Lab #2-Introduction to Data

Anthony Conrardy

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 sub-setting.

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 work space 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 **code book** (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:

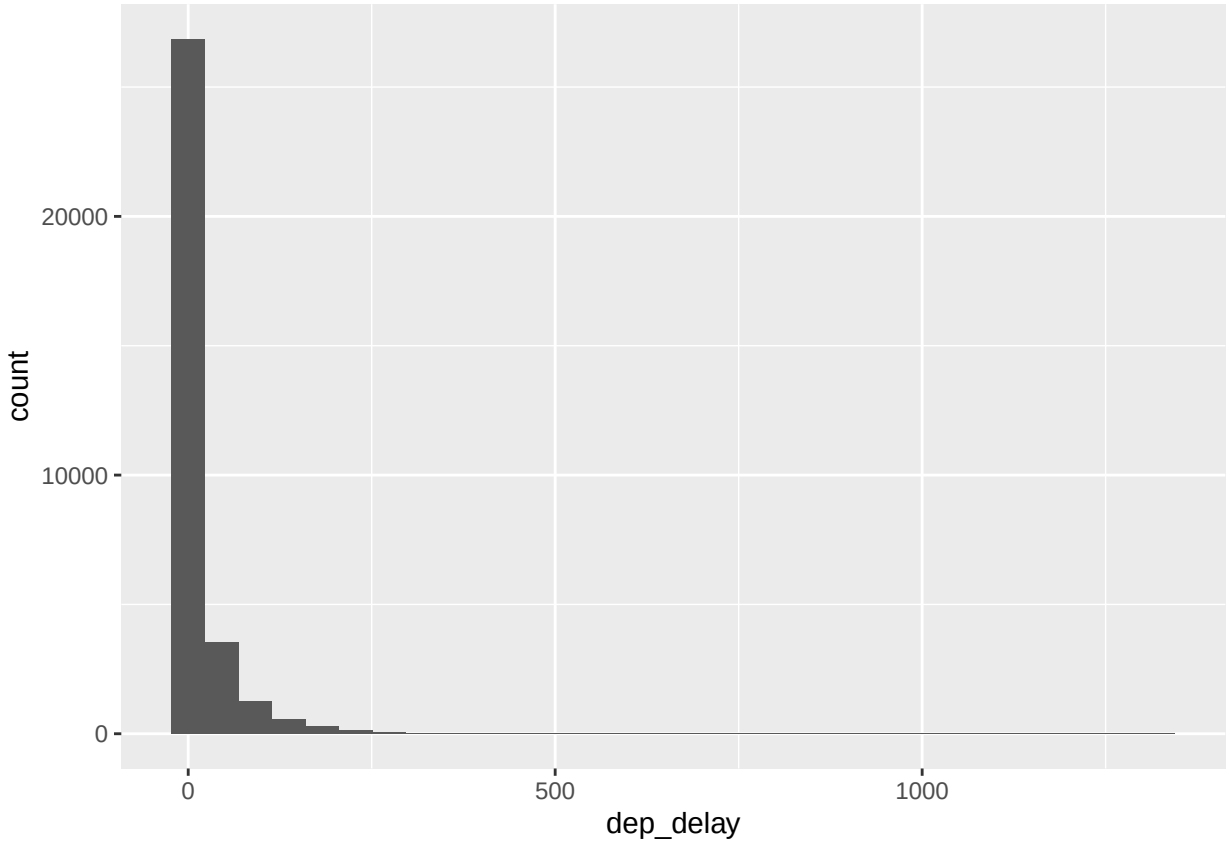
- How delayed were flights that were headed to Los Angeles?
- How do departure delays vary by month?
- Which of the three major NYC airports has the best on time percentage for departing flights?

