AE 04: NYC flights + data wrangling

Gracie Carlaw

```r
library(tidyverse)
library(nycflights13)
```

Exercise 1

Your turn: Fill in the blanks:

The `flights` data frame has 336776 rows. Each row represents a different flight.

Exercise 2

Your turn: What are the names of the variables in `flights`.

```r
names(flights)
```

```
 [1] "year"           "month"          "day"            "dep_time"      
 [5] "sched_dep_time" "dep_delay"      "arr_time"       "sched_arr_time"
 [9] "arr_delay"      "carrier"        "flight"         "tailnum"       
[13] "origin"         "dest"           "air_time"       "distance"      
[17] "hour"           "minute"         "time_hour"     
```

Exercise 3 - `select()`

- Make a data frame that only contains the variables `dep_delay` and `arr_delay`.

```r
select(flights, "dep_delay", "arr_delay")
```

```
# A tibble: 336,776 x 2
   dep_delay arr_delay
       <dbl>     <dbl>
 1         2        11
 2         4        20
 3         2        33
 4        -1       -18
 5        -6       -25
 6        -4        12
 7        -5        19
 8        -3       -14
 9        -3        -8
10        -2         8
# i 336,766 more rows
```

- Make a data frame that keeps every variable except `dep_delay`.

```
select(flights, -dep_delay)
```

```
# A tibble: 336,776 x 18
    year month   day dep_time sched_dep_time arr_time sched_arr_time arr_delay
   <int> <int> <int>    <int>          <int>    <int>          <int>     <dbl>
 1  2013     1     1      517            515      830            819        11
 2  2013     1     1      533            529      850            830        20
 3  2013     1     1      542            540      923            850        33
 4  2013     1     1      544            545     1004           1022       -18
 5  2013     1     1      554            600      812            837       -25
 6  2013     1     1      554            558      740            728        12
 7  2013     1     1      555            600      913            854        19
 8  2013     1     1      557            600      709            723       -14
 9  2013     1     1      557            600      838            846        -8
10  2013     1     1      558            600      753            745         8
# i 336,766 more rows
# i 10 more variables: carrier <chr>, flight <int>, tailnum <chr>,
#   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
#   minute <dbl>, time_hour <dttm>
```

- Make a data frame that includes all variables between `year` through `dep_delay` (inclusive). These are all variables that provide information about the departure of each flight.

```
select(flights, year:dep_delay)
```

```
# A tibble: 336,776 x 6
    year month   day dep_time sched_dep_time dep_delay
   <int> <int> <int>    <int>          <int>     <dbl>
 1  2013     1     1      517            515         2
 2  2013     1     1      533            529         4
 3  2013     1     1      542            540         2
 4  2013     1     1      544            545        -1
 5  2013     1     1      554            600        -6
 6  2013     1     1      554            558        -4
 7  2013     1     1      555            600        -5
 8  2013     1     1      557            600        -3
 9  2013     1     1      557            600        -3
10  2013     1     1      558            600        -2
# i 336,766 more rows
```

- Use the `select` helper `contains()` to make a data frame that includes the variables associated with the arrival, i.e., contains the string `"arr\_"` in the name.

```
flights %>%
select(contains("arr"))
```

```
# A tibble: 336,776 x 4
   arr_time sched_arr_time arr_delay carrier
      <int>          <int>     <dbl> <chr>
 1      830            819        11 UA
 2      850            830        20 UA
 3      923            850        33 AA
 4     1004           1022       -18 B6
 5      812            837       -25 DL
 6      740            728        12 UA
 7      913            854        19 B6
 8      709            723       -14 EV
 9      838            846        -8 B6
10      753            745         8 AA
# i 336,766 more rows
```

Exercise 4 - `slice()`

- Display the first five rows of the `flights` data frame.

```r
flights %>%
  slice_head(n = 5)
```

```
# A tibble: 5 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      517            515         2      830            819
2  2013     1     1      533            529         4      850            830
3  2013     1     1      542            540         2      923            850
4  2013     1     1      544            545        -1     1004           1022
5  2013     1     1      554            600        -6      812            837
# i 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>
```

- Display the last two rows of the `flights` data frame.

```r
flights %>%
  slice_tail(n = 2)
```

```
# A tibble: 2 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     9    30       NA           1159        NA       NA           1344
2  2013     9    30       NA            840        NA       NA           1020
# i 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>
```

