Introduction to data

Koohyar Pooladvand

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:

```r
?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.

```r
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:

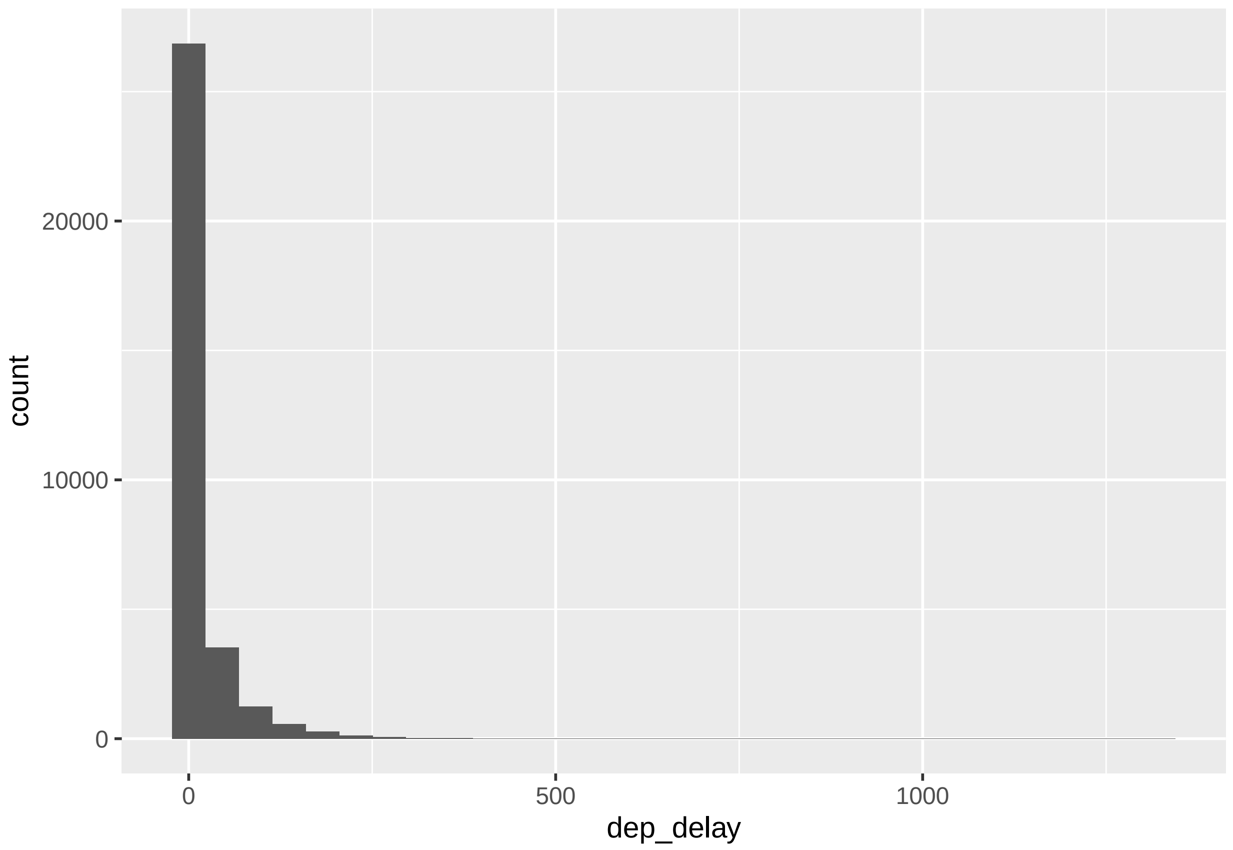
- How delayed were flights that were headed to Los Angeles?
- How do departure delays vary by month?
- Which of the three major NYC airports has the best on time percentage for departing flights?

