AE 04: NYC flights + data wrangling

James Turner

2025-02-04

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

Exercise 1

Your turn: Fill in the blanks:

The `flights` data frame has *337,776* rows. Each row represents a *Flight*.

```r
glimpse(flights)
```

```
Rows: 336,776
Columns: 19
$ 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, 554, 555, 557, 557, 558, 558, ~
$ sched_dep_time <int> 515, 529, 540, 545, 600, 558, 600, 600, 600, 600, 600, ~
$ dep_delay      <dbl> 2, 4, 2, -1, -6, -4, -5, -3, -3, -2, -2, -2, -2, -2, -1~
$ arr_time       <int> 830, 850, 923, 1004, 812, 740, 913, 709, 838, 753, 849,~
$ sched_arr_time <int> 819, 830, 850, 1022, 837, 728, 854, 723, 846, 745, 851,~
$ arr_delay      <dbl> 11, 20, 33, -18, -25, 12, 19, -14, -8, 8, -2, -3, 7, -1~
$ carrier        <chr> "UA", "UA", "AA", "B6", "DL", "UA", "B6", "EV", "B6", "~
$ flight         <int> 1545, 1714, 1141, 725, 461, 1696, 507, 5708, 79, 301, 4~
$ tailnum        <chr> "N14228", "N24211", "N619AA", "N804JB", "N668DN", "N394~
$ origin         <chr> "EWR", "LGA", "JFK", "JFK", "LGA", "EWR", "EWR", "LGA",~
$ dest           <chr> "IAH", "IAH", "MIA", "BQN", "ATL", "ORD", "FLL", "IAD",~
$ air_time       <dbl> 227, 227, 160, 183, 116, 150, 158, 53, 140, 138, 149, 1~
$ distance       <dbl> 1400, 1416, 1089, 1576, 762, 719, 1065, 229, 944, 733, ~
$ hour           <dbl> 5, 5, 5, 5, 6, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 5, 6, 6, 6~
$ minute         <dbl> 15, 29, 40, 45, 0, 58, 0, 0, 0, 0, 0, 0, 0, 0, 0, 59, 0~
$ time_hour      <dttm> 2013-01-01 05:00:00, 2013-01-01 05:00:00, 2013-01-01 0~
```

```r
view(flights)
```

Exercise 2

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

```r
colnames(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
delays <- flights %>%
  select(c(dep_delay, arr_delay))
```

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

```r
depart_on_time <- flights %>%
  select(-dep_delay)
```

- 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.

```r
departures <- flights %>%
  select(c(year:dep_delay))
```

- 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.

```
# add code here
```

Exercise 4 - `slice()`

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

```
# add code here
```

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

```
# add code here
```

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.

```
# add code here
```

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

Add your response here.

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

```
# add code here
```

- **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?

```
# add code here
```

Exercise 6 - `filter()`

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

```
# add code here
```

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

```
# add code here
```

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

Add response here.

```
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.

```
# add code here
```

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

```
# add code here
```

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

```
# add code here
```

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.

```
# add code here
```

- **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?

```
# add code here
```

- 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.

```
# add code here
```

Exercise 9 - `summarize()`

- Find mean arrival delay for all flights.

```
# add code here
```

Exercise 10 - `group_by()`

- Find mean arrival delay for for each month.

```
# add code here
```

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

```
# add code here
```

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.*

```
# add code here
```

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.

```
# add code here
```