Exercise 5 - `arrange()`

- Let's arrange the data by departure delay, so the flights with the shortest departure delays will be at the top of the data frame.

```r
flights %>%
  arrange(dep_delay)
```

```
# A tibble: 336,776 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    12     7     2040           2123       -43       40           2352
 2  2013     2     3     2022           2055       -33     2240           2338
 3  2013    11    10     1408           1440       -32     1549           1559
 4  2013     1    11     1900           1930       -30     2233           2243
 5  2013     1    29     1703           1730       -27     1947           1957
 6  2013     8     9      729            755       -26     1002            955
 7  2013    10    23     1907           1932       -25     2143           2143
 8  2013     3    30     2030           2055       -25     2213           2250
 9  2013     3     2     1431           1455       -24     1601           1631
10  2013     5     5      934            958       -24     1225           1309
# i 336,766 more rows
# i 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>
```

- Question: What does it mean for the `dep_delay` to have a negative value?

It likely means that the flight departed early.

- Arrange the data by descending departure delay, so the flights with the longest departure delays will be at the top.

```r
flights %>%
  arrange(desc(dep_delay))
```

```
# A tibble: 336,776 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     9      641            900      1301     1242           1530
 2  2013     6    15     1432           1935      1137     1607           2120
 3  2013     1    10     1121           1635      1126     1239           1810
 4  2013     9    20     1139           1845      1014     1457           2210
 5  2013     7    22      845           1600      1005     1044           1815
 6  2013     4    10     1100           1900       960     1342           2211
 7  2013     3    17     2321            810       911      135           1020
 8  2013     6    27      959           1900       899     1236           2226
 9  2013     7    22     2257            759       898      121           1026
10  2013    12     5      756           1700       896     1058           2020
# i 336,766 more rows
# i 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>
```

- **Your turn:** Create a data frame that only includes the plane tail number (`tailnum`), carrier (`carrier`), and departure delay for the flight with the longest departure delay. What is the plane tail number (`tailnum`) for this flight?

```r
flights %>%
  select("tailnum", "carrier") %>%
  arrange(desc("dep_delay")) %>%
  slice_head(n = 1)
```

```
# A tibble: 1 x 2
  tailnum carrier
  <chr>   <chr>
1 N14228  UA
```

The tail number for this flight is N14228.

Exercise 6 - `filter()`

- Filter for all rows where the destination airport is RDU.

```r
flights%>%
  filter(dest == "RDU")
```

```
# A tibble: 8,163 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      800            810       -10      949            955
 2  2013     1     1      832            840        -8     1006           1030
 3  2013     1     1      851            851         0     1032           1036
 4  2013     1     1      917            920        -3     1052           1108
 5  2013     1     1     1024           1030        -6     1204           1215
 6  2013     1     1     1127           1129        -2     1303           1309
 7  2013     1     1     1157           1205        -8     1342           1345
 8  2013     1     1     1240           1235         5     1415           1415
 9  2013     1     1     1317           1325        -8     1454           1505
10  2013     1     1     1449           1450        -1     1651           1640
# i 8,153 more rows
# i 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>
```

- Filter for all rows where the destination airport is RDU and the arrival delay is less than 0.

```
flights %>%
  filter(dest == "RDU" & arr_delay < 0)
```

```
# A tibble: 4,232 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      800            810       -10      949            955
 2  2013     1     1      832            840        -8     1006           1030
 3  2013     1     1      851            851         0     1032           1036
 4  2013     1     1      917            920        -3     1052           1108
 5  2013     1     1     1024           1030        -6     1204           1215
 6  2013     1     1     1127           1129        -2     1303           1309
 7  2013     1     1     1157           1205        -8     1342           1345
 8  2013     1     1     1317           1325        -8     1454           1505
 9  2013     1     1     1505           1510        -5     1654           1655
10  2013     1     1     1800           1800         0     1945           1951
# i 4,222 more rows
# i 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>
```

- **Your turn:** Describe what the code is doing in words.

