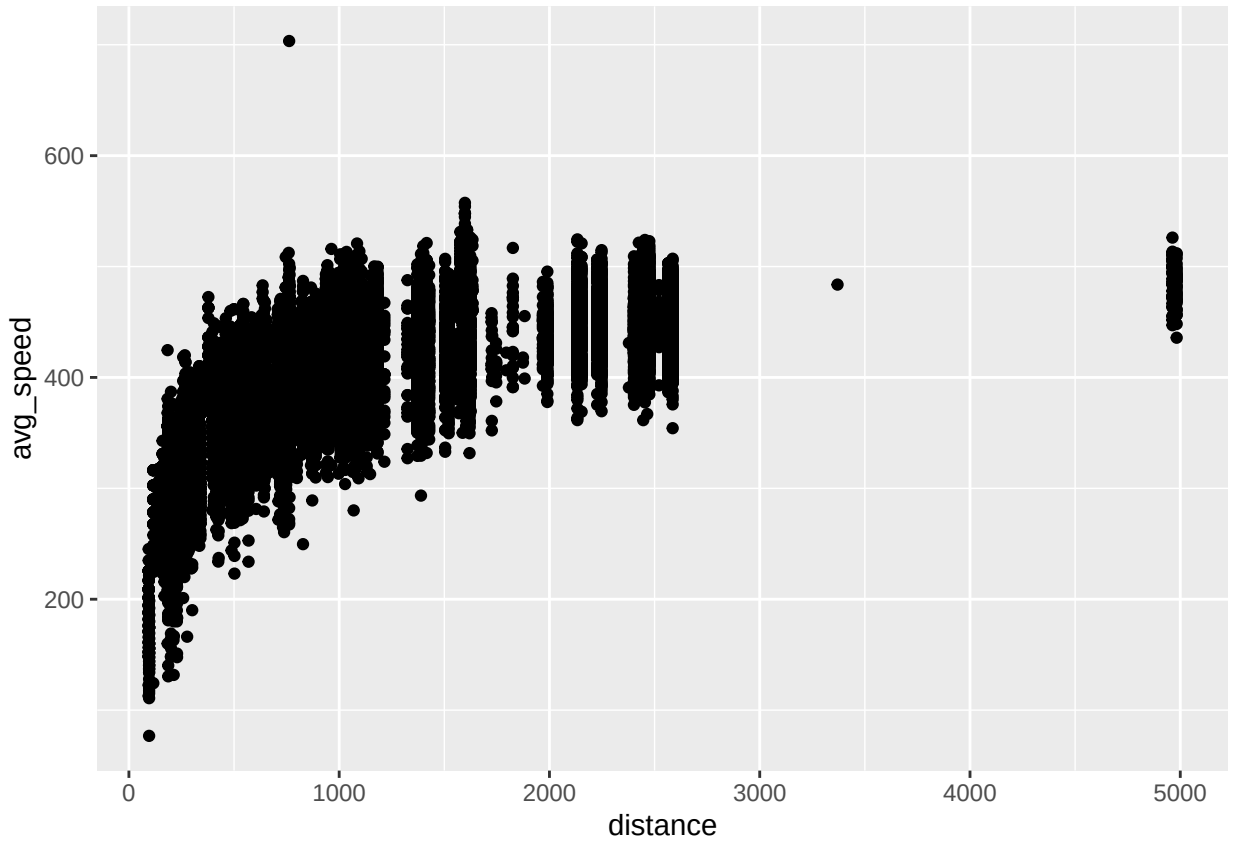
---

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

WJ Response:

The following chunk produces a scatter plot relating a flight's average speed to its distance traveled.

```
ggplot(data = nycflights, aes(x=distance, y=avg_speed)) +
  geom_point()
```



The histogram has some interesting features:

- There are clear vertical lines of points, representing the fixed distances that occur between airports. Each vertical line represents the distribution of average speed values for a specific flight (i.e. LGA –> SFO).
- Really short flights (< 500 miles) appear to have slower average speeds. This is likely due to the fact that the time it takes for the flight to enter and exit cruising altitude (when the plane is slower moving) is a larger percentage of the total flight time.
- As distance increases average flight speed increases, but only up until the flights are less than 1,000 miles. At that point the distribution of flight speeds tend to be relatively similar. This is even true for the flight with the longest distance traveled (~ 5000 miles). This behavior is likely due to the fact that as flight distance (and thus flight time) increases, the time spent at cruising altitude (in which the plane's speed does not fluctuate much, and is at its peak) is an increasingly large percentage of the total flight time.
- There is one outlier value that has an average flight speed greater than 700 miles per hour. Further investigation would need to figure out the reason for this particular value, or if it's an error.

