Relational Data with dplyr Lab

Anthony Tetreault

2025-03-21

1. Identify the primary keys in the following datasets. Be sure to show that you have the primary key by showing there are no duplicate entries.

```
# Lahman::Batting
bat1 <- tibble(Lahman::Batting)

bat1 %>% count(playerID, yearID, stint) %>% filter(n>1)
```

```
## # A tibble: 0 x 4
## # i 4 variables: playerID <chr>, yearID <int>, stint <int>, n <int>
```

```
# Complex key of playerID + yearID + stint
# Doesn't include teamID because some players played on two teams in a year.
```

```
# babynames::babynames
babies <- tibble(babynames::babynames)

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

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

```
# Complex key of year + sex + name
```

```
# nasaweather::atmos
nw <- tibble(nasaweather::atmos)

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

```
## # A tibble: 0 x 5
## # i 5 variables: lat <dbl>, long <dbl>, year <int>, month <int>, n <int>
```

```
# Complex key of lat + long + year + month
```

2. What is the relationship between the "Batting", "Managers", and "Salaries" tables in the "Lahman" package? What are the keys for each dataset and how do they relate to each other?

- The primary keys for each dataset are:

  - Batting: (playerID, yearID, stint)
  - Managers: (playerID, yearID, teamID, inseason)
  - Salaries: (yearID, teamID, playerID)

- The relationships between the datasets are:
  - Batting connects to Salaries on (playerID, yearID, teamID)
  - Salaries connects to Managers on (playerID, yearID, teamID)
  - Managers does not directly connect to Batting on all three keys (playerID, teamID, yearID), but they do share the playerID and yearID keys.

3. Load the "nycflights13" library. Use an appropriate join to add a column containing the airline name to the "flights" dataset present in the library. Be sure to put the carrier code and name in the first two columns of the result so we can see them. Save the result as "flights2".

```
library(nycflights13)
flights2 <- flights %>%
    left_join(airlines, by = "carrier", keep = FALSE) %>%
    select(carrier, name, everything())
flights2
```

```
## # A tibble: 336,776 x 20
##    carrier name      year month   day dep_time sched_dep_time dep_delay arr_time
##    <chr>   <chr>    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1 UA      United ~  2013     1     1      517            515         2      830
##  2 UA      United ~  2013     1     1      533            529         4      850
##  3 AA      America~  2013     1     1      542            540         2      923
##  4 B6      JetBlue~  2013     1     1      544            545        -1     1004
##  5 DL      Delta A~  2013     1     1      554            600        -6      812
##  6 UA      United ~  2013     1     1      554            558        -4      740
##  7 B6      JetBlue~  2013     1     1      555            600        -5      913
##  8 EV      Express~  2013     1     1      557            600        -3      709
##  9 B6      JetBlue~  2013     1     1      557            600        -3      838
## 10 AA      America~  2013     1     1      558            600        -2      753
## # i 336,766 more rows
## # i 11 more variables: sched_arr_time <int>, arr_delay <dbl>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

4. Use an appropriate join to add the airport name to the "flights2" dataset you got above. The codes and names of the airports are in the "airports" dataset of the "nycflights13" package. Put the carrier and carrier name first followed by the destination and destination name, then everything else.

```
flights3 <- flights2 %>%
    left_join(airports %>% select(faa, name), join_by("dest" == "faa"), keep = FALSE) %>%
    rename(airline = name.x, dest.name = name.y) %>%
    select(carrier, airline, dest, dest.name, everything())
flights3
```

