Introduction to data

Alyssa Gurkas

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.

Set Working Directory Below I set my working directory so the Rmd knows to pull files from the correct folder as question 9 loads a graphic from the Lab 2 folder.

```
setwd("C:/Users/AGURKAS/OneDrive - Environmental Protection Agency (EPA)/Profile/Documents/Masters Prog
```

```
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:

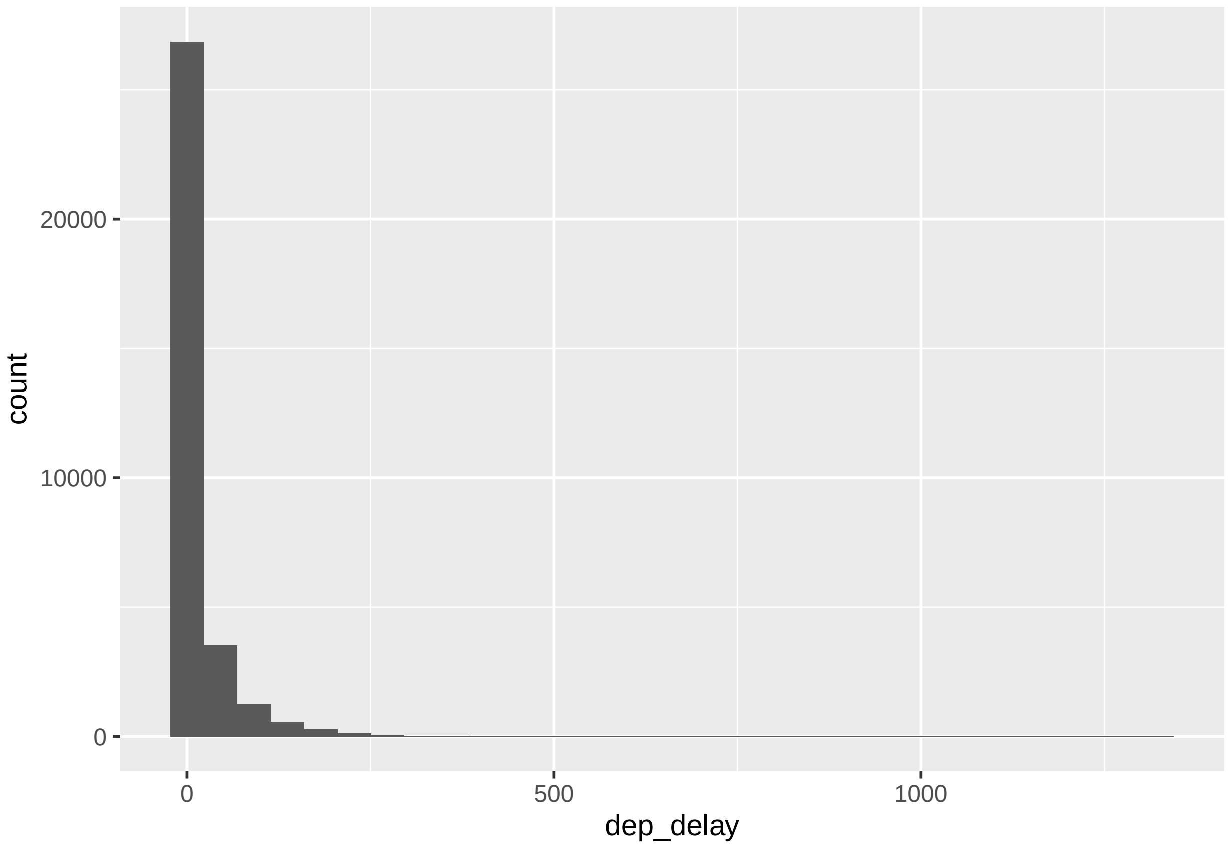
- How delayed were flights that were headed to Los Angeles?
- How do departure delays vary by month?
- Which of the three major NYC airports has the best on time percentage for departing flights?

Analysis

Departure delays

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

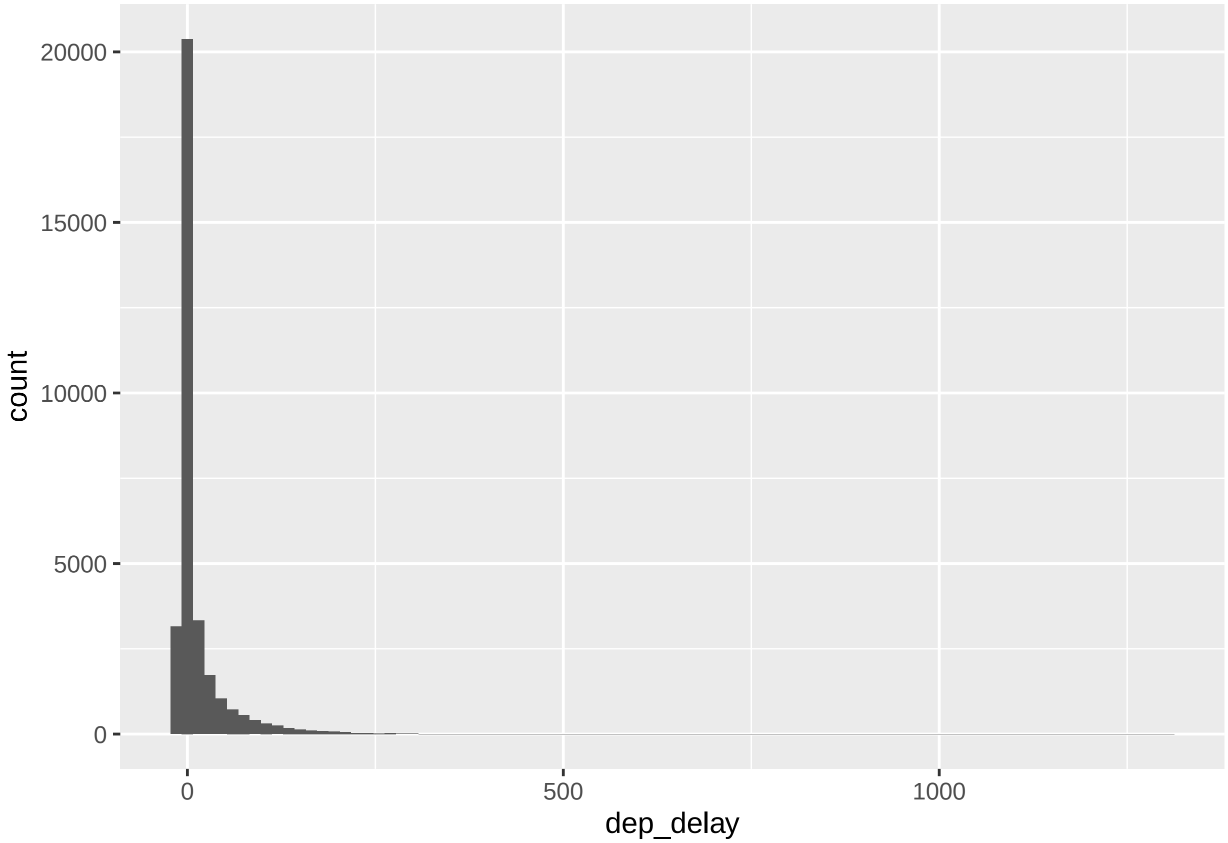
```
ggplot(data = nycflights, aes(x = dep_delay)) +
  geom_histogram()
```



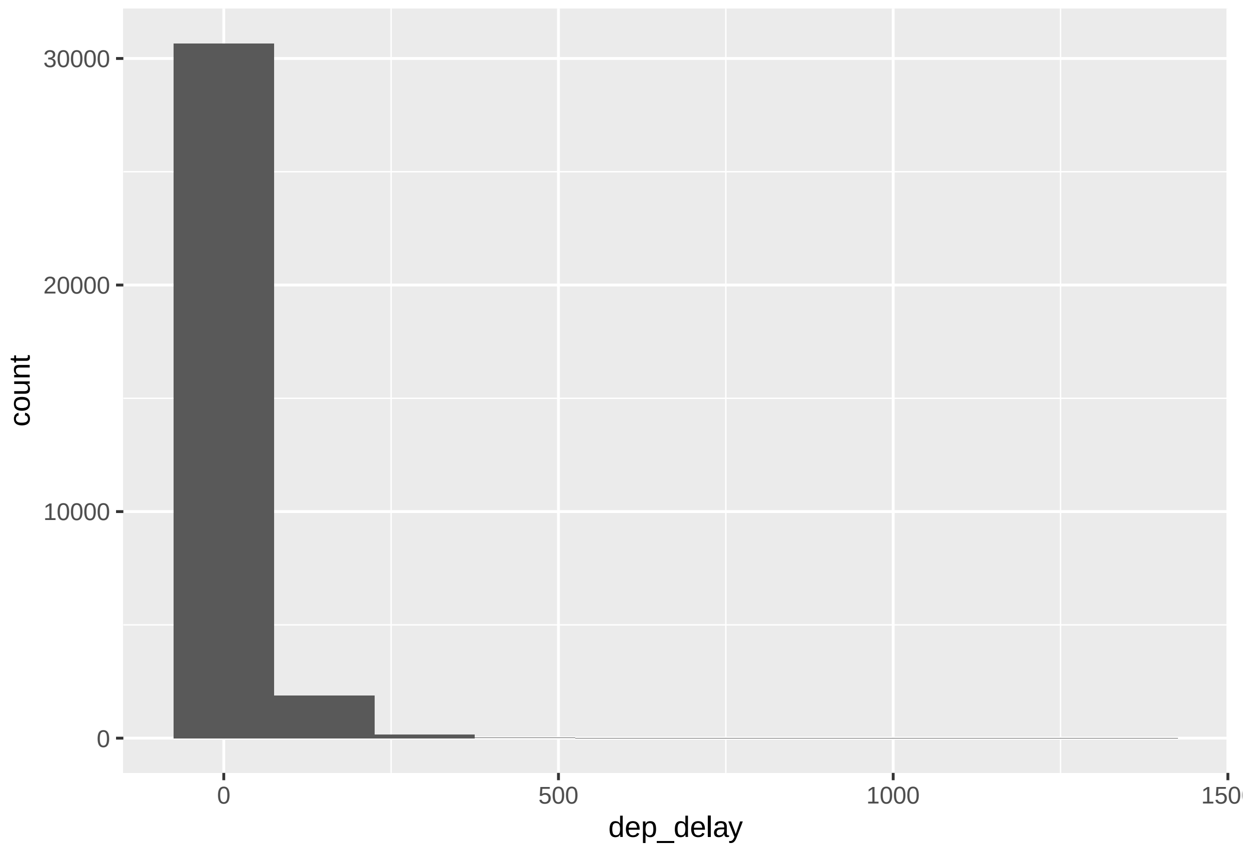
This function says to plot the `dep_delay` variable from the `nycflights` data frame on the x-axis. It also defines a `geom` (short for geometric object), which describes the type of plot you will produce.

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)
```



