

Introduction to data

First thing's first, I need to set the working directory:

```
setwd("F:/git/cuny-msds/data606/labs/lab-2")
```

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, ~
## $ 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:

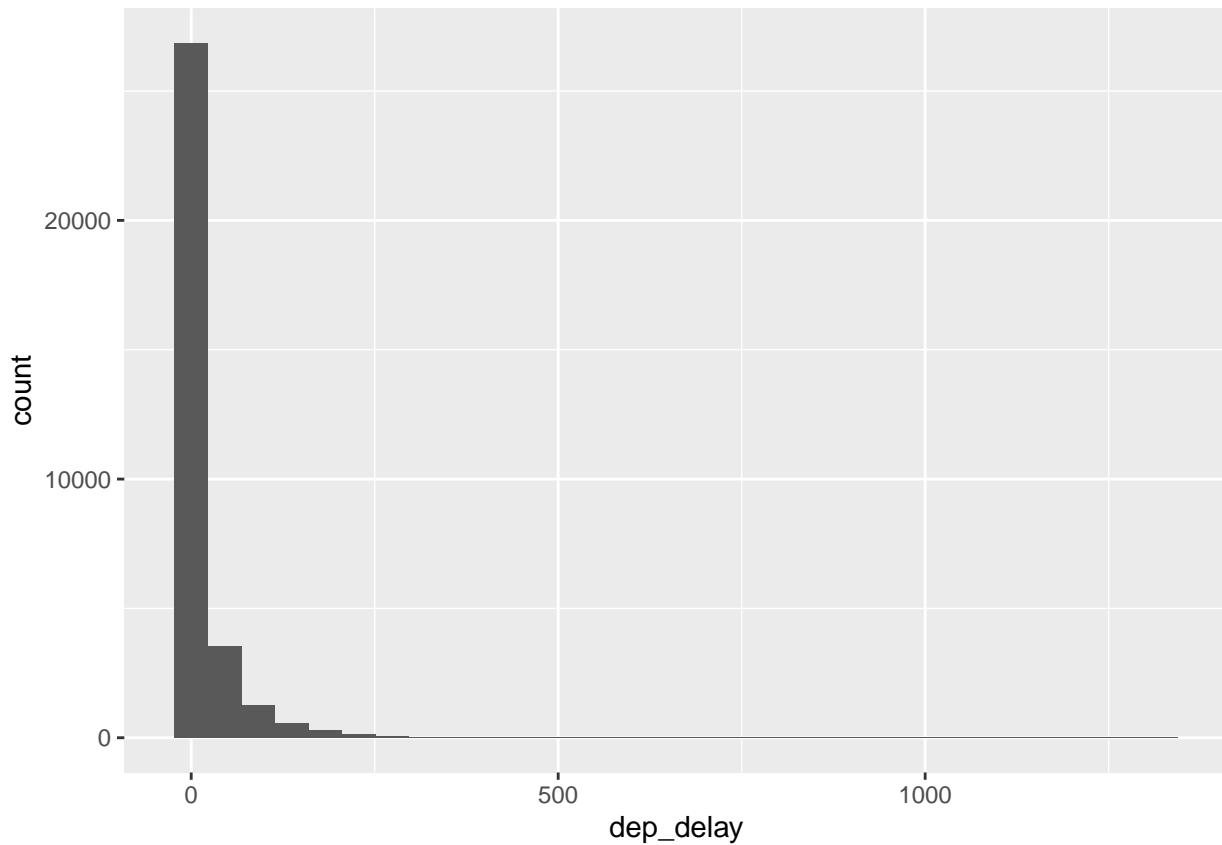
- How delayed were flights that were headed to Los Angeles?
- How do departure delays vary by month?
- Which of the three major NYC airports has the best on time percentage for departing flights?

Analysis

Departure delays

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

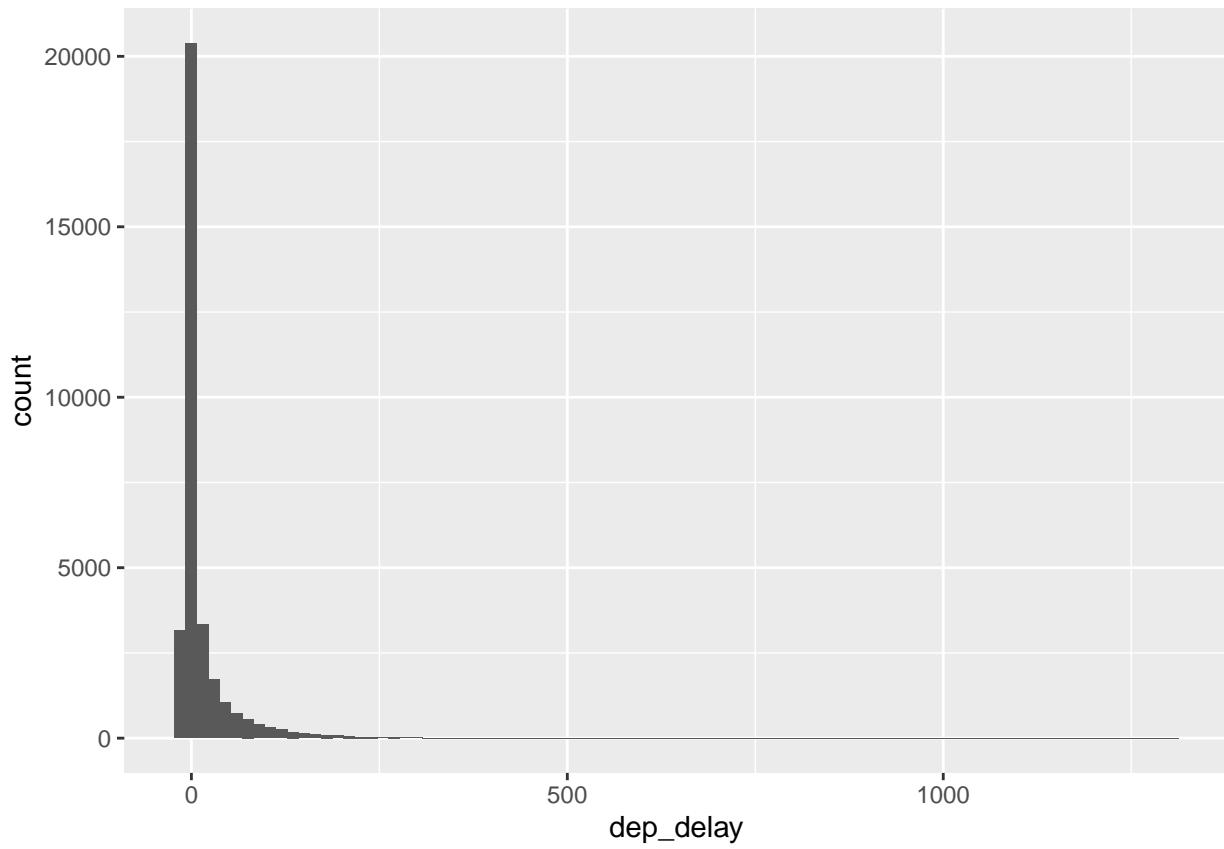
```
ggplot(data = nycflights, aes(x = dep_delay)) +  
  geom_histogram()
```



