In-class worksheet 7

Feb 12, 2019

In this worksheet, we will use the libraries tidyverse and nycflights13:

```
library(tidyverse)
theme_set(theme_bw(base_size=12)) # set default ggplot2 theme
library(nycflights13)
```

The nycflights13 package contains information about all planes departing fron New York City in 2013.

1. Joining tables

The following two tables list the population size and area (in sq miles) of three major Texas cities each:

```
population <- read_csv(file =
"city,year,population
Houston,2014,2239558
San Antonio,2014,1436697
Austin,2014,912791
Austin,2010,790390")
population
```

```
## # A tibble: 4 x 3
##   city         year population
##   <chr>       <int>      <int>
## 1 Houston      2014    2239558
## 2 San Antonio  2014    1436697
## 3 Austin       2014     912791
## 4 Austin       2010     790390
```

```
area <- read_csv(file =
"city,area
Houston,607.5
Dallas,385.6
Austin,307.2")
area
```

```
## # A tibble: 3 x 2
##   city     area
##   <chr>   <dbl>
## 1 Houston  608.
## 2 Dallas   386.
## 3 Austin   307.
```

Combine these two tables using the functions `left_join()`, `right_join()`, and `inner_join()`. How do these join functions differ in their results?

```
left_join(population, area)
```

```
## Joining, by = "city"
```

```
## # A tibble: 4 x 4
##   city         year population  area
##   <chr>       <int>      <int> <dbl>
## 1 Houston      2014    2239558  608.
## 2 San Antonio  2014    1436697   NA
## 3 Austin       2014     912791  307.
## 4 Austin       2010     790390  307.
```

The function `left_join()` keeps the left table as is and fills in data from the right table where available. Missing values are listed as `NA`.

```
inner_join(population, area)
```

```
## Joining, by = "city"
```

```
## # A tibble: 3 x 4
##   city     year population  area
##   <chr>   <int>      <int> <dbl>
## 1 Houston  2014    2239558  608.
## 2 Austin   2014     912791  307.
## 3 Austin   2010     790390  307.
```

The function `inner_join()` only keeps the rows for which there is data in both tables.

```
right_join(population, area)
```

```
## Joining, by = "city"
```

```
## # A tibble: 4 x 4
##   city     year population  area
##   <chr>   <int>      <int> <dbl>
## 1 Houston  2014    2239558  608.
## 2 Dallas     NA         NA  386.
## 3 Austin   2014     912791  307.
## 4 Austin   2010     790390  307.
```

The functioin `right_join()` keeps the right table as is and fills in data form the left table where available. It is equivalent to `left_join()` with the arguments listed in the opposite order:

```
left_join(area, population)
```

```
## Joining, by = "city"
```

```
## # A tibble: 4 x 4
##   city     area  year population
##   <chr>   <dbl> <int>      <int>
## 1 Houston  608.  2014    2239558
## 2 Dallas   386.    NA         NA
## 3 Austin   307.  2014     912791
## 4 Austin   307.  2010     790390
```

2. Relationship between arrival delay and age of plane

The table `flights` from nycflights13 contains information about individual departures:

```
flights
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1      517            515         2      830
##  2  2013     1     1      533            529         4      850
##  3  2013     1     1      542            540         2      923
##  4  2013     1     1      544            545        -1     1004
##  5  2013     1     1      554            600        -6      812
##  6  2013     1     1      554            558        -4      740
##  7  2013     1     1      555            600        -5      913
##  8  2013     1     1      557            600        -3      709
##  9  2013     1     1      557            600        -3      838
## 10  2013     1     1      558            600        -2      753
## # … with 336,766 more rows, and 12 more variables: sched_arr_time <int>,
## #   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>
```

