

AE 04: NYC flights + data wrangling

Hannah Andronyk

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

Exercise 1

Your turn: Fill in the blanks:

The `flights` data frame has 336,776 rows. Each row represents a flight.

Exercise 2

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

```
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~
$ day       <int> 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, 5, 6, 6, 6~
$ minute    <dbl> 15, 29, 40, 45, 0, 58, 0, 0, 0, 0, 0, 0, 0, 0, 59, 0~
$ time_hour <dtm> 2013-01-01 05:00:00, 2013-01-01 05:00:00, 2013-01-01 0~
```

```
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`.

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

print(delays)
```

```
# 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`.

```
not_dep_delay <- flights %>%
  select(-dep_delay)

print(not_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 <dtm>
```

- 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.

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

print(rangeflightdf)
```

```
# 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.

```

arrival <- flights %>%
  select(contains("arr_"))

print(arrival)

```

```

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

```

Exercise 4 - `slice()`

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

```

first5 <- flights %>%
  slice(1:5)

print(first5)

```

```

# 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 <dtm>

```

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

```

last2 <- flights %>%
  tail(2)

print(last2)

```

```

# 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 <dtm>

```

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.

```

depart_delays1 <- flights %>%
  arrange(dep_delay)

print(depart_delays1)

```

```

# 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 <dtm>

```

- Question: What does it mean for the `dep_delay` to have a negative value? It means this flight came early by x amount of time, we know this because this value is the difference between scheduled departure time and departure time.
- Arrange the data by descending departure delay, so the flights with the longest departure delays will be at the top.

```

depart_delays2 <- flights %>%
  arrange(-dep_delay)

print(depart_delays2)

```

```

# 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 <dtm>

```

- **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? N384HA

```
tailnum_delay <- depart_delays2 %>%
  select(c(tailnum, carrier, dep_delay)) %>%
  slice(1)

print(tailnum_delay)
```

```
# A tibble: 1 x 3
  tailnum carrier dep_delay
  <chr>    <chr>      <dbl>
1 N384HA  HA           1301
```

Exercise 6 - filter()

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

```
RDU_airport1 <- flights %>%
  filter(dest == "RDU")

print(RDU_airport1)
```

```
# 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 <dtm>
```

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

```
RDU_airport2 <- flights %>%
  filter(dest == "RDU") %>%
  filter(arr_delay < 0)

print(RDU_airport2)
```

```
# 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 <dtm>
```

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

The code is filtering the flights data set for “RDU” and “GSO” in the dest column and keeping rows for these values that correspond to less than 0 rows in arr_delay or dep_delay columns.

```
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 <dtm>

```

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.

```

freq_flights <- flights %>%
  count(dest)

print(freq_flights)

```

```
# A tibble: 105 x 2
  dest      n
  <chr> <int>
1 ABQ    254
2 ACK    265
3 ALB    439
4 ANC      8
5 ATL  17215
6 AUS   2439
7 AVL    275
8 BDL    443
9 BGR    375
10 BHM    297
# i 95 more rows
```

- In which month was there the fewest number of flights? How many flights were there in that month? February had the fewest number of flights with a total of 24951 flights

```
freq_month <- flights %>%
  count(month) %>%
  arrange(n)

print(freq_month)
```

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

- **Your turn:** On which date (month + day) was there the largest number of flights? How many flights were there on that day? The largest number of flights was on November 27th with a total of 1014 flights

```
freq_day <- flights %>%
  count(month, day) %>%
  arrange(-n)

print(freq_day)
```

```
# 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
```

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.

```
flight_timehr <- flights %>%
  mutate(air_time = air_time/60) %>%
  mutate(mph = distance/air_time)

print(flight_timehr)
```

```
# A tibble: 336,776 x 20
  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 12 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 <dtm>, 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? 0.087

```

month_prop <- flights %>%
  count(month) %>%
  mutate(prop_flight = n/sum(n))

print(month_prop)

```

```

# A tibble: 12 x 3
  month      n prop_flight
  <int> <int>      <dbl>
1     1 27004     0.0802
2     2 24951     0.0741
3     3 28834     0.0856
4     4 28330     0.0841
5     5 28796     0.0855
6     6 28243     0.0839
7     7 29425     0.0874
8     8 29327     0.0871
9     9 27574     0.0819
10    10 28889     0.0858
11    11 27268     0.0810
12    12 28135     0.0835

```

- 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.

```

RDU_flights <- flights %>%
  mutate(rdu_bound = if_else(dest == "RDU", "Yes", "No")) %>%
  group_by(origin) %>%

```

```
mutate(prop_rdu = sum(rdu_bound == "Yes")/n()) %>%
count(prop_rdu)

print(RDU_flights)
```

```
# A tibble: 3 x 3
# Groups:   origin [3]
  origin prop_rdu      n
  <chr>    <dbl> <int>
1 EWR      0.0123 120835
2 JFK      0.0279 111279
3 LGA      0.0342 104662
```

Exercise 9 - summarize()

- Find mean arrival delay for all flights.

```
mean_arr_delay <- flights %>%
  summarize(mean_arr_time = mean(arr_delay, na.rm = TRUE))

print(mean_arr_delay)
```

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

Exercise 10 - group_by()

- Find mean arrival delay for for each month.
- **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
```