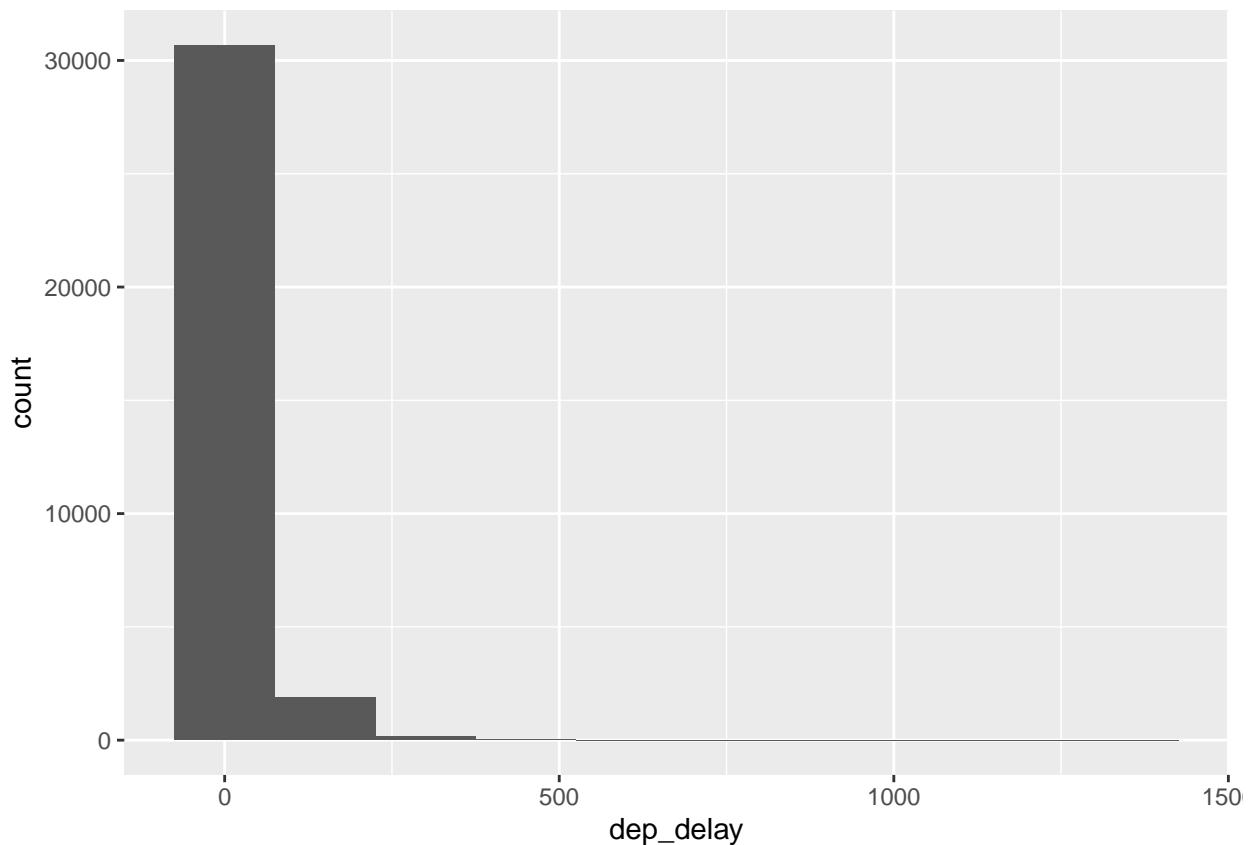
This function says to plot the `dep_delay` variable from the `nycflights` data frame on the x-axis. It also defines a `geom` (short for geometric object), which describes the type of plot you will produce.

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?

Insert your answer here

By showing a bin size of 15 and 150 it's pretty clear that the size of the bins can hide some critical information about the distribution of data. When looking at the graph with a bin size of 15 we can see that the vast majority of flights depart without a delay ($x=0$) but there is a still a significant amount of flights in the bins adjacent to it. Contrasting with the graph where the bin size is 150, a substantial portion of flights are aggregated into that first bin, illustrating the role that bin size has on presenting or concealing features within the data set.