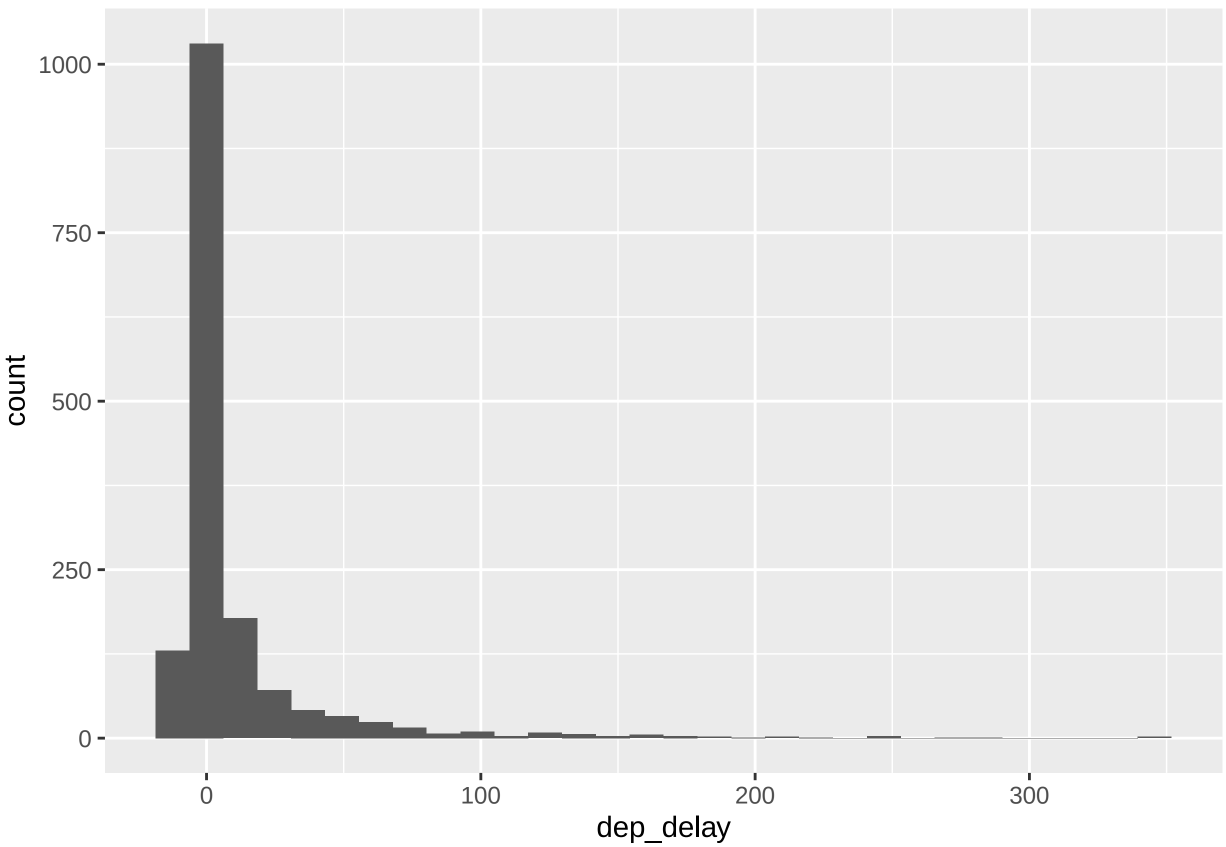
1. Look carefully at these three histograms. How do they compare? Are features revealed in one that are obscured in another?

