Problem Set 3a

2023-12-08

Data import and tidying

1.

```r
library("tidyverse")
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
mtcars
```

```
##                      mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
## AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
```

```
## Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
## Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

We can check is an object is a tibble by using the function "is_tibble()"

```r
is_tibble(mtcars)
```

```
## [1] FALSE
```

2.

```r
df <- data.frame(abc = 1, xyz = "a")
df$x
```

```
## [1] "a"
```

```r
df[, "xyz"]
```

```
## [1] "a"
```

```r
df[, c("abc", "xyz")]
```

```
##   abc xyz
## 1   1   a
```

```r
table <- as_tibble(df)
table$x
```

```
## Warning: Unknown or uninitialised column: `x`.
```

```
## NULL
```

```r
table[, "xyz"]
```

```
## # A tibble: 1 x 1
##   xyz
##   <chr>
## 1 a
```

```r
table[, c("abc", "xyz")]
```

```
## # A tibble: 1 x 2
##     abc xyz
##   <dbl> <chr>
## 1     1 a
```

R has an interesting feature which is match the column based on what the first few letters are. Which is why df$x is shown as df$xyz. Although helpful, not the safest if you happen to be dealing with something important.

Generally, if we use "[" we get a vector returned if it's only column. If there's more than one column, then it will return a dataframe. So keeping this in mind is important.

Data import

1.

I would use the read_delim() function with the argument that I'm interested to separate the words with: read_delim(file_name, delim = "|")

2.

The most important argument for the function read_fwf() would be col_positions

3.

read_csv(file.csv, ",", quote = "”')

Parsing vectors

1.

```
# Commenting out to export the document but it does throw an error
# locale(decimal_mark = ".", grouping_mark = ".")
```

If decimal and grouping marks are set to same, then as we can see it will throw an error If decimal mark is set to the comma, then grouping mark will be set to the period

```
locale(decimal_mark = ",")
```

```
## <locale>
## Numbers:  123.456,78
## Formats:  %AD / %AT
## Timezone: UTC
## Encoding: UTF-8
## <date_names>
## Days:   Sunday (Sun), Monday (Mon), Tuesday (Tue), Wednesday (Wed), Thursday
##         (Thu), Friday (Fri), Saturday (Sat)
## Months: January (Jan), February (Feb), March (Mar), April (Apr), May (May),
##         June (Jun), July (Jul), August (Aug), September (Sep), October
##         (Oct), November (Nov), December (Dec)
## AM/PM:  AM/PM
```

But if grouping is set to a period, then decimal mark is set to a comma

```
locale(grouping_mark = ".")
```

```
## <locale>
## Numbers:  123.456,78
## Formats:  %AD / %AT
## Timezone: UTC
## Encoding: UTF-8
## <date_names>
## Days:   Sunday (Sun), Monday (Mon), Tuesday (Tue), Wednesday (Wed), Thursday
##         (Thu), Friday (Fri), Saturday (Sat)
## Months: January (Jan), February (Feb), March (Mar), April (Apr), May (May),
##         June (Jun), July (Jul), August (Aug), September (Sep), October
```

```
##         (Oct), November (Nov), December (Dec)
## AM/PM:   AM/PM
```

2.

Most common encodings in Europe are UTF-8 and ASCII. In Asia ISO and Windows standards are used. There are more such as JIS X 0208, GB 2312, and EUC-KR.

Spreading and gathering

1.

```r
stocks <- tibble(
year = c(2015, 2015, 2016, 2016),
half = c(1,2,1,2),
return = c(1.88, 0.59, 0.92, 0.17)
)
stocks %>%
spread(year, return) %>%
gather("year", "return", `2015`:`2016`)
```