Analysis

Departure delays

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

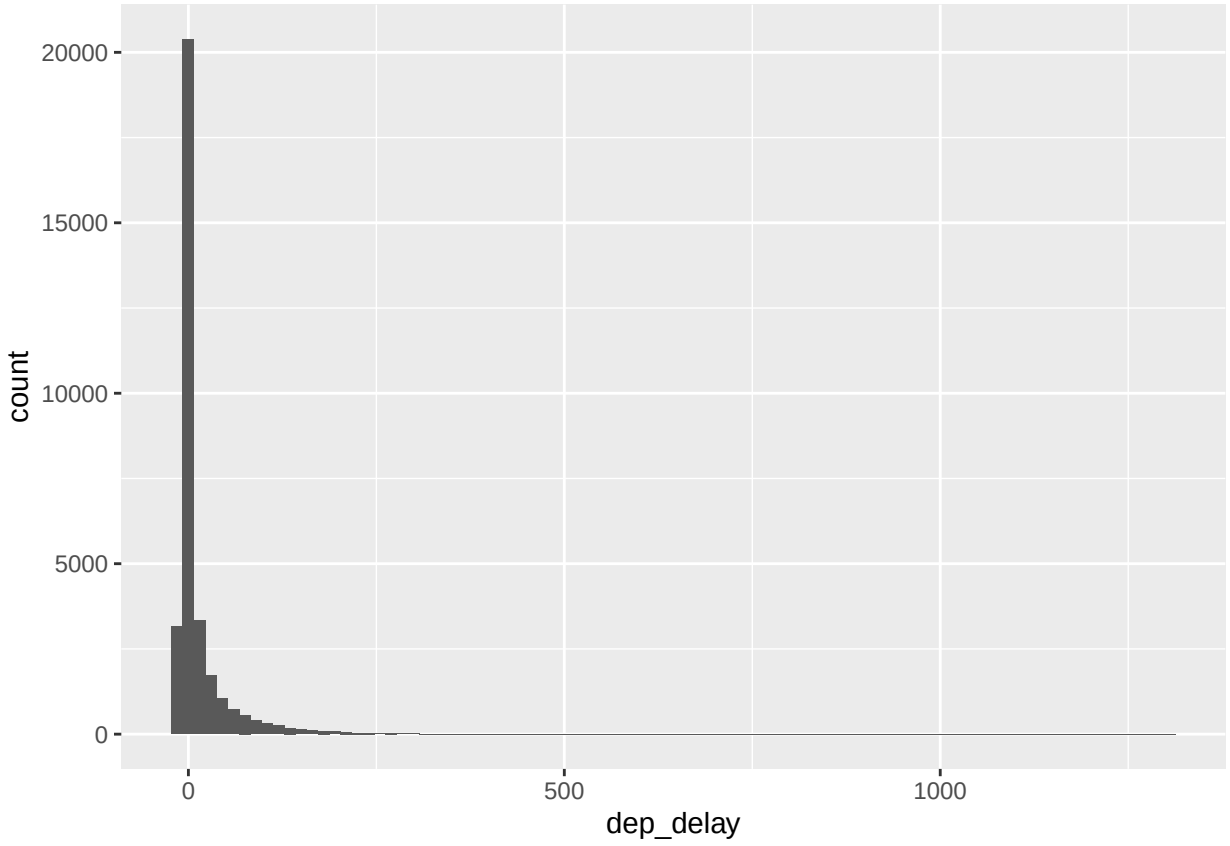
```
ggplot(data = nycflights, aes(x = dep_delay)) +
  geom_histogram()
```



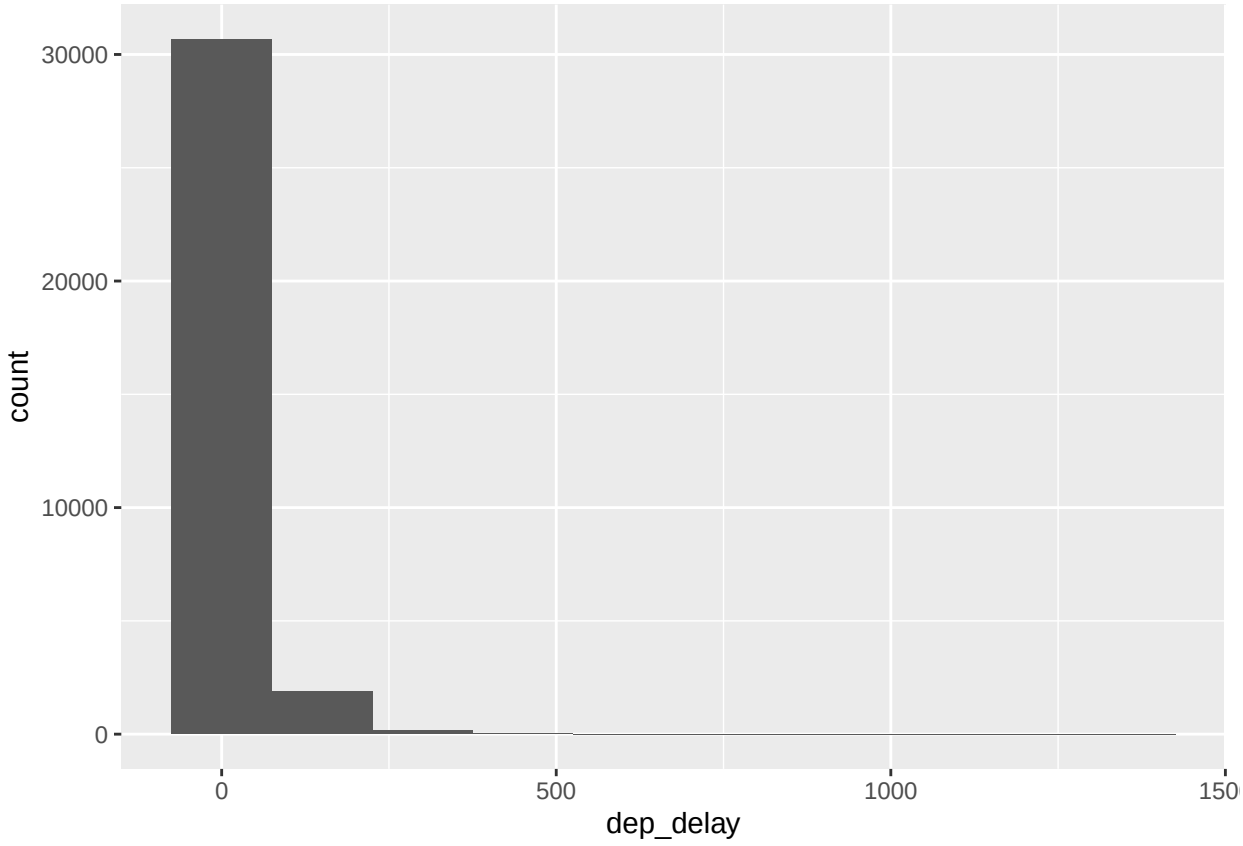
This function says to plot the `dep_delay` variable from the `nycflights` data frame on the x-axis. It also defines a `geom` (short for geometric object), which describes the type of plot you will produce.

