L08 Relational Data

Data Science I (STAT 301-1)

Contents

Overview

The goal of this lab is to learn and understand the concepts of **relational data**. It is extremely rare that data analyses involve one all-encompassing dataset; we usually want to combine information from multiple data tables/sources to answer interesting questions. The collection of data tables/sources is called **relational data** because it is the relations connecting the datasets together that are important.

See two-table verbs in `dplyr` for more information concerning relational data. Relational Database Wikipedia Page

Datasets

All datasets are found within the R packages, which students should be able to identify and download when needed.

Exercises

Please complete the following exercises. Be sure your solutions are clearly indicated and that the document is neatly formatted.

```
library(tidyverse)
library(nycflights13)
library(Lahman)
library(babynames)
library(nasaweather)
library(fueleconomy)
library(maps)
```

Load Packages

Exercise 1 (Website: 13.2.1 Ex. 1)

Imagine you wanted to draw (approximately) the route each plane flies from its origin to its destination. What variables would you need? What tables would you need to combine? **You would need the `faa`**

code from the `airports` table to identify the origin and destination airport, the `tailnum` from the `planes` table, and perhaps variables like `air_time` or `distance` from the `flights` table.

Exercise 2 (Website: 13.2.1 Ex. 2)

A relationship between `weather` and `airports` is possible. What is the relationship and how would it appear in a diagram (i.e., which variables should be matched)? **`origin` from `weather` and `faa` from `airports` are related, as they both describe airports. They both connect to the `origin` variable of `flights`.**

Exercise 3 (Website: 13.2.1 Ex. 3)

`weather` only contains information for the origin (NYC) airports. If it contained weather records for all airports in the USA, what additional relation would it define with `flights`? **`weather` would likely also contain a `dest` variable that would relate to `flights` if it contained information for all airports in the U.S.**

Exercise 4 (Website: 13.3.1 Ex. 1)

Add a surrogate key to `flights`.

```
flights1 <- flights %>%
  mutate(surr_key = row_number())

flights1 %>%
  select(surr_key, year, month, day, origin, dest)
```

```
## # A tibble: 336,776 x 6
##    surr_key  year month   day origin dest
##       <int> <int> <int> <int> <chr>  <chr>
##  1        1  2013     1     1 EWR    IAH
##  2        2  2013     1     1 LGA    IAH
##  3        3  2013     1     1 JFK    MIA
##  4        4  2013     1     1 JFK    BQN
##  5        5  2013     1     1 LGA    ATL
##  6        6  2013     1     1 EWR    ORD
##  7        7  2013     1     1 EWR    FLL
##  8        8  2013     1     1 LGA    IAD
##  9        9  2013     1     1 JFK    MCO
## 10       10  2013     1     1 LGA    ORD
## # ... with 336,766 more rows
```

Exercise 5 (Website: 13.3.1 Ex. 2)

For each of the following datasets, identify any key column[s] and specify whether they are primary or foreign keys. You might need to install some packages and read some documentation.

- `Lahman::Batting`,

```
Batting %>%
  count(playerID, yearID, teamID, lgID, stint) %>%
  filter(n > 1)
```

```
## [1] playerID yearID   teamID   lgID     stint    n
## <0 rows> (or 0-length row.names)
```

```
People %>%
  count(playerID) %>%
  filter(n > 1)
```

```
## [1] playerID n
## <0 rows> (or 0-length row.names)
```

playerID, yearID, teamID, lgID, and stint comprise the primary key for Batting. They also form a foreign key for Pitching and Fielding. playerID is a foreign key for People. NOTE: I only analyzed the main tables from the library, as there are dozens of additional ones that would take too long to figure out. * babynames::babynames

```
babynames %>%
  count(name, year, sex) %>%
  filter(n > 1)
```

```
## # A tibble: 0 x 4
## # ... with 4 variables: name <chr>, year <dbl>, sex <chr>, n <int>
```

name, year, and sex comprise a primary key for babynames. year and sex comprise a foreign key for applicants, and year is a foreign key for births. There is no foreign key for lifetables. * nasaweather::atmos

```
atmos %>%
  count(lat, long, year, month) %>%
  filter(n > 1)

borders %>%
  count(lat, long, country, group) %>%
  filter(n > 1)

elev %>%
  count(lat, long) %>%
  filter(n > 1)

glaciers %>%
  count(lat, long) %>%
  filter(n > 1)

storms %>%
  count(lat, long, year, month, hour, name, seasday) %>%
  filter(n > 1)
```

lat, long, year, and month uniquely identify data in atmos and thus are primary keys. There are no foreign keys for borders. lat and long are foreign keys for elev, and form a foreign key along with id in glaciers. However, id in glaciers does uniquely identify data by itself, so lat and long are not needed. lat, long, year, and month, form part of a foreign key for storms, along with hour, name, and seasday. * fueleconomy::vehicles

```
vehicles %>%
  count(id) %>%
  filter(n > 1)
```

```
## # A tibble: 0 x 2
## # ... with 2 variables: id <dbl>, n <int>
```

```
common %>%
  count(make, model) %>%
  filter(n > 1)
```

```
## # A tibble: 0 x 3
## # ... with 3 variables: make <chr>, model <chr>, n <int>
```

`id` is a primary key for `vehicles` and `make` and `model` form a foreign key for `common`. * `ggplot2::diamonds`

```
diamonds %>%
  count(carat, cut, color, clarity, depth, price, x, y, z, table) %>%
  filter(n > 1)
```

```
## # A tibble: 143 x 11
##    carat cut       color clarity depth price     x     y     z table     n
##    <dbl> <ord>     <ord> <ord>   <dbl> <int> <dbl> <dbl> <dbl> <dbl> <int>
##  1  0.3  Good      J     VS1      63.4   394  4.23  4.26  2.69    57     2
##  2  0.3  Very Good G     VS2      63     526  4.29  4.31  2.71    55     2
##  3  0.3  Very Good J     VS1      63.4   506  4.26  4.23  2.69    57     2
##  4  0.3  Premium   D     SI1      62.2   709  4.31  4.28  2.67    58     2
##  5  0.3  Ideal     G     VS2      63     675  4.31  4.29  2.71    55     2
##  6  0.3  Ideal     G     IF       62.1   863  4.32  4.35  2.69    55     2
##  7  0.3  Ideal     H     SI1      62.2   450  4.26  4.29  2.66    57     2
##  8  0.3  Ideal     H     SI1      62.2   450  4.27  4.28  2.66    57     2
##  9  0.31 Good      D     SI1      63.5   571  4.29  4.31  2.73    56     2
## 10  0.31 Very Good D     SI1      63.5   732  4.31  4.29  2.73    56     2
## # ... with 133 more rows
```

Even when using every single variable in the `diamonds` dataset, there is no unique identifier for observations, and thus no primary key for `diamonds`.