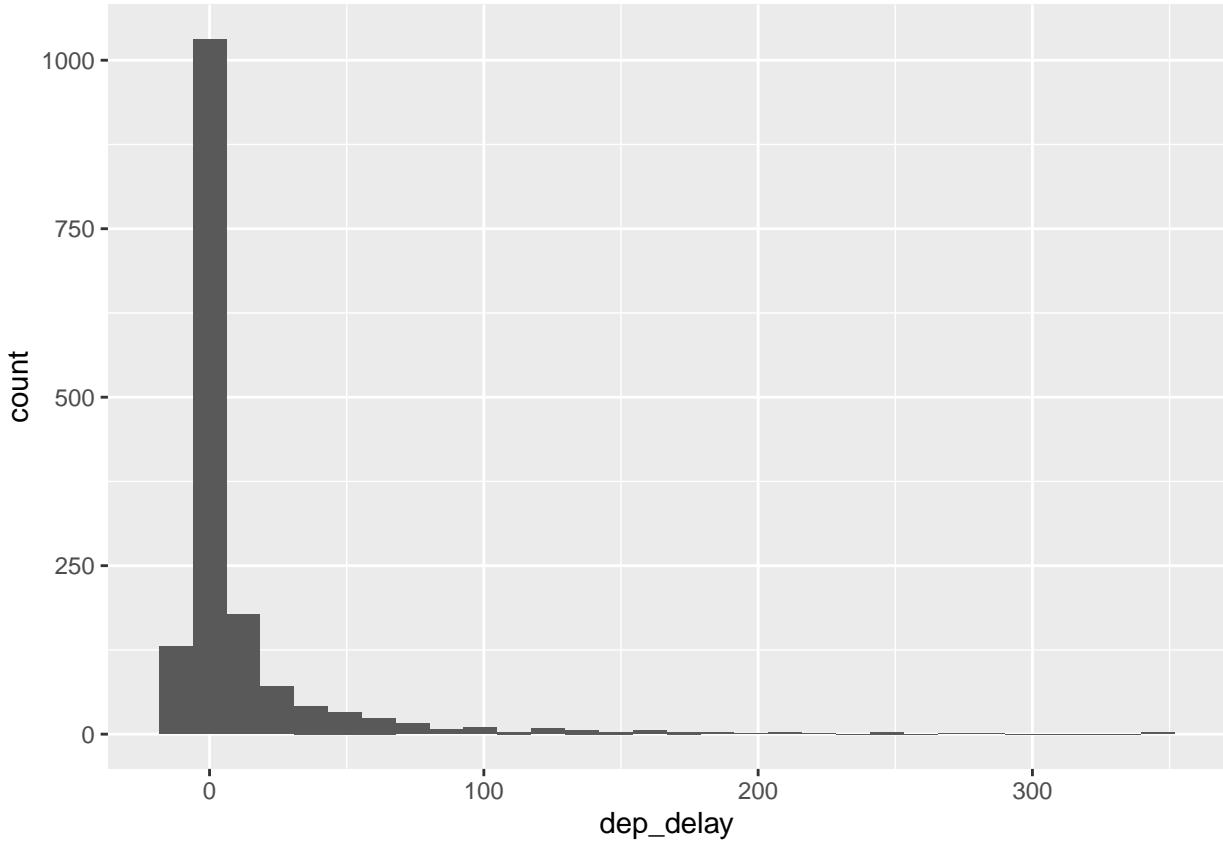
End of your answer

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?

Insert your answer here

```

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

sfo_feb_flights %>%
  summarise(
    n = n()
  )

```

```
## # A tibble: 1 x 1
##      n
##   <int>
## 1     68

nrow(sfo_feb_flights)
```

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

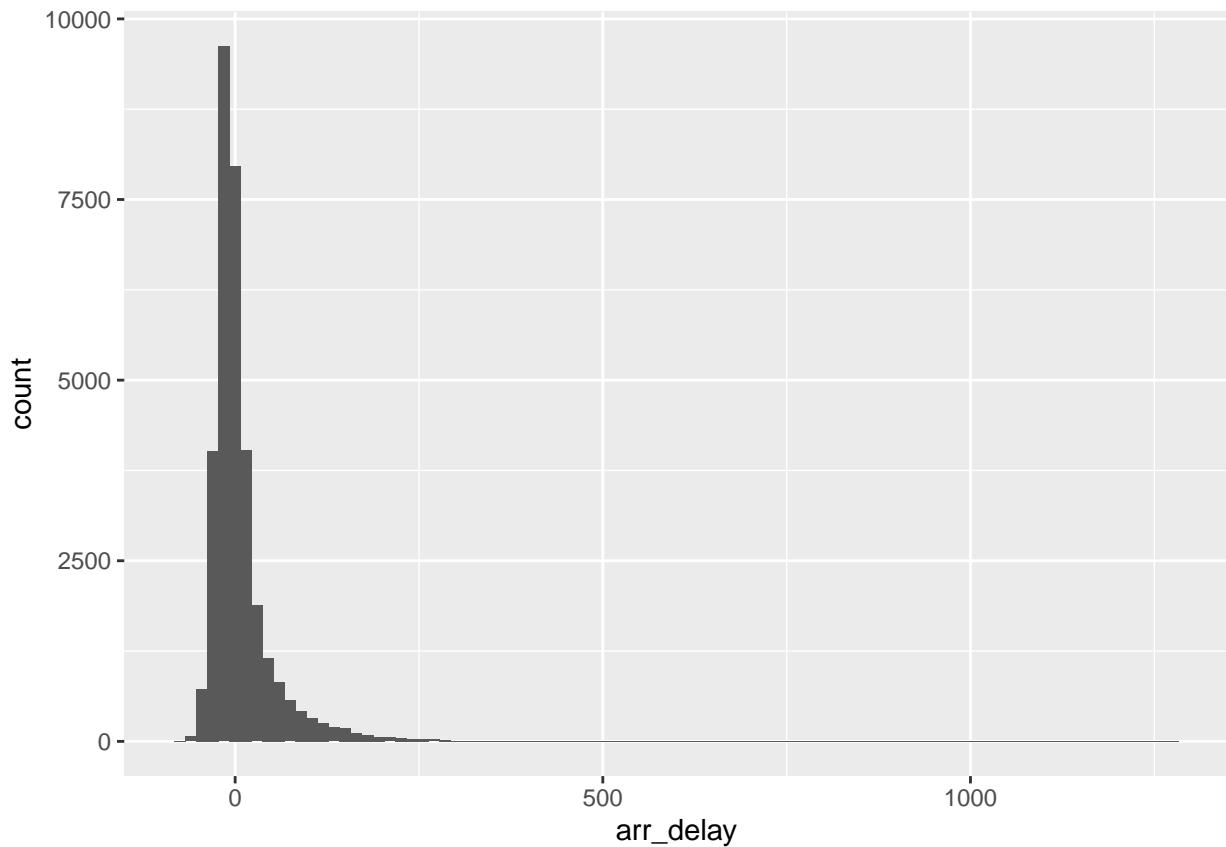
From the cell above, we can see that there are 68 flights in February 2013 that went from any NYC airport to SFO.