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 bin width 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)
```

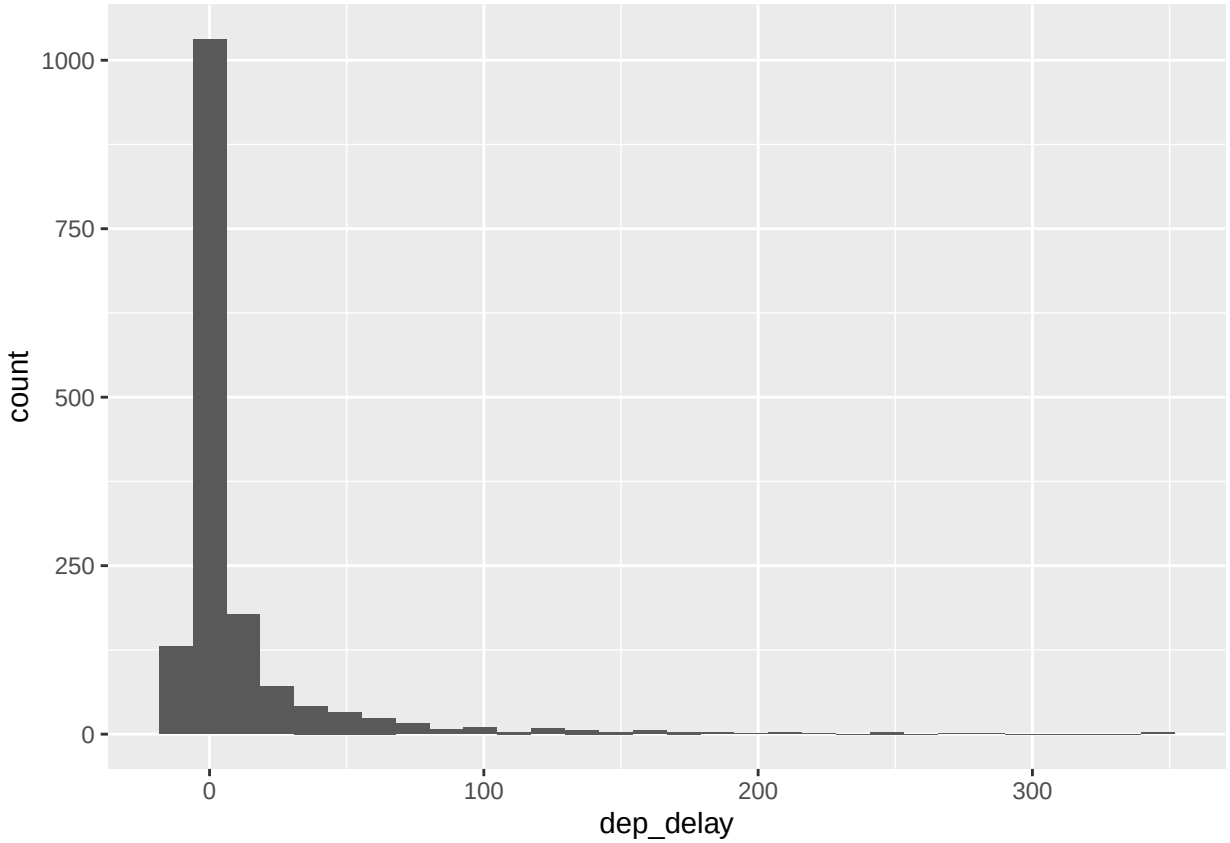


1. Look carefully at these three histograms. How do they compare? Are features revealed in one that are obscured in another?

In general, all three plots indicate a decreasing frequency of delays with respect to time. The question is how granular one wishes to view the data. The first binwidth is not specified, but ggplot should default to 30, The second plot is set to 15, which provides a narrower view of the data by decreasing the width so there are more bins in which the observations may fall, and increasing the number of bars observed on the plot (maximum count on the plot also decreases). Increasing the size of the binwidth to 150, effectively reduces the number of bars to three (3) and reduces the detail of the data. So, the binwidth=150 makes it appear that a significant number of cases in the first bin account for over 30,000 cases of delays, while the binwidth=15 indicates that almost 20,000+ of the cases occur in a much narrower range of minutes and that almost ~3,000 cases have either an on-time or early departure The binwidth=30 (default) does provide a "middle ground" view, but it does not provide the more enlightening view of binwidth=15. The minimum of the data set is -21 minutes, while the maximum is 1301 minutes, and the curve seems to be similar to an exponential decay curve.

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:

```r
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:

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

```
## Rows: 68
## Columns: 16
## $ year      <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, ~
## $ month     <int> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, ~
## $ day       <int> 18, 3, 15, 18, 24, 25, 7, 15, 13, 8, 11, 13, 25, 20, 12, 27,~
## $ dep_time  <int> 1527, 613, 955, 1928, 1340, 1415, 1032, 1805, 1056, 656, 191~
## $ dep_delay <dbl> 57, 14, -5, 15, 2, -10, 1, 20, -4, -4, 40, -2, -1, -6, -7, 2~
## $ arr_time  <int> 1903, 1008, 1313, 2239, 1644, 1737, 1352, 2122, 1412, 1039, ~
## $ arr_delay <dbl> 48, 38, -28, -6, -21, -13, -10, 2, -13, -6, 2, -5, -30, -22,~
## $ carrier   <chr> "DL", "UA", "DL", "UA", "UA", "UA", "B6", "AA", "UA", "DL", ~
## $ tailnum   <chr> "N711ZX", "N502UA", "N717TW", "N24212", "N76269", "N532UA", ~
## $ flight    <int> 1322, 691, 1765, 1214, 1111, 394, 641, 177, 642, 1865, 272, ~
## $ origin    <chr> "JFK", "JFK", "JFK", "EWR", "EWR", "JFK", "JFK", "JFK", "JFK~
## $ dest      <chr> "SFO", "SFO", "SFO", "SFO", "SFO", "SFO", "SFO", "SFO", "SFO~
## $ air_time  <dbl> 358, 367, 338, 353, 341, 355, 359, 338, 347, 361, 332, 351, ~
## $ distance  <dbl> 2586, 2586, 2586, 2565, 2565, 2586, 2586, 2586, 2586, 2586, ~
## $ hour      <dbl> 15, 6, 9, 19, 13, 14, 10, 18, 10, 6, 19, 8, 10, 18, 7, 17, 1~
## $ minute    <dbl> 27, 13, 55, 28, 40, 15, 32, 5, 56, 56, 10, 33, 48, 49, 23, 2~
```

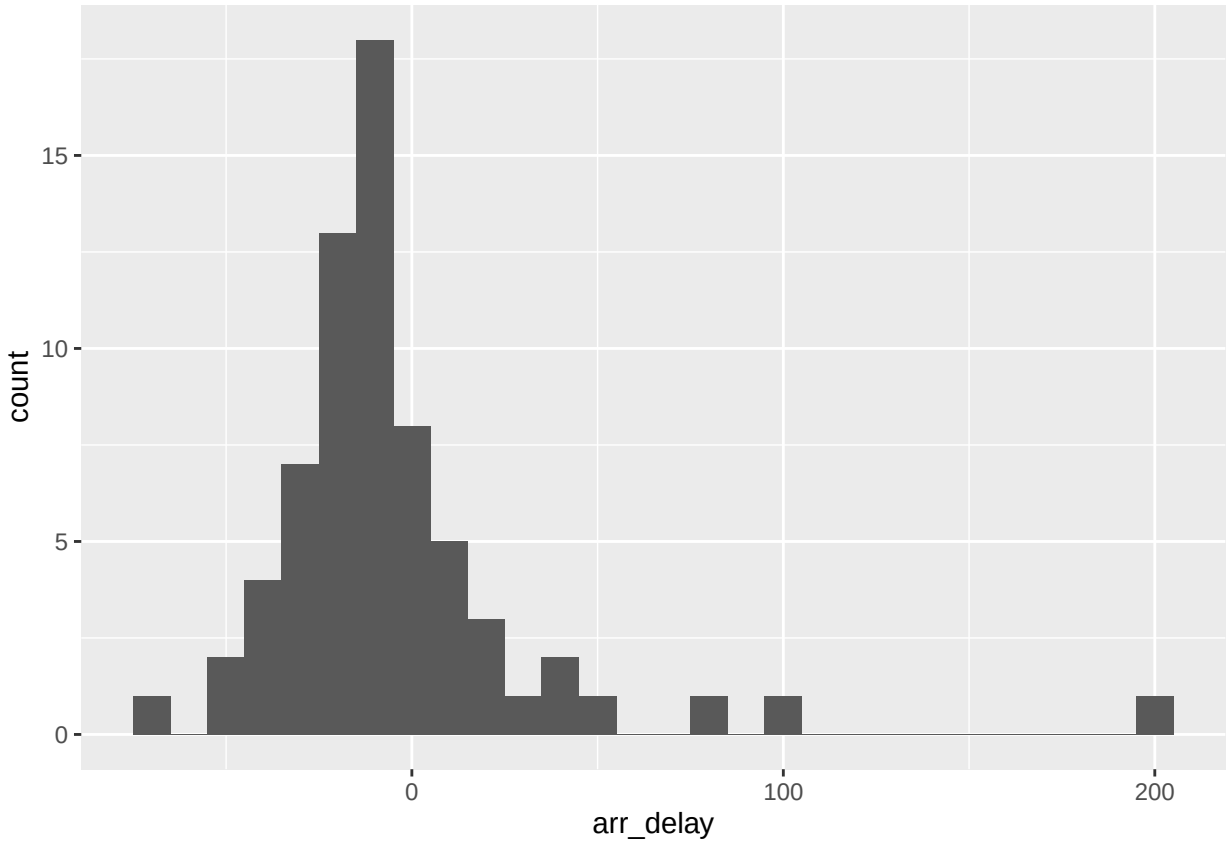
Note that you can separate the conditions using commas if you want flights that are both headed to SFO **and** in February. If you are interested in either flights headed to SFO **or** in February, you can use the | instead of the comma.