The code below is finding rows (flights) within the “flights” dataset where the destination is either “RDU” or “GSO” and the arrival and deparure delays are both less than 0.

```
flights |>
  filter(
    dest %in% c("RDU", "GSO"),
    arr_delay < 0 | dep_delay < 0
  )
```

```
# A tibble: 6,203 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      800            810       -10      949            955
 2  2013     1     1      832            840        -8      1006           1030
 3  2013     1     1      851            851         0      1032           1036
 4  2013     1     1      917            920        -3      1052           1108
 5  2013     1     1     1024           1030        -6      1204           1215
 6  2013     1     1     1127           1129        -2      1303           1309
 7  2013     1     1     1157           1205        -8      1342           1345
 8  2013     1     1     1317           1325        -8      1454           1505
 9  2013     1     1     1449           1450        -1      1651           1640
10  2013     1     1     1505           1510        -5      1654           1655
# i 6,193 more rows
# i 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>
```

Hint: Logical operators in R:

operator	definition
`<`	is less than?
`<=`	is less than or equal to?
`>`	is greater than?
`>=`	is greater than or equal to?
`==`	is exactly equal to?
`!=`	is not equal to?
`x & y`	is x AND y?
`x | y`	is x OR y?
`is.na(x)`	is x NA?
`!is.na(x)`	is x not NA?
`x %in% y`	is x in y?
`!(x %in% y)`	is x not in y?
`!x`	is not x? (only makes sense if `x` is `TRUE` or `FALSE`)

Exercise 7 - `count()`

- Create a frequency table of the destination locations for flights from New York.

```
flights %>%
  count(origin, dest, sort = TRUE)
```

```
# A tibble: 224 x 3
   origin dest      n
   <chr>  <chr> <int>
 1 JFK    LAX   11262
 2 LGA    ATL   10263
 3 LGA    ORD    8857
 4 JFK    SFO    8204
 5 LGA    CLT    6168
 6 EWR    ORD    6100
 7 JFK    BOS    5898
 8 LGA    MIA    5781
 9 JFK    MCO    5464
10 EWR    BOS    5327
# i 214 more rows
```

- In which month was there the fewest number of flights? How many flights were there in that month?

```
flights %>%
  count(month, sort = TRUE)
```

```
# A tibble: 12 x 2
   month     n
   <int> <int>
 1     7 29425
 2     8 29327
 3    10 28889
 4     3 28834
 5     5 28796
 6     4 28330
 7     6 28243
 8    12 28135
 9     9 27574
10    11 27268
11     1 27004
12     2 24951
```

The fewest flights were in February with 24951 flights.

- **Your turn:** On which date (month + day) was there the largest number of flights? How many flights were there on that day?

```
flights %>%
  count(month, day, sort = TRUE)
```

```
# A tibble: 365 x 3
   month   day     n
   <int> <int> <int>
 1    11    27  1014
 2     7    11  1006
 3     7     8  1004
 4     7    10  1004
 5    12     2  1004
 6     7    18  1003
 7     7    25  1003
 8     7    12  1002
 9     7     9  1001
10     7    17  1001
# i 355 more rows
```

The day with the most flights was November 27th with 1014 flights.

Exercise 8 - `mutate()`

- Convert `air_time` (minutes in the air) to hours and then create a new variable, `mph`, the miles per hour of the flight.

```r
flights %>%
  mutate(air_time / 60,
    mph = distance/air_time)
```

```
# A tibble: 336,776 x 21
    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      517            515         2      830            819
 2  2013     1     1      533            529         4      850            830
 3  2013     1     1      542            540         2      923            850
 4  2013     1     1      544            545        -1     1004           1022
 5  2013     1     1      554            600        -6      812            837
 6  2013     1     1      554            558        -4      740            728
 7  2013     1     1      555            600        -5      913            854
 8  2013     1     1      557            600        -3      709            723
 9  2013     1     1      557            600        -3      838            846
10  2013     1     1      558            600        -2      753            745
# i 336,766 more rows
# i 13 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>, `air_time/60` <dbl>, mph <dbl>
```

- **Your turn:** First, count the number of flights each month, and then calculate the proportion of flights in each month. What proportion of flights take place in July?

```
flights %>%
  count(month, sort = TRUE) %>%
  mutate(month_prop = n/sum(n)) %>%
  filter(month == 7)
```

```
# A tibble: 1 x 3
  month     n month_prop
  <int> <int>      <dbl>
1     7 29425     0.0874
```

The proportion of flights that take place in July is 0.08737262, or 8.73%

- Create a new variable, `rdu_bound`, which indicates whether the flight is to RDU or not. Then, for each departure airport (`origin`), calculate what proportion of flights originating from that airport are to RDU.

```
flights %>%
  mutate(rdu_bound = if_else(dest == "RDU", "YES", "NO")) %>%
  count(rdu_bound, origin, sort = TRUE) %>%
  filter(rdu_bound == "YES") %>%
  mutate(prop_rdu_bound = n/sum(n))
```

```
# A tibble: 3 x 4
  rdu_bound origin     n prop_rdu_bound
  <chr>     <chr>  <int>          <dbl>
1 YES       LGA     3581          0.439
2 YES       JFK     3100          0.380
3 YES       EWR     1482          0.182
```

