Introduction to data

Alex Ptacek

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)
library(scales)
```

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:

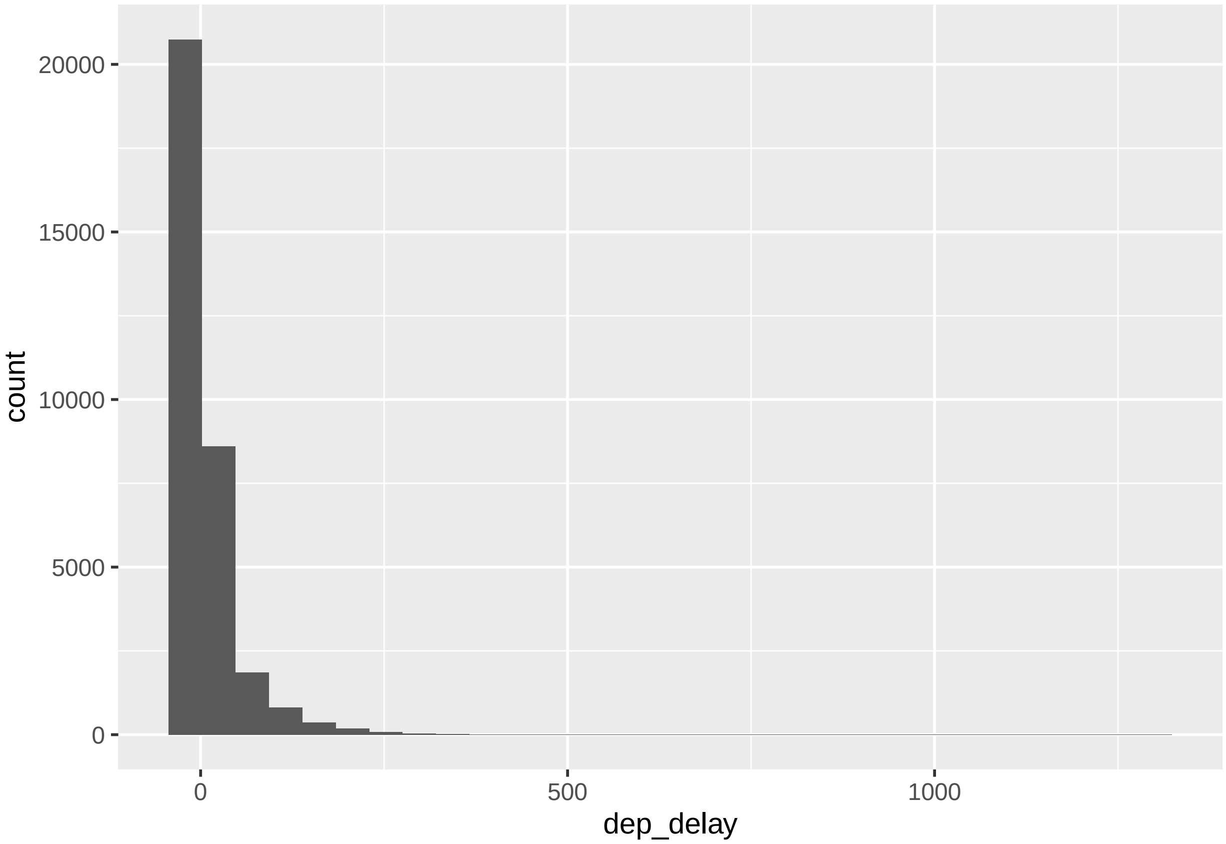
- How delayed were flights that were headed to Los Angeles?
- How do departure delays vary by month?
- Which of the three major NYC airports has the best on time percentage for departing flights?

Analysis

Departure delays