Analysis

Departure delays

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

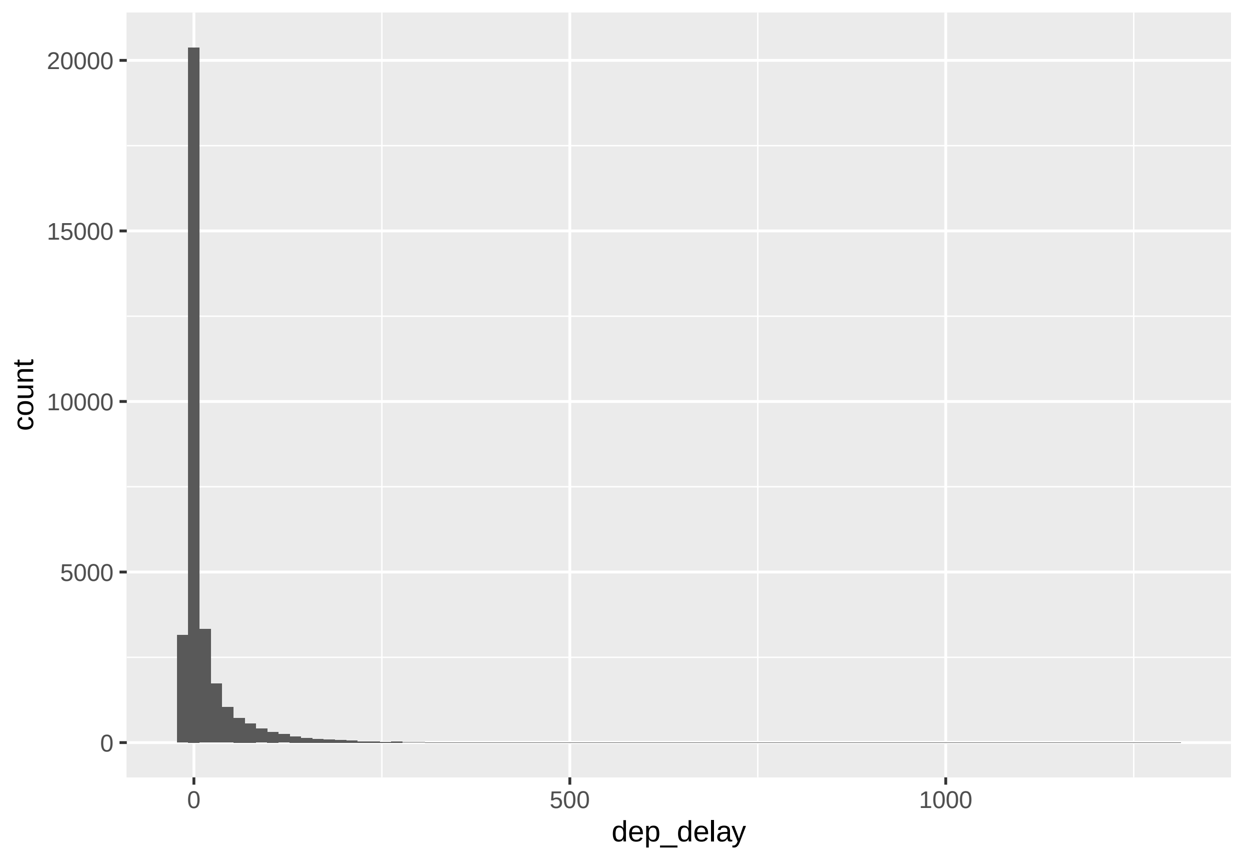
```
ggplot(data = nycflights, aes(x = dep_delay)) +
  geom_histogram()
```



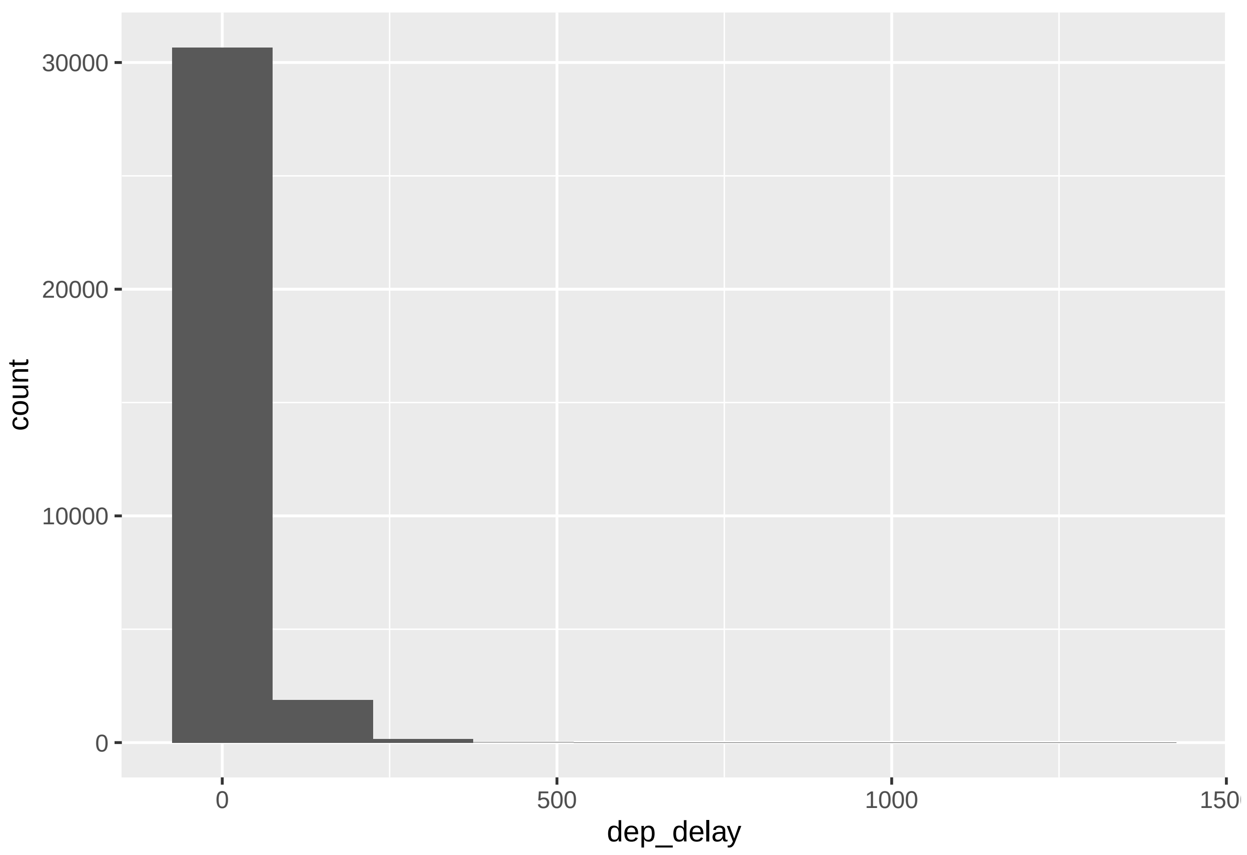
This function says to plot the `dep_delay` variable from the `nycflights` data frame on the x-axis. It also defines a `geom` (short for geometric object), which describes the type of plot you will produce.