Exercise 6 (Website: 13.3.1 Ex. 3)

Draw a diagram illustrating the connections between the `Batting`, `Master`, and `Salaries` tables in the `Lahman` package. Draw another diagram that shows the relationship between `Master`, `Managers`, `AwardsManagers`.

You do not have to include the drawings. We trust that you will do this and then check it against the solution set. You are welcome to challenge yourself by scanning your drawings and including them in your lab submission, but it is not required.

Exercise 7 (Website: 13.4.6 Ex. 1)

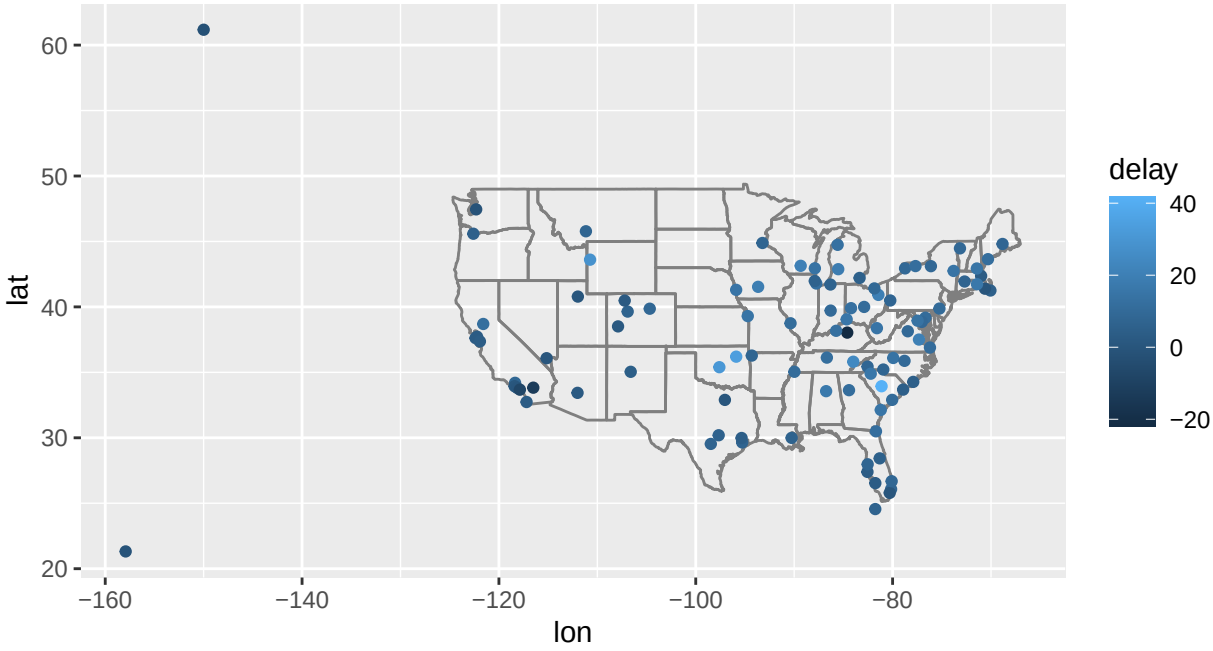
Compute the average arrival delay by destination, then join on the `airports` data frame so you can show the spatial distribution of delays. As a resource, here is a template for drawing a map of the United States (you'll be replacing the first two lines of code):

```
flights <- nycflights13::flights

avg_dest_delay <- flights %>%
  group_by(dest) %>%
  summarise(delay = mean(arr_delay, na.rm = TRUE)) %>%
  inner_join(airports, by = c(dest = "faa"))

avg_dest_delay %>%
```

```
  ggplot(aes(lon, lat, colour = delay)) +
  borders("state") +
  geom_point() +
  coord_quickmap()
```



Don't worry if you don't understand what `semi_join()` does — you don't need it for this problem. Consider using the size or color of the points to display the average delay for each airport.

Exercise 8 (Website: 13.4.6 Ex. 2)

Add the location for both the origin *and* destination (i.e. the `lat` and `lon`) to `flights`.

```
airport_location <- airports %>%
  select(faa, lat, lon)

flights %>%
  select(year:day, hour, origin, dest) %>%
  left_join(airport_location, by = c("origin" = "faa")) %>%
  left_join(airport_location, by = c("dest" = "faa"))
```

```
## # A tibble: 336,776 x 10
##     year month   day  hour origin dest  lat.x lon.x lat.y lon.y
##    <int> <int> <int> <dbl> <chr>  <chr> <dbl> <dbl> <dbl> <dbl>
##  1  2013     1     1     5 EWR    IAH    40.7 -74.2  30.0 -95.3
##  2  2013     1     1     5 LGA    IAH    40.8 -73.9  30.0 -95.3
##  3  2013     1     1     5 JFK    MIA    40.6 -73.8  25.8 -80.3
##  4  2013     1     1     5 JFK    BQN    40.6 -73.8  NA    NA
##  5  2013     1     1     6 LGA    ATL    40.8 -73.9  33.6 -84.4
##  6  2013     1     1     5 EWR    ORD    40.7 -74.2  42.0 -87.9
##  7  2013     1     1     6 EWR    FLL    40.7 -74.2  26.1 -80.2
##  8  2013     1     1     6 LGA    IAD    40.8 -73.9  38.9 -77.5
##  9  2013     1     1     6 JFK    MCO    40.6 -73.8  28.4 -81.3
## 10  2013     1     1     6 LGA    ORD    40.8 -73.9  42.0 -87.9
```

```
## # ... with 336,766 more rows
```

Exercise 9 (Website: 13.4.6 Ex. 3)