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?

There are a total of 68 flights that meet the criteria of arriving in San Francisco (SFO) and in the month of 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.

```
ggplot(data = sfo_feb_flights, aes(x = arr_delay)) +
  geom_histogram(binwidth = 10)
```




```
sfo_feb_flights |> summarise(medianad = median(arr_delay),
                              iqrad = IQR(arr_delay),
                              minad = min(arr_delay),
                              maxad = max(arr_delay),
                              n = n())
```

```
## # A tibble: 1 x 5
##   medianad iqrad minad maxad     n
##      <dbl> <dbl> <dbl> <dbl> <int>
## 1      -11  23.2   -66   196    68
```

I selected a binwidth = 10 to get a relatively good view of the data distribution. Since there are only 68 observations, we must try and keep the bins to a functional number to reflect a proper distribution. The distribution appears to be fairly "normal", but one can see that there are indeed substantial outliers that would "pull" a mean result to the right. Therefore, the best option would be to report the median value. We can also observe that we have negative arrival delays, which could indicate that the flight arrived early and that the airline was able to make up time during in-flight travel.

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_delay`s of flights in in the `sfo_feb_flights` data frame, grouped by carrier. Which carrier has the most variable arrival delays?

```
## # A tibble: 5 x 4
##   carrier median_ad iqr_ad n_flights
##   <chr>       <dbl>  <dbl>     <int>
## 1 AA            5     17.5        10
## 2 B6          -10.5   12.2         6
## 3 DL          -15     22          19
## 4 UA          -10     22          21
## 5 VX          -22.5   21.2        12
```

For the purposes of this exercise, the IQR indicates how spread out the middle 50% of the observations occur. So the larger the IQR spread, the larger the variability. In this case that occurs with Virgin America (VX) airlines, which is interesting since it indicates that 75% of the flights have a negative arrival delay or an early arrival.

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?

It depends on how skewed the data might be and the carrier involved. If all carriers are "equal", meaning that there are no other factors influencing the carrier other than that which affects all carriers equally, then one could look at the entire population of flights and look at the distribution of delays to see if it assumes the shape of a normal curve and see if it might be skewed left or right due to outliers. If not such skewing exists, then a lowest mean approach might be appropriate. If skewing is present, then a lowest median approach might be better suited. However, in either case, this does not take into account time of day variation when more flights are taking off and landing, which could increase the probability of greater delays. Finally, if all carriers are not "equal", meaning there are internal issues particular to the carrier that might cause delays (i.e. shortage of flight crews), then an individualized approach for each carrier might be necessary and then following the above approach of mean vs. median.

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:

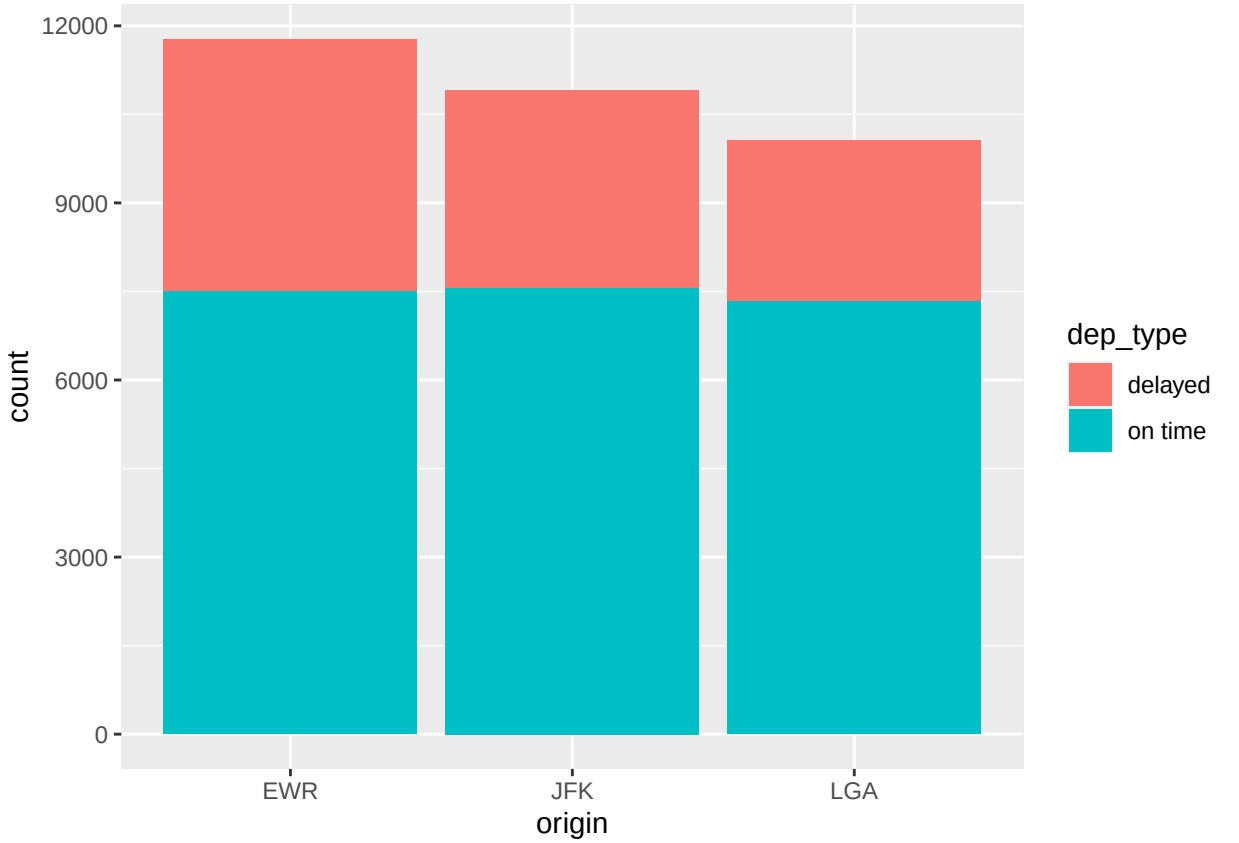
```
nycflights %>%
  group_by(origin) %>%
  summarise(ot_dep_rate = sum(dep_type == "on time") / n()) %>%
  arrange(desc(ot_dep_rate))
```

```
## # 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?

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()
```



If you were strictly going on time departure percentage, then the largest percentage would be LGA at 72.8% (0.728) of flights. This can easily be identified in the results of the initial calculation. However, that result does not consider other factors such as month of travel, time of day, or carrier involved. While the bar chart is interesting, it would be helpful to add in the percentage values into the chart. At first glance, it looks like all on time departure levels are somewhat equal, when they are smaller in percentage comparison to all flights from the different airports.

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.

The following code chunk calculates the air speed from the air_time and distance variables, and then converts it to miles per hour and places it into a column named air_speed. It rounds it to the nearest whole number and moves the three columns to the front of the data frame.

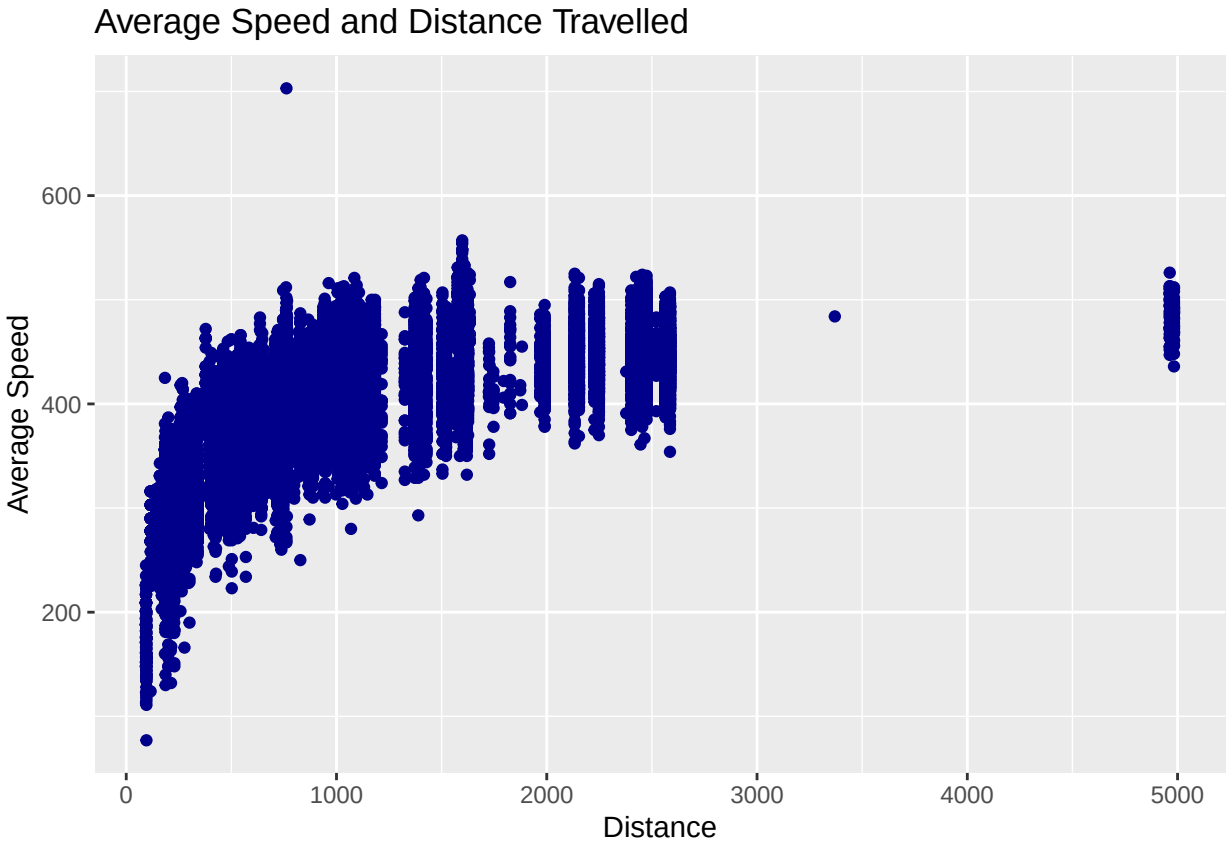
```
nycflights <- nycflights |>
  mutate(avg_speed = round(distance / (air_time / 60), 0))
nycflights <-  nycflights |> relocate(air_time, distance, avg_speed)
nycflights
```

```
## # A tibble: 32,735 x 18
##    air_time distance avg_speed  year month   day dep_time dep_delay arr_time
##       <dbl>    <dbl>     <dbl> <int> <int> <int>    <int>     <dbl>    <int>
##  1      313     2475       474  2013     6    30      940        15     1216
##  2      216     1598       444  2013     5     7     1657        -3     2104
##  3      376     2475       395  2013    12     8      859        -1     1238
##  4      135     1005       447  2013     5    14     1841        -4     2122
##  5       50      296       355  2013     7    21     1102        -3     1230
##  6      138      733       319  2013     1     1     1817        -3     2008
##  7      240     1411       353  2013    12     9     1259        14     1617
##  8       48      228       285  2013     8    13     1920        85     2032
##  9      148     1096       444  2013     9    26      725       -10     1027
## 10      110      820       447  2013     4    30     1323        62     1549
## # i 32,725 more rows
## # i 9 more variables: arr_delay <dbl>, carrier <chr>, tailnum <chr>,
## #   flight <int>, origin <chr>, dest <chr>, hour <dbl>, minute <dbl>,
## #   dep_type <chr>
```