End of your answer

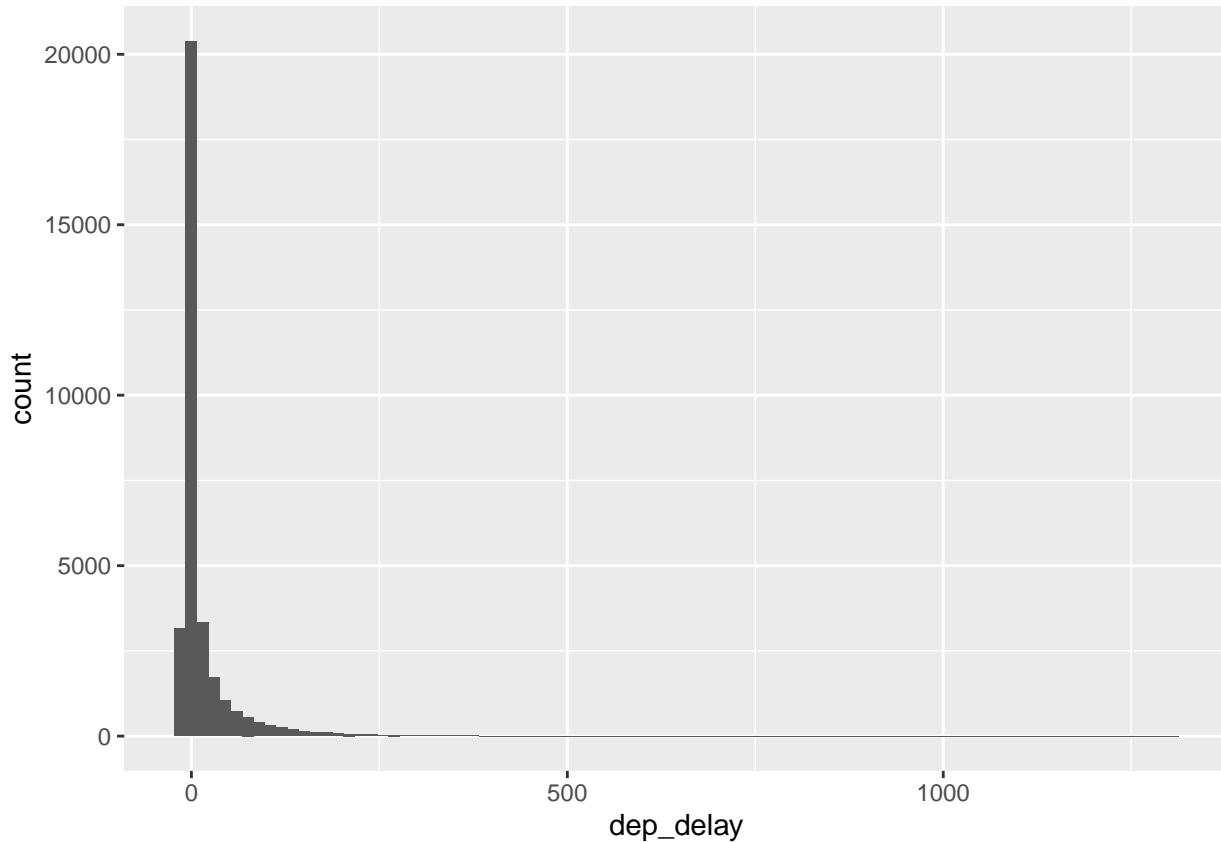
3. Describe the distribution of the **arrival** delays of these flights using a histogram and appropriate summary statistics. **Hint:** The summary statistics you use should depend on the shape of the distribution.

Insert your answer here

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



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



Looking at the tail of the distribution of arrival delays for all flights in the data set, I can see that there are longer arrival delays than departure delays in the bins further on the x-axis. Additionally, the modal bin (the bin with the most entries) for dep_delay is 0 while the modal bin for arr_delay is negative. So it looks like the spread is a bit greater with the arr_delay. Let's more rigorously define this by using summarise:

```
nycflights %>%
  summarise(
    mean_dd = mean(dep_delay),
    mean_ad = mean(arr_delay),
    median_dd = median(dep_delay),
    median_ad = median(arr_delay),
    sd_dd = sd(dep_delay),
    sd_ad = sd(arr_delay)
  )

## # A tibble: 1 x 6
##   mean_dd mean_ad median_dd median_ad sd_dd sd_ad
##     <dbl>    <dbl>      <dbl>     <dbl> <dbl> <dbl>
## 1    12.7     7.10      -2        -5  40.4  44.7
```

When looking at this data, we can see that the mean arr_delay and median arr_delay are lower than their respective dep_delays ($12.7 - 7.1 = 5.6$ and $(-2) - (-5) = 3$, respectively). When looking at the standard

deviation, we can see that the arr_delay is more dispersed from the mean than dep_delays (40.4 vs 44.7) which aligns with the visual finding we've had when comparing the histograms of arr_delay and dep_delay.

End of your answer

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.

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

Insert your answer here

```
sfo_feb_flights %>%
  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         5       17.5
## 2 B6        -10.5   12.2
## 3 DL        -15      22
## 4 UA        -10      22
## 5 VX        -22.5   21.2
```

According to the summary from the code above, there is a tie between both Delta and United Airlines. Both of these airlines have an interquartile range of 22 minutes. Interestingly enough, their median arrival delay is -15 and -10 minutes, respectively. So half of their flights are at least 10 minutes early.

