Lab 7: Relational Data

Sunghwu Song

March 19, 2021

Introduction

The main purpose of this lab is to practice data join skills from Chapter 10. The functions and their purposes are listed as follows:

- `inner_join()` Keeps observations appear in both datasets.
- `left_join()` Keeps all observations in left dataset.
- `right_join()` Keeps all observations in right dataset.
- `full_join()` Keeps all observations in both datasets.
- `semi_join()` Keeps all observations in left dataset that have a match in right dataset.
- `anti_join()` Drops all observations in left dataset that have a match in right dataset.

You will need to modify the code chunks so that the code works within each of chunk (usually this means modifying anything in ALL CAPS). You will also need to modify the code outside the code chunk. When you get the desired result for each step, change `Eval=F` to `Eval=T` and knit the document to HTML to make sure it works. After you complete the lab, you should submit your HTML file of what you have completed to Sakai before the deadline.

Excercises

Part 1

In part 1, you will practice the skills using the datasets from the R package `Lahman`. This database includes data related to baseball teams. It includes summary statistics about how the players performed on offense and defense for several years. It also includes personal information about the players.

The `Batting` data frame contains the offensive statistics for all players for many years. You can see, for example, the top 10 hitters in 2016 by running this code: (For more details of the dataset run `?Batting` in console.)

```
top <- Batting %>%
  filter(yearID == 2016) %>%
  arrange(desc(HR)) %>%
  slice(1:10)

top
```

```
##     playerID yearID stint teamID lgID   G  AB   R   H X2B X3B HR RBI SB CS BB
## 1  trumbma01   2016     1    BAL   AL 159 613  94 157  27   1 47 108  2  0 51
## 2   cruzne02   2016     1    SEA   AL 155 589  96 169  27   1 43 105  2  0 62
## 3  daviskh01   2016     1    OAK   AL 150 555  85 137  24   2 42 102  1  2 42
## 4  doziebr01   2016     1    MIN   AL 155 615 104 165  35   5 42  99 18  2 61
## 5  encared01   2016     1    TOR   AL 160 601  99 158  34   0 42 127  2  0 87
## 6  arenano01   2016     1    COL   NL 160 618 116 182  35   6 41 133  2  3 68
## 7  cartech02   2016     1    MIL   NL 160 549  84 122  27   1 41  94  3  1 76
## 8  frazito01   2016     1    CHA   AL 158 590  89 133  21   0 40  98 15  5 64
## 9  bryankr01   2016     1    CHN   NL 155 603 121 176  35   3 39 102  8  5 75
## 10  canoro01   2016     1    SEA   AL 161 655 107 195  33   2 39 103  0  1 47
##     SO IBB HBP SH SF GIDP
## 1  170   1   3  0  0   14
## 2  159   5   9  0  7   15
## 3  166   0   8  0  5   19
## 4  138   6   8  2  5   12
## 5  138   3   5  0  8   22
## 6  103  10   2  0  8   17
## 7  206   1   9  0 10   18
## 8  163   1   4  1  7   11
## 9  154   5  18  0  3    3
## 10 100   8   8  0  5   18
```

But who are these players? We see an ID, but not the names. The player names are in this table

```
head(Master,5)
```

```
##    playerID birthYear birthMonth birthDay birthCountry birthState  birthCity
## 1 aardsda01      1981         12       27          USA         CO     Denver
## 2 aaronha01      1934          2        5          USA         AL     Mobile
## 3 aaronto01      1939          8        5          USA         AL     Mobile
## 4  aasedo01      1954          9        8          USA         CA     Orange
## 5  abadan01      1972          8       25          USA         FL Palm Beach
##   deathYear deathMonth deathDay deathCountry deathState deathCity nameFirst
## 1        NA         NA       NA         <NA>       <NA>      <NA>     David
## 2        NA         NA       NA         <NA>       <NA>      <NA>      Hank
## 3      1984          8       16          USA         GA   Atlanta    Tommie
## 4        NA         NA       NA         <NA>       <NA>      <NA>       Don
## 5        NA         NA       NA         <NA>       <NA>      <NA>      Andy
##   nameLast      nameGiven weight height bats throws      debut  finalGame
## 1  Aardsma    David Allan    215     75    R      R 2004-04-06 2015-08-23
## 2    Aaron    Henry Louis    180     72    R      R 1954-04-13 1976-10-03
## 3    Aaron     Tommie Lee    190     75    R      R 1962-04-10 1971-09-26
## 4     Aase Donald William    190     75    R      R 1977-07-26 1990-10-03
## 5     Abad  Fausto Andres    184     73    L      L 2001-09-10 2006-04-13
##    retroID   bbrefID  deathDate  birthDate
## 1 aardd001 aardsda01       <NA> 1981-12-27
## 2 aaroh101 aaronha01       <NA> 1934-02-05
## 3 aarot101 aaronto01 1984-08-16 1939-08-05
## 4 aased001  aasedo01       <NA> 1954-09-08
## 5 abada001  abadan01       <NA> 1972-08-25
```

We can see column names `nameFirst` and `nameLast` in table `Master`.