```
## # A tibble: 336,776 x 21
##    carrier airline     dest  dest.name  year month   day dep_time sched_dep_time
##    <chr>   <chr>       <chr> <chr>     <int> <int> <int>    <int>          <int>
```

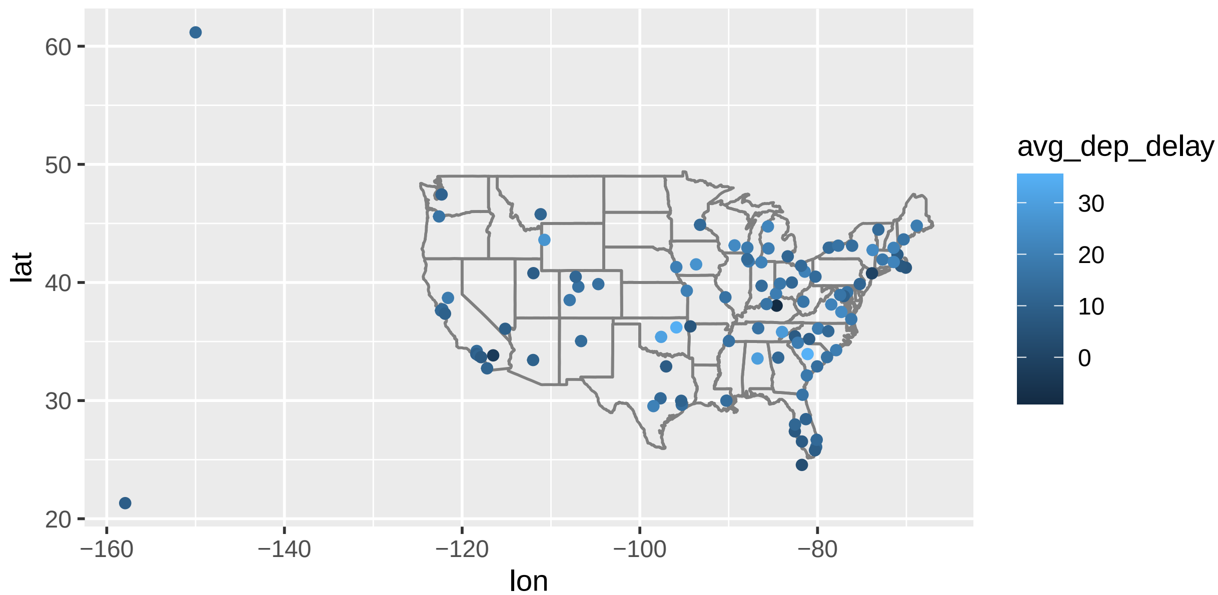
```
##    1 UA      United Air~ IAH   George B~  2013     1     1      517              515
##    2 UA      United Air~ IAH   George B~  2013     1     1      533              529
##    3 AA      American A~ MIA   Miami In~  2013     1     1      542              540
##    4 B6      JetBlue Ai~ BQN   <NA>       2013     1     1      544              545
##    5 DL      Delta Air ~ ATL   Hartsfie~  2013     1     1      554              600
##    6 UA      United Air~ ORD   Chicago ~  2013     1     1      554              558
##    7 B6      JetBlue Ai~ FLL   Fort Lau~  2013     1     1      555              600
##    8 EV      ExpressJet~ IAD   Washingt~  2013     1     1      557              600
##    9 B6      JetBlue Ai~ MCO   Orlando ~  2013     1     1      557              600
##   10 AA      American A~ ORD   Chicago ~  2013     1     1      558              600
## # i 336,766 more rows
## # i 12 more variables: dep_delay <dbl>, arr_time <int>, sched_arr_time <int>,
## #   arr_delay <dbl>, flight <int>, tailnum <chr>, origin <chr>, air_time <dbl>,
## #   distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

5. The "nycflights13" library and the code to create spatial map is provided for you. Now compute the average delay by destination, then join on the airports dataframe so you can show the spatial distribution of delays.

- Use the size or colour of the points to display the average delay for each airport.
- Add the location of the origin and destination (i.e. the lat and lon) to flights.
- Compute the average delay by destination.

```r
avg_delay_w_loc <- flights %>%
    select(dest, dep_delay, arr_delay) %>%
    group_by(dest) %>%
    summarize(avg_dep_delay = mean(dep_delay, na.rm = TRUE),
              avg_arr_delay = mean(arr_delay, na.rm = TRUE)
    )  %>%
    mutate(avg_dep_delay = ifelse(is.na(avg_dep_delay), 0, avg_dep_delay)) %>%
    left_join(airports, by = c("dest" = "faa"))
avg_delay_w_loc %>%
    ggplot(aes(lon,lat, color= avg_dep_delay)) +
    borders("state") +
    geom_point() +
    coord_quickmap()
```

```
## Warning: Removed 4 rows containing missing values or values outside the scale range
## (‘geom_point()‘).
```



6. Use a set operation function to find which airport codes from flights are not in the airports dataset.

```
flco <- unique(c(flights %>% pull("origin"), flights %>% pull("dest"))) # produce a vector to be used i
apco <- airports %>% pull(faa) # produce a vector to be used in setdiff
setdiff(flco, apco)
```

```
## [1] "BQN" "SJU" "STT" "PSE"
```