End of your answer

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 `descending` order

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

```
## # A tibble: 12 x 4
##   month mean_dd median_dd max_dd
##   <int>   <dbl>     <dbl>   <dbl>
## 1     7    20.8      0     392
## 2     6    20.4      0     803
## 3    12    17.4      1     849
## 4     4    14.6     -2     427
## 5     3    13.5     -1     393
## 6     5    13.3     -1     351
## 7     8    12.6     -1     436
## 8     2    10.7     -2     319
## 9     1    10.2     -2    1301
## 10    9     6.87    -3     473
## 11   11     6.10    -2     413
## 12   10     5.88    -3     272
```

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?

Insert your answer here

Because the mean is calculated using every data point it is a comprehensive view of the entire group and it will show any large changes. Although, this is also its weakness as it is very sensitive to outliers and doesn't represent typical values in the dataset. For example, the maximum departure delay is 1301 minutes (almost a whole day) which greatly impacts the mean and would drastically skew the data if it was the only delay for that given month.

On the other hand, the median is robust to these outliers and provides the typical value within the data set. If the data spread of values is large, the median is a good measure to provide something meaningful. Although, the median will not provide any information about the magnitude of entries beyond the middle point which could mean that an atypical value can be incredibly far from the middle.

Choosing between these two to minimize the chances of a delay would come down to your risk tolerance and whether the duration of the delay matters more than the chances of being affected by a delay.

End of your answer

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 or 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:

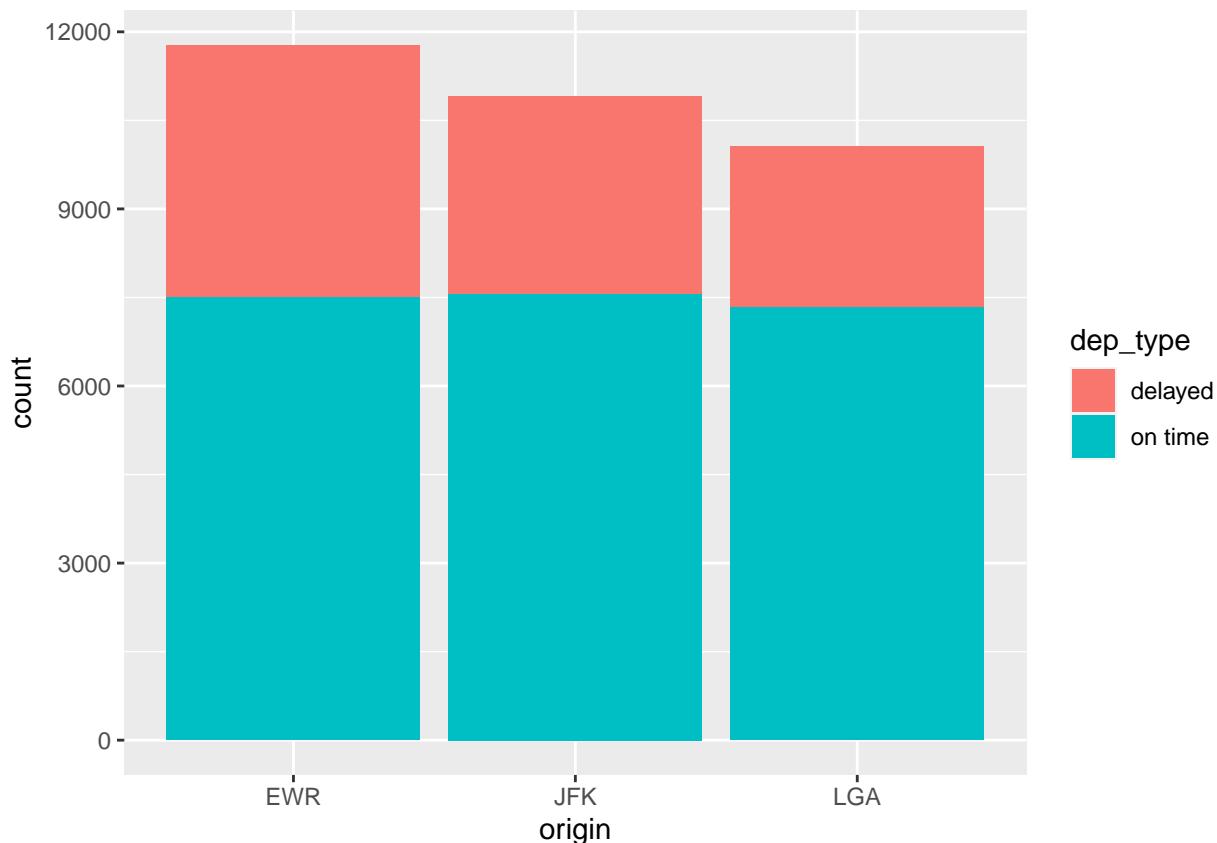
```
nycflights %>%
  group_by(origin) %>%
  summarise(ot_dep_rate = sum(dep_type == "on time") / n()) %>%
  arrange(desc(ot_dep_rate))
```

```
## # 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?

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()
```



Insert your answer here

If I had lounge access and wanted to lounge around for as long as possible, I would pick EWR as that airport has the worse on time rate.

Otherwise (and realistically) I'll pick LGA as LGA has an on time rate of 72.8% which is much greater than the others. This can also be visually seen using the segmented bar plot. By noticing that the count of "on time" departures is similar, we can simply choose the airport with the least amount of flights.

End of your answer

More Practice

- 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.

Insert your answer here

Using the mutate function:

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

```

nycflights %>%
  summarise(
    mean_mph      = mean(avg_speed),
    median_mph   = median(avg_speed)
  )

## # A tibble: 1 x 2
##   mean_mph median_mph
##       <dbl>     <dbl>
## 1     394.     404.

