Flights of New York

Gowri Prathap

Type the following command in the Console window to view the dataset:

View(flights)

Important questions about the dataset:

1. How many rows and columns does this dataset have? This data set has are 336776 rows and 19 columns.
2. What does a single row in this dataset represent? Each row in this dataset represents data for each flight that departed NYC (i.e. JFK, LGA or EWR) in 2013.
3. What is the difference between the information contained in the dep_time and sched_dep_time columns? The dep_time gives the actual departing time of the flights, while sched_dep_time gives the scheduled departing time of the flights.
4. Which columns contain information about dates and times? Column 3 (day) gives the date and columns 15-19 give the air times in hours, minutes and so on. Columns 4-9 give the timings of the departure, arrival, and so on.
5. Airplanes are reused across many different flights. Which columns would be helpful to use in identifying individual airplanes? The 'flight' column and 'tailnum' column which gives the flight number and plane tail number respectively will help identify individual flights.

Important functions:

select() function

The select() function selects columns from a dataset.

```
flights %>%
  select(year, month)
```

```
## # A tibble: 336,776 x 2
##     year month
##    <int> <int>
##  1  2013     1
##  2  2013     1
##  3  2013     1
##  4  2013     1
##  5  2013     1
##  6  2013     1
##  7  2013     1
##  8  2013     1
##  9  2013     1
## 10  2013     1
```

```
## # ... with 336,766 more rows
```

This command displays the columns year and month since it was included in the select() function. The symbol %>% is called the pipe and is used to pass a dataset through a chain of commands.

```
flights %>%
  select(year:day)
```

```
## # A tibble: 336,776 x 3
##     year month   day
##    <int> <int> <int>
##  1  2013     1     1
##  2  2013     1     1
##  3  2013     1     1
##  4  2013     1     1
##  5  2013     1     1
##  6  2013     1     1
##  7  2013     1     1
##  8  2013     1     1
##  9  2013     1     1
## 10  2013     1     1
## # ... with 336,766 more rows
```

The colon prints all columns between including the two columns mentioned with the colon. Here it prints all the columns form year to month since the command says year:day.

arrange() function

The arrange() function sorts columns with textual data (chr data type) into alphabetical order and sorts numerical data into numerical order.

```
flights %>%
  arrange(month, day)
```

```
## # 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>
```

```
flights %>%
  arrange(day, month)
```

```
## # 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>
```

Based on the output, it looks like both the month and day columns were sorted. Day column was sorted first. If we reverse the order of columns in arrange(), the first piece of code does not make a difference, since days were arranged first, and days always change before months so it does not differ from the original dataset. The second piece of code makes a difference if reversed because it sorts the months first, in the order 1, 2, 3,...etc. and days only after that. Hence it is different from the first dataset.

The column that gives the departure delay is 'dep_delay'.

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

```
## # 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     9      641            900      1301     1242
##  2  2013     6    15     1432           1935      1137     1607
##  3  2013     1    10     1121           1635      1126     1239
##  4  2013     9    20     1139           1845      1014     1457
##  5  2013     7    22      845           1600      1005     1044
##  6  2013     4    10     1100           1900       960     1342
##  7  2013     3    17     2321            810       911      135
##  8  2013     6    27      959           1900       899     1236
##  9  2013     7    22     2257            759       898      121
## 10  2013    12     5      756           1700       896     1058
## # ... 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>
```

The first row of the arranged dataset has the longest departure delay, which is flight 51 tail number N384HA.

mutate() function

mutate() lets us transform a dataset by applying the same operation to each row in the dataset and appending the results as a new column.

```
flights %>%
  mutate(
    average_speed = distance / (air_time * 60)
  )
```

```
## # A tibble: 336,776 x 20
##     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 13 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>, average_speed <dbl>
```

The new column shows up as the last column of the dataset. The name of this new column is average_speed. The part of the code under mutate() where the column is declared, where the column name is given, is taken as the column name. Here, under mutate() it is given as average_speed = the formula, and hence average_speed is taken as the name of the column.

```
flights %>%
  mutate(
    dep_time_hour = dep_time %/% 100,
    dep_time_minute = dep_time %% 100,
    dep_time_minutes_midnight = dep_time_hour * 60 + dep_time_minute
  )
```

```
## # A tibble: 336,776 x 22
##     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 15 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>, dep_time_hour <dbl>,
## #   dep_time_minute <dbl>, dep_time_minutes_midnight <dbl>
```

We used the dep_time_hour and dep_time_minute columns to compute the number of minutes since midnight.

filter() function

To find all the flights operated by United Airlines (UA) that arrived early.

```
flights %>%
  filter(
    arr_delay < 0
  )
```

```
## # A tibble: 188,933 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      544            545        -1     1004
##  2  2013     1     1      554            600        -6      812
##  3  2013     1     1      557            600        -3      709
##  4  2013     1     1      557            600        -3      838
##  5  2013     1     1      558            600        -2      849
##  6  2013     1     1      558            600        -2      853
##  7  2013     1     1      558            600        -2      923
##  8  2013     1     1      559            559         0      702
##  9  2013     1     1      559            600        -1      854
## 10  2013     1     1      600            600         0      851
## # ... with 188,923 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>
```

```
flights %>%
  filter(
    carrier == "UA"
  )
```

```
## # A tibble: 58,665 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      554            558        -4      740
##  4  2013     1     1      558            600        -2      924
##  5  2013     1     1      558            600        -2      923
##  6  2013     1     1      559            600        -1      854
##  7  2013     1     1      607            607         0      858
##  8  2013     1     1      611            600        11      945
##  9  2013     1     1      623            627        -4      933
## 10  2013     1     1      628            630        -2     1016
## # ... with 58,655 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>
```

This code can be combined into a single block of code:

```
flights %>%
  filter(
    arr_delay < 0,
    carrier == "UA"
  )
```

```
## # A tibble: 34,642 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      558            600        -2      923
##  2  2013     1     1      559            600        -1      854
##  3  2013     1     1      607            607         0      858
##  4  2013     1     1      643            646        -3      922
##  5  2013     1     1      644            636         8      931
##  6  2013     1     1      646            645         1      910
##  7  2013     1     1      646            645         1     1023
##  8  2013     1     1      656            700        -4      948
##  9  2013     1     1      659            700        -1      959
## 10  2013     1     1      701            700         1     1123
## # ... with 34,632 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>
```

Using group_by() and summarize()

To calculate the mean (average) arrival delay for each airline carrier:

```
flights %>%
  group_by(carrier) %>%
  summarize(
```

```
    average_arr_delay = mean(arr_delay, na.rm = TRUE)
  )
```

```
## # A tibble: 16 x 2
##    carrier average_arr_delay
##    <chr>               <dbl>
##  1 9E                  7.38
##  2 AA                  0.364
##  3 AS                 -9.93
##  4 B6                  9.46
##  5 DL                  1.64
##  6 EV                 15.8
##  7 F9                 21.9
##  8 FL                 20.1
##  9 HA                 -6.92
## 10 MQ                 10.8
## 11 OO                 11.9
## 12 UA                  3.56
## 13 US                  2.13
## 14 VX                  1.76
## 15 WN                  9.65
## 16 YV                 15.6
```

Airline carrier FL had the longest arrival delays on average. The airline carrier AS had the shortest arrival delays on average.