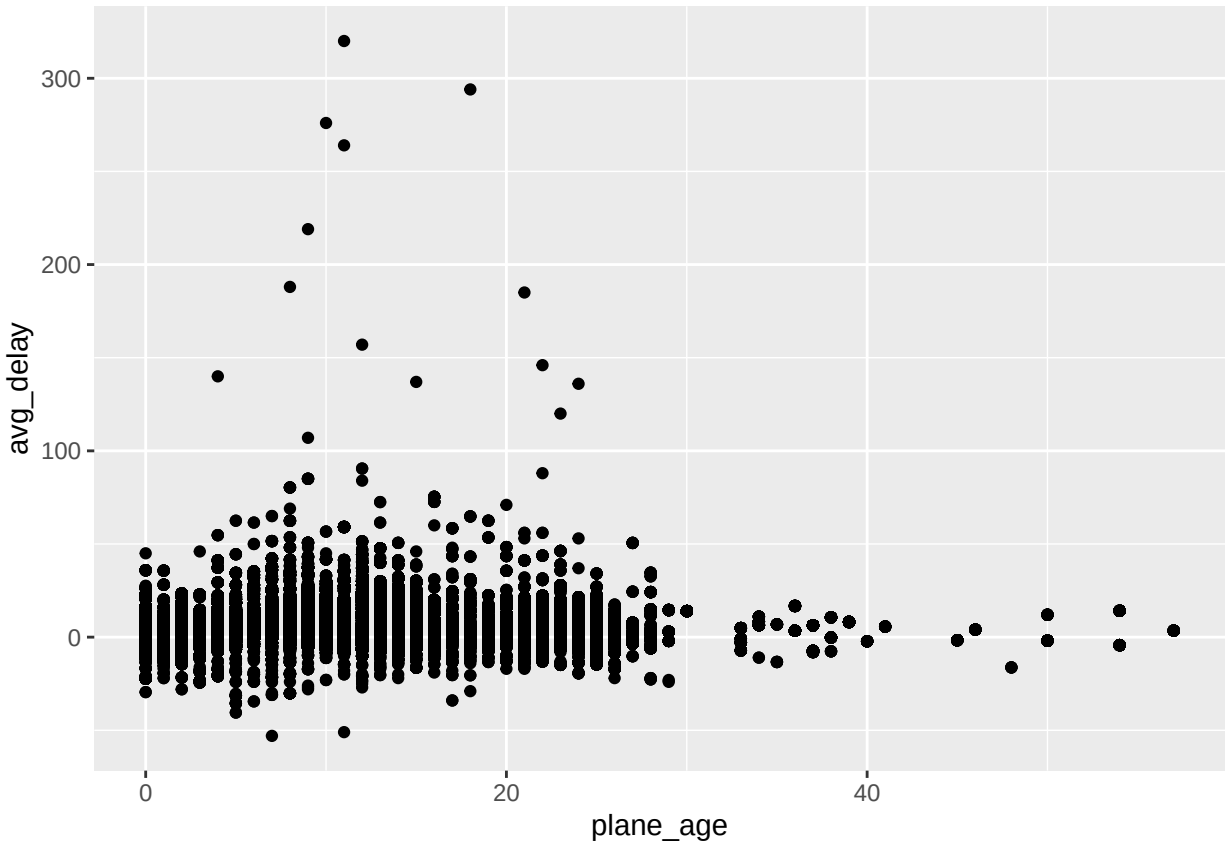
Is there a relationship between the age of a plane and its average arrival delay?

```
planes2 <- planes %>%
  mutate(plane_age = 2013 - year)

flights3 <- flights %>%
  inner_join(planes2, by = 'tailnum') %>%
  group_by(tailnum) %>%
  mutate(avg_delay = mean(arr_delay, na.rm = TRUE)) %>%
  glimpse()
```

```
## Rows: 284,170
## Columns: 29
## Groups: tailnum [3,322]
## $ year.x         <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013...
## $ month          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ day            <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ dep_time       <int> 517, 533, 542, 544, 554, 554, 555, 557, 557, 558, 55...
## $ sched_dep_time <int> 515, 529, 540, 545, 600, 558, 600, 600, 600, 600, 60...
## $ dep_delay      <dbl> 2, 4, 2, -1, -6, -4, -5, -3, -3, -2, -2, -2, -2, 0, ...
## $ arr_time       <int> 830, 850, 923, 1004, 812, 740, 913, 709, 838, 849, 8...
## $ sched_arr_time <int> 819, 830, 850, 1022, 837, 728, 854, 723, 846, 851, 8...
## $ arr_delay      <dbl> 11, 20, 33, -18, -25, 12, 19, -14, -8, -2, -3, 7, -1...
## $ carrier        <chr> "UA", "UA", "AA", "B6", "DL", "UA", "B6", "EV", "B6"...
## $ flight         <int> 1545, 1714, 1141, 725, 461, 1696, 507, 5708, 79, 49,...
## $ tailnum        <chr> "N14228", "N24211", "N619AA", "N804JB", "N668DN", "N...
## $ origin         <chr> "EWR", "LGA", "JFK", "JFK", "LGA", "EWR", "EWR", "LG...
## $ dest           <chr> "IAH", "IAH", "MIA", "BQN", "ATL", "ORD", "FLL", "IA...
## $ air_time       <dbl> 227, 227, 160, 183, 116, 150, 158, 53, 140, 149, 158...
## $ distance       <dbl> 1400, 1416, 1089, 1576, 762, 719, 1065, 229, 944, 10...
## $ hour           <dbl> 5, 5, 5, 5, 6, 5, 6, 6, 6, 6, 6, 6, 6, 5, 6, 6, 6, 6...
## $ minute         <dbl> 15, 29, 40, 45, 0, 58, 0, 0, 0, 0, 0, 0, 0, 59, 0, 0...
## $ time_hour      <dttm> 2013-01-01 05:00:00, 2013-01-01 05:00:00, 2013-01-0...
## $ year.y         <int> 1999, 1998, 1990, 2012, 1991, 2012, 2000, 1998, 2004...
## $ type           <chr> "Fixed wing multi engine", "Fixed wing multi engine"...
## $ manufacturer   <chr> "BOEING", "BOEING", "BOEING", "AIRBUS", "BOEING", "B...
## $ model          <chr> "737-824", "737-824", "757-223", "A320-232", "757-23...
## $ engines        <int> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2...
## $ seats          <int> 149, 149, 178, 200, 178, 191, 200, 55, 200, 200, 200...
## $ speed          <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ...
## $ engine         <chr> "Turbo-fan", "Turbo-fan", "Turbo-fan", "Turbo-fan", ...
## $ plane_age      <dbl> 14, 15, 23, 1, 22, 1, 13, 15, 9, 2, 6, 15, NA, 5, 5,...
## $ avg_delay      <dbl> 3.71171171, 7.70000000, 7.65217391, -1.86046512, 2.6...
```

```
flights3 %>%
  ggplot(mapping = aes(x = plane_age, y = avg_delay)) +
  geom_point()
```