```

With that we can see that the mean avg_speed is 394 MPH and the median avg_speed is 404 MPH

End of your answer

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

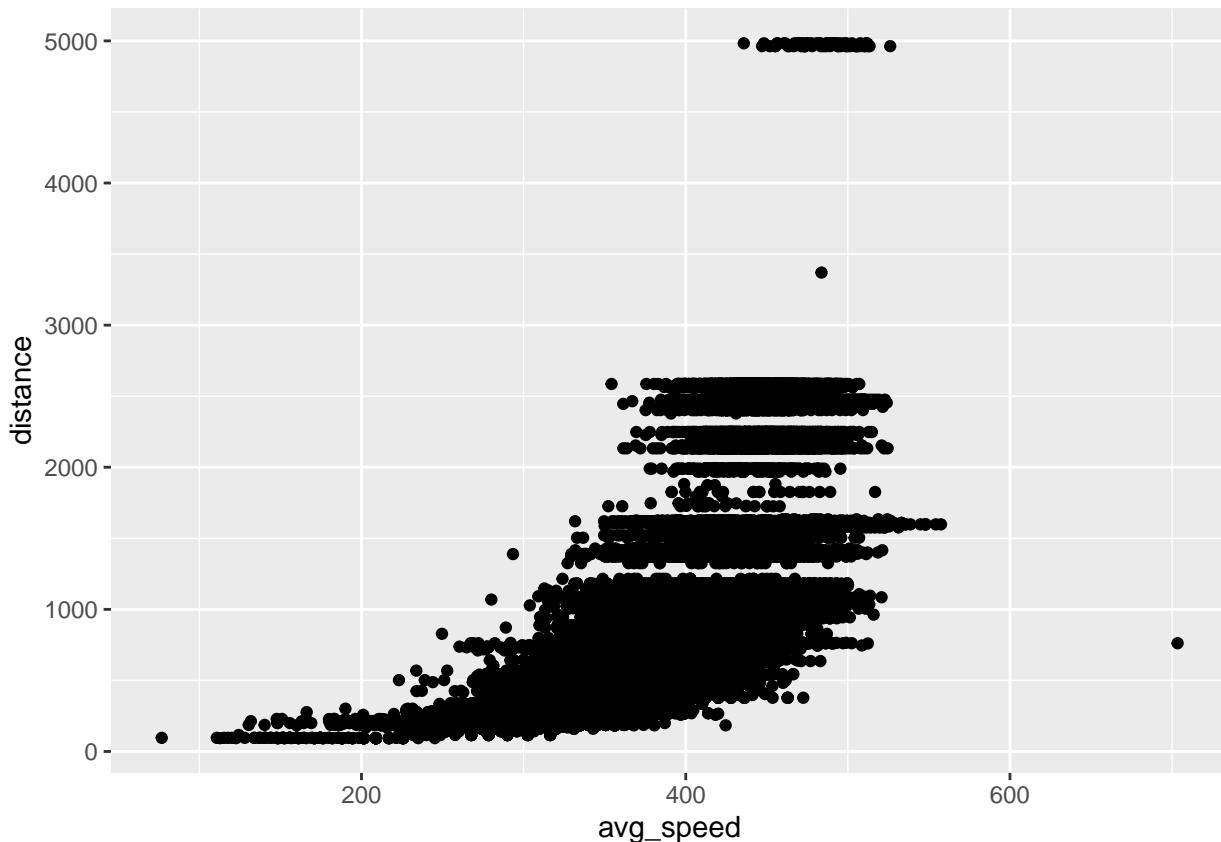
Insert your answer here

Creating the plot:

```

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

```



Looking at the graph, it seems that the trend is that as distance increases, the average speed increases until it they seem to converge to a maximum value. That being said, the bands within each distance seems to be pretty wide although the trend seems to be upwards.

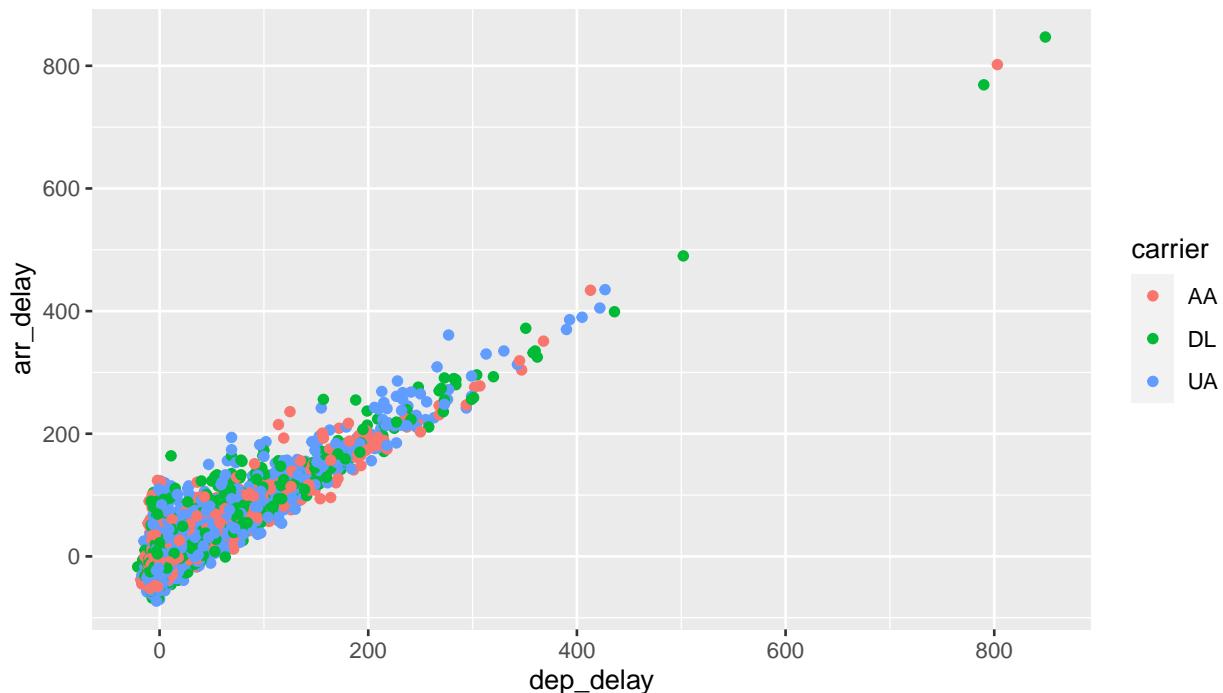
```
nycflights %>%
  filter(avg_speed > 600)
```

```
## # A tibble: 1 x 18
##   year month   day dep_time dep_delay arr_time arr_delay carrier tailnum flight
##   <int> <int> <int>    <dbl>     <int>     <dbl> <chr>   <chr>   <int>
## 1  2013      5     25     1709        9     1923     -14  DL     N666DN  1499
## # i 8 more variables: origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## # hour <dbl>, minute <dbl>, dep_type <chr>, avg_speed <dbl>
```

There is an outlier on a flight to Atlanta that was really going fast. The US Standard Atmosphere tables show that the speed of sound at this elevation (35000) is 660 MPH, suggesting that this flight had bad data or there was an incredibly strong tailwind.