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

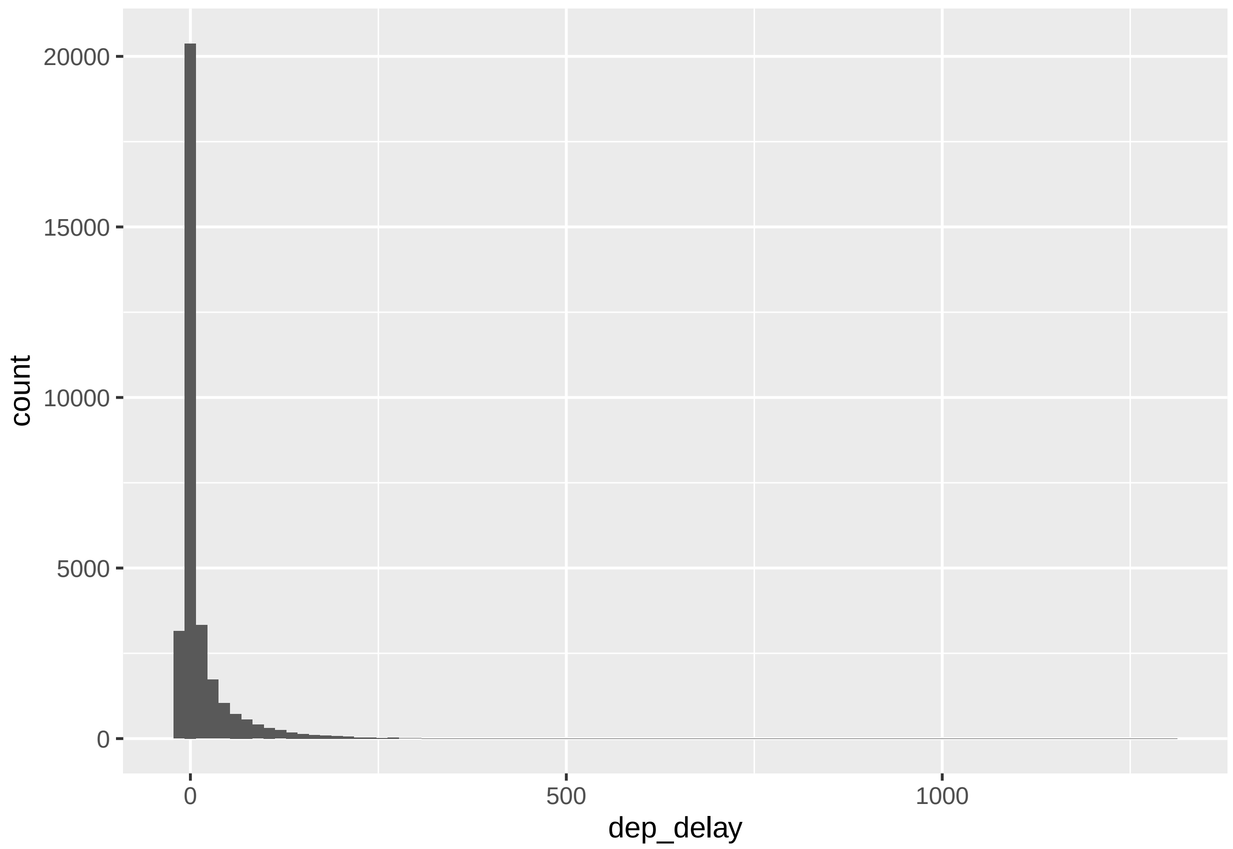
```
ggplot(data = nycflights, aes(x = dep_delay)) +
  geom_histogram()
```



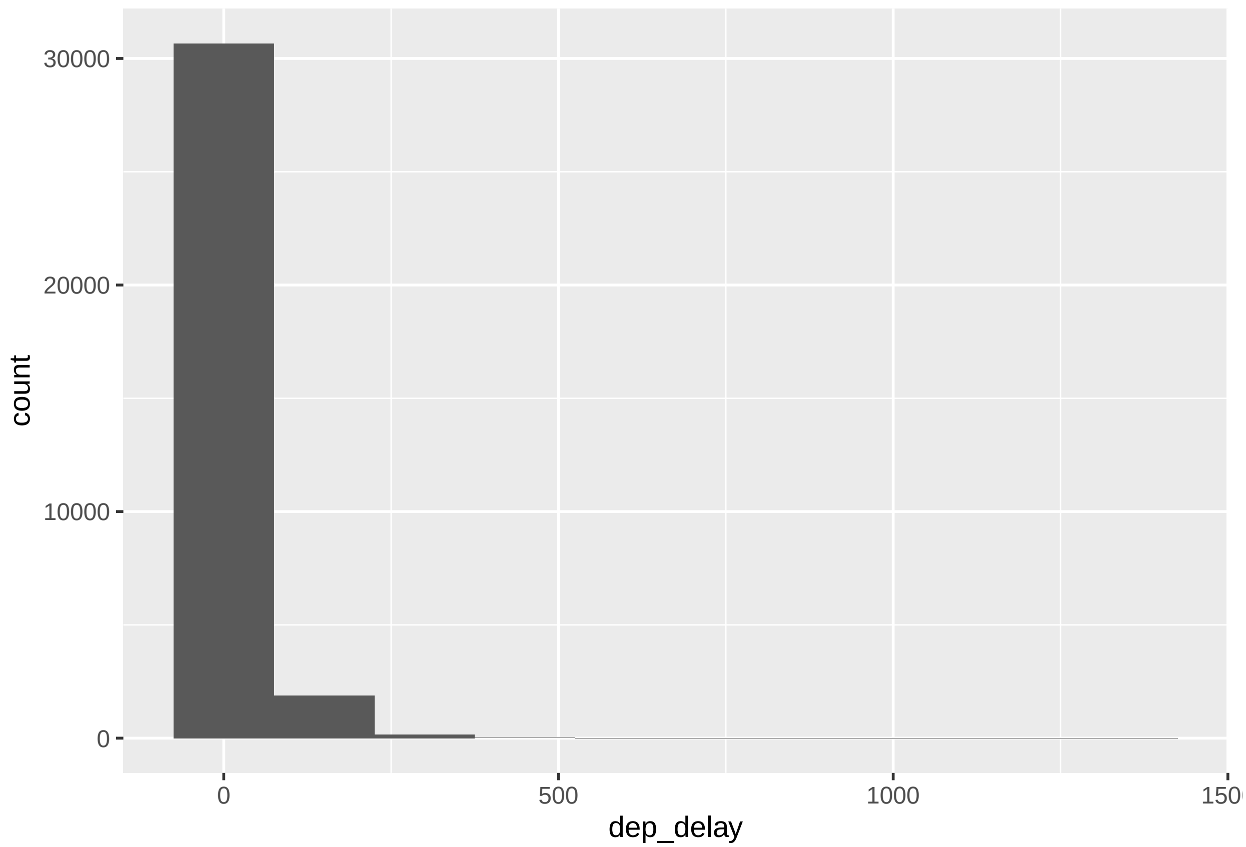
This function says to plot the `dep_delay` variable from the `nycflights` data frame on the x-axis. It also defines a `geom` (short for geometric object), which describes the type of plot you will produce.

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)
```