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

The following code chunk plots the distance vs. Average Speed. The calculation of speed from the distance and air time lends itself to a number of results that make it appear to be different speeds for the same distance. The code also re-labels the axis and puts a title to the plot.

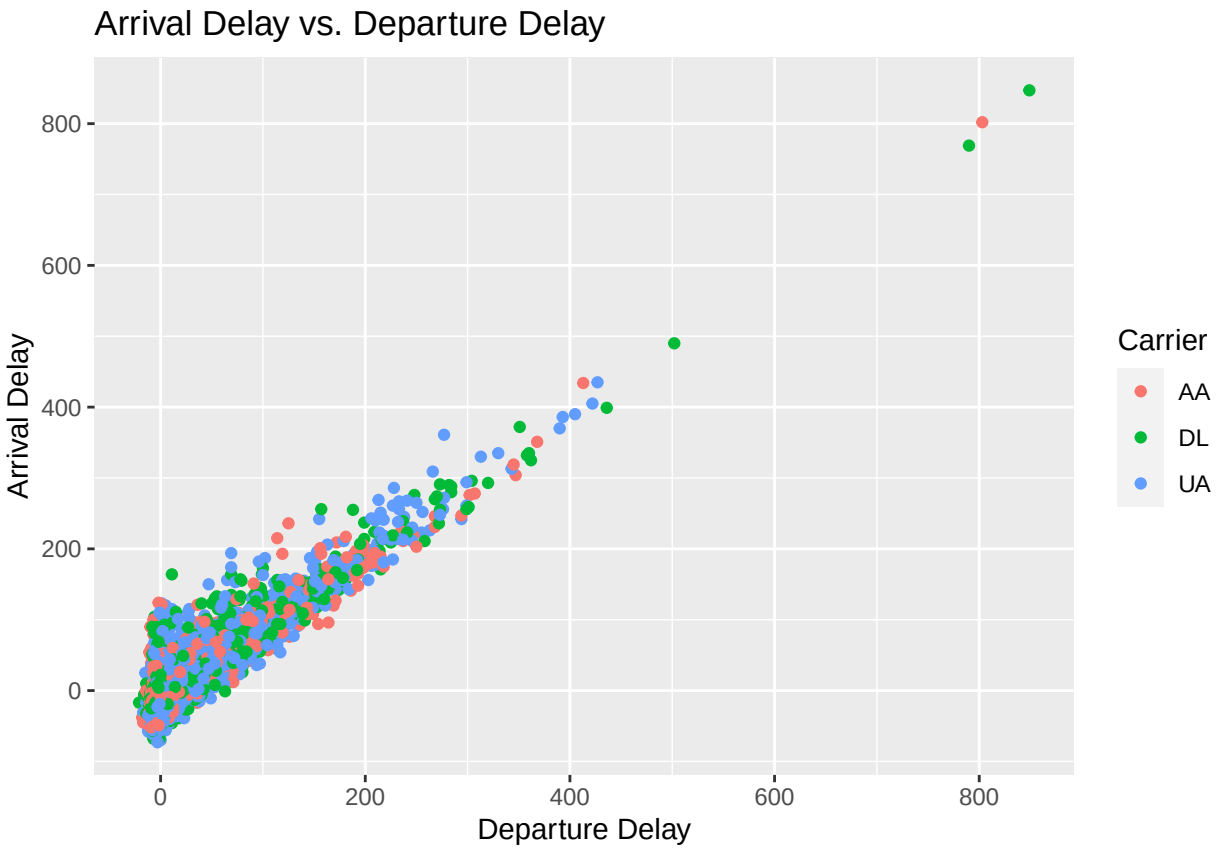
```
ggplot(
  data = nycflights,
  mapping = aes(x = distance, y = avg_speed)
) +
  geom_point(color = "dark blue") +
  labs(title = "Average Speed and Distance Travelled",
       x = "Distance",
       y = "Average Speed")
```