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)
```



```r
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

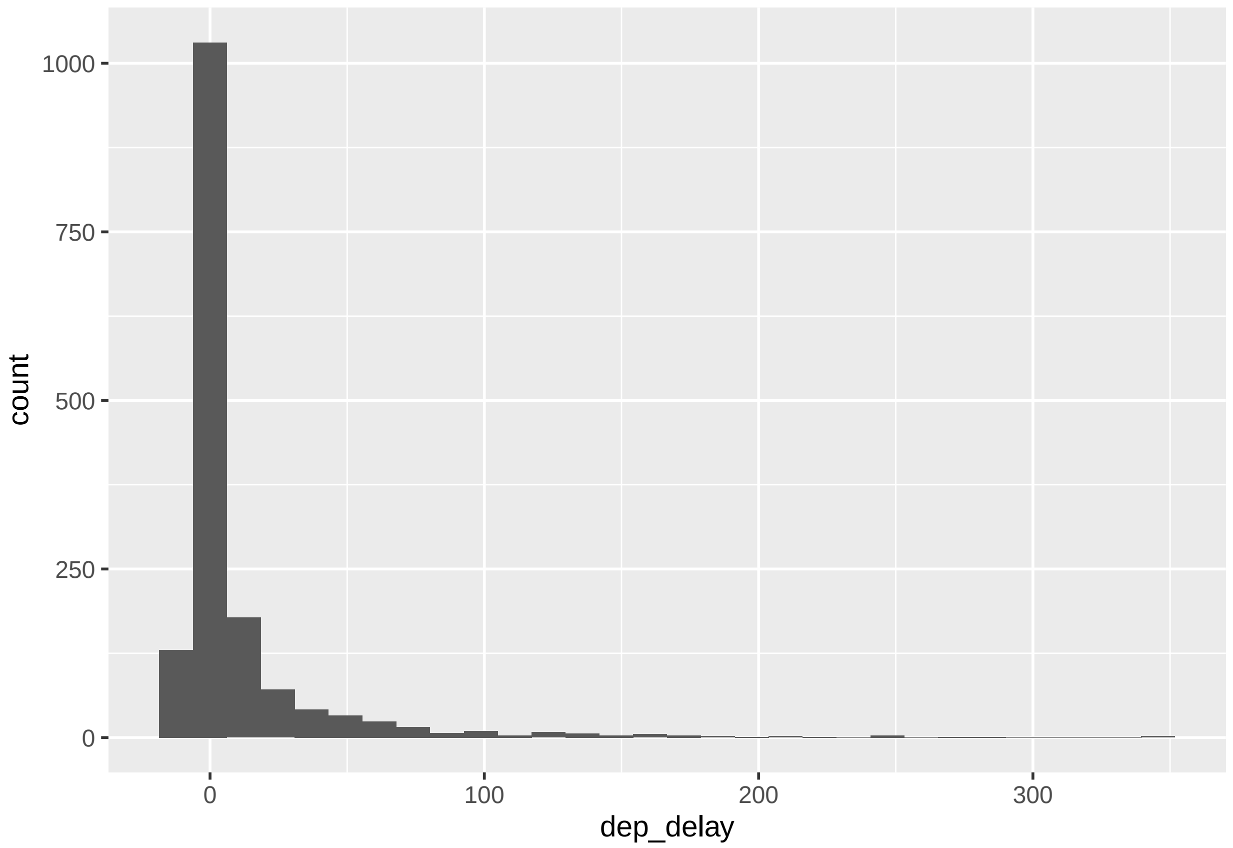
Answer (Koohyar)

Not exactly. The difference lies in how more details are revealed with a smaller bin width. When the bin width is set to 15 minutes, you can observe finer granularity: this narrower bin width provides more detailed results compared to the default and 150-minute bins. Since the majority of delays are small, a larger bin width tends to obscure the details in the lower range. However, we must exercise caution when selecting the appropriate bin width. A very small bin width may lose its practical significance—people don't typically concern themselves with 5-minute intervals. Remember, histograms serve to give us an overall sense of data distribution. Hence, choosing the right bin width is crucial in defining the value of histograms.

End of 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.

```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?

Insert your answer here

Koohyar'a Answer

```
sfo_feb_flights <- nycflights|>
  filter(dest=="SFO", month==2)
#how many fligth meets these crtieria
print("How many Flight meet destination SFO during Feb?")
```

```
## [1] "How many Flight meet destination SFO during Feb?"
```

```r
nrow(sfo_feb_flights)
```

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

```r
#summarise (sfo_feb_flights, n=n(sfo_feb_flights))

#extra parctices
#fLIGHT TO lax all around a year
lax_feb_flights <- nycflights|>
  filter(dest=="LAX", month==2)
print("How many Flight meet destination LAX during Feb?")
```

```
## [1] "How many Flight meet destination LAX during Feb?"
```

```r
nrow(lax_feb_flights)
```

```
## [1] 95
```

```r
#flight to sfo or lax in feb

#fLIGHT TO lax all around a year
laxorsfo_feb_flights <- nycflights|>
  filter(dest=="LAX" & month==2 | dest=="SFO" & month==2)
print("How many Flight meet destination SFO or LAX during Feb?")
```