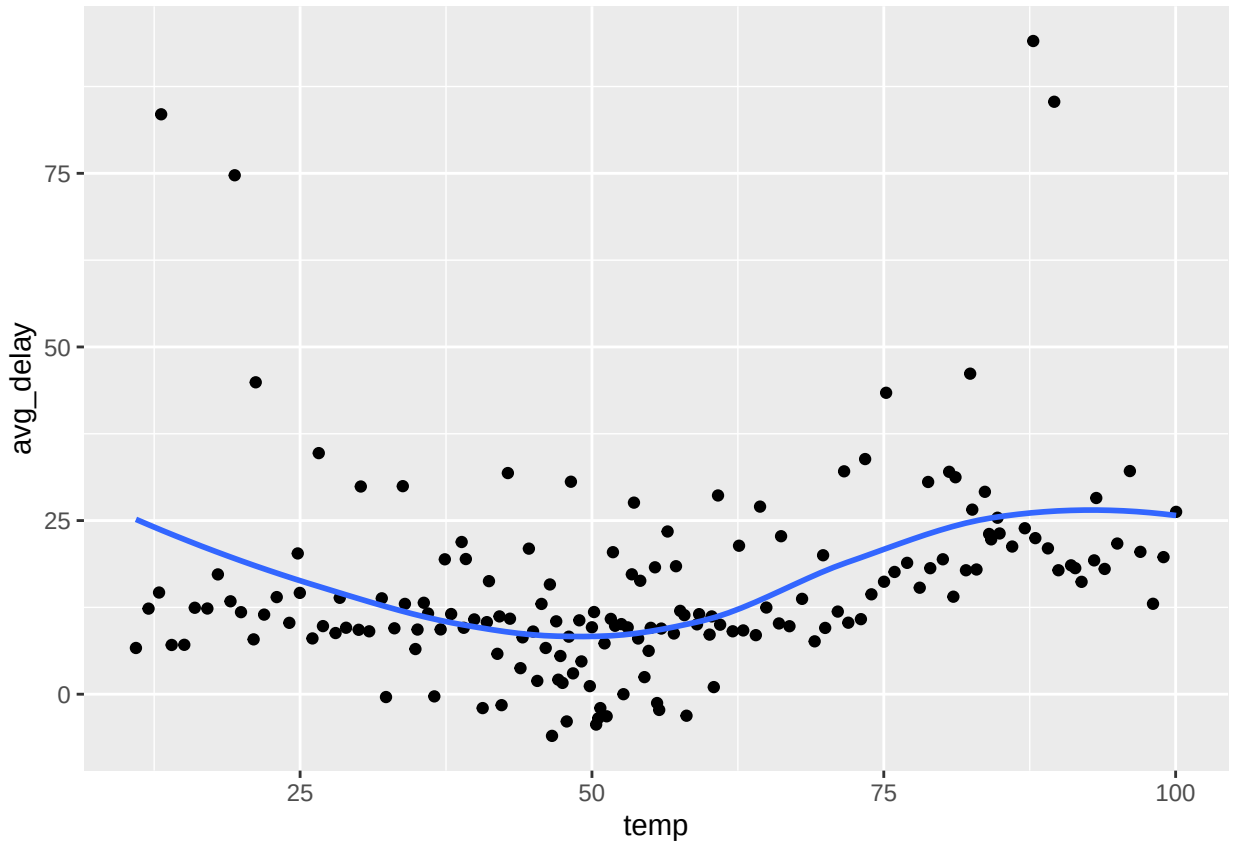
There doesn't appear to be any relationship between average arrival delay and plane age.

Exercise 10 (Website: 13.4.6 Ex. 4)

What weather conditions make it more likely to see a departure delay?

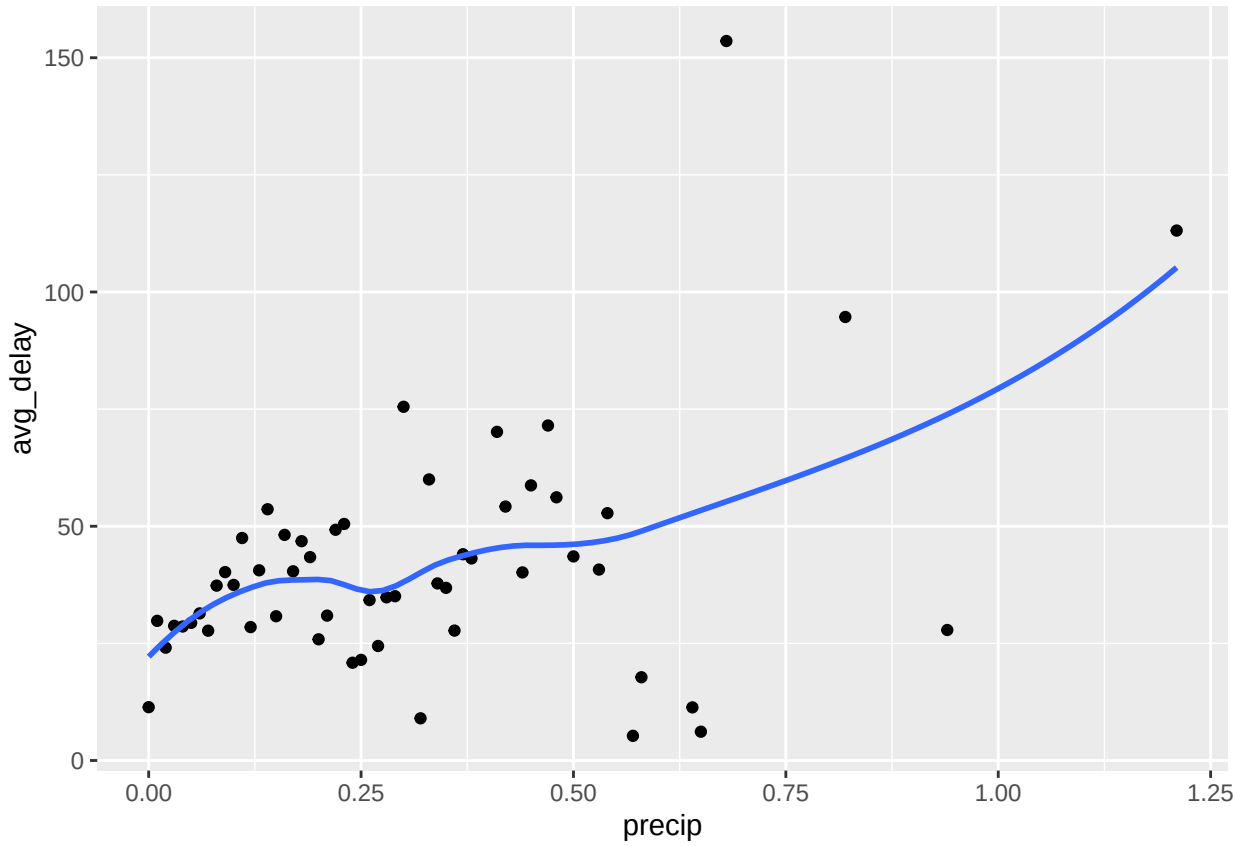
```
flight_weather <- flights %>%
  inner_join(weather, by = c(
    "origin" = "origin",
    "year" = "year",
    "month" = "month",
    "day" = "day",
    "hour" = "hour"
  ))

flight_weather %>%
  group_by(temp) %>%
  summarise(avg_delay = mean(dep_delay, na.rm = TRUE)) %>%
  ggplot(mapping = aes(x = temp, y = avg_delay)) +
  geom_point() +
  geom_smooth(se = FALSE)
```



