Data Wrangling I

[INSERT YOUR NAME]

2021-09-06

To demonstrate data wrangling we will use `flights`, a tibble in the **nycflights13** R package. It includes characteristics of all flights departing from New York City (JFK, LGA, EWR) in 2013.

```
library(tidyverse)
library(nycflights13) #includes flights data
```

The data frame has over 336,000 observations (rows), 336776 observations to be exact, so we will **not** view the entire data frame. Instead we'll use the commands below to help us explore the data.

```
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~
```

```
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"
```

```
head(flights)
```

```
## # A tibble: 6 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
## 6  2013     1     1      554            558        -4      740            728
## # ... with 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>
```

The `head()` function returns "A tibble: 6 x 19" and then the first six rows of the `flights` data.

Tibble vs. data frame

A **tibble** is an opinionated version of the `R` data frame. In other words, all tibbles are data frames, but not all data frames are tibbles!

There are two main differences between a tibble and a data frame:

1. When you print a tibble, the first ten rows and all of the columns that fit on the screen will display, along with the type of each column.

Let's look at the differences in the output when we type `flights` (tibble) in the console versus typing `cars` (data frame) in the console.

2. Second, tibbles are somewhat more strict than data frames when it comes to subsetting data. You will get an error message if you try to access a variable that doesn't exist in a tibble. You will get `NULL` if you try to access a variable that doesn't exist in a data frame.

```
flights$apple
```

```
## Warning: Unknown or uninitialised column: ‘apple‘.

## NULL
```

```
cars$apple
```

```
## NULL
```

Data wrangling with `dplyr`

dplyr is the primary package in the tidyverse for data wrangling. Click here for the dplyr reference page. Click here for the dplyr cheatsheet.

Quick summary of key dplyr functions[1]:

Rows:

- `filter()`:chooses rows based on column values.
- `slice()`: chooses rows based on location.
- `arrange()`: changes the order of the rows
- `sample_n()`: take a random subset of the rows

Columns:

- `select()`: changes whether or not a column is included.
- `rename()`: changes the name of columns.
- `mutate()`: changes the values of columns and creates new columns.

Groups of rows:

- `summarise()`: collapses a group into a single row.
- `count()`: count unique values of one or more variables.
- `group_by()`: perform calculations separately for each value of a variable

`select()`

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

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

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

[1] From dplyr vignette

```
# add code here
```

- 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.

```
## add code
```

- 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
```

The pipe

Before looking at more data wrangling functions, let's introduce the pipe. The **pipe**, `%>%`, is a technique for passing information from one process to another. We will use `%>%` mainly in dplyr pipelines to pass the output of the previous line of code as the first input of the next line of code.

When reading code "in English", say "and then" whenever you see a pipe.

Question 1 (4 minutes) The following code is equivalent to which line of code? Submit your response in Ed Discussion: https://edstem.org/us/courses/8027/discussion/590071

```
flights %>%
  select(dep_delay, arr_delay) %>%
  head()
```

```
## # A tibble: 6 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
```

`slice()`

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

```
flights %>%
  slice(1:5)
```

```
## # 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
## # ... with 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>
```

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

```
flights %>%
  slice((n()-1):n())
```

```
## # 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
## # ... with 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>
```

`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. **What does it mean for the `dep_delay` to have a negative value?**

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

- Now let's arrange the data by descending departure delay, so the flights with the longest departure delays will be at the top.

```
## add code
```

- **Question 2 (5 minutes)**: 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? Submit your response on Ed Discussion: https://edstem.org/us/courses/8027/discussion/590079

```
## add code
```

filter()

- Filter the data frame by selecting the rows where the destination airport is RDU.

```
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
## # ... with 8,153 more rows, and 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>
```

- We can also filter using more than one condition. Here we select 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
## # ... with 4,222 more rows, and 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>
```

We can do more complex tasks using logical operators:

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?

The final operator only makes sense if `x` is logical (TRUE / FALSE).

- **Question 3 (4 minutes)**: Describe what the code is doing in words. Submit your response in Ed Discussion: https://edstem.org/us/courses/8027/discussion/590083

```
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
## # ... with 6,193 more rows, and 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>
```

`count()`

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

```
flights %>%
  count(dest)
```

```
## # 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
## # ... with 95 more rows
```

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

```
## add code
```

- **Question 4 (5 minutes)**: On which date (month + day) was there the largest number of flights? How many flights were there on that day? Submit your response on Ed Discussion: https://edstem.org/us/courses/8027/discussion/590086

```
## add code
```

`mutate()`

Use `mutate()` to create a new variable.

- In the code chunk below, `air_time` (minutes in the air) is converted to hours, and then new variable `mph` is created, corresponding to the miles per hour of the flight.

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

```
## # A tibble: 336,776 x 4
##    air_time distance hours   mph
##       <dbl>    <dbl> <dbl> <dbl>
##  1      227     1400 3.78   370.
##  2      227     1416 3.78   374.
##  3      160     1089 2.67   408.
##  4      183     1576 3.05   517.
##  5      116      762 1.93   394.
##  6      150      719 2.5    288.
##  7      158     1065 2.63   404.
##  8       53      229 0.883  259.
##  9      140      944 2.33   405.
## 10      138      733 2.3    319.
## # ... with 336,766 more rows
```

- **Question (4 minutes)**: Create a new variable to calculate the percentage of flights in each month. What percentage of flights take place in July?

```r
## add code
```

summarize()/ summarise()

summarise() collapses the rows into summary statistics and removes columns irrelevant to the calculation.

Be sure to name your columns!

```r
flights %>%
  summarise(mean_dep_delay = mean(dep_delay))
```

```
## # A tibble: 1 x 1
##   mean_dep_delay
##            <dbl>
## 1             NA
```

Question: Why did this code return NA?

Let's fix it

```r
flights %>%
  summarize(mean_dep_delay = mean(dep_delay, na.rm = TRUE))
```

```
## # A tibble: 1 x 1
##   mean_dep_delay
##            <dbl>
## 1           12.6
```

group_by()

group_by() is used for grouped operations. It's very powerful when paired with summarise() to calculate summary statistics by group.

Here we find the mean and standard deviation of departure delay for each month.

```r
flights %>%
  group_by(month) %>%
  summarize(mean_dep_delay = mean(dep_delay, na.rm = TRUE),
            sd_dep_delay = sd(dep_delay, na.rm = TRUE))
```

```
## # A tibble: 12 x 3
##    month mean_dep_delay sd_dep_delay
##    <int>          <dbl>        <dbl>
##  1     1           10.0         36.4
##  2     2           10.8         36.3
##  3     3           13.2         40.1
##  4     4           13.9         43.0
##  5     5           13.0         39.4
##  6     6           20.8         51.5
##  7     7           21.7         51.6
##  8     8           12.6         37.7
##  9     9           6.72         35.6
## 10    10           6.24         29.7
## 11    11           5.44         27.6
## 12    12          16.6          41.9
```

- **Question 5 (4 minutes)**: What is the median departure delay for each airport in NYC (`origin`)? Which airport has the shortest median departure delay? Submit your response on Ed Discussion: https://edstem.org/us/courses/8027/discussion/590091

```
## add code
```

Additional Practice

(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.*

(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.