```
## [1] "How many Flight meet destination SFO or LAX during Feb?"
```

```r
nrow(laxorsfo_feb_flights)
```

```
## [1] 163
```

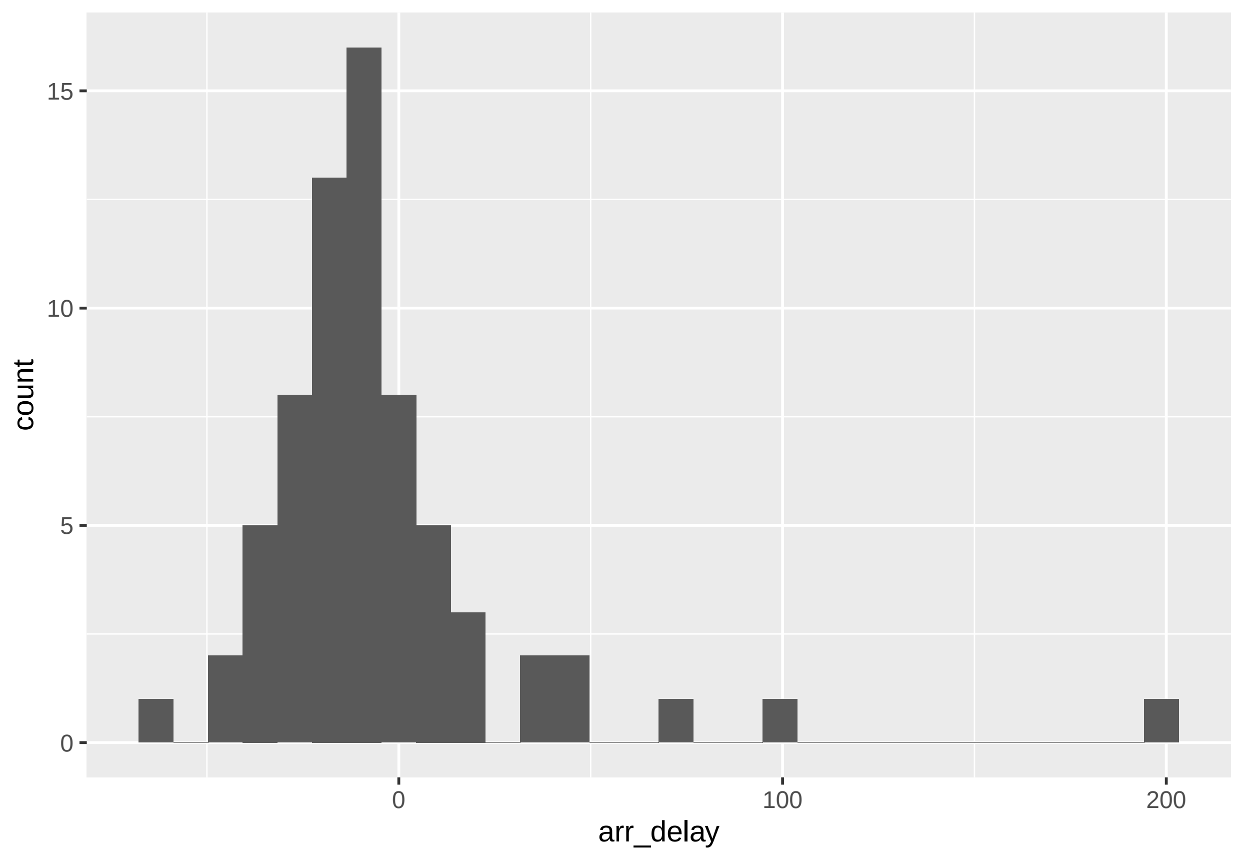
End of Answer 2

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

Koohyar's Answer

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



```
print("Sumamry of flights to SFO with actual delay > 5 minutes")
```

```
## [1] "Sumamry of flights to SFO with actual delay > 5 minutes"
```

```
sfo_feb_flights %>%
  filter(arr_delay>5)%>%
  summarise(
    mean_1 = mean(arr_delay),
    median_1 = median(arr_delay),
    max_1 = max(arr_delay),
    n = n()
  )
```

```
## # A tibble: 1 x 4
##   mean_1 median_1 max_1     n
##    <dbl>    <dbl> <dbl> <int>
## 1   42.3       21   196    15
```

```
print("summary without considering histogram")
```

```
## [1] "summary without considering histogram"
```

```r
sfo_feb_flights %>%
  summarise(mean_dd   = mean(arr_delay),
            median_dd = median(arr_delay),
            IQR_dd = IQR(arr_delay),
            min_dd = min(arr_delay),
            max_dd = max(arr_delay),
            n      = n()
            )
```