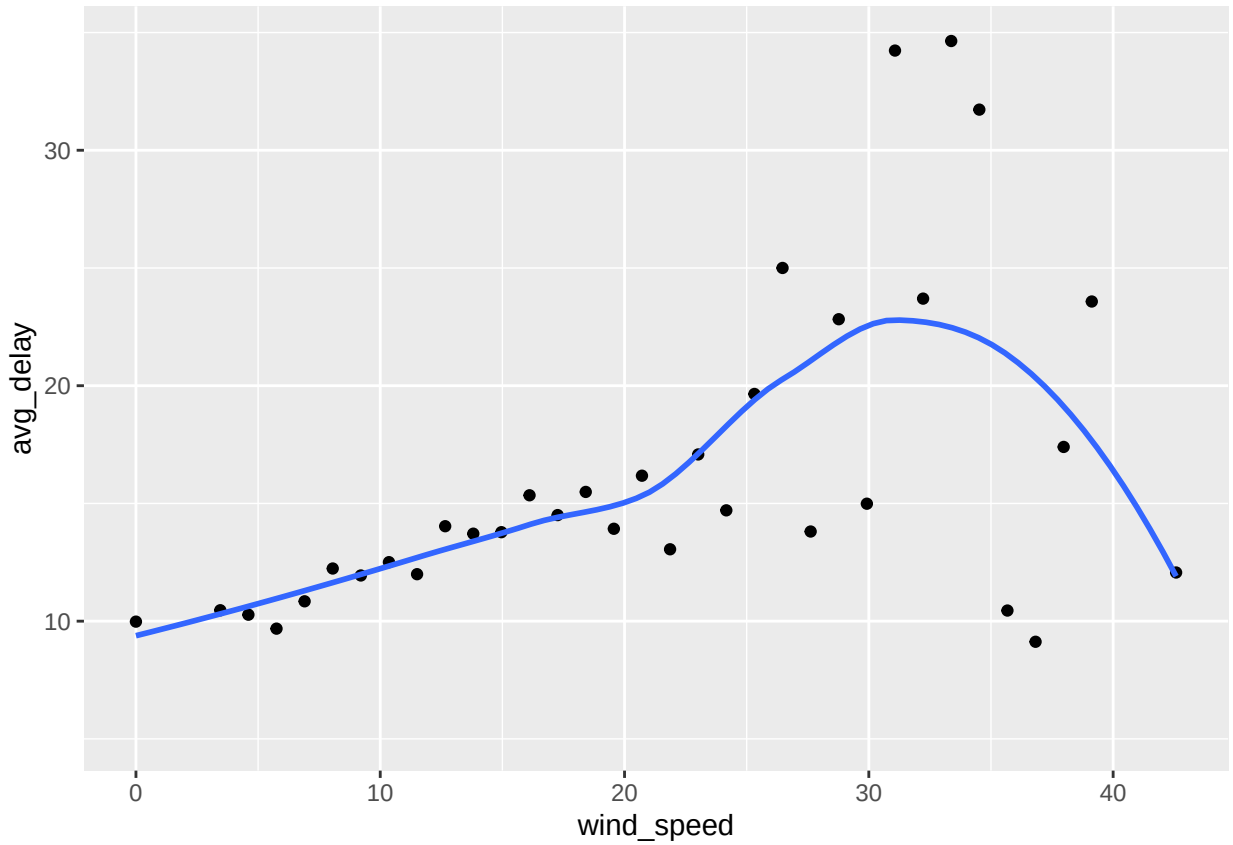

```
#Extreme hot and cold tend to have longer delays.

flight_weather %>%
  group_by(precip) %>%
  summarise(avg_delay = mean(dep_delay, na.rm = TRUE)) %>%
  ggplot(mapping = aes(x = precip, y = avg_delay)) +
  geom_point() +
  geom_smooth(se = FALSE)
```