```
## # A tibble: 4 x 3
##    half year  return
##   <dbl> <chr>  <dbl>
## 1     1 2015    1.88
## 2     2 2015    0.59
## 3     1 2016    0.92
## 4     2 2016    0.17
```

2.

The key variable is the column names, and is moved as a character column. So it doesn't make sense to treat column names as anything else. We could fix this with "convert = TRUE".

3.

Gather cannot find the columns values

4.

There is a redundant entry with Phillip Woods' age. We can fix it by using a separate column id that identify one of the ages as a unique entry.

5.

It's best to gather here based on gender

-> preg %>% gather(gender, values, -pregnant)

Separating and uniting

1.

x has vectors with 3 and 4 characters but we specify 3 columns. We can give a fourth column like below:

```
tibble(x = c("a,b,c", "d,e,f,g", "h,i,j")) %>%
  separate(x, c("one", "two", "three", "four"))
```

```
## Warning: Expected 4 pieces. Missing pieces filled with `NA` in 2 rows [1, 3].
```

```
## # A tibble: 3 x 4
##   one   two   three four
##   <chr> <chr> <chr> <chr>
## 1 a     b     c     <NA>
## 2 d     e     f     g
## 3 h     i     j     <NA>
```

```
tibble(x = c("a,b,c", "d,e,f,g", "h,i,j")) %>%
  separate(x, c("one", "two", "three", "four"), fill = "right")
```

```
## # A tibble: 3 x 4
##   one   two   three four
##   <chr> <chr> <chr> <chr>
## 1 a     b     c     <NA>
## 2 d     e     f     g
## 3 h     i     j     <NA>
```

2.

unite and separate have columns and create new ones. remove allows to get rid of original columns that we unite or separate on.

Missing Values

1.

In spread(), the NA values will be replaced with fill value. But in complete(), NAs can be replaced by different values. We can provide the fill argument with values to replace NAs with.

2.

The NA values are basically replaced by the next or previous non NA value.

Question 2: Relational data and data types

Relational data

1.

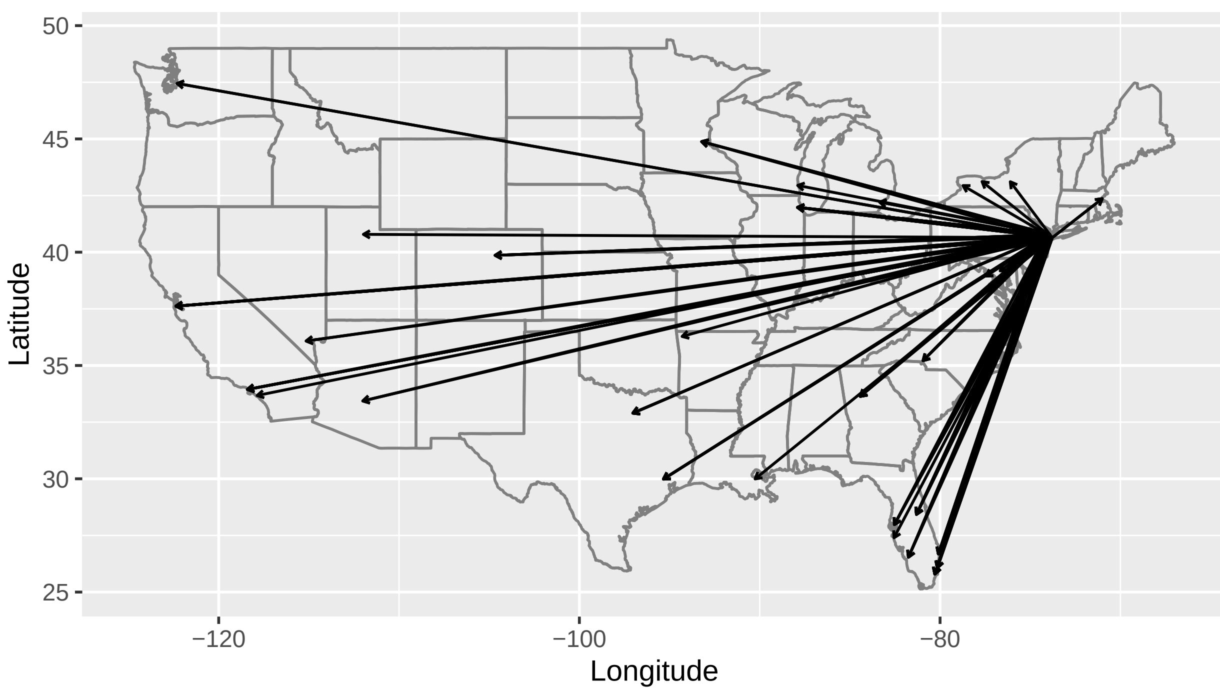
```r
library("tidyverse")
library("nycflights13")
library("viridis")
```