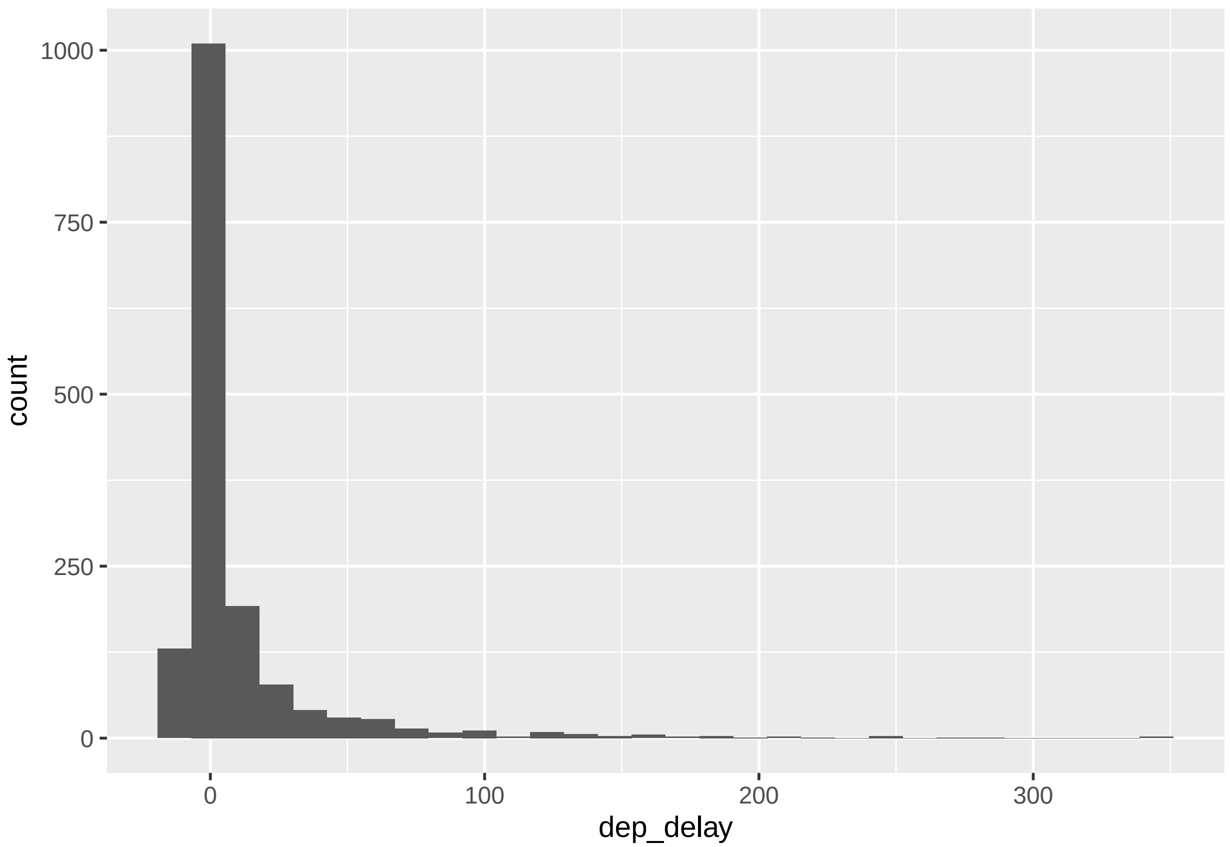
1. Look carefully at these three histograms. How do they compare? Are features revealed in one that are obscured in another?