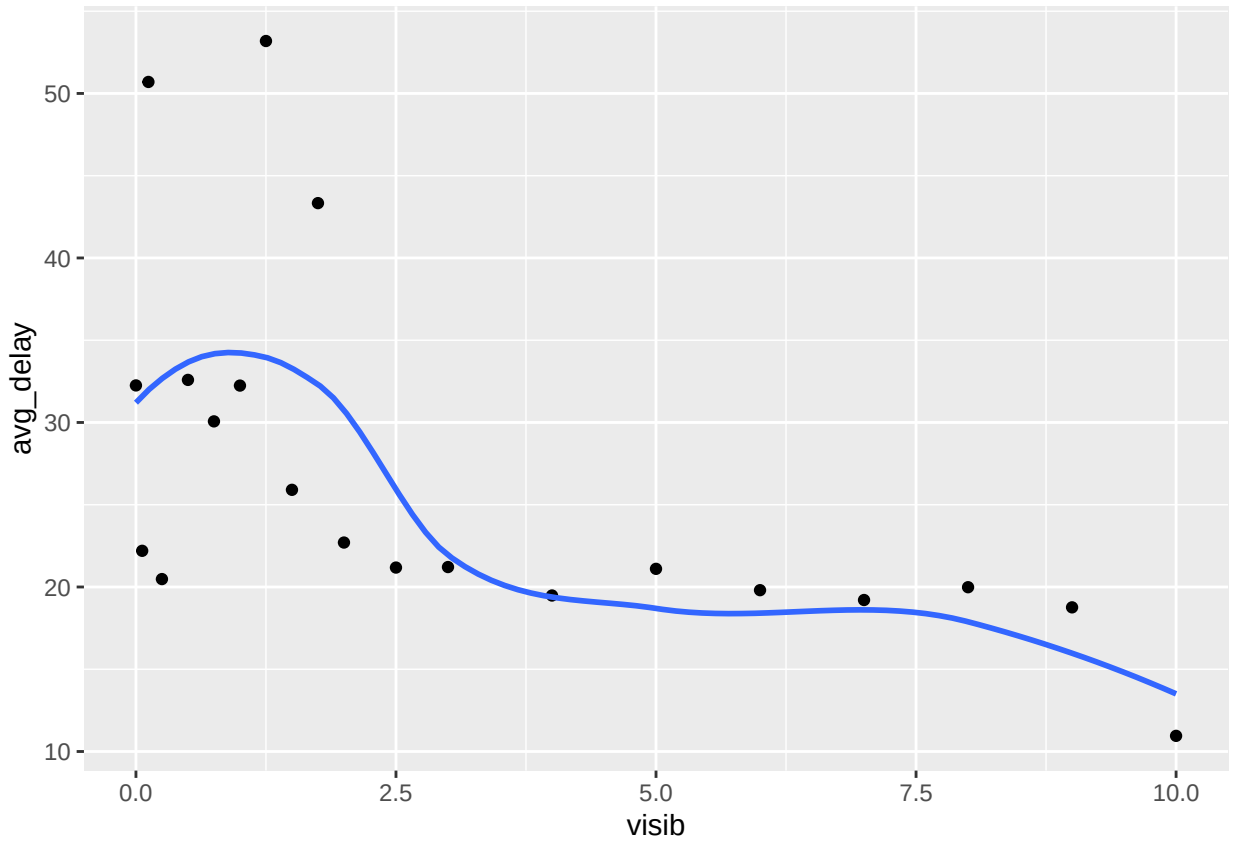

```
#More precipitation leads to longer delay times.

flight_weather %>%
  group_by(wind_speed) %>%
  summarise(avg_delay = mean(dep_delay, na.rm = TRUE)) %>%
  ggplot(mapping = aes(x = wind_speed, y = avg_delay)) +
  geom_point() +
  geom_smooth(se = FALSE)
```



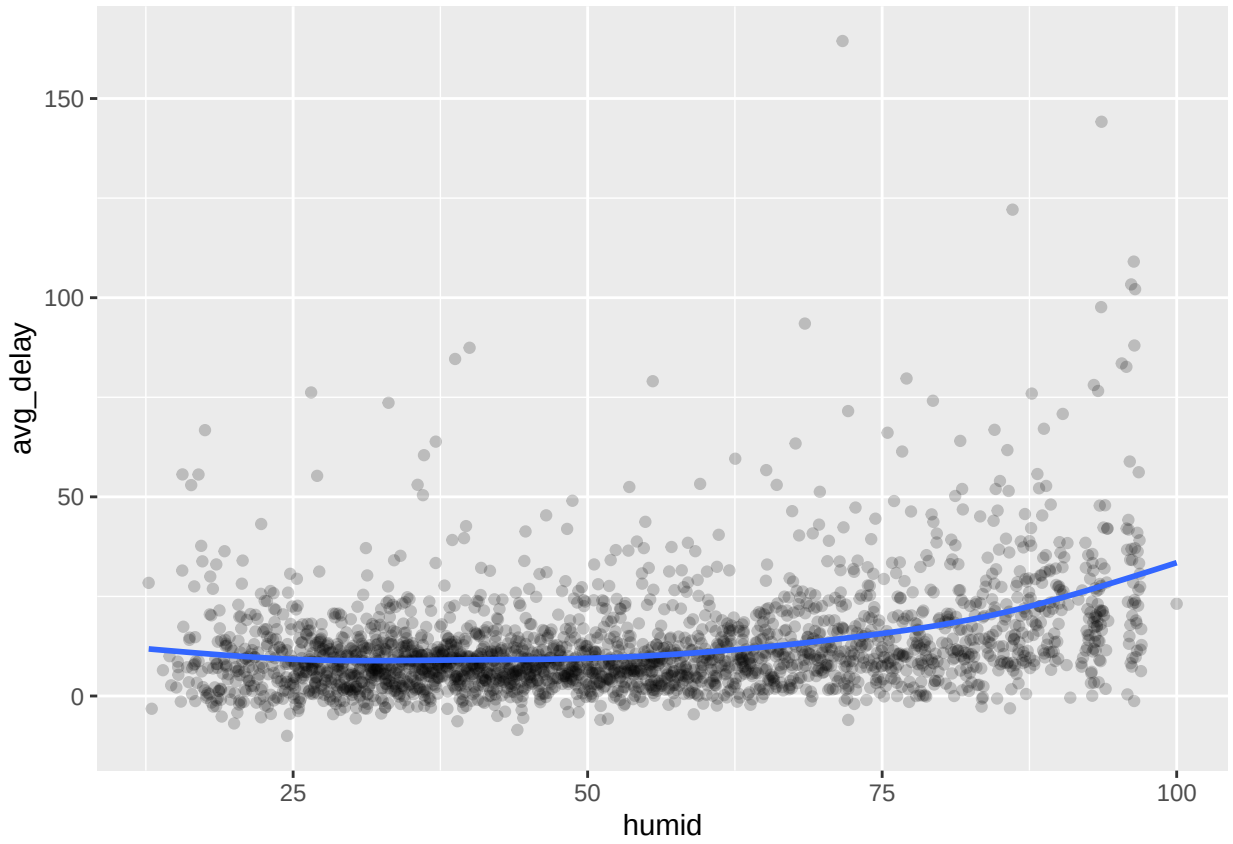

```
#Higher wind speed leads to longer delay times

flight_weather %>%
  group_by(visib) %>%
  summarise(avg_delay = mean(dep_delay, na.rm = TRUE)) %>%
  ggplot(mapping = aes(x = visib, y = avg_delay)) +
  geom_point() +
  geom_smooth(se = FALSE)
```




```
#Higher visibility leads to shorter delay times

flight_weather %>%
  group_by(humid) %>%
  summarise(avg_delay = mean(dep_delay, na.rm = TRUE)) %>%
  ggplot(mapping = aes(x = humid, y = avg_delay)) +
  geom_point(alpha = 0.2) +
  geom_smooth(se = FALSE)
```




```
#Higher humidity at extreme values gives longer delay times.
```

