Data Transformation

A gentle introduction to Data Proprocessing

David Raj M

Contents

1 Introduction

Visualization is an important tool for generating insight, but it's rare that you get the data in exactly the right form you need to make the graph you want. Often you'll need to create some new variables or summaries to answer your questions with your data, or maybe you just want to rename the variables or reorder the observations to make the data a little easier to work with.

We will be using **`dplyr`** package, as you might have already known, we will use `tidyverse` so that `dplyr` comes by default.

```
# install.packages("nycflights13")
library(nycflights13)
library(tidyverse)
```

```
── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0
──
✔ dplyr     1.1.4     ✔ readr     2.1.6
✔ forcats   1.0.1     ✔ stringr   1.6.0
✔ ggplot2   4.0.1     ✔ tibble    3.3.1
✔ lubridate 1.9.4     ✔ tidyr     1.3.2
✔ purrr     1.2.1
── Conflicts ────────────────────────────────────────── tidyverse_conflicts()
──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
```

```
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors
```

1.1 nycflights13

To explore the basic dplyr verbs, we will use `nycflights13::flights`. This dataset contains all 336,776 flights that departed from New York City in 2013. The data comes from the US Bureau of Transportation Statistics and is documented in `?flights`.

```
flights
```

```
# A tibble: 336,776 × 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
 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 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>
```

`flights` is a tibble, a special type of data frame used by the tidyverse to avoid some common gotchas. The most important difference between tibbles and data frames is the way tibbles print; they are designed for large datasets, so they only show the first few rows and only the columns that fit on one screen.

```
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…
```

You're about to learn the primary dplyr verbs (functions), which will allow you to solve the vast majority of your data manipulation challenges. But before we discuss their individual differences, it's worth stating what they have in common:

- The first argument is always a data frame.
- The subsequent arguments typically describe which columns to operate on using the variable names (without quotes).
- The output is always a new data frame.

```
flights |>
  filter(dest == "IAH") |>
  group_by(year, month, day) |>
  summarize(
```

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

```
`summarise()` has grouped output by 'year', 'month'. You can override using
the
`.groups` argument.
```

```
# A tibble: 365 × 4
# Groups:   year, month [12]
    year month   day arr_delay
   <int> <int> <int>     <dbl>
 1  2013     1     1     17.8
 2  2013     1     2      7
 3  2013     1     3     18.3
 4  2013     1     4     -3.2
 5  2013     1     5     20.2
 6  2013     1     6      9.28
 7  2013     1     7     -7.74
 8  2013     1     8      7.79
 9  2013     1     9     18.1
10  2013     1    10      6.68
# ℹ 355 more rows
```

dplyr's verbs are organized into four groups based on what they operate on: **rows**, **columns**, **groups**, or **tables**.

2 Basic Functions of dplyr

2.1 filter()

```
flights |>
  filter(dep_delay > 120)
```

```
# A tibble: 9,723 × 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      848           1835       853     1001           1950
 2  2013     1     1      957            733       144     1056            853
 3  2013     1     1     1114            900       134     1447           1222
 4  2013     1     1     1540           1338       122     2020           1825
 5  2013     1     1     1815           1325       290     2120           1542
 6  2013     1     1     1842           1422       260     1958           1535
 7  2013     1     1     1856           1645       131     2212           2005
 8  2013     1     1     1934           1725       129     2126           1855
 9  2013     1     1     1938           1703       155     2109           1823
```

```
10  2013     1     1     1942           1705       157     2124           1830
# ℹ 9,713 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>
```

```
flights |>
  filter(month == 1 & day == 1)
```

```
# A tibble: 842 × 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
 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 832 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>
```

```
flights |>
  filter(month == 1 | month == 2)
```

```
# A tibble: 51,955 × 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
 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 51,945 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>
```

```
flights |>
  filter(month %in% c(1, 2))
```

```
# A tibble: 51,955 × 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
 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 51,945 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>
```

```
jan1 <- flights |>
  filter(month == 1 & day == 1)
```

2.2 mutate()

The job of `mutate()` is to add new columns that are calculated from the existing columns.

```
flights |>
  mutate(
    gain = dep_delay - arr_delay,
    speed = distance / air_time * 60
  )
```

```
# A tibble: 336,776 × 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>, gain <dbl>, speed <dbl>
```