Exercise 1 Response Note: dep_delay is in minutes.

Response: The histograms are displaying the same data, but visualize the data differently through bin sizes. These histograms include on time and delayed flights. The first histogram uses the default bin size and it appears that most flights are on time or within 250 minutes of the estimated departure time. The second histogram creates a bin width of 15 minutes, which creates a similar graph but highlights that most departures are within ~20 minutes of the estimated departure time. The third histogram changes the bin width to 150 minutes which creates a clearer graph, showing that the majority of flights are within 150 minutes of the estimated departure time.

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.

```r
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?

```


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

nrow(sfo_feb_flights)
```

Exercise 2 Response

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

There are 68 flights that meet these criteria.

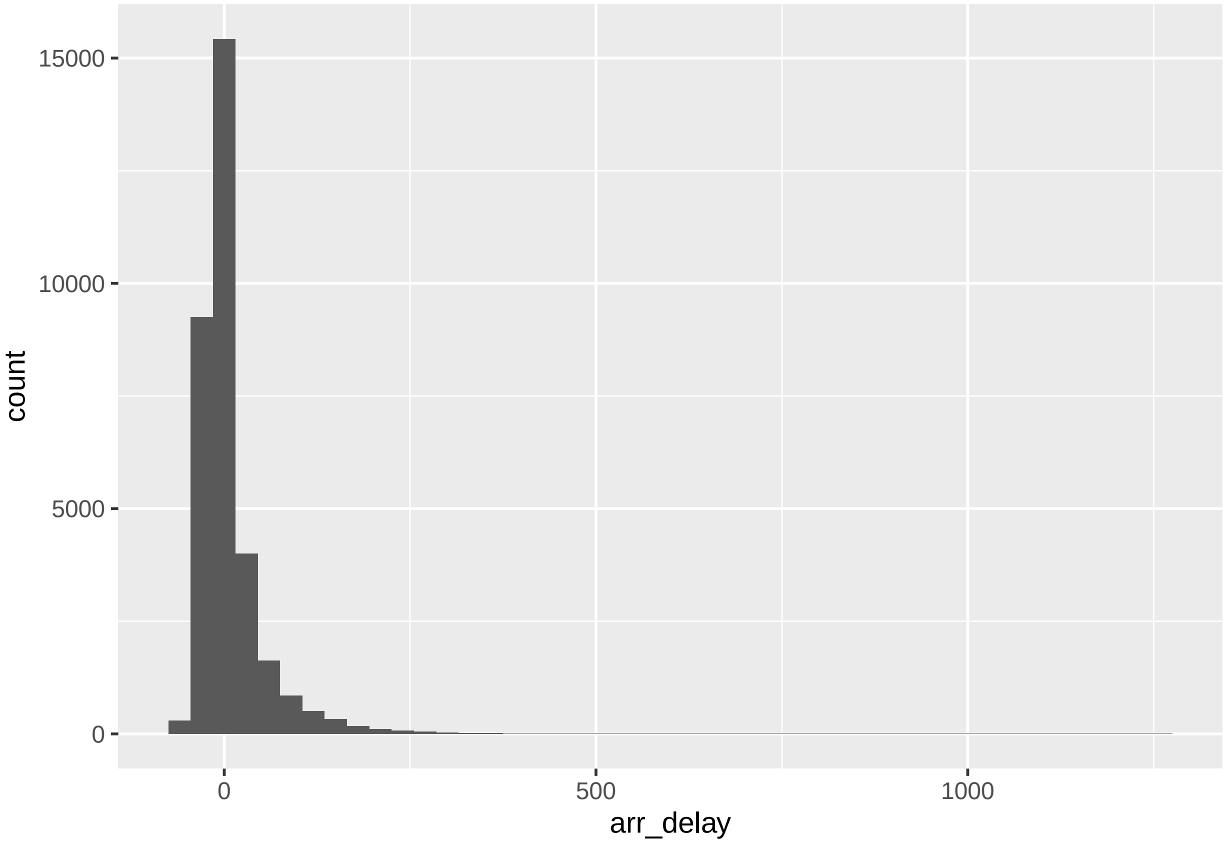
3. Describe the distribution of the **arrival** delays of these flights using a histogram and appropriate summary statistics. **Hint:** The summary statistics you use should depend on the shape of the distribution.