---

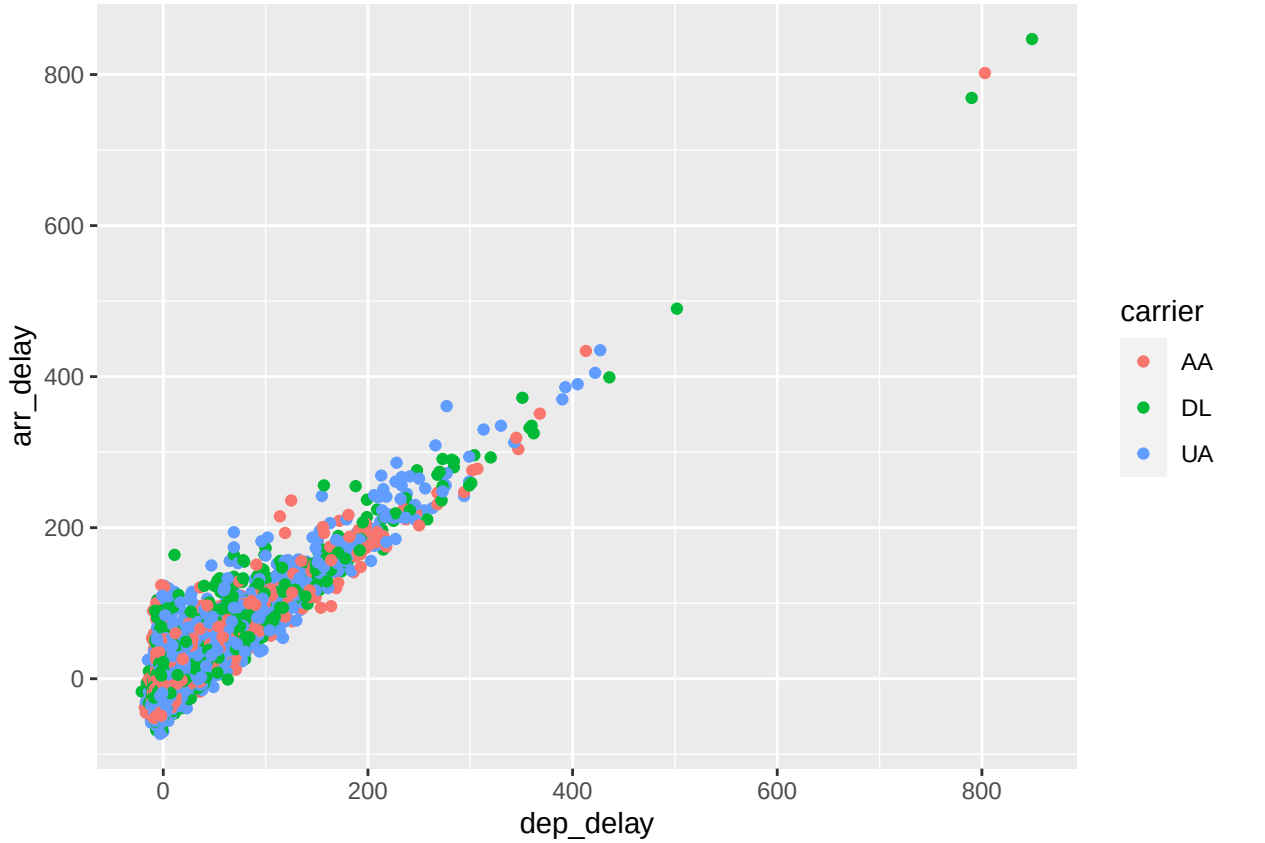
9. Replicate the following plot. **Hint:** The data frame plotted only contains flights from American Airlines, Delta Airlines, and United Airlines, and the points are `color`ed by `carrier`. Once you replicate the plot, determine (roughly) what the cutoff point is for departure delays where you can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

WJ Response:

Code to recreate plot seen in original .Rmd file:

```
plot_df <- nycflights %>%
  filter(carrier == 'AA' | carrier == 'DL' | carrier == 'UA')

ggplot(plot_df, aes(x=dep_delay, y=arr_delay, color=carrier)) +
  geom_point()
```



Looking at the output of the plot shown above we can estimate that so long as your departure time is within an ~60 minutes of when it was scheduled, you can still expect to arrive at your destination on time.