```
## Loading required package: viridisLite
```

This would require the variables latitude and longitude of the origin and destination airports of each flight. As for tables, we would require the flights and airports tables.

```r
flights_latlon <- flights %>%
  inner_join(select(airports, origin = faa, origin_lat = lat, origin_lon = lon),
    by = "origin"
  ) %>%
  inner_join(select(airports, dest = faa, dest_lat = lat, dest_lon = lon),
    by = "dest"
  )
```

```r
flights_latlon %>%
  slice(1:100) %>%
  ggplot(aes(
    x = origin_lon, xend = dest_lon,
    y = origin_lat, yend = dest_lat
  )) +
  borders("state") +
  geom_segment(arrow = arrow(length = unit(0.1, "cm"))) +
  coord_quickmap() +
  labs(y = "Latitude", x = "Longitude")
```



2. The airports$faaisaforeignkeyinweather$origin. The relationship in pictorial representation should look like a direct connection from the faa column of airports to the origin column in weather column.

3.

First of all, having weather for all airports would provide the weather for the destination of each flight. The additional relation it would define with flights would be that this would prove the weather at the destination airport at the time of the flight take off.

4.

```r
special_days <- tribble( ~year, ~month, ~day, ~holiday,
  2013, 01, 01, "New Years Day",
  2013, 11, 29, "Thanksgiving Day",
  2013, 12, 25, "Christmas Day"
)
special_days
```

```
## # A tibble: 3 x 4
##    year month   day holiday
##   <dbl> <dbl> <dbl> <chr>
## 1  2013     1     1 New Years Day
## 2  2013    11    29 Thanksgiving Day
## 3  2013    12    25 Christmas Day
```

Keys

1.