Question 1

1. Use the `left_join` function to create a data frame called `top1`, which contains information of the 10 top home run hitters. The table should have the following columns: `playerID`, `nameFirst`, `nameLast`, and number of home runs (`HR`). (1 Point)

```
top1 = top %>%
  left_join(Master, by = c("playerID" = "playerID")) %>%
  select('playerID', 'nameFirst', 'nameLast', 'HR')

top1
```

```
##     playerID nameFirst    nameLast HR
## 1  trumbma01      Mark      Trumbo 47
## 2   cruzne02    Nelson        Cruz 43
## 3  daviskh01     Khris       Davis 42
## 4  doziebr01     Brian      Dozier 42
## 5  encared01     Edwin Encarnacion 42
## 6  arenano01     Nolan     Arenado 41
## 7  cartech02     Chris      Carter 41
## 8  frazito01      Todd     Frazier 40
## 9  bryankr01      Kris      Bryant 39
## 10  canoro01  Robinson        Cano 39
```

Question 2

Data `Salaries` contains the baseball player salary data.

```
head(Salaries,5)
```

```
##   yearID teamID lgID  playerID salary
## 1   1985    ATL   NL barkele01 870000
## 2   1985    ATL   NL bedrost01 550000
## 3   1985    ATL   NL benedbr01 545000
## 4   1985    ATL   NL  campri01 633333
## 5   1985    ATL   NL ceronri01 625000
```

2. You may be curious about the salaries of the top 10 hitters in 2016 (4 Points):

- Now create a new data frame called `top2` by adding top 10 hitters' salaries to `top1` and including only `nameFirst`, `nameLast`, `teamID`, `HR`, and `salary` columns.
- Rename the columns to `FirstName`, `LastName`, `Team`, `Homeruns` and `Salary` respectively.
- Arrange the data frame by `Salary` in descending order.

Note that salaries are different every year so make sure to filter for the year 2016. This time, please use `right_join` to complete the exercise.

```
top2 = Salaries %>%
  filter(yearID == 2016) %>%
  right_join(top1, by="playerID") %>%
  select('FirstName' = 'nameFirst', "LastName" = "nameLast", "Team" = "teamID", "Homeruns" = "HR", "Sal
```

```
  arrange(desc(Salary))

top2
```

```
##    FirstName    LastName Team Homeruns   Salary
## 1   Robinson        Cano  SEA       39 24000000
## 2     Nelson        Cruz  SEA       43 14250000
## 3      Edwin Encarnacion  TOR       42 10000000
## 4       Mark      Trumbo  BAL       47  9150000
## 5       Todd     Frazier  CHA       40  8250000
## 6      Nolan     Arenado  COL       41  5000000
## 7      Brian      Dozier  MIN       42  3000000
## 8      Chris      Carter  MIL       41  2500000
## 9       Kris      Bryant  CHN       39   652000
## 10     Khris       Davis  OAK       42   524500
```

Part 2

In this part, we will explore relational data from `nycflights13`, which contains four data frames related to the `flights` table that you used in previous assignments.

Question 3

Data `airports` gives information about each airport, such as latitude and longitude, identified by the `faa` airport code.

```
head(airports,5)
```

```
## # A tibble: 5 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_Y~
## 2 06A   Moton Field Municipal Airp~     32.5 -85.7   264    -6 A     America/Chica~
## 3 06C   Schaumburg Regional             42.0 -88.1   801    -6 A     America/Chica~
## 4 06N   Randall Airport                 41.4 -74.4   523    -5 A     America/New_Y~
## 5 09J   Jekyll Island Airport           31.1 -81.4    11    -5 A     America/New_Y~
```