```
## # A tibble: 1 x 6
##   mean_dd median_dd IQR_dd min_dd max_dd     n
##     <dbl>     <dbl>  <dbl>  <dbl>  <dbl> <int>
## 1    -4.5       -11   23.2    -66    196    68
```

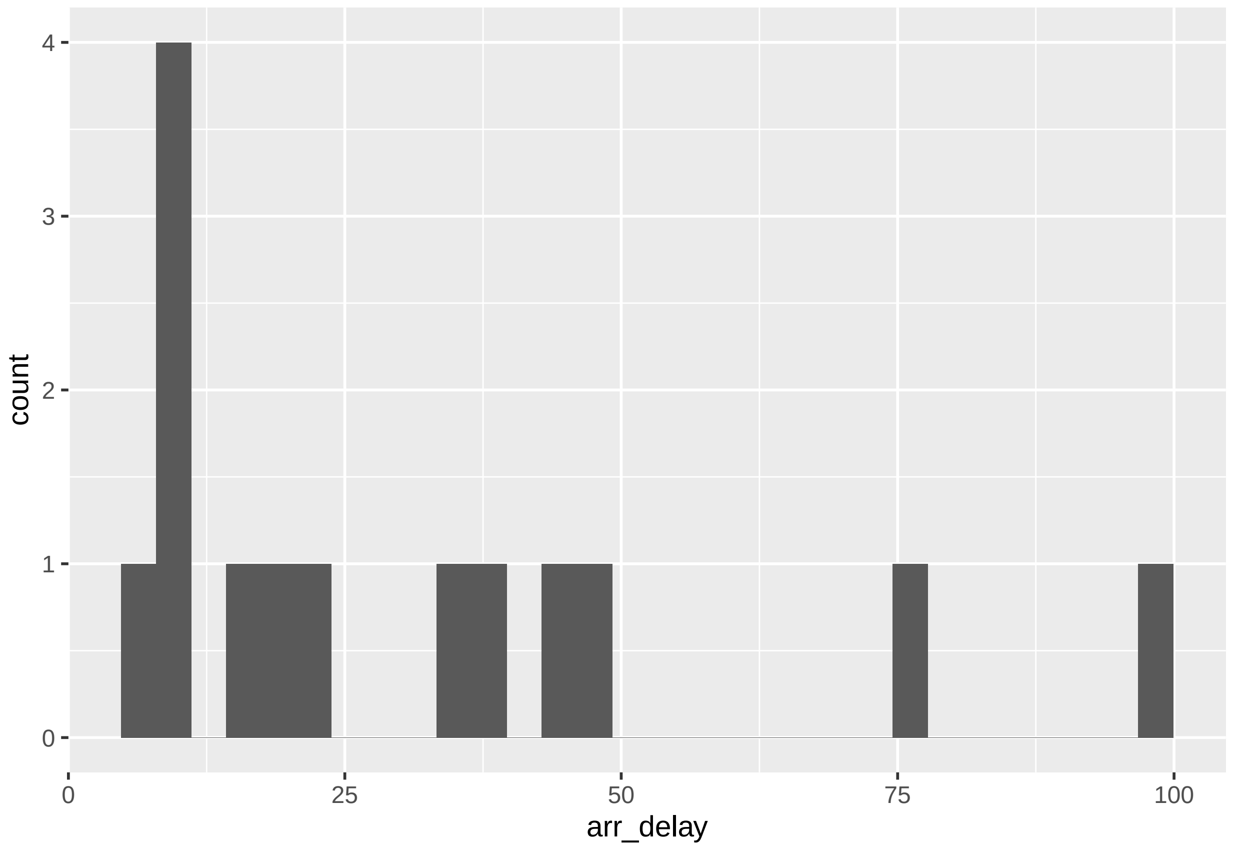
```r
print("summary considering histogram with arr_delay>5 & arr_delay<150")
```

```
## [1] "summary considering histogram with arr_delay>5 & arr_delay<150"
```

```r
sfo_feb_flights %>%
  filter(arr_delay>5 & arr_delay<150)%>%
  summarise(mean_dd   = mean(arr_delay),
            median_dd = median(arr_delay),
            IQR_dd = IQR(arr_delay),
            SD_dd  = sd(arr_delay),
            min_dd = min(arr_delay),
            max_dd = max(arr_delay),
            n      = n()
            )
```

```
## # A tibble: 1 x 7
##   mean_dd median_dd IQR_dd SD_dd min_dd max_dd     n
##     <dbl>     <dbl>  <dbl> <dbl>  <dbl>  <dbl> <int>
## 1    31.3      19.5   33.8  28.0      7     99    14
```

```r
sfo_feb_flights %>%
  filter(arr_delay>5 & arr_delay<150)%>%
  ggplot(aes(x = arr_delay)) +
  geom_histogram(bins=30)
```



Examining the histograms reveals some intriguing insights. Firstly, a significant number of flights actually arrived earlier than scheduled, so they should not be classified as delays. Furthermore, the histograms exhibit skewness toward early arrivals rather than delays. In reality, a delay of around 5 minutes should not be considered significant. Additionally, the histogram distribution spans from 50 minutes to 200 minutes, with only a few outliers—specifically, a handful of flights experiencing delays longer than 150 minutes.

end of 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

Koohyar's answer

```
print("summary of arrival flight without considering removing early arrivals")
```

```
## [1] "summary of arrival flight without considering removing early arrivals"
```

```
sfo_feb_flights %>%
  group_by(origin)%>%
  summarise(median_dd = median(arr_delay),
            IQR_dd = IQR(arr_delay),
            n      = n()
            )
```

```
## # A tibble: 2 x 4
##   origin median_dd IQR_dd     n
##   <chr>      <dbl>  <dbl> <int>
## 1 EWR        -15.5   17.5     8
## 2 JFK        -10.5   22.8    60
```

```
print("summary of arrival flight considering arr_delay>5 & arr_delay<150")
```

```
## [1] "summary of arrival flight considering arr_delay>5 & arr_delay<150"
```

```
sfo_feb_flights %>%
  filter(arr_delay>5 & arr_delay<150)%>%
  group_by(origin)%>%
  summarise(median_dd = median(arr_delay),
            IQR_dd = IQR(arr_delay),
            n      = n()
            )
```

```
## # A tibble: 2 x 4
##   origin median_dd IQR_dd     n
##   <chr>      <dbl>  <dbl> <int>
## 1 EWR            7      0     1
## 2 JFK           21     34    13
```

End of 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 `desc`ending order

```r
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?