Individual planes are indicated by their tail number (`tailnum` in the table). Calculate the mean arrival delay (`arr_delay`) for each tail number. Do you notice anything unusual in the result? Try to calculate the mean with and without adding the option `na.rm = TRUE` .

```
# without na.rm = TRUE:
flights %>%
  group_by(tailnum) %>%
  summarize(mean_delay = mean(arr_delay))
```

```
## # A tibble: 4,044 x 2
##    tailnum mean_delay
##    <chr>        <dbl>
##  1 <NA>        NA
##  2 D942DN      31.5
##  3 N0EGMQ      NA
##  4 N10156      NA
##  5 N102UW       2.94
##  6 N103US      -6.93
##  7 N104UW      NA
##  8 N10575      NA
##  9 N105UW      -0.267
## 10 N107US      -5.73
## # … with 4,034 more rows
```

```
# with na.rm = TRUE:
flights %>%
  group_by(tailnum) %>%
  summarize(mean_delay = mean(arr_delay, na.rm = TRUE))
```

```
## # A tibble: 4,044 x 2
##    tailnum mean_delay
##    <chr>        <dbl>
##  1 <NA>       NaN
##  2 D942DN      31.5
##  3 N0EGMQ       9.98
##  4 N10156      12.7
##  5 N102UW       2.94
##  6 N103US      -6.93
##  7 N104UW       1.80
##  8 N10575      20.7
##  9 N105UW      -0.267
## 10 N107US      -5.73
## # … with 4,034 more rows
```

The option `na.rm = TRUE` removes missing values before averaging. Without this option, many of the averages end up as missing values (`NA`).

Information about individual planes is availabe in the table `planes` :

```
planes
```

```
## # A tibble: 3,322 x 9
##    tailnum  year type       manufacturer  model  engines seats speed engine
##    <chr>   <int> <chr>      <chr>         <chr>    <int> <int> <int> <chr>
##  1 N10156   2004 Fixed win… EMBRAER       EMB-1…       2    55    NA Turbo…
##  2 N102UW   1998 Fixed win… AIRBUS INDUS… A320-…       2   182    NA Turbo…
##  3 N103US   1999 Fixed win… AIRBUS INDUS… A320-…       2   182    NA Turbo…
##  4 N104UW   1999 Fixed win… AIRBUS INDUS… A320-…       2   182    NA Turbo…
##  5 N10575   2002 Fixed win… EMBRAER       EMB-1…       2    55    NA Turbo…
##  6 N105UW   1999 Fixed win… AIRBUS INDUS… A320-…       2   182    NA Turbo…
##  7 N107US   1999 Fixed win… AIRBUS INDUS… A320-…       2   182    NA Turbo…
##  8 N108UW   1999 Fixed win… AIRBUS INDUS… A320-…       2   182    NA Turbo…
##  9 N109UW   1999 Fixed win… AIRBUS INDUS… A320-…       2   182    NA Turbo…
## 10 N110UW   1999 Fixed win… AIRBUS INDUS… A320-…       2   182    NA Turbo…
## # … with 3,312 more rows
```

In particular, this table lists the year each individual plane was manufactured. Make a combined table that holds tail number, mean arrival delay, and year of manufacture for each plane. Then plot mean arrival delay vs. year of manufacture.

```
delay_year <-
  flights %>%
  group_by(tailnum) %>%
  summarize(mean_delay = mean(arr_delay, na.rm = TRUE)) %>% # calculate mean de
lay
  left_join(planes) %>% # combine with planes
  select(tailnum, mean_delay, year)
```

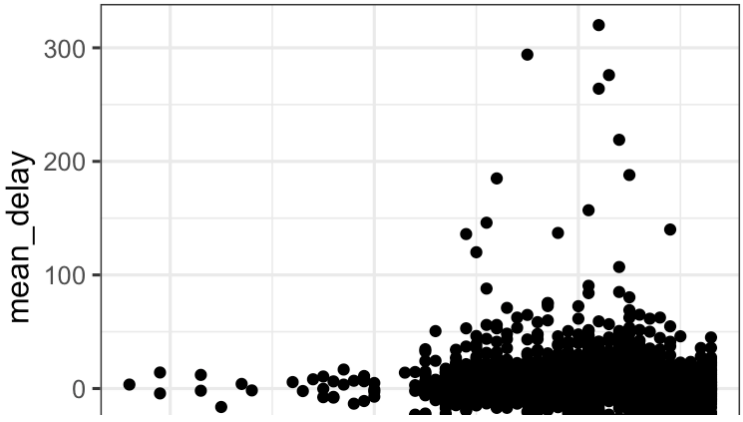
```
## Joining, by = "tailnum"
```

```
delay_year
```

```
## # A tibble: 4,044 x 3
##    tailnum mean_delay  year
##    <chr>        <dbl> <int>
##  1 <NA>       NaN        NA
##  2 D942DN      31.5      NA
##  3 N0EGMQ       9.98     NA
##  4 N10156      12.7    2004
##  5 N102UW       2.94   1998
##  6 N103US      -6.93   1999
##  7 N104UW       1.80   1999
##  8 N10575      20.7    2002
##  9 N105UW      -0.267  1999
## 10 N107US      -5.73   1999
## # … with 4,034 more rows
```

```
ggplot(delay_year, aes(x = year, y = mean_delay)) +
  geom_point()
```

```
## Warning: Removed 798 rows containing missing values (geom_point).
```