```
nycflights %>%
  summarise(mean_arr_d   = mean(arr_delay),
            median_arr_d = median(arr_delay),
            max_arr_d = max(arr_delay),
            min_arr_d = min(arr_delay),
            sd = sd(arr_delay),
            iqr = IQR(arr_delay),
            n         = n())
```

Exercise 3 Response

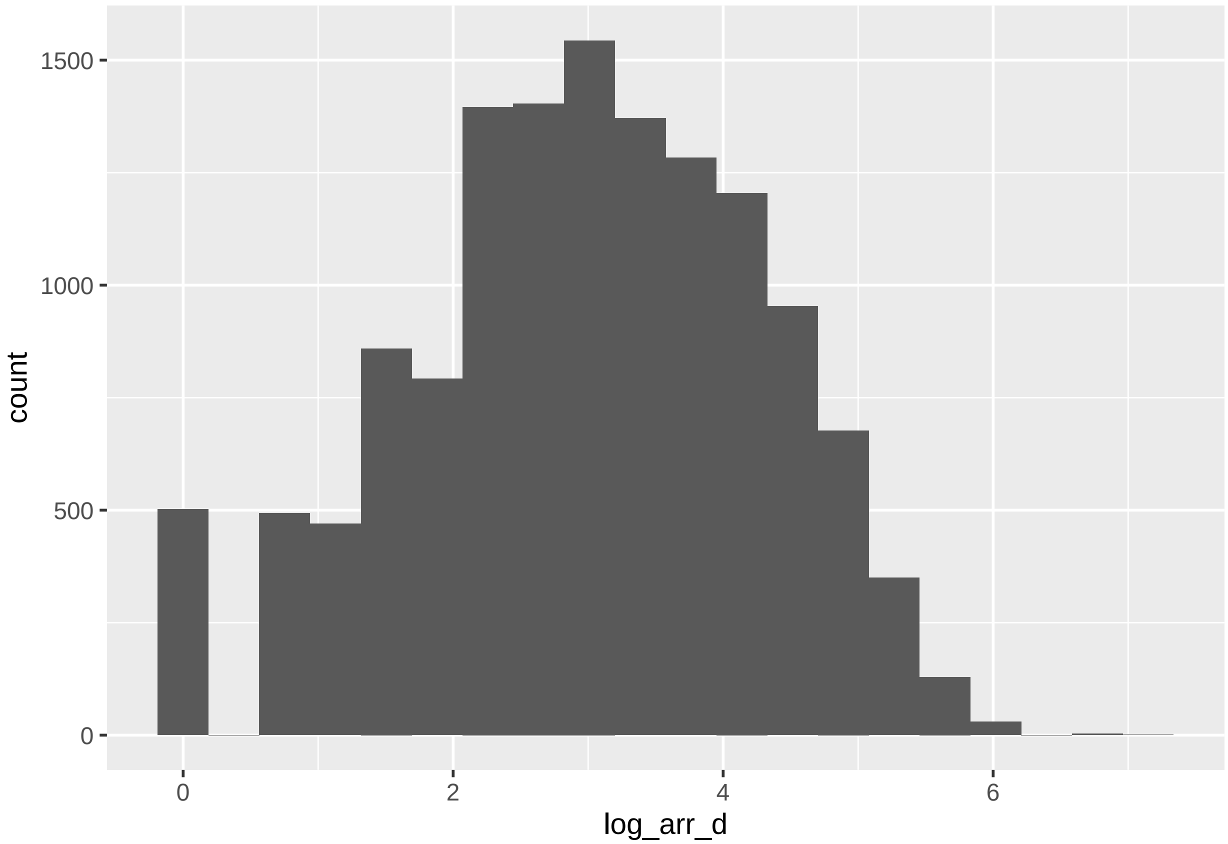
```
## # A tibble: 1 x 7
##   mean_arr_d median_arr_d max_arr_d min_arr_d    sd   iqr     n
##        <dbl>        <dbl>     <dbl>     <dbl> <dbl> <dbl> <int>
## 1       7.10           -5      1272       -73  44.7    31 32735
```

```
ggplot(data = nycflights, aes(x=arr_delay)) +
  geom_histogram(binwidth=30)
```



This data frame is right skewed (or positively skewed) because the mean is greater than the median. Median and IQR are more robust to skewness and outliers than mean and standard deviation therefore, for this data set it is more helpful to use median and IQR to describe the center and spread. As shown in the summarize statement, the median arrival delay is 7.10 minutes, the IQR is 31, and there are 32735 observations in this dataset. If I wanted to transform the data, I could use scale_x_log10() however the number of bins or bin width could impact the interpretation of this data. Another way to transform the data would be:

```r
nycflights %>%
  mutate(log_arr_d = log(arr_delay)) %>%
  ggplot(aes(x=log_arr_d)) +
  geom_histogram(bins=20)
```



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:

```r
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.