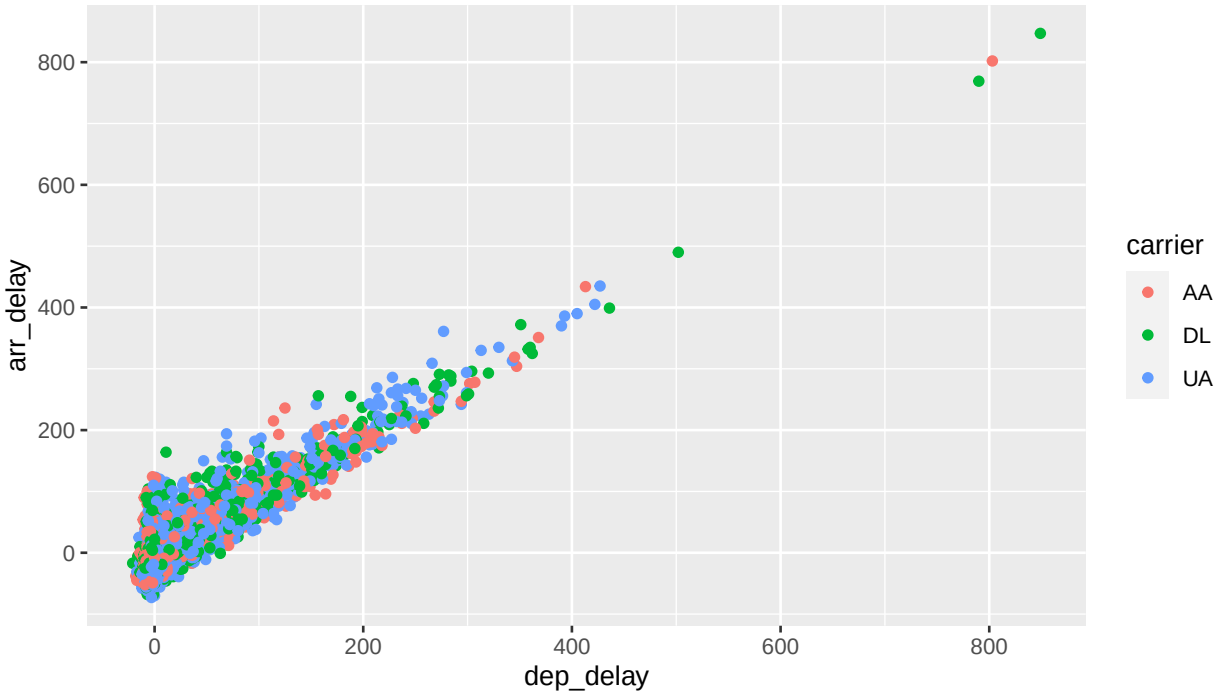


9. Replicate the following plot. **Hint:** The data frame plotted only contains flights from American Airlines, Delta Airlines, and United Airlines, and the points are `colored` by `carrier`. Once you replicate the plot, determine (roughly) what the cutoff point is for departure delays where you can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

Thank you for providing the code and the pass on this question. However, I felt somewhat guilty about not trying to achieve the same type of plot a little differently and then adding something to it to try and expand on the exercise to learn something.

```
nyc_aar_dep_del <- nycflights |> select(dep_delay, arr_delay, carrier)
nyc_aar_dep_del_sel <-  nyc_aar_dep_del |> filter((carrier == "AA" | carrier == "DL" | carrier == "UA"))
ggplot(nyc_aar_dep_del_sel, aes(x = dep_delay, y = arr_delay, color = carrier)) +
  geom_point() +
 labs(title = "Arrival Delay vs. Departure Delay",
       x = "Departure Delay",
       y = "Arrival Delay",
      color = "Carrier")
```


Arrival Delay vs. Departure Delay
Arrival Delay
Departure Delay
0
200
400
600
800
Carrier
AA
DL
UA


arr_delay
dep_delay
0
200
400
600
800
carrier
AA
DL
UA