STAT167 Lab #5 - Spring 2025

Ethan Choi

2025/5/2

Contents

Discussion/Lab #5 instructions

This week, we will continue reviewing data transformation and manipulation using `dplyr`.

- First, download the `rmd` file from Canvas.
- Open this `rmd` file in RStudio and click `Knit -> Knit to PDF` to render it to PDF format. You need to have `LaTex` installed on the computer to render it to PDF format. If not, you can also render it to HTML format.
- Read this `rmd` file and the rendered `pdf/html` file side-by-side, to see how this document was generated!
- Be sure to play with this document! Change it. Break it. Fix it. The best way to learn R Markdown (or really almost anything) is to try, fail, then find out what you did wrong.
- Read over the example code and the output. If you have any questions about certain functions or parameters, it is the time to ask!

- There are some exercises through out this document. Replace **INSERT_YOUR_ANSWER** with your own answers. Knit the file, and check your results.

Please comment your R code thoroughly, and follow the R coding style guideline (https://google.github.io/styleguide/Rguide.xml). Partial credit will be deducted for insufficient commenting or poor coding styles.

Lab submission guideline

- After you completed all exercises, save your file to `FirstnameLastname-SID-lab5.rmd` and save the rendered pdf file to `FirstnameLastname-SID-lab5.pdf`. If you can not knit it to pdf, knit it to html first and then print/save it to pdf format.
- Submit **BOTH your source `rmd` file and the knitted `pdf` file** to **GradeScope**. Do NOT create a zip file.
- You can submit multiple times, you last submission will be graded.

Lecture Review - data transformation with `dplyr`

Load the `tidyverse` package

```
# install the tidyverse package first if you have not done it yet.
#install.packages("tidyverse") # you can comment out this line after you have installed `tidyverse`
library(tidyverse)
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

The `nycflights13::flights` data set

This data frame contains all 336,776 flights that departed from New York City in 2013. https://nycflights13.tidyverse.org

```
# You need to install `nycflights13` first, then you can comment out the following line
#install.packages("nycflights13")
library(nycflights13)
```

```
#?flights # full documentation
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~
```

`arrange()` - reorder the rows by their column values

```
# sort flights by date
arrange(flights, year, month, day)
## # 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     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>

# sort flight by departure delay in a descending order
arrange(flights, desc(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     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>
```

Exercise #1

(a) Find the flights that traveled the shortest distance

```
flights |>
  arrange(distance) |>
  select(year, month, day, tailnum, carrier, flight, origin, dest, distance) |>
  head()
```

```
## # A tibble: 6 x 9
##    year month   day tailnum carrier flight origin dest  distance
##   <int> <int> <int> <chr>   <chr>    <int> <chr>  <chr>    <dbl>
## 1  2013     7    27 <NA>    US        1632 EWR    LGA         17
## 2  2013     1     3 N13989  EV        3833 EWR    PHL         80
## 3  2013     1     4 N14972  EV        4193 EWR    PHL         80
## 4  2013     1     4 N15983  EV        4502 EWR    PHL         80
## 5  2013     1     4 N27962  EV        4645 EWR    PHL         80
## 6  2013     1     5 N14902  EV        4193 EWR    PHL         80
```

(b) Find the flights that had the shortest air time

```
flights |>
  filter(!is.na(air_time)) |>
  arrange(air_time) |>
  select(year, month, day, tailnum, carrier, flight, origin, dest, air_time) |>
  head()
```

```
## # A tibble: 6 x 9
##    year month   day tailnum carrier flight origin dest  air_time
##   <int> <int> <int> <chr>   <chr>    <int> <chr>  <chr>    <dbl>
## 1  2013     1    16 N16911  EV        4368 EWR    BDL         20
## 2  2013     4    13 N12167  EV        4631 EWR    BDL         20
## 3  2013    12     6 N27200  EV        4276 EWR    BDL         21
## 4  2013     2     3 N13913  EV        4619 EWR    PHL         21
## 5  2013     2     5 N13955  EV        4368 EWR    BDL         21
## 6  2013     2    12 N12921  EV        4619 EWR    PHL         21
```

`mutate()` - create new columns

```
# create two new variables, gain and speed, and add them after the day column
mutate(flights,
       gain = arr_delay - dep_delay,
       speed = distance / air_time * 60,
       after = day)
## # A tibble: 336,776 x 22
##     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 14 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>,
## #   after <int>