Insert your answer here

Answer: The smaller the bin, the smaller the bars in the histogram. Both histograms show us there is a right skew to the `dep_delay` distribution. The histogram with binwidth size set to 15 also reveals that there is a range of 15 minutes delay time with a drastically high distribution (>20,000 count), even compared to the neighboring high distribution bins.

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

Answer: There are 68 flights in `nycflights` that headed to SFO in February.

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

sfo_feb_flights |> count()
```

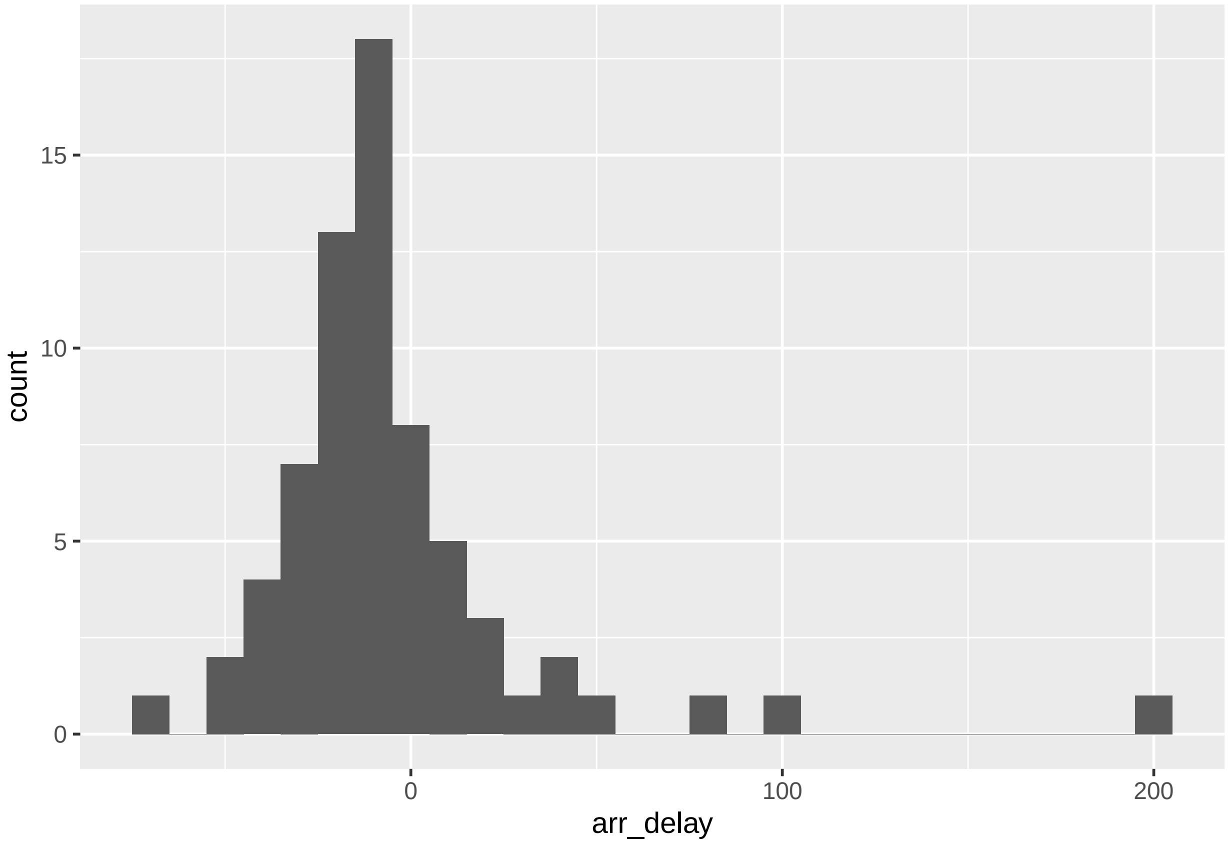
```
## # A tibble: 1 x 1
##       n
##   <int>
## 1    68
```

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

Answer: `sfo-feb-flights` looks like a right skewed distribution. However, only 4% of flights had >50 minutes of `arr_delay`. Based on the histogram, the mean seems to fall right in the area of highest distribution, so the >50 values may not be having a huge effect. However, based on the comparison between median, mean, and std_dev, the >50 values did possibly have a large effect, because there's a 21% difference between median and mean, an std_dev is quite high. Further investigation may be needed.