```r
flights %>%
  arrange(year, month, day, sched_dep_time, carrier, flight) %>%
  mutate(flight_id = row_number()) %>%
  glimpse()
```

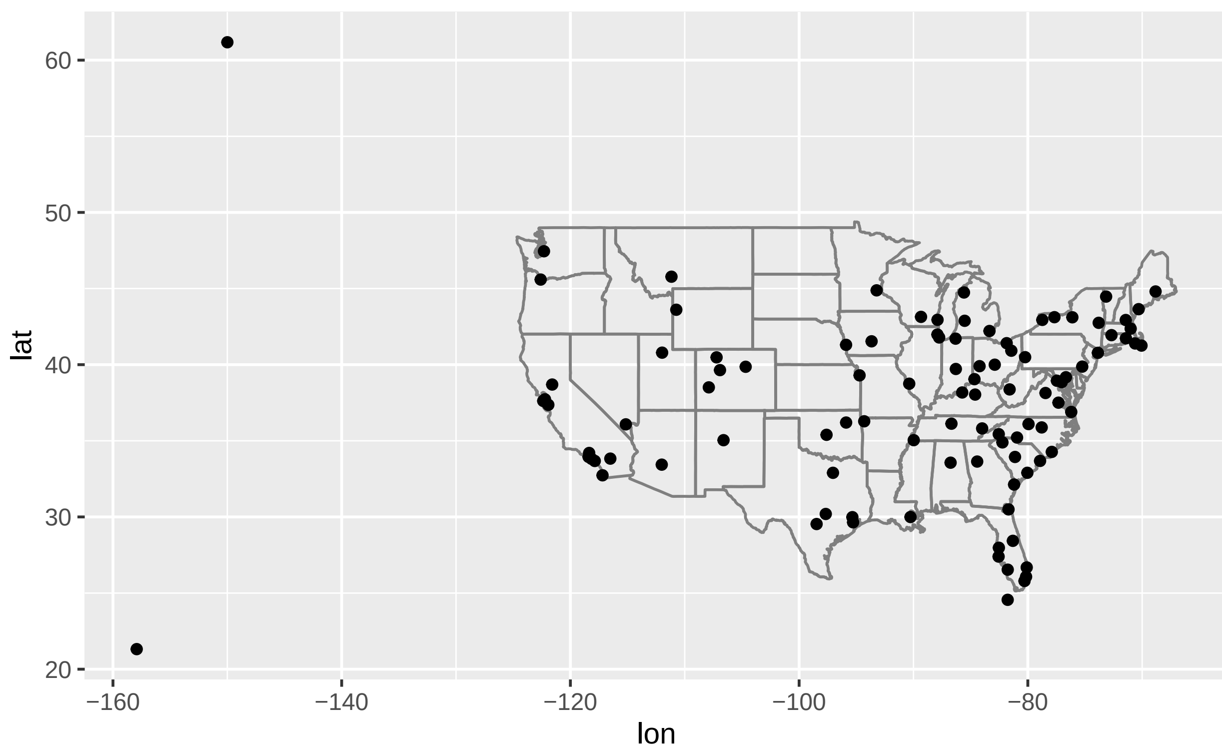
```
## Rows: 336,776
## Columns: 20
## $ year           <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2~
## $ month          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ day            <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ dep_time       <int> 517, 533, 542, 544, 554, 559, 558, 559, 558, 558, 557, ~
## $ sched_dep_time <int> 515, 529, 540, 545, 558, 559, 600, 600, 600, 600, 600, ~
## $ dep_delay      <dbl> 2, 4, 2, -1, -4, 0, -2, -1, -2, -2, -3, NA, 1, 0, -5, -~
## $ arr_time       <int> 830, 850, 923, 1004, 740, 702, 753, 941, 849, 853, 838,~
## $ sched_arr_time <int> 819, 830, 850, 1022, 728, 706, 745, 910, 851, 856, 846,~
## $ arr_delay      <dbl> 11, 20, 33, -18, 12, -4, 8, 31, -2, -3, -8, NA, -6, -7,~
## $ carrier        <chr> "UA", "UA", "AA", "B6", "UA", "B6", "AA", "AA", "B6", "~
## $ flight         <int> 1545, 1714, 1141, 725, 1696, 1806, 301, 707, 49, 71, 79~
## $ tailnum        <chr> "N14228", "N24211", "N619AA", "N804JB", "N39463", "N708~
## $ origin         <chr> "EWR", "LGA", "JFK", "JFK", "EWR", "JFK", "LGA", "LGA",~
## $ dest           <chr> "IAH", "IAH", "MIA", "BQN", "ORD", "BOS", "ORD", "DFW",~
## $ air_time       <dbl> 227, 227, 160, 183, 150, 44, 138, 257, 149, 158, 140, N~
```