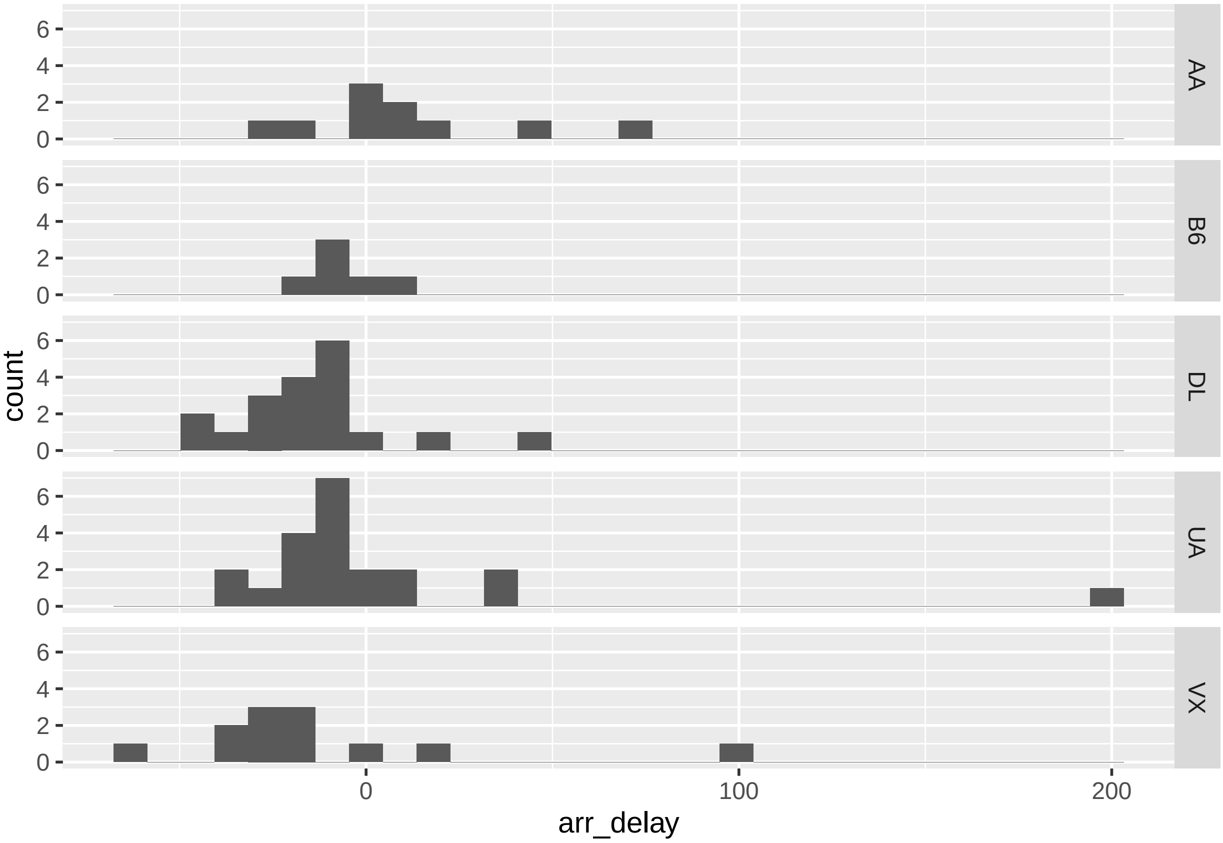
4. Calculate the median and interquartile range for `arr_delay`s of flights in in the `sfo_feb_flights` data frame, grouped by carrier. Which carrier has the most variable arrival delays?

```
sfo_feb_flights %>%
  group_by(carrier) %>%
  summarise(median_ad = median(arr_delay), iqr_ad = IQR(arr_delay), var_ad = var(arr_delay))
```

Exercise 4 Response

```
## # A tibble: 5 x 4
##   carrier median_ad iqr_ad var_ad
##   <chr>       <dbl>  <dbl>  <dbl>
## 1 AA            5     17.5   868.
## 2 B6          -10.5   12.2   121.
## 3 DL          -15     22     485.
## 4 UA          -10     22    2335.
## 5 VX          -22.5   21.2  1669.
```

```
ggplot(data = sfo_feb_flights, aes(x=arr_delay)) +
  geom_histogram() +
  facet_grid(rows=vars(`carrier`))
```



Delta Air Lines Inc. (DL) and United Air Lines Inc. (UA) have the same interquartile range for the arrival delays. This indicates that DL and UA have the most arrival delays. When plotting the histograms for all of the carrier arrival delays, it shows that UA has one outlier that may be skewing the IQR result. **Does this mean that DL should be considered the airline carrier with most variable arrival delays? When calculating which carrier has the most variable arrival delays, should negative values**

be filtered out? If we wanted to filter for positive values to determine the carrier with the most variable delays we could complete the following:

```
sfo_feb_flights %>%
  filter(arr_delay > 0) %>%
  group_by(carrier) %>%
  summarise(median_ad = median(arr_delay), iqr_ad = IQR(arr_delay))
```

```
## # A tibble: 5 x 3
##   carrier median_ad iqr_ad
##   <chr>       <dbl>  <dbl>
## 1 AA            8     26.5
## 2 B6            7      4
## 3 DL           34.5   13.5
## 4 UA           21.5   29.5
## 5 VX           57.5   41.5
```

When filtering for only positive values and reviewing the interquartile range of all the carriers, it seems that Virgin America has the most variable 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?

Exercise 5 Response Using the lowest median means you calculating the 50th percentile time, so if the future is like the past, you can expect the delay time would be a similar value. However, the median does not account for outliers.