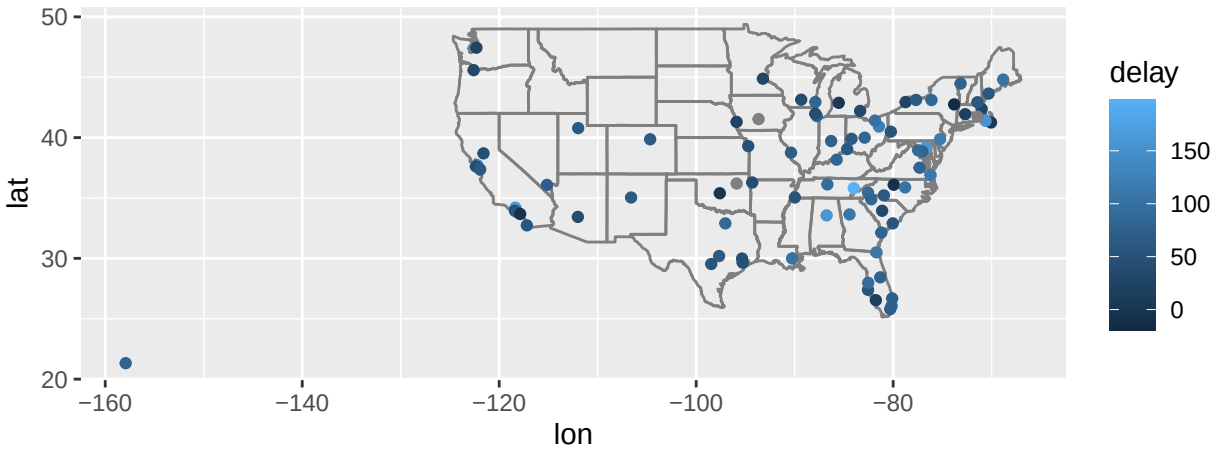
From this analysis, we see that extreme hot and cold, high precipitation, high wind speeds, low visibility, and higher humidity at extreme values tend to give longer delay times.

Exercise 11 (Website: 13.4.6 Ex. 5)

What happened on June 13, 2013? To find out, display the spatial pattern of delays on that date, and use Google to cross-reference with weather information.

```
june13_delays <- flights %>%
  filter(month == 6, day == 13) %>%
  group_by(dest) %>%
  summarise(delay = mean(arr_delay, na.rm = TRUE)) %>%
  inner_join(airports, by = c(dest = "faa"))

june13_delays %>%
  ggplot(aes(lon, lat, colour = delay)) +
  borders("state") +
  geom_point() +
  coord_quickmap()
```



On June 13, 2013, there was a large derecho that swept through the south-eastern U.S. We see huge delays in places like Tennessee, Alabama, Georgia, and Maryland, where the derecho casued the most damage.

Exercise 12 (Website: 13.5.1 Ex. 1)

What does it mean for a flight to have a missing `tailnum`? What do the tail numbers that don't have a matching record in `planes` have in common? (Hint: one variable explains ~90% of the problems.)

```
flights %>%
  filter(is.na(tailnum)) %>%
  select(month, day, tailnum, dep_time, arr_time)
```

```
## # A tibble: 2,512 x 5
##    month   day tailnum dep_time arr_time
##    <int> <int> <chr>      <int>    <int>
##  1     1     2 <NA>          NA       NA
##  2     1     2 <NA>          NA       NA
##  3     1     3 <NA>          NA       NA
##  4     1     3 <NA>          NA       NA
##  5     1     4 <NA>          NA       NA
##  6     1     4 <NA>          NA       NA
##  7     1     5 <NA>          NA       NA
##  8     1     7 <NA>          NA       NA
##  9     1     8 <NA>          NA       NA
## 10     1     9 <NA>          NA       NA
## # ... with 2,502 more rows
```

Flights with a missing tail number also have missing departure and arrive times, meaning they were cancelled.

```
flights %>%
  anti_join(planes, by = 'tailnum') %>%
  count(carrier) %>%
  arrange(desc(n))
```

```
## # A tibble: 10 x 2
##    carrier     n
##    <chr>   <int>
##  1 MQ      25397
##  2 AA      22558
```

```
##  3 UA       1693
##  4 9E       1044
##  5 B6        830
##  6 US        699
##  7 FL        187
##  8 DL        110
##  9 F9         50
## 10 WN         38
```

Seems like MQ and AA have by far the greatest number of missing tail numbers. After some googling this is because AA and MQ report by fleet number, not by tail number.