Exercise 9 - `summarize()`

- Find mean arrival delay for all flights.

```r
flights %>%
  summarize(
    avg_delay = mean(arr_delay, na.rm = TRUE))
```

```
# A tibble: 1 x 1
  avg_delay
      <dbl>
1      6.90
```

Exercise 10 - `group_by()`

- Find mean arrival delay for for each month.

```r
flights %>%
  group_by(month) %>%
  summarize(
    avg_delay = mean(arr_delay, na.rm = TRUE))
```

```
# A tibble: 12 x 2
   month avg_delay
   <int>     <dbl>
 1     1     6.13
 2     2     5.61
 3     3     5.81
 4     4    11.2
 5     5     3.52
 6     6    16.5
 7     7    16.7
 8     8     6.04
 9     9    -4.02
10    10    -0.167
11    11     0.461
12    12    14.9
```

- **Your turn:** What is the median departure delay for each airports around NYC (`origin`)? Which airport has the shortest median departure delay?

```
flights %>%
  group_by(origin) %>%
  summarize(
    avg_dep_delay = median(dep_delay, na.rm = TRUE)) %>%
  slice_min(avg_dep_delay, n = 1)
```

```
# A tibble: 1 x 2
  origin avg_dep_delay
  <chr>          <dbl>
1 LGA               -3
```

Additional Practice

Try these on your own, either in class if you finish early, or after class.

1. Create a new dataset that only contains flights that do not have a missing departure time. Include the columns `year`, `month`, `day`, `dep_time`, `dep_delay`, and `dep_delay_hours` (the departure delay in hours). *Hint: Note you may need to use `mutate()` to make one or more of these variables.*

```
flights %>%
  mutate(dep_delay_hours = dep_delay/60) %>%
  select(year, month, day, dep_time, dep_delay, dep_delay_hours) %>%
   filter(!is.na(dep_time))
```

```
# A tibble: 328,521 x 6
    year month   day dep_time dep_delay dep_delay_hours
   <int> <int> <int>    <int>     <dbl>           <dbl>
 1  2013     1     1      517         2          0.0333
 2  2013     1     1      533         4          0.0667
 3  2013     1     1      542         2          0.0333
 4  2013     1     1      544        -1         -0.0167
 5  2013     1     1      554        -6         -0.1
 6  2013     1     1      554        -4         -0.0667
 7  2013     1     1      555        -5         -0.0833
 8  2013     1     1      557        -3         -0.05
 9  2013     1     1      557        -3         -0.05
10  2013     1     1      558        -2         -0.0333
# i 328,511 more rows
```

2. For each airplane (uniquely identified by `tailnum`), use a `group_by()` paired with `summarize()` to find the sample size, mean, and standard deviation of flight distances. Then include only the top 5 and bottom 5 airplanes in terms of mean distance traveled per flight in the final data frame.

```r
flights %>%
  group_by(tailnum) %>%
  summarize(mean = mean(distance),
            standard_deviation = sd(distance, na.rm = TRUE),
            n=n()) %>%
  arrange(mean) %>%
  filter(row_number() > max(row_number()) - 5 | row_number() <= 5)
```

```
# A tibble: 10 x 4
   tailnum  mean standard_deviation     n
   <chr>   <dbl>              <dbl> <int>
 1 N955UW   173.               32.9   225
 2 N948UW   174.               32.7   232
 3 N959UW   174.               34.3   213
 4 N956UW   174.               31.4   222
 5 N945UW   176.               31.2   285
 6 N390HA  4983                 0      20
 7 N391HA  4983                 0      21
 8 N392HA  4983                 0      13
 9 N393HA  4983                 0      10
10 N395HA  4983                 0       7
```

NOTE: The last filter to have top 5 and bottom 5 row numbers I found from an online forum (https://stackoverflow.com/questions/56809476/dplyr-filter-top-and-bottom-rows-by-value-simultaneously-on-grouped-data). I struggled to find an easier way to do it, but that was the only thing I could find that worked. I just wanted to acknowledge that that line was not my doing.