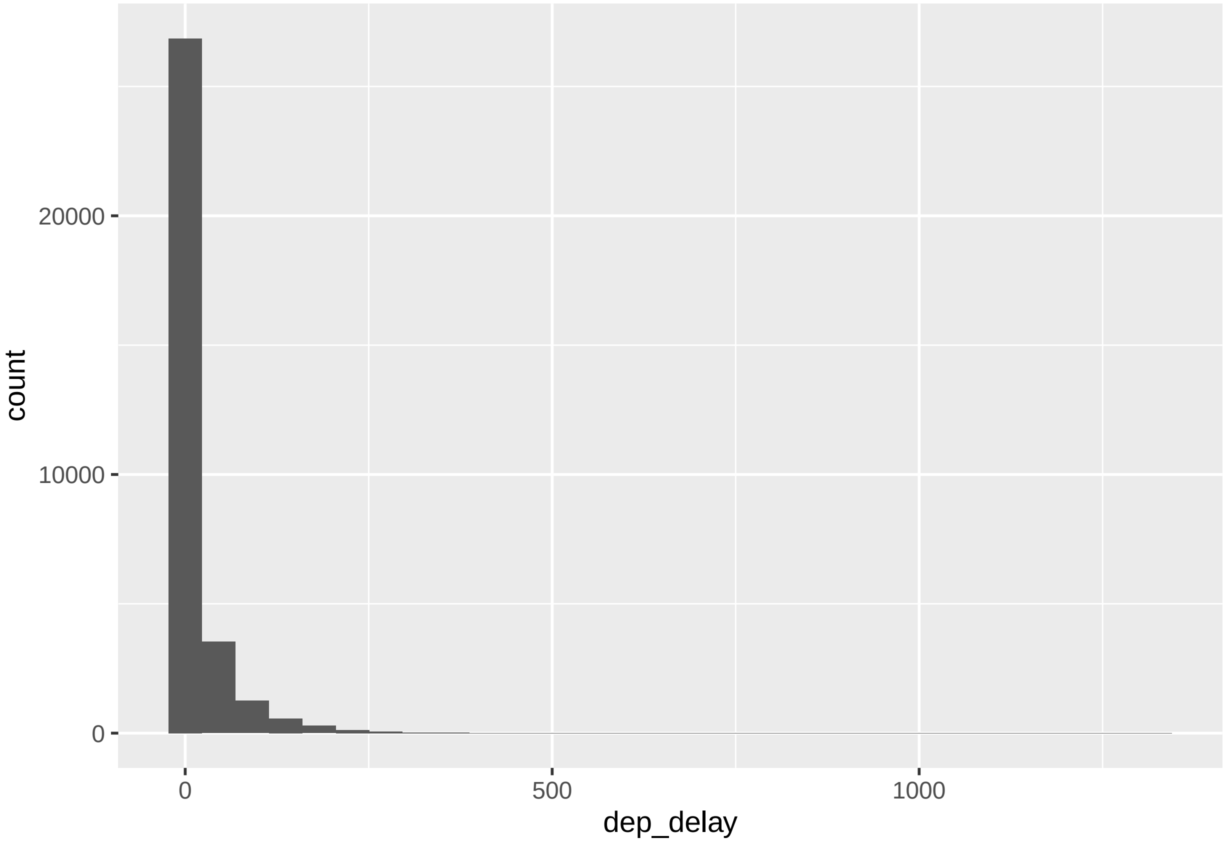
The mean does account for outliers in the data set. If the sample is representative of the population (in this case flight delays), it may be a good estimate of the expected departure delay. However, if an outlier is uncommon and not representative of common flight delays, it may provide a poor estimate for future flight delays.

Below is code to calculate the mean and median departure delays by month:

```
nycflights %>%
  group_by(month) %>%
  summarise(mean_dd = mean(dep_delay), median_dd = median(dep_delay)) %>%
  arrange(month)
```

```
## # A tibble: 12 x 3
##    month mean_dd median_dd
##    <int>   <dbl>     <dbl>
##  1     1   10.2         -2
##  2     2   10.7         -2
##  3     3   13.5         -1
##  4     4   14.6         -2
##  5     5   13.3         -1
##  6     6   20.4          0
##  7     7   20.8          0
##  8     8   12.6         -1
##  9     9    6.87        -3
## 10    10    5.88        -3
## 11    11    6.10        -2
## 12    12   17.4          1
```

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



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.

```r
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:

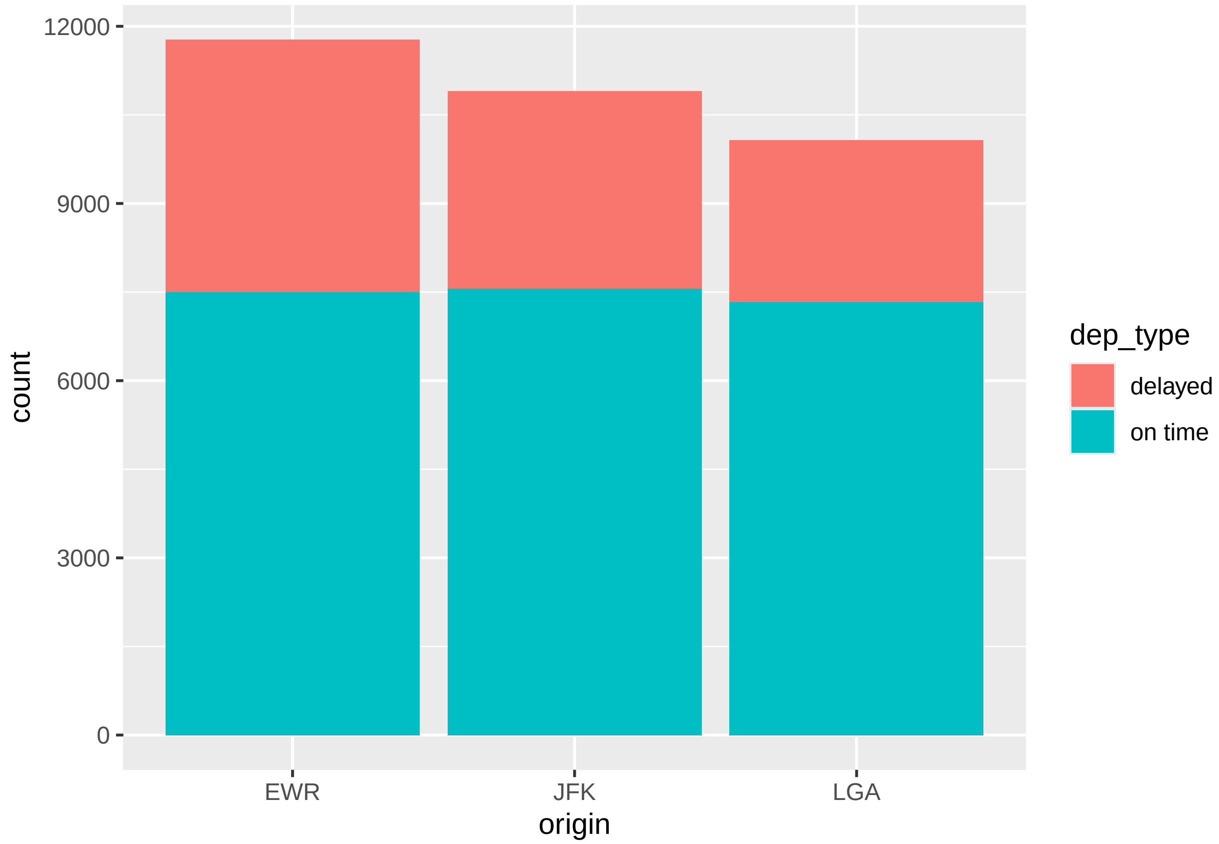
```r
nycflights %>%
  group_by(origin) %>%
  summarise(ot_dep_rate = sum(dep_type == "on time") / n()) %>%
  arrange(desc(ot_dep_rate))
```