3. Relationship between arrival delay and temperature

Now calculate the mean arrival delay for each day of the year, and store in a variable called `daily_delays`.

```
daily_delays <-
  flights %>%
  group_by(year, month, day) %>%
  summarize(mean_delay = mean(arr_delay, na.rm = TRUE))
daily_delays
```

```
## # A tibble: 365 x 4
## # Groups:   year, month [12]
##     year month   day mean_delay
##    <int> <int> <int>      <dbl>
##  1  2013     1     1     12.7
##  2  2013     1     2     12.7
##  3  2013     1     3      5.73
##  4  2013     1     4     -1.93
##  5  2013     1     5     -1.53
##  6  2013     1     6      4.24
##  7  2013     1     7     -4.95
##  8  2013     1     8     -3.23
##  9  2013     1     9     -0.264
## 10  2013     1    10     -5.90
## # … with 355 more rows
```

We want to correlate these delay values with the temperature of each day. The data frame `weather` holds temperature measurements for each hour of each day:

```
weather
```

```
## # A tibble: 26,115 x 15
##    origin  year month   day  hour  temp  dewp humid wind_dir wind_speed
##    <chr>  <dbl> <dbl> <int> <int> <dbl> <dbl> <dbl>    <dbl>      <dbl>
##  1 EWR     2013     1     1     1  39.0  26.1  59.4      270      10.4
##  2 EWR     2013     1     1     2  39.0  27.0  61.6      250       8.06
##  3 EWR     2013     1     1     3  39.0  28.0  64.4      240      11.5
##  4 EWR     2013     1     1     4  39.9  28.0  62.2      250      12.7
##  5 EWR     2013     1     1     5  39.0  28.0  64.4      260      12.7
##  6 EWR     2013     1     1     6  37.9  28.0  67.2      240      11.5
##  7 EWR     2013     1     1     7  39.0  28.0  64.4      240      15.0
##  8 EWR     2013     1     1     8  39.9  28.0  62.2      250      10.4
##  9 EWR     2013     1     1     9  39.9  28.0  62.2      260      15.0
## 10 EWR     2013     1     1    10  41    28.0  59.6      260      13.8
## # … with 26,105 more rows, and 5 more variables: wind_gust <dbl>,
## #   precip <dbl>, pressure <dbl>, visib <dbl>, time_hour <dttm>
```

First, calculate the mean temperature for each day, and store in a variable called `mean_temp` :

```
mean_temp <-
  weather %>%
  group_by(year, month, day) %>%
  summarize(mean_temp = mean(temp))
mean_temp
```

```
## # A tibble: 364 x 4
## # Groups:   year, month [12]
##     year month   day mean_temp
##    <dbl> <dbl> <int>     <dbl>
##  1  2013     1     1      37.0
##  2  2013     1     2      28.7
##  3  2013     1     3      30.0
##  4  2013     1     4      34.9
##  5  2013     1     5      37.2
##  6  2013     1     6      40.1
##  7  2013     1     7      40.6
##  8  2013     1     8      40.1
##  9  2013     1     9      43.2
## 10  2013     1    10      43.8
## # … with 354 more rows
```