Insert your answer here

Koohyar Answer

Median reports the actual delay or average of two actual ones vs mean reports the average of central tendency which most likely is between two actual values. On the other hand, if there is some skewness toward the lower or higher or a few outlines for those with a low number of events, the mean data can be a lower value or a higher one compared to what can happen in reality. This means outlines can significantly affect the outcome.

In summary, if data is balanced and there are many events, the mean and median should be closed, but if there are a few outliers on both higher or lower ends and the data is not extensive, the median most likely presents the better representation of data for a person to decide when to go a light.

Our previous observation above showed skewness in the data and that is a reason to lean toward the median to make the decision not the mean.

End of 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 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:

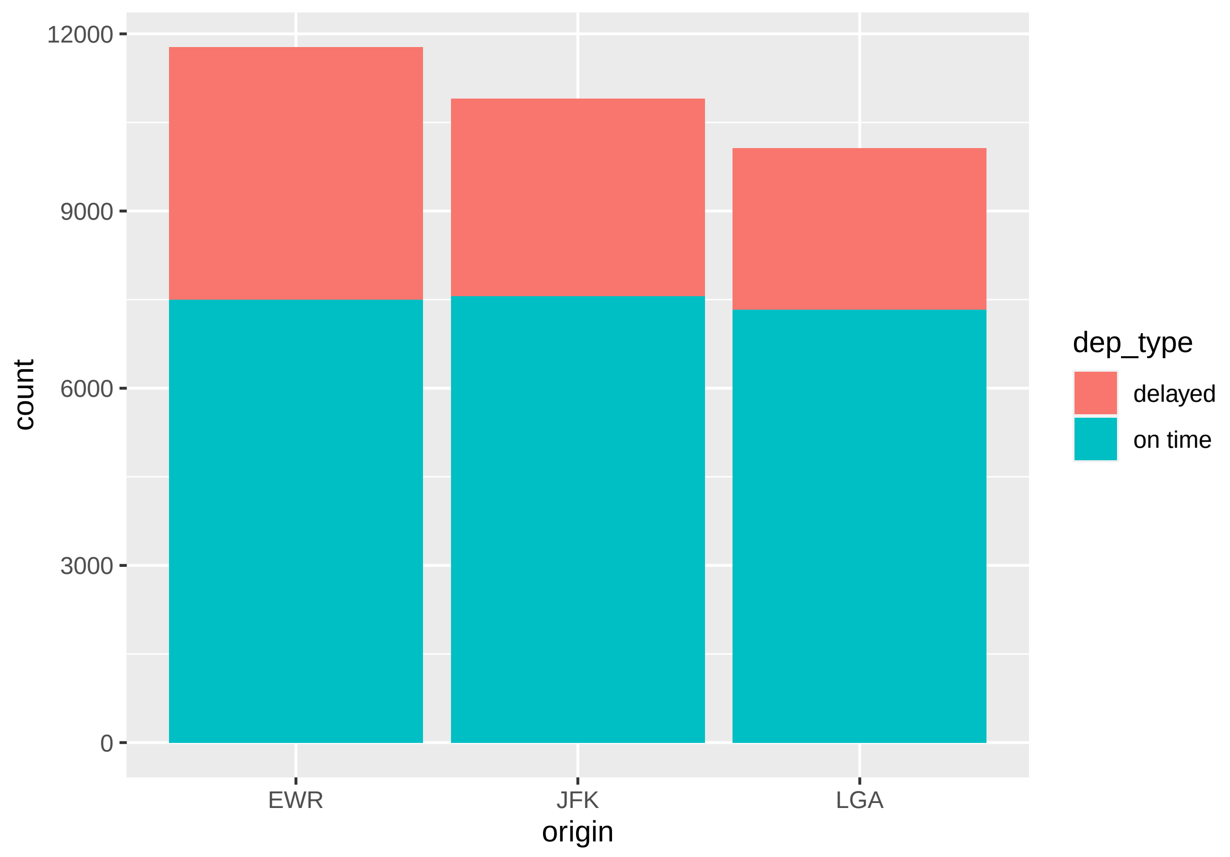
```r
nycflights %>%
  group_by(origin) %>%
  summarise(ot_dep_rate = sum(dep_type == "on time") / n()) %>%
  arrange(desc(ot_dep_rate))
```

```
## # 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?
7. **`Koohyar's Answer: Those are very close, it seems EWR is the best, but data based on the mean is close.`**

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()
```