```
## # 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?

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

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



Exercise 6 Response If I were selecting an airport simply based on the percentage of on time departures, I would select LGA. LGA has the highest percentage of on time departures with 72.8% of flights departing on time. Below is a table of the departures:

```
ot_d_dep <- nycflights %>%
  group_by(origin) %>%
  summarise(`On Time Rate` = sum(dep_type == "on time") / n(), `Delayed Rate` = sum(dep_type == "delaye
  rename("Origin" = "origin")

flextable(ot_d_dep) %>%
  theme_vanilla()
```

Origin	On Time Rate	Delayed Rate
EWR	0.6369892	0.3630108
JFK	0.6935854	0.3064146
LGA	0.7279229	0.2720771

More Practice

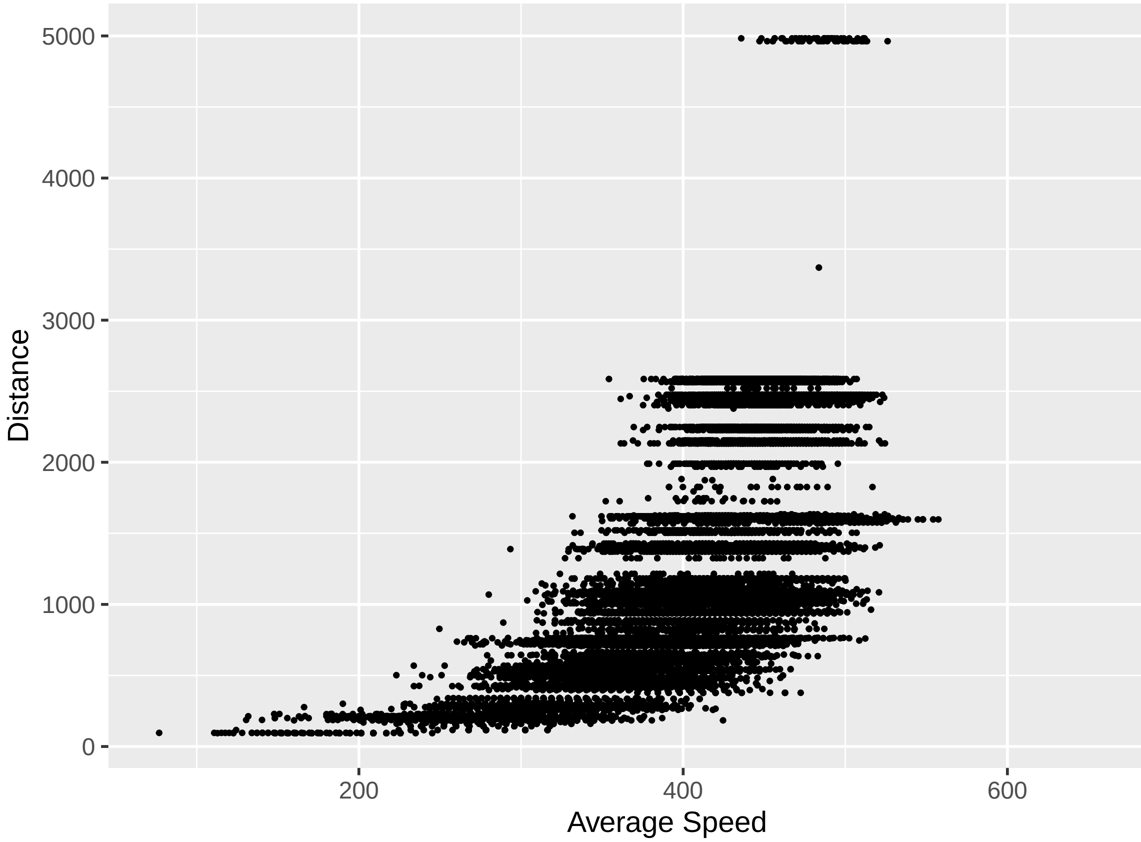
7. Mutate the data frame so that it includes a new variable that contains the average speed, `avg_speed` traveled by the plane for each flight (in mph). **Hint:** Average speed can be calculated as distance divided by number of hours of travel, and note that `air_time` is given in minutes.

```
nycflights <- nycflights %>%
  mutate(avg_speed = distance/(air_time/60))
# mutate nycflights df to include variable "avg_speed" distance/# hrs of travel. Note* convert air_time
```

Exercise 7 Response

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

```
ggplot(data = nycflights, aes(avg_speed,distance)) +
  geom_point(size=.5) +
  xlab("Average Speed") +
  ylab("Distance")
```