```
## $ distance       <dbl> 1400, 1416, 1089, 1576, 719, 187, 733, 1389, 1028, 1005~
## $ hour           <dbl> 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6~
## $ minute         <dbl> 15, 29, 40, 45, 58, 59, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ time_hour      <dttm> 2013-01-01 05:00:00, 2013-01-01 05:00:00, 2013-01-01 0~
## $ flight_id      <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, ~
```

Mutating joins

1.

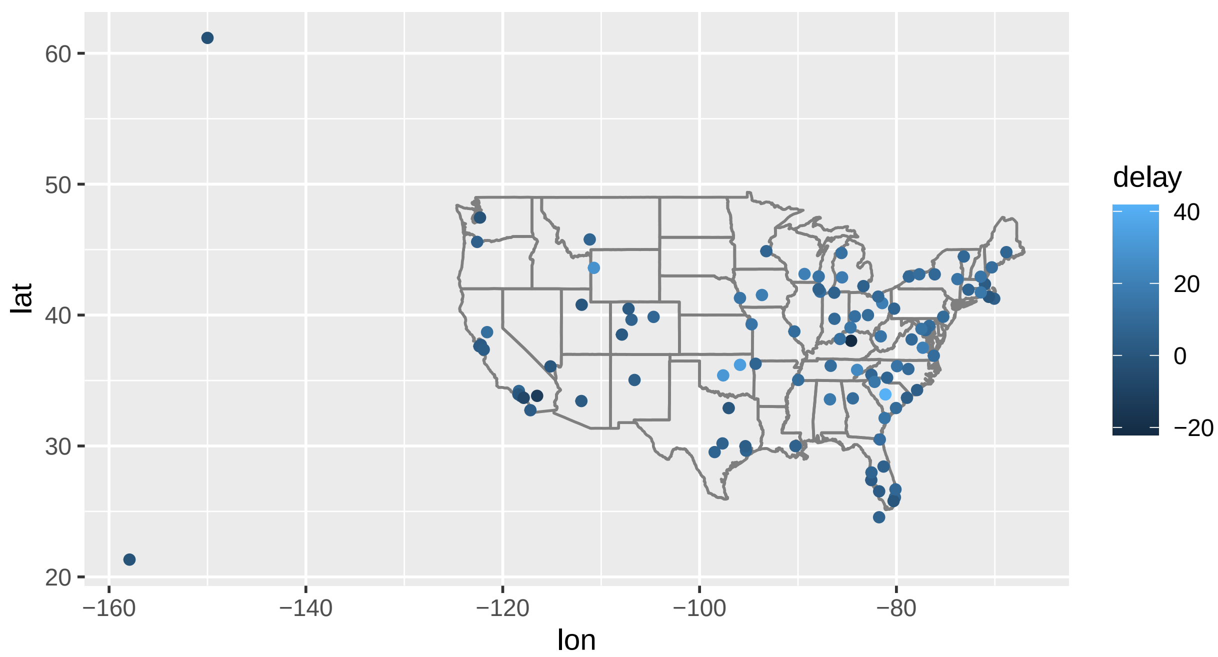
```
airports %>%
  semi_join(flights, c("faa" = "dest")) %>%
  ggplot(aes(lon, lat)) +
  borders("state") +
  geom_point() +
  coord_quickmap()
```




```
avg_dest_delays <-
  flights %>%
  group_by(dest) %>%
  # arrival delay NA's are cancelled flights
  summarise(delay = mean(arr_delay, na.rm = TRUE)) %>%
  inner_join(airports, by = c(dest = "faa"))