```
#Create a histogram to examine the distribution of arrival delays
sfo_feb_flights |>
  ggplot(aes(x = arr_delay)) +
  geom_histogram(binwidth = 10)
```



```
#Let's see how much of our data is skewed to the right
sfo_feb_flights |>
  filter(arr_delay > 50) |>
  summarise("arr_delay > 50" = n())
```

```
## # A tibble: 1 x 1
##   `arr_delay > 50`
##              <int>
## 1                3
```

```
#Calculate the percentage of total flights made up of these flights
percent(3/68)
```

```
## [1] "4%"
```

```
#Let's see if the differences in mean, std, and median reflect this finding
sfo_feb_flights |>
  summarise(med_arr_delay = median(arr_delay),
            mean_arr_delay = mean(arr_delay),
            std_dev = sd(arr_delay))
```

```
## # A tibble: 1 x 3
##   med_arr_delay mean_arr_delay std_dev
##           <dbl>          <dbl>   <dbl>
## 1           -11           -4.5    36.3
```

```
#Calculate the difference between median and mean
percent((-11 - -4.5)/(-11 + -4.5)/2)
```

```
## [1] "21%"
```

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

Answer: Delta and United Airlines are tied for the highest variable arrival delays because they have the highest IQR at 22 minutes.

```
sfo_feb_flights |>
  group_by(carrier) |>
  summarise(median_arr_delay = median(arr_delay),
            iqr_arr_delay = IQR(arr_delay),
            n_flights = n())
```

```
## # A tibble: 5 x 4
##   carrier median_arr_delay iqr_arr_delay 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
```

Departure delays by month

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

Answer: I expected November and December to have the highest delays. I was not super surprised that June and July turned out to be at the top, but did not expect November to be the second lowest.

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),
            median_dd = median(dep_delay)) %>%
  arrange(desc(mean_dd))
```

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

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

Answer: Choosing the lowest mean departure delay means that you're either equally likely to get a very negative delay as a very high delay or an average delay, because those values would average out to a low mean, or you're >50% likely to get an average delay and <50% likely to get a high delay (or low delay), because the more delays around the median means less outliers. Essentially, with just seeing the mean and not the deviation, we don't know how normal the distrubtion is. If the distribution is normal, choosing the lowest mean will give us a 50% chance to get an average or lower delay and 50% chance to get an average or higher delay. If the distribution is skewed, we are >50% likely to get a delay around average. Choosing the lowest median gives you a good chance of getting a low departure delay, because the delays are distributed around the median, meaning the most of the of the values fall around that amount. However, with this method, we don't have any idea what outliers are at play.

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"))