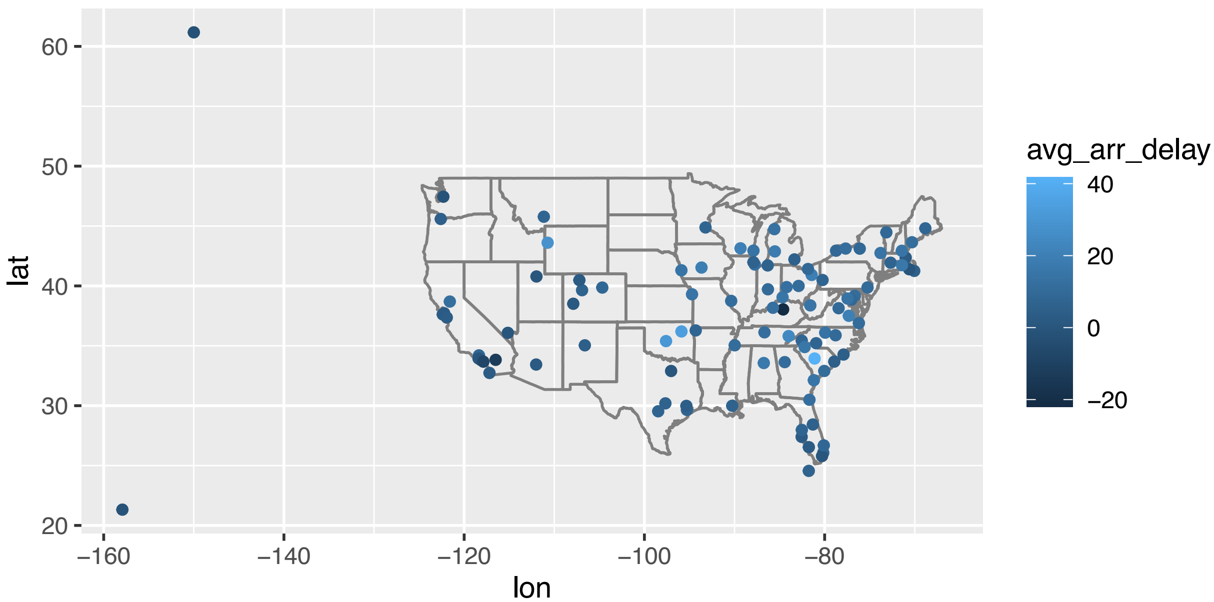
3. Based on `flights`, compute the average arrival delay by destination (`dest`) and ignore missing values, then join on the `airports` data frame then show the spatial distribution of delays. (3 Points)

```
delay = flights %>%
  group_by(dest) %>%
  summarise(avg_arr_delay=mean(arr_delay, na.rm = TRUE), .groups='drop') %>%
  inner_join(airports,by=c("dest" = "faa")) %>%
  select("avg_arr_delay", "lat", "lon")
```

Question 4

4. Draw a scatterplot with dots representing destination locations and colors of dots representing average arrival delay on US map. (4 Points) (For `coord_quickmap` to work, you need to install `maps` packages. If you haven't installed the package before, please run `install.packages('maps')` in Console.)

```r
delay %>%
  ggplot(aes(x = lon, y = lat, color=avg_arr_delay)) +
    borders("state") +
    geom_point() +
    coord_quickmap()
```



Question 5

Data `planes` gives information about each plane, identified by its `tailnum`. Note that `year` column in `planes` represents the year a plane was manufactured, which is different from `year` column in `flights`.

```r
head(planes,5)
```

```
## # A tibble: 5 x 9
##   tailnum  year type           manufacturer   model  engines seats speed engine
##   <chr>   <int> <chr>          <chr>          <chr>    <int> <int> <int> <chr>
## 1 N10156   2004 Fixed wing mu~ EMBRAER        EMB-1~       2    55    NA Turbo--
## 2 N102UW   1998 Fixed wing mu~ AIRBUS INDUST~ A320-~       2   182    NA Turbo--
## 3 N103US   1999 Fixed wing mu~ AIRBUS INDUST~ A320-~       2   182    NA Turbo--
## 4 N104UW   1999 Fixed wing mu~ AIRBUS INDUST~ A320-~       2   182    NA Turbo--
## 5 N10575   2002 Fixed wing mu~ EMBRAER        EMB-1~       2    55    NA Turbo--
```

5. Use the `planes` data to calculate the `age` of planes, assuming current year is 2013. Keep only `tailnum` and `age` in the output table `plane_ages`. (1 Point)

```r
plane_ages <-
  planes %>%
  mutate(age = 2013-year) %>%
  select(tailnum, age)
```