avg_dest_delays %>%
```

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



2.

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

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

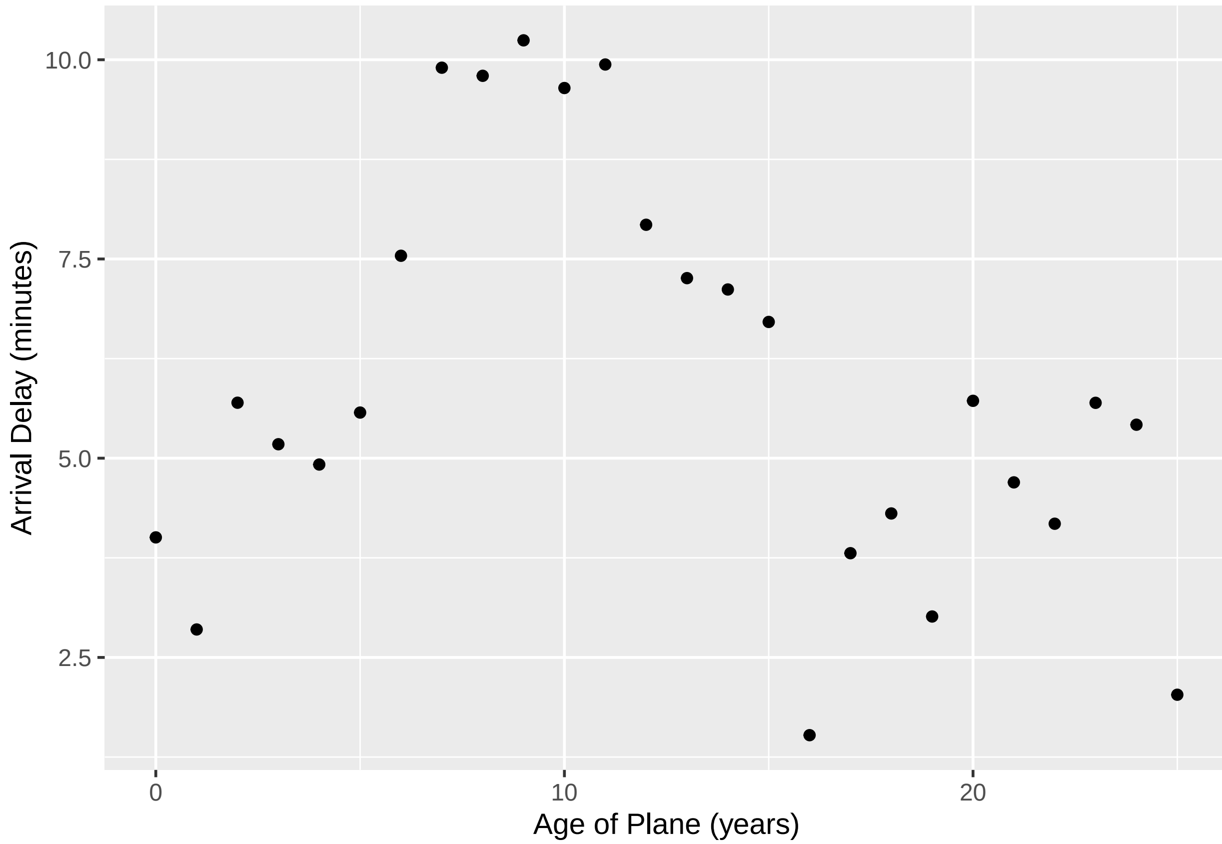
```
##  2  2013     1     1     5 LGA    IAH     40.8 -73.9  30.0 -95.3
##  3  2013     1     1     5 JFK    MIA     40.6 -73.8  25.8 -80.3
##  4  2013     1     1     5 JFK    BQN     40.6 -73.8  NA    NA
##  5  2013     1     1     6 LGA    ATL     40.8 -73.9  33.6 -84.4
##  6  2013     1     1     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
## # i 336,766 more rows
```

3.

We could try to find out by plotting a graph between delay and age of planes

```
plane_cohorts <- inner_join(flights,
  select(planes, tailnum, plane_year = year),
  by = "tailnum"
) %>%
  mutate(age = year - plane_year) %>%
  filter(!is.na(age)) %>%
  mutate(age = if_else(age > 25, 25L, age)) %>%
  group_by(age) %>%
  summarise(
    dep_delay_mean = mean(dep_delay, na.rm = TRUE),
    dep_delay_sd = sd(dep_delay, na.rm = TRUE),
    arr_delay_mean = mean(arr_delay, na.rm = TRUE),
    arr_delay_sd = sd(arr_delay, na.rm = TRUE),
    n_arr_delay = sum(!is.na(arr_delay)),
    n_dep_delay = sum(!is.na(dep_delay))
  )