Insert your answer here

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

Koohyar's Answer

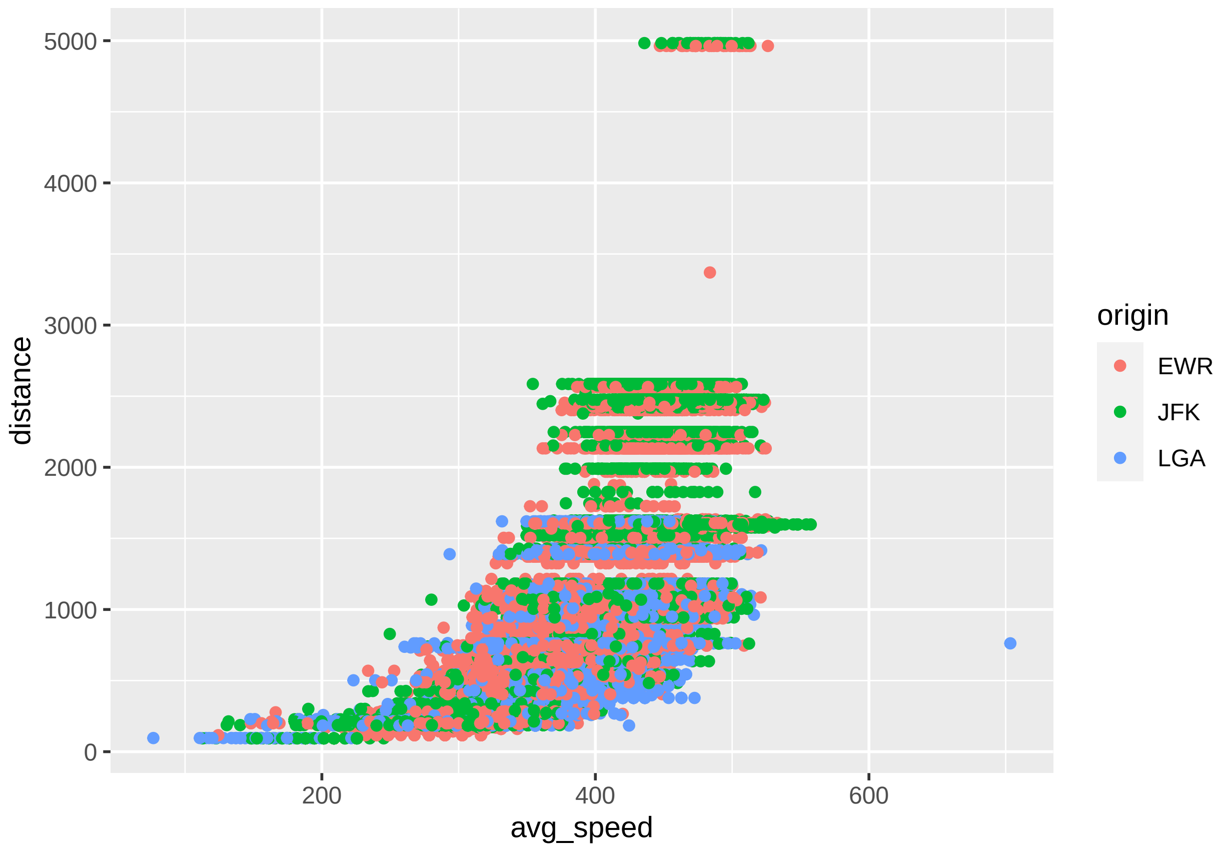
```
nycflights <- nycflights|>mutate(avg_speed=distance*60/air_time)
```

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

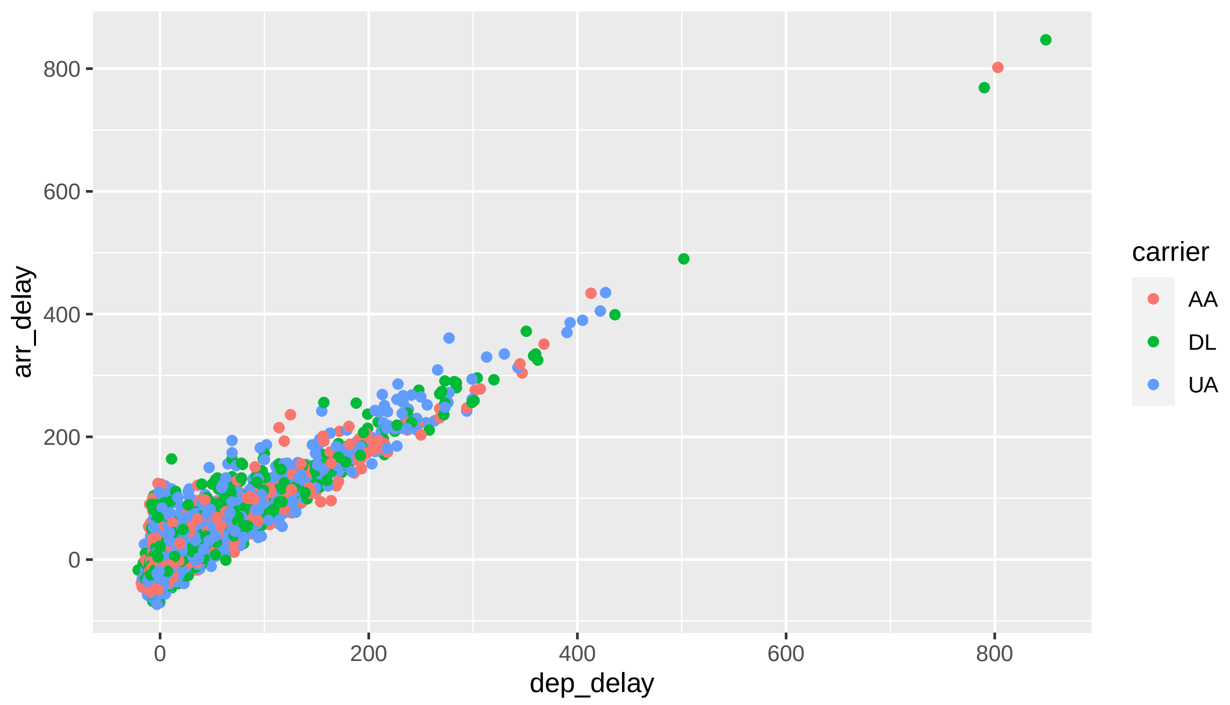
Koohyar's Answer

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



Looking at the graph, it is evident that, except for a very few outliers, speed generally increases as distance increases. However, the data is somewhat scattered. This suggests that there might be another variable at play, such as the type of airplanes or their specific models, influencing this distribution. Additionally, there is one flight from LGA (LaGuardia Airport) that stands out as surprisingly fast—either it is an error or a hyper-sonic aircraft.