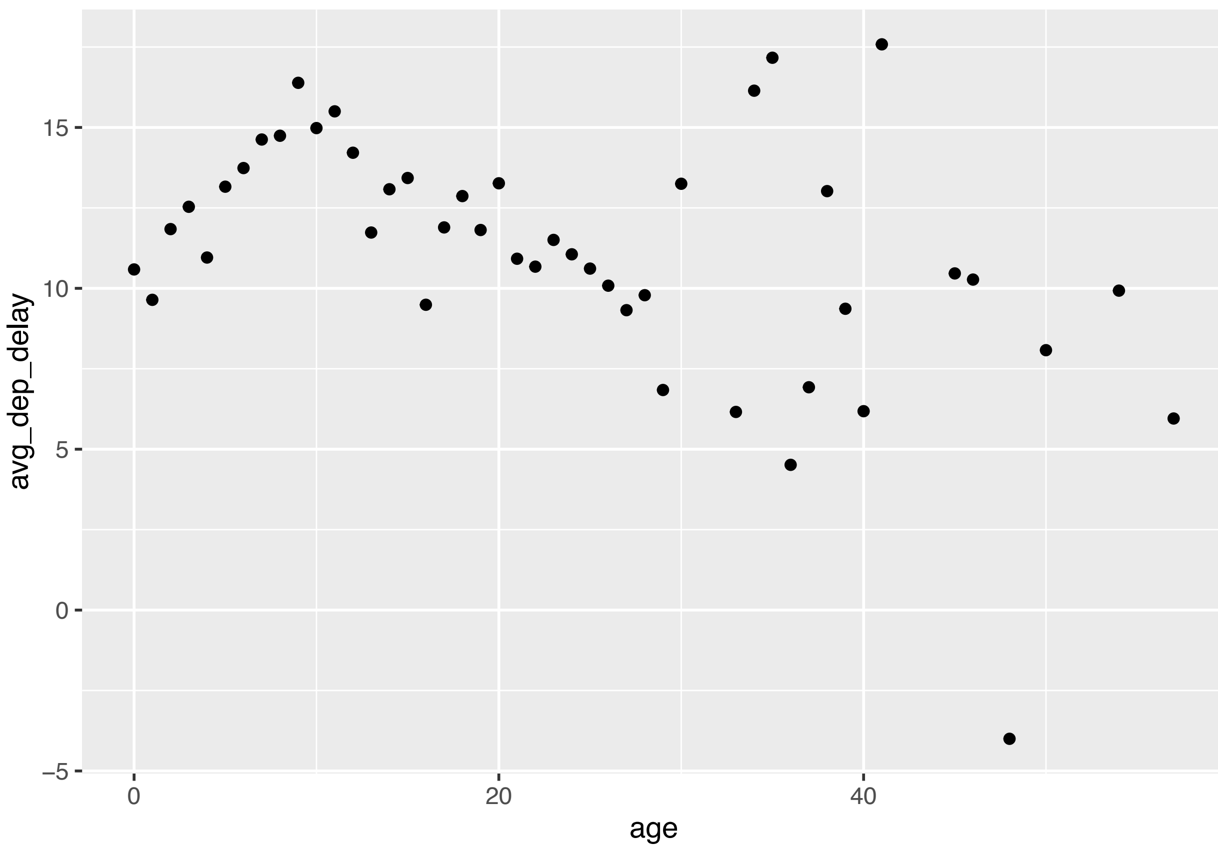
Question 6

6. Is there a relationship between the age of a plane and its delays? (4 Points)

- Join the `plane_ages` with `flights`, keeping observations with matches in both datasets.
- Summarize the average departure delay by plane `age` and ignore missing values.
- Draw a scatterplot of plane age vs. average departure delay.

```
plane_ages %>%
  inner_join(flights, by = 'tailnum') %>%
  group_by(age) %>%
  summarise(avg_dep_delay = mean(dep_delay, na.rm = TRUE), .groups='drop') %>%
  ggplot(aes(x = age, y = avg_dep_delay)) +
  geom_point()
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```



Question 7

7. What does it mean for a flight to have a missing `tailnum`? (1 Point)

```
flights %>%
  filter(is.na(tailnum))
```

```
## # A tibble: 2,512 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     2       NA           1545        NA       NA           1910
##  2  2013     1     2       NA           1601        NA       NA           1735
##  3  2013     1     3       NA            857        NA       NA           1209
##  4  2013     1     3       NA            645        NA       NA            952
##  5  2013     1     4       NA            845        NA       NA           1015
##  6  2013     1     4       NA           1830        NA       NA           2044
##  7  2013     1     5       NA            840        NA       NA           1001
##  8  2013     1     7       NA            820        NA       NA            958
##  9  2013     1     8       NA           1645        NA       NA           1838
## 10  2013     1     9       NA            755        NA       NA           1012
## # ... with 2,502 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>
```

Answer: Flights with missing tailnum have no airtime, arrival time, departial time, and other variables that relate to its flights being in the air. Thus the flights with missing tailnum were probably cancelled.

Question 8

8. 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. Check the documentation of `planes` for help.) (2 Points)

```
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
```

Answer: A majority of planes without missing tailnumbers belong to the MQ and AA carriers.