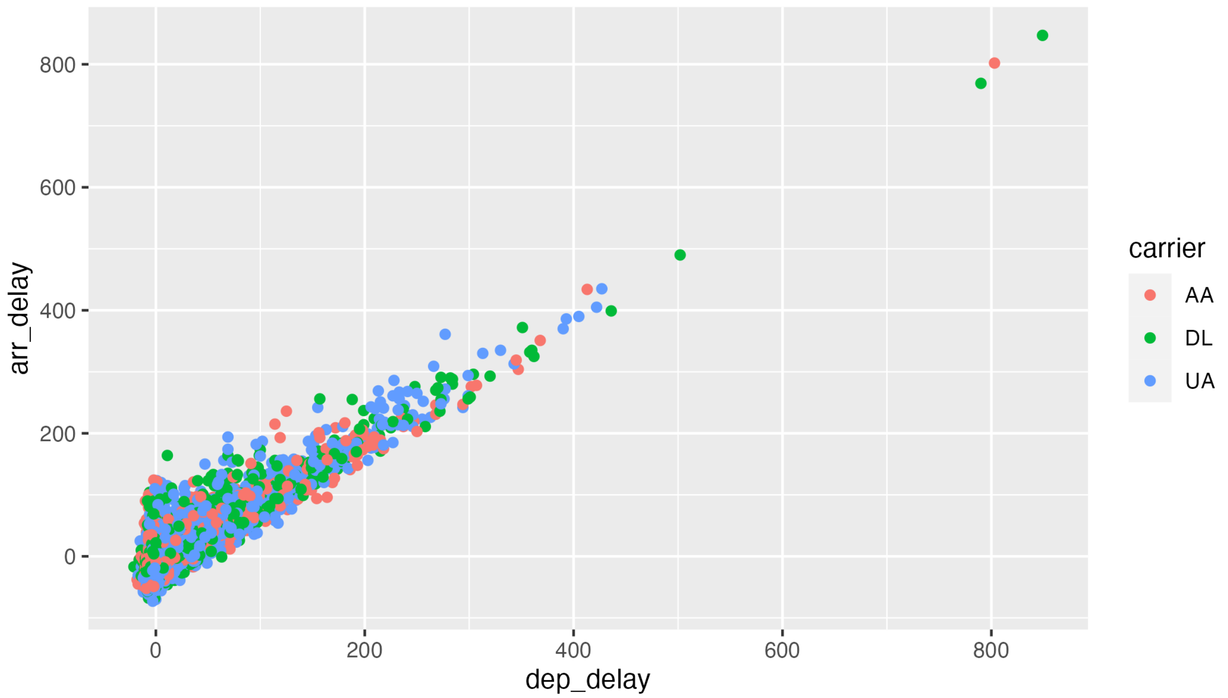


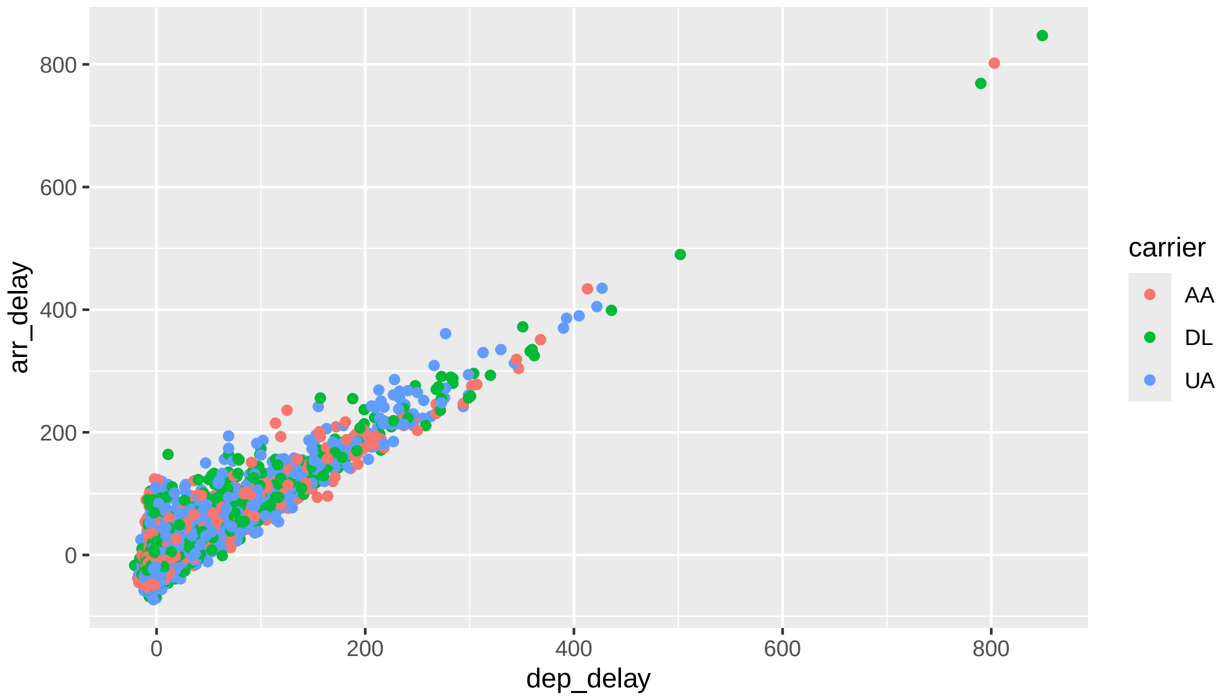
Exercise 8 Response

```
# make scatterplot w/ 'avg_speed' vs. 'distance'.
```

This is an exponential relationship. As flight distance increases, the average speed increases too in a non-linear fashion. The flights generally do not exceed 550 mph.

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 `colored` 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.





```
## # A tibble: 1 x 18
##    year month   day dep_time dep_delay arr_time arr_delay carrier tailnum flight
##   <int> <int> <int>    <int>     <dbl>    <int>     <dbl> <chr>   <chr>    <int>
## 1  2013     3     8     1709        40     1937         0 UA      N77296    1624
## # i 8 more variables: origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, dep_type <chr>, avg_speed <dbl>
```

The maximum delay that did not relate in an arrival delay was 40 minutes. This indicates

that 40 minutes is roughly the cutoff point for departure delays where you can still expect to get to your destination on time.