End of your answer

- 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.



Insert your answer here

```
big_3_flights <- nycflights %>%
  filter(
    carrier == "AA" |
```

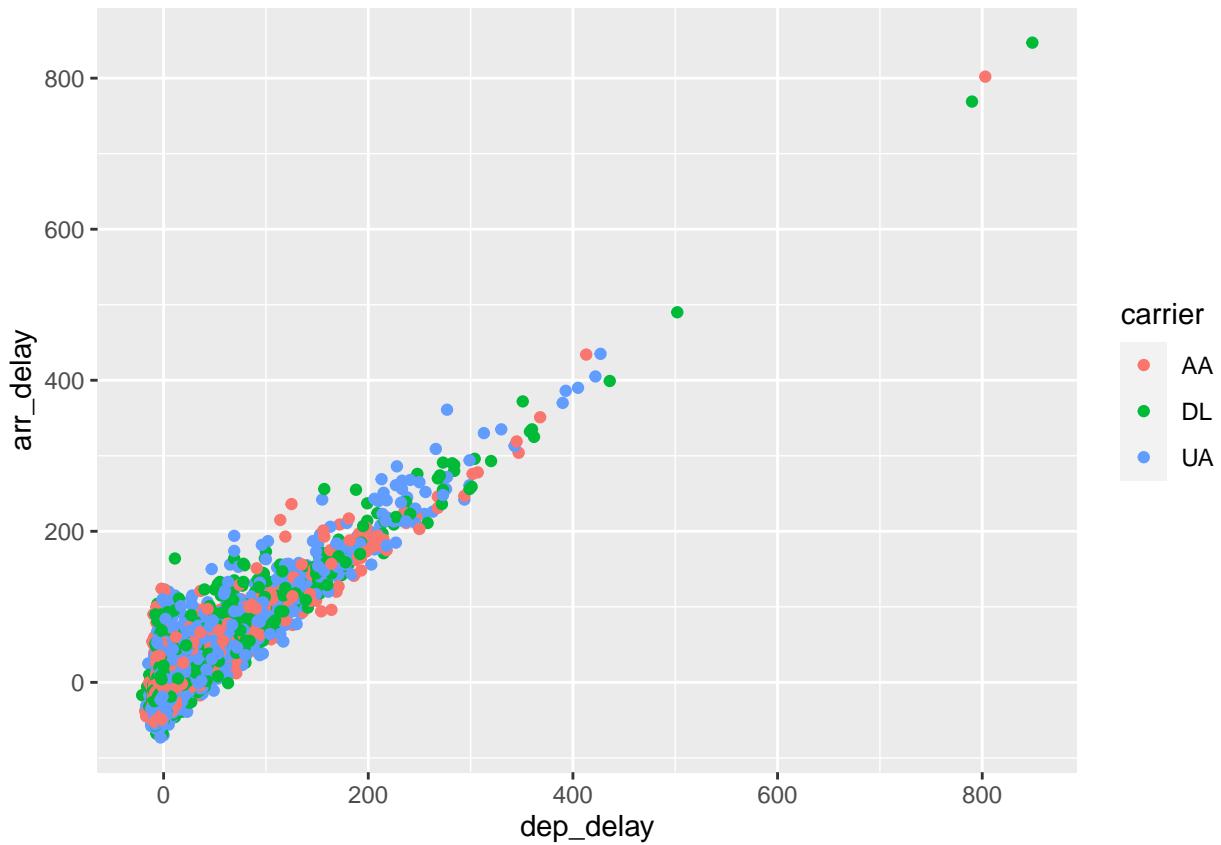
```

    carrier == "UA" |
    carrier == "DL"
)

big_3_plot = ggplot(
  data = big_3_flights,
  aes(x=dep_delay, y=arr_delay, color=carrier)
) +
  geom_point()

big_3_plot