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



Insert your answer here

Koohyar's Answer

In this case, we want to examine the distribution of departure delays (**`dep_delay`**) when the arrival delay (**`arr_delay`**) is less than a threshold. For this example, I assume a threshold of 5 minutes, which has been used previously. This threshold represents the amount of time an airplane or carrier can compensate if the flight is delayed.

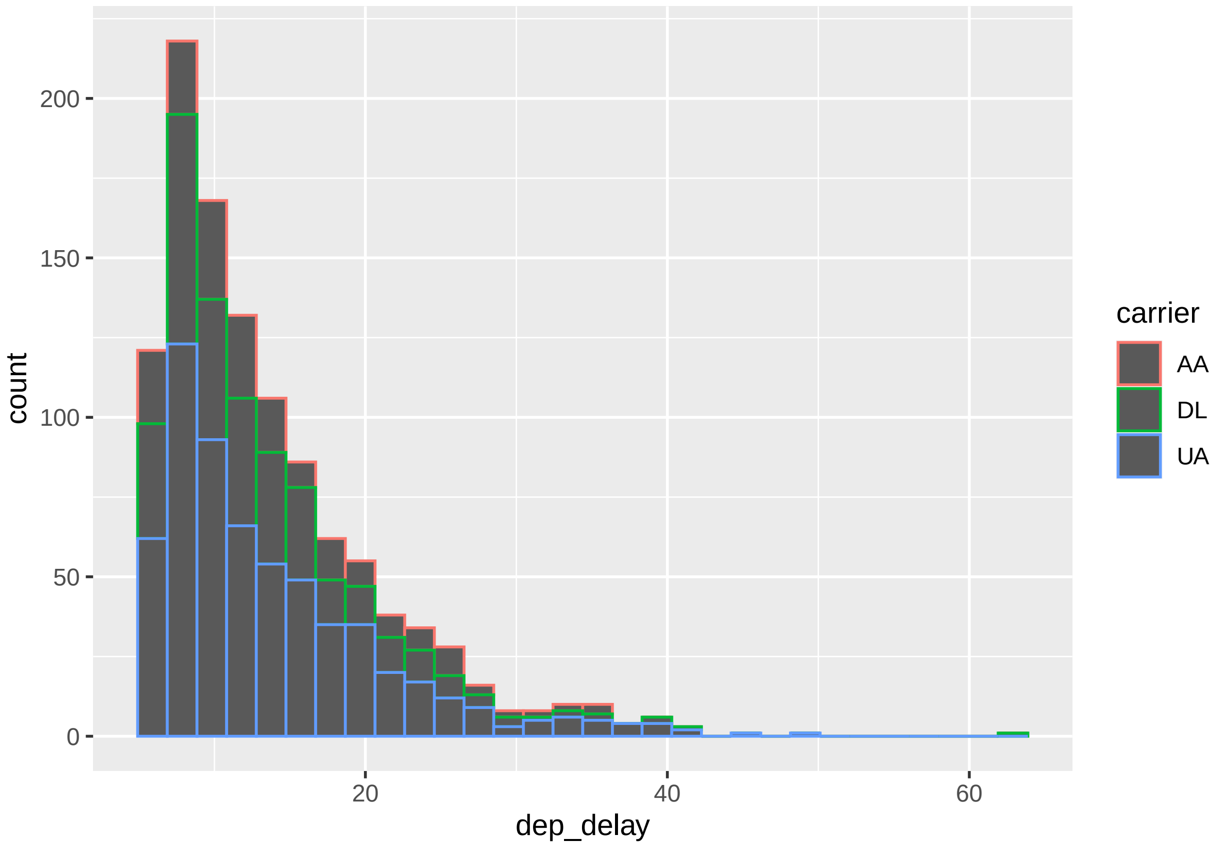
Upon analyzing the graph, it appears that when **`arr_delay`** is approximately 40 minutes or less, the corresponding **`dep_delay`** can be as low as 5 minutes. To further validate this observation, let's calculate the median or mean of **`dep_delay`** for cases where **`arr_delay`** is less than 5 minutes.

```
dl_aa_ua_1 <- dl_aa_ua %>%
  filter(arr_delay<5 )

# Amount the arrival delay of 5 minites and less, we want to know which flight has dep_delay of more th
dl_aa_ua_1 %>% filter(dep_delay>5)%>%
  summarize(
    mean_delay    = mean(dep_delay),
    median_deleay = median(dep_delay),
    IQR_delay     = IQR(dep_delay),
    n_delay       = n()
  )
```

```
## # A tibble: 1 x 4
##   mean_delay median_deleay IQR_delay n_delay
##        <dbl>         <dbl>     <dbl>   <int>
## 1       13.4            11         9    1116
```

```r
dl_aa_ua_1 %>% filter(dep_delay>5 )%>%
  ggplot(aes(x=dep_delay, color = carrier))+geom_histogram(bins=30)
```



The average recovery time is approximately 13 minutes, with a median of 11 minutes. Technically, it appears that they only recover about 10 to 15 minutes but they can also recover as large as 40 minutes or even one example of ~65 minutes.

End of Answer

End of Assignment