By default, `mutate()` adds new columns on the right-hand side of your dataset, which makes it difficult to see what's happening here. We can use the `.before` argument to instead add the variables to the left-hand side.

```
flights |>
  mutate(
    gain = dep_delay - arr_delay,
    speed = distance / air_time * 60,
    .before = 1
  )
```

```
# A tibble: 336,776 × 21
    gain speed  year month   day dep_time sched_dep_time dep_delay arr_time
   <dbl> <dbl> <int> <int> <int>    <int>          <int>     <dbl>    <int>
 1    -9  370.  2013     1     1      517            515         2      830
 2   -16  374.  2013     1     1      533            529         4      850
 3   -31  408.  2013     1     1      542            540         2      923
 4    17  517.  2013     1     1      544            545        -1     1004
 5    19  394.  2013     1     1      554            600        -6      812
 6   -16  288.  2013     1     1      554            558        -4      740
 7   -24  404.  2013     1     1      555            600        -5      913
 8    11  259.  2013     1     1      557            600        -3      709
 9     5  405.  2013     1     1      557            600        -3      838
10   -10  319.  2013     1     1      558            600        -2      753
# ℹ 336,766 more rows
# ℹ 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 `.` indicates that `.before` is an argument to the function, not the name of a third new variable we are creating. You can also use `.after` to add after a variable, and in both `.before` and `.after` you can use the variable name instead of a position. For example, we could add the new variables after `day`:

```
flights |>
  mutate(
    gain = dep_delay - arr_delay,
    speed = distance / air_time * 60,
    .after = day
  )
```

```
# A tibble: 336,776 × 21
    year month   day  gain speed dep_time sched_dep_time dep_delay arr_time
   <int> <int> <int> <dbl> <dbl>    <int>          <int>     <dbl>    <int>
 1  2013     1     1    -9  370.      517            515         2      830
 2  2013     1     1   -16  374.      533            529         4      850
 3  2013     1     1   -31  408.      542            540         2      923
 4  2013     1     1    17  517.      544            545        -1     1004
 5  2013     1     1    19  394.      554            600        -6      812
 6  2013     1     1   -16  288.      554            558        -4      740
 7  2013     1     1   -24  404.      555            600        -5      913
 8  2013     1     1    11  259.      557            600        -3      709
 9  2013     1     1     5  405.      557            600        -3      838
10  2013     1     1   -10  319.      558            600        -2      753
# ℹ 336,766 more rows
# ℹ 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>
```

2.3 select()

`select()` allows you to rapidly zoom in on a useful subset using operations based on the names of the variables:

```
# Select columns by name
flights |>
  select(year, month, day)
```

```
# A tibble: 336,776 × 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
# ℹ 336,766 more rows
```

```
# Select all columns between year and day (inclusive)
flights |>
  select(year:day)
```

```
# A tibble: 336,776 × 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
# ℹ 336,766 more rows
```

```
# Select all columns except those from year to day (inclusive)
flights |>
  select(!year:day)
```

```
# A tibble: 336,776 × 16
   dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier
      <int>          <int>     <dbl>    <int>          <int>     <dbl> <chr>
 1      517            515         2      830            819        11 UA
 2      533            529         4      850            830        20 UA
 3      542            540         2      923            850        33 AA
 4      544            545        -1     1004           1022       -18 B6
 5      554            600        -6      812            837       -25 DL
 6      554            558        -4      740            728        12 UA
 7      555            600        -5      913            854        19 B6
 8      557            600        -3      709            723       -14 EV
 9      557            600        -3      838            846        -8 B6
10      558            600        -2      753            745         8 AA
# ℹ 336,766 more rows
# ℹ 9 more variables: flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
#   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

```
# Select all columns that are characters
flights |>
  select(where(is.character))
```

```
# A tibble: 336,776 × 4
   carrier tailnum origin dest
   <chr>   <chr>   <chr>  <chr>
 1 UA      N14228  EWR    IAH
 2 UA      N24211  LGA    IAH
 3 AA      N619AA  JFK    MIA
 4 B6      N804JB  JFK    BQN
 5 DL      N668DN  LGA    ATL
 6 UA      N39463  EWR    ORD
 7 B6      N516JB  EWR    FLL
 8 EV      N829AS  LGA    IAD
 9 B6      N593JB  JFK    MCO
10 AA      N3ALAA  LGA    ORD
# ℹ 336,766 more rows
```

There are a number of helper functions you can use within `select()`:

- `starts_with("abc")`: matches names that begin with "abc".
- `ends_with("xyz")`: matches names that end with "xyz".
- `contains("ijk")`: matches names that contain "ijk".
- `num_range("x", 1:3)`: matches `x1`, `x2` and `x3`.

See `?select` for more details.