```



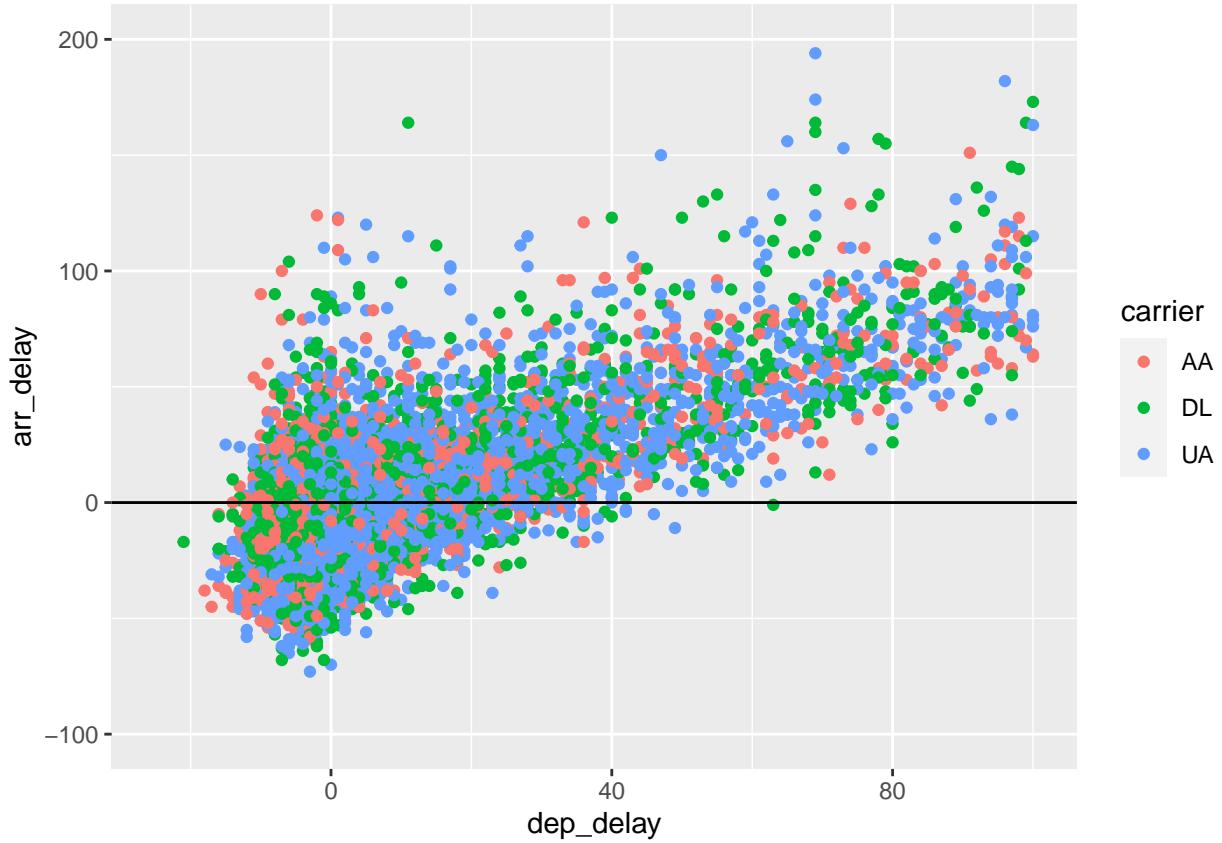
This graph doesn't really help much as it has a lot of values outside of the range we're interested in. I'll be setting an xlim and ylim so that we can zoom into the range we care about and adding a line at y=0 so it's easier to visualize.

```

zoomed_big_3_plot = big_3_plot +
  xlim(-25,100) +
  ylim(-100, 200) +
  geom_hline(yintercept=0)

zoomed_big_3_plot

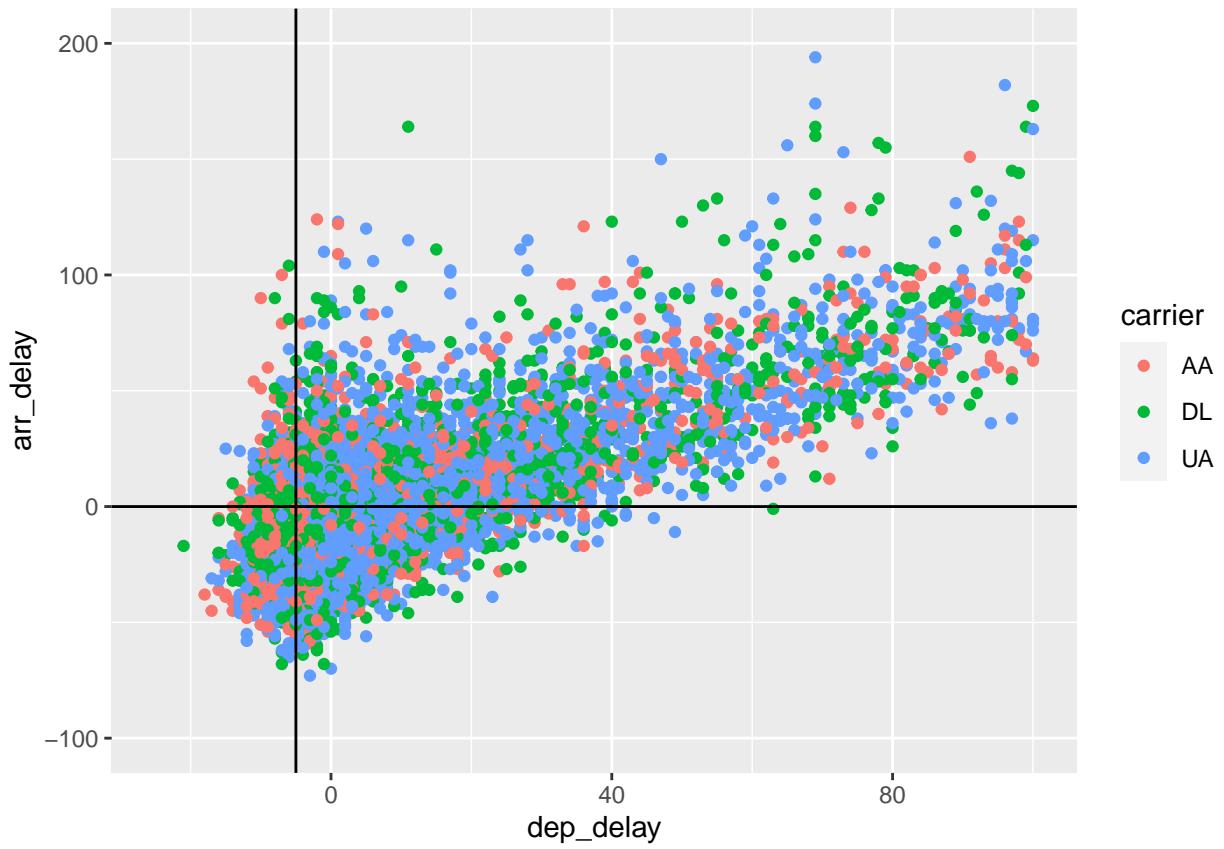
```



Zooming in, we can see that there's still a huge spread and it seems like a lot of flights which even departed early are delayed on arrival. A scatter plot with this many data points doesn't really help to understand how many observations are concentrated in that area. But I would say that if your flight departed 5 minutes early then you have a great chance to arrive on time.

Let's put a vertical line on that $x=-5$

```
zoomed_big_3_plot +
  geom_vline(xintercept=-5)
```



Looking at that bottom left quadrant, we can definitely see that the majority of flights that depart 5 minutes early arrive on time or early, but this would need to be more rigorously analyzed to be meaningful.

End of your answer