#personal test
nycflights_table <- nycflights |>
  mutate(dep_type = ifelse(dep_delay < 5, "on_time", "delayed")) |>
  group_by(origin, dep_type) |>
  summarise(n = n()) |>
  pivot_wider(names_from = dep_type, values_from = n) |>
  mutate(on_time_rate = percent(on_time/(on_time + delayed), accuracy = .1)) |>
  arrange(desc(on_time_rate))
```

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:

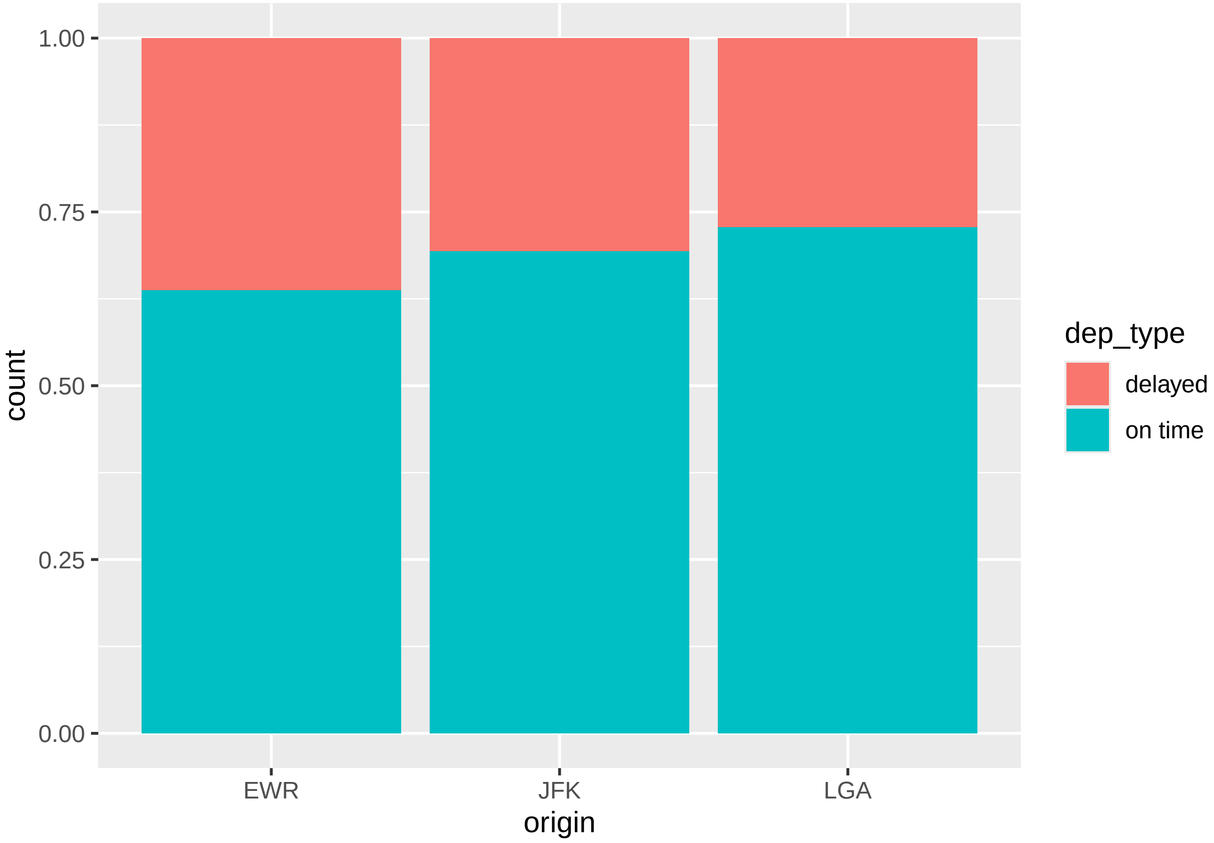
```
nycflights %>%
  group_by(origin) %>%
  summarise(ot_dep_rate = sum(dep_type == "on time") / n()) %>%
  arrange(desc(ot_dep_rate))
```