ggplot(plane_cohorts, aes(x = age, y = arr_delay_mean)) +
  geom_point() +
  scale_x_continuous("Age of Plane (years)", breaks = seq(0, 30, by = 10)) +
  scale_y_continuous("Arrival Delay (minutes)")
```

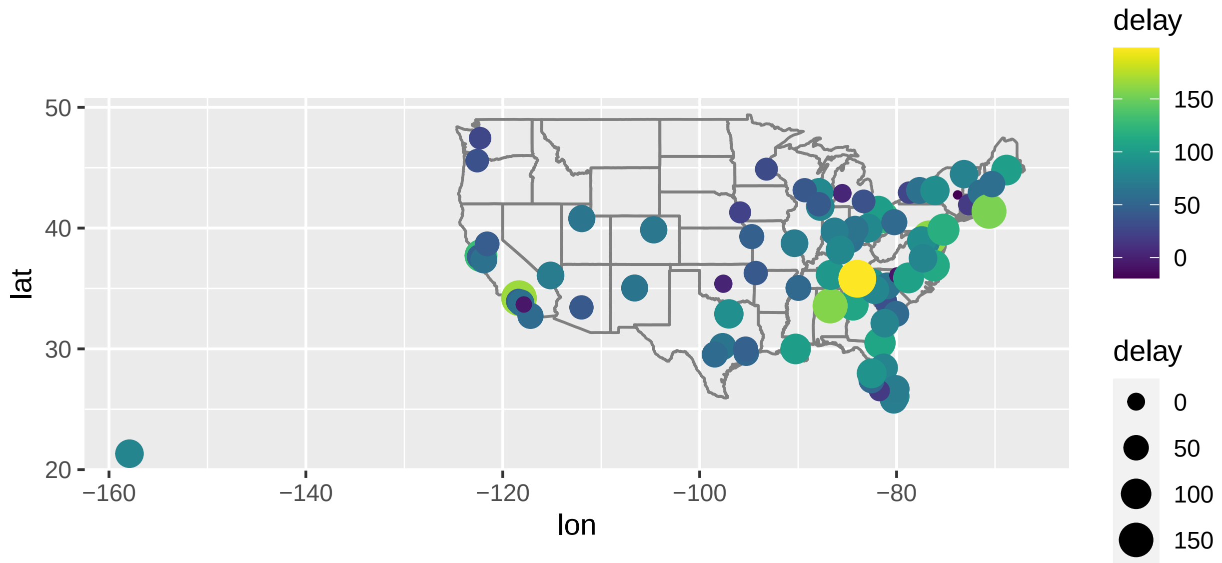


It looks like until 10 years, the delay is increasing and after that it's decreasing.

4.

```
flights %>%
  filter(year == 2013, month == 6, day == 13) %>%
  group_by(dest) %>%
  summarise(delay = mean(arr_delay, na.rm = TRUE)) %>%
  inner_join(airports, by = c("dest" = "faa")) %>%
  ggplot(aes(y = lat, x = lon, size = delay, colour = delay)) +
  borders("state") +
  geom_point() +
  coord_quickmap() +
  scale_colour_viridis()
```

```
## Warning: Removed 3 rows containing missing values (`geom_point()`).
```



According to Google, it looks like there were a lot of storms especially in the southeast and midwest.

Filtering Joins

1.

```
planes_gte100 <- flights %>%
  filter(!is.na(tailnum)) %>%
  group_by(tailnum) %>%
  count() %>%
  filter(n >= 100)
planes_gte100
```

```
## # A tibble: 1,217 x 2
## # Groups:   tailnum [1,217]
##    tailnum     n
##    <chr>   <int>
##  1 N0EGMQ    371
##  2 N10156    153
##  3 N10575    289
##  4 N11106    129
##  5 N11107    148
##  6 N11109    148
##  7 N11113    138
##  8 N11119    148
```

```
##   9 N11121    154
## 10 N11127    124
## # i 1,207 more rows
```

2.

```r
# Output being displayed when executing on R studio but not when knitting document for some odd reason

#weather_most_delayed <- semi_join(weather, day,
#                                   by = c("origin", "year",
#                                          "month", "day", "hour"))
#weather_most_delayed
```

It looks like a lot of these have an average wind speed of about 10 and pressure is mostly around 1000.

3.