You can rename variables as you `select()` them by using `=`. The new name appears on the left-hand side of the `=`, and the old variable appears on the right-hand side

```
flights |>
  select(tail_num = tailnum)
```

```
# A tibble: 336,776 × 1
   tail_num
   <chr>
 1 N14228
 2 N24211
 3 N619AA
 4 N804JB
 5 N668DN
 6 N39463
 7 N516JB
 8 N829AS
```

```
 9 N593JB
10 N3ALAA
# i 336,766 more rows
```

2.4 arange()

`arrange()` changes the order of the rows based on the value of the columns. It takes a data frame and a set of column names (or more complicated expressions) to order by. If you provide more than one column name, each additional column will be used to break ties in the values of the preceding columns. For example, the following code sorts by the departure time, which is spread over four columns.

```
flights |>
  arrange(year, month, day, dep_time)
```

```
# A tibble: 336,776 × 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
 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 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>
```

You can use `desc()` on a column inside of `arrange()` to re-order the data frame based on that column in descending (big-to-small) order. For example, this code orders flights from most to least delayed:

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

```
# A tibble: 336,776 × 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>
```

2.5 distinct()

`distinct()` finds all the unique rows in a dataset, so technically, it primarily operates on the rows. Most of the time, however, you'll want the distinct combination of some variables, so you can also optionally supply column names:

```
# Remove duplicate rows, if any
flights |>
  distinct()
```

```
# A tibble: 336,776 × 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
 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 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>
```

```
# Find all unique origin and destination pairs
flights |>
  distinct(origin, dest)
```

```
# A tibble: 224 × 2
   origin dest
   <chr>  <chr>
 1 EWR    IAH
 2 LGA    IAH
 3 JFK    MIA
 4 JFK    BQN
 5 LGA    ATL
 6 EWR    ORD
 7 EWR    FLL
 8 LGA    IAD
 9 JFK    MCO
10 LGA    ORD
# ℹ 214 more rows
```

Alternatively, if you want to keep the other columns when filtering for unique rows, you can use the `.keep_all = TRUE` option.

```
flights |>
  distinct(origin, dest, .keep_all = TRUE)
```

```
# A tibble: 224 × 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
 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 214 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>
```

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

```
# A tibble: 224 × 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
```

We've shown you simple examples of the pipe above, but its real power arises when you start to combine multiple verbs. For example, imagine that you wanted to find the fastest flights to Houston's IAH airport: you need to combine `filter()`, `mutate()`, `select()`, and `arrange()`

```
flights |>
  filter(dest == "IAH") |>
  mutate(speed = distance / air_time * 60) |>
  select(year:day, dep_time, carrier, flight, speed) |>
  arrange(desc(speed))
```

```
# A tibble: 7,198 × 7
    year month   day dep_time carrier flight speed
   <int> <int> <int>    <int> <chr>    <int> <dbl>
 1  2013     7     9      707 UA         226  522.
 2  2013     8    27     1850 UA        1128  521.
 3  2013     8    28      902 UA        1711  519.
 4  2013     8    28     2122 UA        1022  519.
 5  2013     6    11     1628 UA        1178  515.
 6  2013     8    27     1017 UA         333  515.
 7  2013     8    27     1205 UA        1421  515.
 8  2013     8    27     1758 UA         302  515.
 9  2013     9    27      521 UA         252  515.
10  2013     8    28      625 UA         559  515.
# ℹ 7,188 more rows
```

2.6 group_by()

```
flights |>
  group_by(year)
```

```
# A tibble: 336,776 × 19
# Groups:   year [1]
    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 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>
```

```
flights |>
  group_by(month) |>
  summarize(
    avg_delay = mean(dep_delay)
  )
```

```
# A tibble: 12 × 2
   month avg_delay
   <int>     <dbl>
 1     1        NA
 2     2        NA
 3     3        NA
 4     4        NA
 5     5        NA
 6     6        NA
 7     7        NA
 8     8        NA
 9     9        NA
10    10        NA
11    11        NA
12    12        NA
```

Uh-oh! Something has gone wrong, and all of our results are `NA`s (pronounced "N-A"), R's symbol for missing value. This happened because some of the observed flights had missing data in the delay column, and so when we calculated the mean including those values, we got an `NA` result.

```
flights |>
  group_by(month) |>
  summarize(
```

```
    avg_delay = mean(dep_delay, na.rm = TRUE)
  )
```

```
# A tibble: 12 × 2
   month avg_delay
   <int>     <dbl>
 1     1     10.0
 2     2     10.8
 3     3     13.2
 4     4     13.9
 5     5     13.0
 6     6     20.8
 7     7     21.7
 8     8     12.6
 9     9      6.72
10    10      6.24
11    11      5.44
12    12     16.6
```