```
## # 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(position = "fill")
```



Insert your answer here

Answer: LGA

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.

Insert your answer here

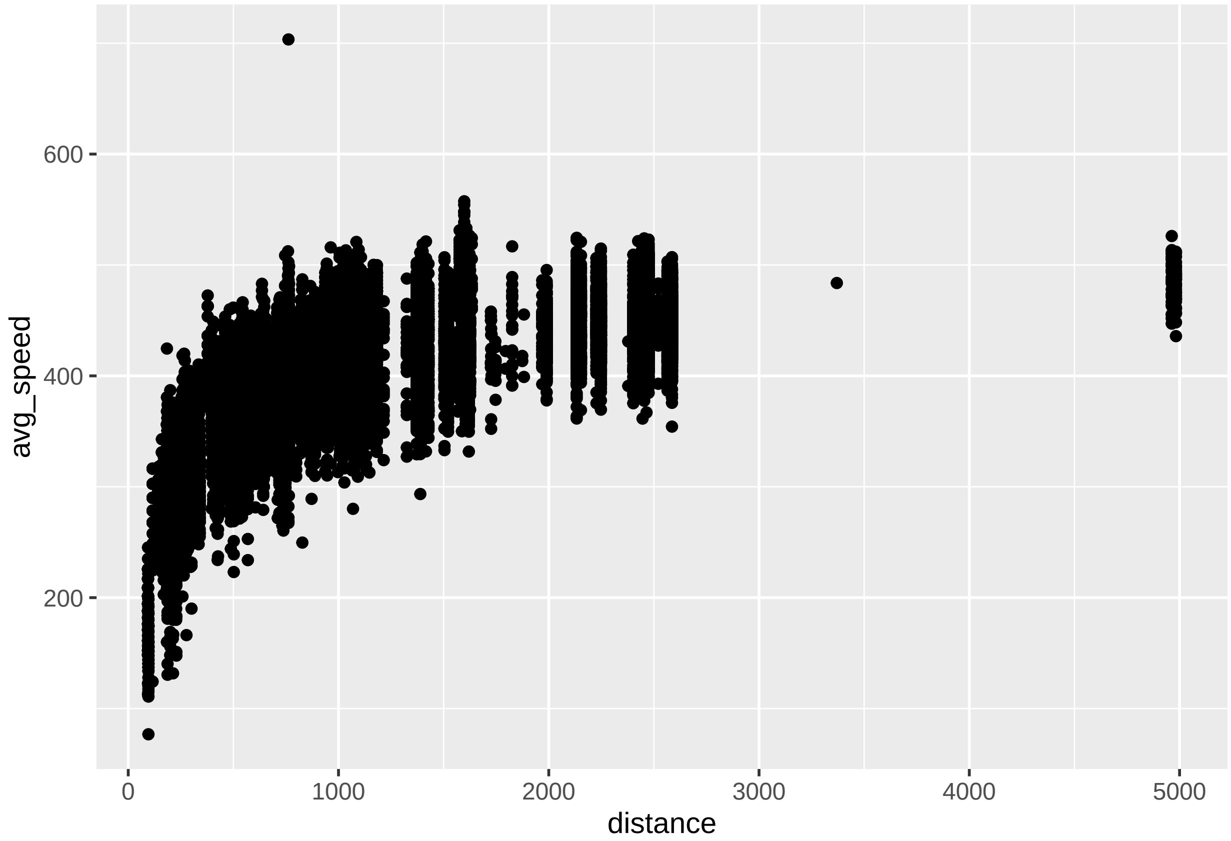
```
nycflights <- nycflights |>
  mutate(avg_speed = distance/air_time*60)
```

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

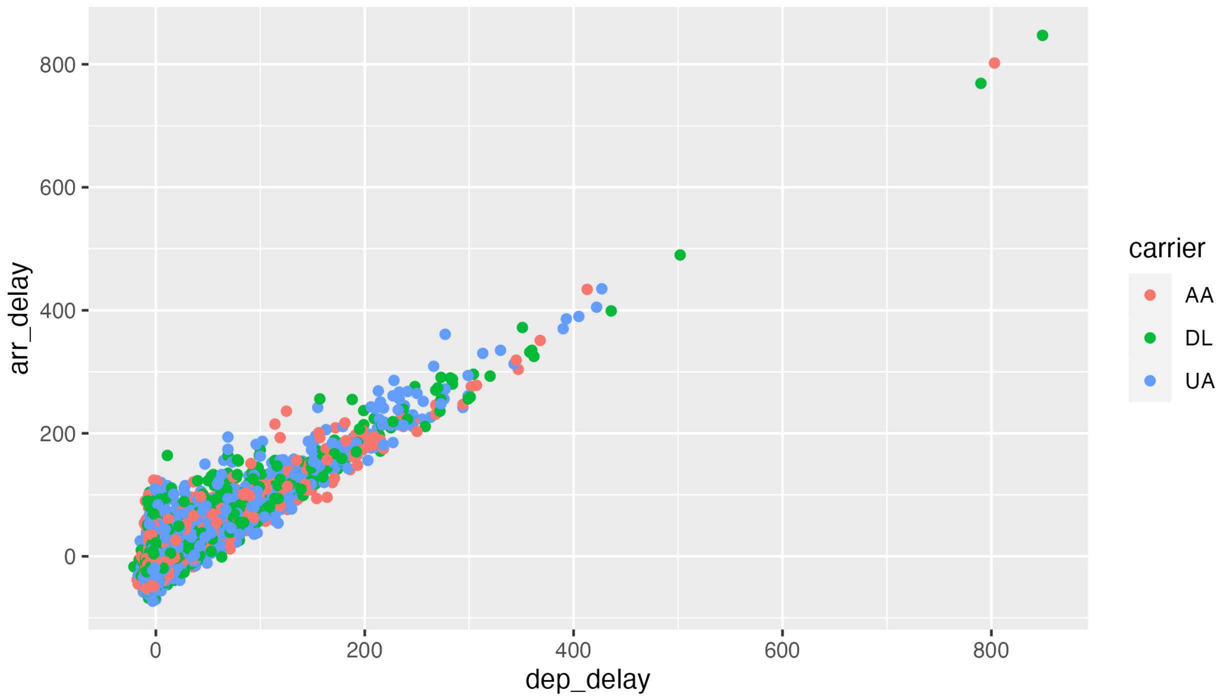
Insert your answer here

Answer: Average speed increases as a square root function of distance. At lower distances, small increases in distance greatly increase average speed, but at higher distances, change in distance has a small effect on average speed.

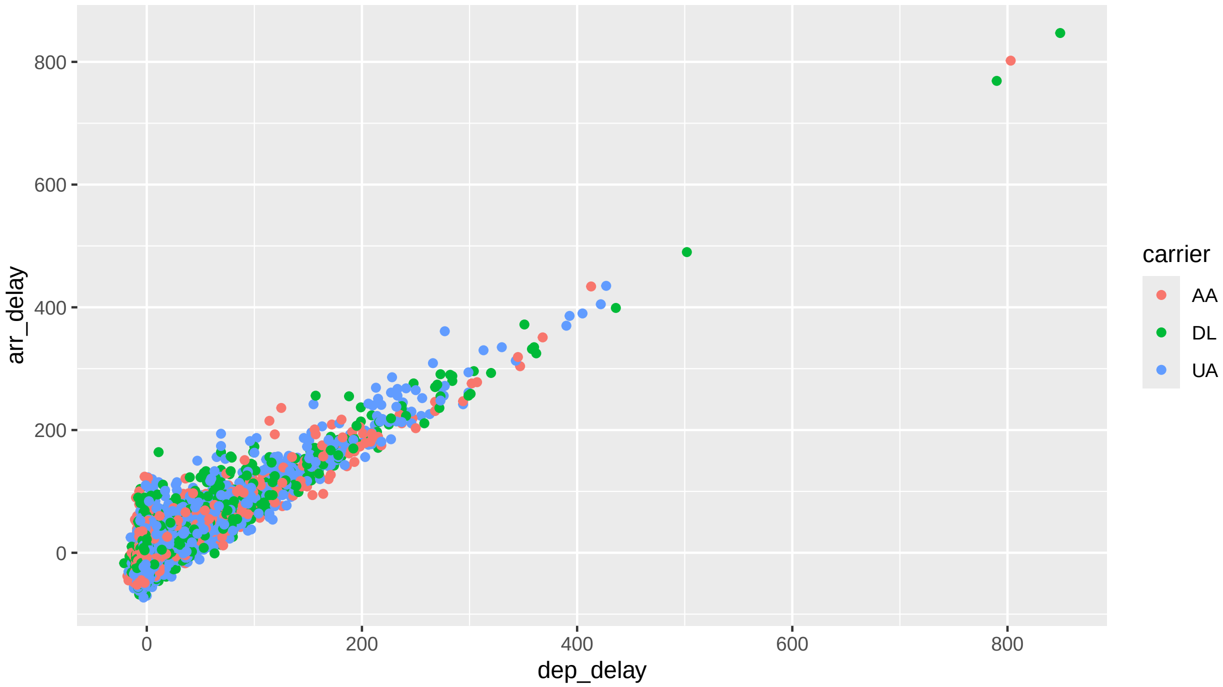
```
nycflights |>
  ggplot(aes(x = distance, y = avg_speed)) +
  geom_point()
```



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.