```r
planes_carriers <-
  flights %>%
  filter(!is.na(tailnum)) %>%
  distinct(tailnum, carrier)

planes_carriers %>%
  count(tailnum) %>%
  filter(n > 1) %>%
  nrow()
```

```
## [1] 17
```

```r
carrier_transfer_tbl <- planes_carriers %>%
  group_by(tailnum) %>%
  filter(n() > 1) %>%
  left_join(airlines, by = "carrier") %>%
  arrange(tailnum, carrier)

carrier_transfer_tbl
```

```
## # A tibble: 34 x 3
## # Groups:   tailnum [17]
##    tailnum carrier name
##    <chr>   <chr>   <chr>
##  1 N146PQ  9E      Endeavor Air Inc.
##  2 N146PQ  EV      ExpressJet Airlines Inc.
##  3 N153PQ  9E      Endeavor Air Inc.
##  4 N153PQ  EV      ExpressJet Airlines Inc.
##  5 N176PQ  9E      Endeavor Air Inc.
##  6 N176PQ  EV      ExpressJet Airlines Inc.
##  7 N181PQ  9E      Endeavor Air Inc.
##  8 N181PQ  EV      ExpressJet Airlines Inc.
##  9 N197PQ  9E      Endeavor Air Inc.
## 10 N197PQ  EV      ExpressJet Airlines Inc.
## # i 24 more rows
```

We reject this hypothesis because we Endeavor Air and ExpressJet use same plane. That's enough to reject this hypothesis.

Strings

1.

```r
paste("abc", "xyz")
```

```
## [1] "abc xyz"
```

```r
paste0("abc", "xyz")
```

```
## [1] "abcxyz"
```

paste separates strings with a space and paste0 doesn't

2.

sep is used to add a string in between arguments. And collapse is used to separate elements from a character vector into a character vector of length one.

3.

It wraps text to fit within a certain width specified.

4.

str_trim() removes the whitespace from a string

Part 2: Your Project

Question 3

Project Proposal: Investigating the Impact of Abortion Rates on Crime Rates

Research Question: Our objective is to assess the robustness of Donohue and Levitt's (2001) study on the impact of legalized abortion on crime rates. Specifically, we aim to answer whether higher abortion rates lead to lower crime rates. We will replicate the original study, explore additional variables, and propose a model that provides a statistically reliable estimate of the causal effect of abortion on crime, considering potential confounders.

Data and Empirical Methods: 1. *Descriptive Statistics and Visualization:* - Obtain descriptive statistics for abortion rates, crime rates, and relevant variables. - Visualize trends over time, addressing concerns about data quality for Alaska, DC, and Hawaii. - Restrict the sample to 1985-1997.

2. *Replication of Donohue and Levitt (2001):*
   - Regress murder rates on abortion rates (a_murd), control variables, state fixed effects, and a linear time trend using OLS.
   - Assess significance and draw conclusions on the impact of abortion rates on murder rates.
3. *Additional Variables:*
   - Add variables suggested by the politician to the regression model from Question 2.
   - Evaluate changes in estimation results.
4. *LASSO Model:*

- Apply LASSO to the regression model from Question 2 to handle a large set of control variables.
- Analyze LASSO estimates and address concerns.

5. *Statistically Reliable Causal Effect Model:*
    - Propose a model that controls for confounders and provides a statistically reliable estimate of the causal effect of abortion on crime.
    - Walk through conceptual steps and code.
    - Compare results with the original Donohue and Levitt paper.
6. *Replication of Conservative Representatives' Analysis:*
    - Replace abortion rates with cellphone penetration rates in the model from Question 2.
    - Assess the conservative representatives' argument and provide counterarguments based on findings.

Data Management: - Systematically handle data read-in, ensuring code takes original data files as input. - Organize data sources using relational database concepts. - Implement systematic cleaning to check for errors, missing values, etc. - Address variable type correctness.

Preliminary Findings: - Preliminary analysis indicates a need to address concerns about data quality and potential changes in estimation results with additional variables. - Replication of the conservative representatives' analysis is essential to assess the plausibility of their argument.

Code Submission: - Provide organized code for data handling, combining sources, cleaning, and regression models.

Conclusion: Our research will contribute to the assessment of the original study's robustness and provide evidence for legislative initiatives related to abortion. The systematic approach to data management and empirical methods will ensure transparency and reliability in our findings.