Exercise 13 (Website: 13.5.1 Ex. 2)

Filter `flights` to show only the flights of planes that flew at least 100 times.

```
more_than_100 <- flights %>%
  group_by(tailnum) %>%
  count() %>%
  filter(n > 100)

flights %>%
  right_join(more_than_100, by = 'tailnum') %>%
  select(tailnum, n)
```

```
## # A tibble: 229,202 x 2
##    tailnum     n
##    <chr>   <int>
##  1 N14228    111
##  2 N24211    130
##  3 N804JB    219
##  4 N39463    107
##  5 N516JB    288
##  6 N829AS    230
##  7 N593JB    294
##  8 N793JB    283
##  9 N657JB    285
## 10 N53441    102
## # ... with 229,192 more rows
```

Exercise 14 (Website: 13.5.1 Ex. 3)

Combine `fueleconomy::vehicles` and `fueleconomy::common` to find the records for only the most common models.

```
vehicles %>%
  semi_join(common, by = c('make', 'model'))
```

```
## # A tibble: 14,531 x 12
##       id make  model   year class   trans  drive     cyl displ fuel    hwy   cty
##    <dbl> <chr> <chr>  <dbl> <chr>   <chr>  <chr>   <dbl> <dbl> <chr> <dbl> <dbl>
##  1  1833 Acura Integ~  1986 Subcom~ Autom~ Front-~     4   1.6 Regu~    28    22
##  2  1834 Acura Integ~  1986 Subcom~ Manua~ Front-~     4   1.6 Regu~    28    23
##  3  3037 Acura Integ~  1987 Subcom~ Autom~ Front-~     4   1.6 Regu~    28    22
##  4  3038 Acura Integ~  1987 Subcom~ Manua~ Front-~     4   1.6 Regu~    28    23
```

```
##  5  4183 Acura Integ~  1988 Subcom~ Autom~ Front-~     4   1.6 Regu~    27    22
##  6  4184 Acura Integ~  1988 Subcom~ Manua~ Front-~     4   1.6 Regu~    28    23
##  7  5303 Acura Integ~  1989 Subcom~ Autom~ Front-~     4   1.6 Regu~    27    22
##  8  5304 Acura Integ~  1989 Subcom~ Manua~ Front-~     4   1.6 Regu~    28    23
##  9  6442 Acura Integ~  1990 Subcom~ Autom~ Front-~     4   1.8 Regu~    24    20
## 10  6443 Acura Integ~  1990 Subcom~ Manua~ Front-~     4   1.8 Regu~    26    21
## # ... with 14,521 more rows
```

Exercise 15 (Website: 13.5.1 Ex. 5)

What does `anti_join(flights, airports, by = c("dest" = "faa"))` tell you? What does `anti_join(airports, flights, by = c("faa" = "dest"))` tell you?

```
anti_join(flights, airports, by = c("dest" = "faa"))
```

```
## # A tibble: 7,602 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      544            545        -1     1004           1022
##  2  2013     1     1      615            615         0     1039           1100
##  3  2013     1     1      628            630        -2     1137           1140
##  4  2013     1     1      701            700         1     1123           1154
##  5  2013     1     1      711            715        -4     1151           1206
##  6  2013     1     1      820            820         0     1254           1310
##  7  2013     1     1      820            820         0     1249           1329
##  8  2013     1     1      840            845        -5     1311           1350
##  9  2013     1     1      909            810        59     1331           1315
## 10  2013     1     1      913            918        -5     1346           1416
## # ... with 7,592 more rows, and 11 more variables: arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

This tells you the airports that are in `flights` but not in `airports`.

```
anti_join(airports, flights, by = c("faa" = "dest"))
```

```
## # A tibble: 1,357 x 8
##    faa   name                      lat    lon   alt    tz dst   tzone
##    <chr> <chr>                   <dbl>  <dbl> <dbl> <dbl> <chr> <chr>
##  1 04G   Lansdowne Airport        41.1  -80.6  1044    -5 A     America/New_Yo~
##  2 06A   Moton Field Municipal A~ 32.5  -85.7   264    -6 A     America/Chicago
##  3 06C   Schaumburg Regional      42.0  -88.1   801    -6 A     America/Chicago
##  4 06N   Randall Airport          41.4  -74.4   523    -5 A     America/New_Yo~
##  5 09J   Jekyll Island Airport    31.1  -81.4    11    -5 A     America/New_Yo~
##  6 0A9   Elizabethton Municipal ~ 36.4  -82.2  1593    -5 A     America/New_Yo~
##  7 0G6   Williams County Airport  41.5  -84.5   730    -5 A     America/New_Yo~
##  8 0G7   Finger Lakes Regional A~ 42.9  -76.8   492    -5 A     America/New_Yo~
##  9 0P2   Shoestring Aviation Air~ 39.8  -76.6  1000    -5 U     America/New_Yo~
## 10 0S9   Jefferson County Intl    48.1 -123.    108    -8 A     America/Los_An~
## # ... with 1,347 more rows
```

This tells you the airports that are in `airports` but not in `flights`.

Exercise 16 (Website: 13.5.1 Ex. 6)

You might expect that there's an implicit relationship between plane and airline, because each plane is flown by a single airline. Confirm or reject this hypothesis using the tools you've learned above.

```
planes_with_mult_carriers <- flights %>%
  filter(!is.na(tailnum)) %>%
  distinct(tailnum, carrier) %>%
  count(tailnum, name = 'n_carriers') %>%
  filter(n_carriers > 1)


flights %>%
  semi_join(planes_with_mult_carriers) %>%
  select(tailnum, carrier) %>%
  count(tailnum, carrier, name = 'num_flights') %>%
  arrange(tailnum) %>%
  left_join(airlines) %>%
  select(tailnum, name, num_flights, carrier)
```

```
## # A tibble: 34 x 4
##    tailnum name                     num_flights carrier
##    <chr>   <chr>                          <int> <chr>
##  1 N146PQ  Endeavor Air Inc.                 8 9E
##  2 N146PQ  ExpressJet Airlines Inc.         36 EV
##  3 N153PQ  Endeavor Air Inc.                 5 9E
##  4 N153PQ  ExpressJet Airlines Inc.         26 EV
##  5 N176PQ  Endeavor Air Inc.                 7 9E
##  6 N176PQ  ExpressJet Airlines Inc.         21 EV
##  7 N181PQ  Endeavor Air Inc.                 4 9E
##  8 N181PQ  ExpressJet Airlines Inc.         35 EV
##  9 N197PQ  Endeavor Air Inc.                 2 9E
## 10 N197PQ  ExpressJet Airlines Inc.         31 EV
## # ... with 24 more rows
```

We see that Endeavor Air and ExpressJet Airlines 17 planes.

Challenge (Website: 13.5.1 Ex. 4) – NOT REQUIRED

Find the 48 hours (over the course of the whole year) that have the worst delays. Cross-reference with the `weather` data. Can you see any patterns?