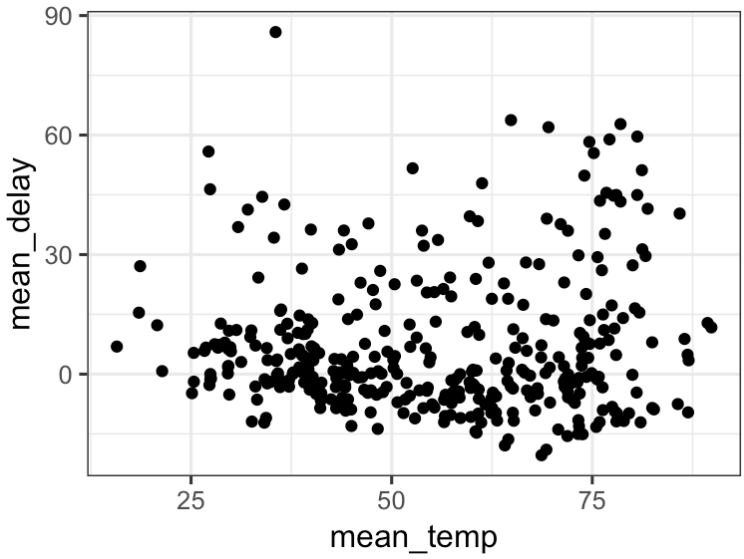
Now combine the mean delay and the mean temperature into one table, and then plot mean delay vs. mean temperature.

```
delay_temp <-
  daily_delays %>%
  left_join(mean_temp) %>%
  select(year, month, day, mean_delay, mean_temp)
```

```
## Joining, by = c("year", "month", "day")
```

```
ggplot(delay_temp, aes(x = mean_temp, y = mean_delay)) +
  geom_point()
```

```
## Warning: Removed 2 rows containing missing values (geom_point).
```



It looks like there is no strong relationship between daily temperature and mean delay.

4. If this was easy

Find out for how many tail numbers in the `flights` data set we have no information in the `planes` data set. What do we have to pay attention to when joining the `flights` and `planes` tables?

```
flights %>%
  left_join(planes, by = "tailnum") %>%
  filter(is.na(type)) %>%
  tally()
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1 52606
```

There are 52606 such flights. It is important here to tell the `left_join()` function to join by `tailnum`, otherwise it tries to join by `tailnum` and `year`, but `year` has different meanings in the two tables.

Calculate the mean arrival delay by plane model and by plane engine. Sort in order of descending mean delay. Remove all tailnumbers for which no plane information is available.

```
# 1. plane model
# we first join the fligths table and the planes table to make a new table hold
ing plane model, engine, and arrival delay
delay_table <-
  flights %>%
  left_join(planes, by = "tailnum") %>%
  filter(!is.na(type)) %>%
  select(model, engine, arr_delay)
delay_table
```

```
## # A tibble: 284,170 x 3
##    model       engine    arr_delay
##    <chr>       <chr>         <dbl>
##  1 737-824     Turbo-fan        11
##  2 737-824     Turbo-fan        20
##  3 757-223     Turbo-fan        33
##  4 A320-232    Turbo-fan       -18
##  5 757-232     Turbo-fan       -25
##  6 737-924ER   Turbo-fan        12
##  7 A320-232    Turbo-fan        19
##  8 CL-600-2B19 Turbo-fan       -14
##  9 A320-232    Turbo-fan        -8
## 10 A320-232    Turbo-fan        -2
## # … with 284,160 more rows
```

```
# we next calculate the mean delay per model
model_delay <-
  delay_table %>%
  group_by(model) %>%
  summarize(mean_delay = mean(arr_delay, na.rm = TRUE)) %>%
  arrange(desc(mean_delay))
model_delay
```

```
## # A tibble: 127 x 2
##    model    mean_delay
##    <chr>         <dbl>
##  1 747-451       120
##  2 757-351        72.5
##  3 A330-223       46.5
##  4 G-IV           41.2
##  5 777-224        40.8
##  6 A319-115       33.5
##  7 A109E          30.6
##  8 A340-313       29.8
##  9 MD-90-30       28.5
## 10 737-76N        28.4
## # … with 117 more rows
```

```
# 2. plane engine
# we go back to the delay_table we created above, and now calculate the mean pe
r engine
engine_delay <-
  delay_table %>%
  group_by(engine) %>%
  summarize(mean_delay = mean(arr_delay, na.rm = TRUE)) %>%
  arrange(desc(mean_delay))
engine_delay
```

```
## # A tibble: 6 x 2
##   engine        mean_delay
##   <chr>              <dbl>
## 1 4 Cycle             9.72
## 2 Turbo-shaft         9.28
## 3 Turbo-fan           7.72
## 4 Reciprocating       5.72
## 5 Turbo-prop          4.89
## 6 Turbo-jet           3.19
```