```r
nycflights |>
  filter(carrier == "UA" | carrier == "DL" | carrier == "AA") |>
  ggplot(aes(x = dep_delay, y = arr_delay, color = carrier)) +
  geom_point()
```



Insert your answer here

Answer: The cutoff point for departure delays where you can still expect to get to your destination on time is about 60 minutes. We can find this answer more accurately with the following transformations.

```
nycflights |>
  filter(carrier == "UA" | carrier == "DL" | carrier == "AA") |>
  filter(arr_delay <= 0) |>
  arrange(desc(dep_delay))
```

```
## # A tibble: 8,708 x 18
##     year month   day dep_time dep_delay arr_time arr_delay carrier tailnum
##    <int> <int> <int>    <int>     <dbl>    <int>     <dbl> <chr>   <chr>
##  1  2013     7    11     2018        63     2210        -1 DL      N3751B
##  2  2013     4    27     2001        49     2155       -11 UA      N34455
##  3  2013     7    15      845        46     1055        -5 UA      N486UA
##  4  2013     8    28     1505        42     1705        -4 UA      N78501
##  5  2013     7     1     1538        42     1721        -3 UA      N36447
##  6  2013     3     8     1709        40     1937         0 UA      N77296
##  7  2013    11    26     2055        40     2348        -6 DL      N188DN
##  8  2013     5     9     1639        39     1842        -4 DL      N3736C
##  9  2013     4    27     1843        38     2111       -15 UA      N24702
## 10  2013     7    14     1247        38     1411        -7 UA      N439UA
## # i 8,698 more rows
## # i 9 more variables: flight <int>, origin <chr>, dest <chr>, air_time <dbl>,
## #   distance <dbl>, hour <dbl>, minute <dbl>, dep_type <chr>, avg_speed <dbl>
```