# If you only want to keep the new variables, use `transmute()`
transmute(flights,
          gain = arr_delay - dep_delay,
          hours = air_time / 60,
          gain_per_hour = gain / hours)
## # A tibble: 336,776 x 3
##     gain hours gain_per_hour
##    <dbl> <dbl>         <dbl>
##  1     9 3.78           2.38
##  2    16 3.78           4.23
##  3    31 2.67          11.6
##  4   -17 3.05          -5.57
##  5   -19 1.93          -9.83
##  6    16 2.5            6.4
##  7    24 2.63           9.11
##  8   -11 0.883        -12.5
##  9    -5 2.33          -2.14
## 10    10 2.3            4.35
## # i 336,766 more rows
```

Exercise #2

Find the fastest flight (i.e., the flight has the highest speed). Report the flight date, actual departure and arrival time, origin, destination, carrier, and flight number.

```r
flights |>
  filter(!is.na(distance), !is.na(air_time)) |>
  mutate(speed = distance / air_time * 60) |>
  arrange(desc(speed)) |>
  select(year, month, day, dep_time, arr_time, origin, dest, carrier, flight, speed) |>
  slice(1)
```

```
## # A tibble: 1 x 10
##    year month   day dep_time arr_time origin dest  carrier flight speed
##   <int> <int> <int>    <int>    <int> <chr>  <chr> <chr>    <int> <dbl>
## 1  2013     5    25     1709     1923 LGA    ATL   DL        1499  703.
```

Grouped summaries with `group_by()` and `summarize()`

```r
# `summarize()` collapses a data frame to a single row:
summarize(flights,
          flight_count = n(),
          avg_delay = mean(dep_delay, na.rm = TRUE))
## # A tibble: 1 x 2
##   flight_count avg_delay
##          <int>     <dbl>
## 1       336776      12.6

# `summarize()` is useful when pair it with `group_by()`
by_day <- group_by(flights, year, month, day)
summarize(by_day,
          flight_count = n(),
          avg_delay = mean(dep_delay, na.rm = TRUE),
          .groups = "keep")
## # A tibble: 365 x 5
## # Groups:   year, month, day [365]
##     year month   day flight_count avg_delay
##    <int> <int> <int>        <int>     <dbl>
##  1  2013     1     1          842     11.5
##  2  2013     1     2          943     13.9
##  3  2013     1     3          914     11.0
##  4  2013     1     4          915      8.95
##  5  2013     1     5          720      5.73
##  6  2013     1     6          832      7.15
##  7  2013     1     7          933      5.42
##  8  2013     1     8          899      2.55
##  9  2013     1     9          902      2.28
## 10  2013     1    10          932      2.84
## # i 355 more rows
```

Combining multiple operations with the pipe |>

Imagine that we want to explore the relationship between the distance and average delay for each destination.

Data transformation & cleaning - step by step

1. Filter out canceled flights;
2. Group non-canceled flights by destination;
3. Summarize grouped data to compute the number of flights, average distance, and average arrival delay per destination;
4. Filter out outliers (longest distances - HNL and ANC);
5. Filter out noisy points (small flight counts - rare destinations).

```
# original code - step by step
not_canceled <- filter(flights, !is.na(dep_delay), !is.na(arr_delay))
not_canceled_by_dest <-  group_by(not_canceled, dest)
delays_by_dest <- summarize(not_canceled_by_dest, flight_count = n(),
                            avg_dist = mean(distance), avg_delay = mean(arr_delay))
delays_by_dest_filtered <- filter(delays_by_dest,
                                  dest != "HNL", dest != "ANC")
delays_by_dest_cleaned <- filter(delays_by_dest_filtered, flight_count >= 10)
```

```
# alternative code using piping |>
not_canceled <- filter(flights, !is.na(dep_delay), !is.na(arr_delay))
delays_by_dest_cleaned <- not_canceled |>
  group_by(dest) |>
  summarize(flight_count = n(), avg_dist = mean(distance), avg_delay = mean(arr_delay)) |>
  filter(dest != "HNL", dest != "ANC", flight_count >= 10)
```

```
delays_by_dest_cleaned
## # A tibble: 101 x 4
##    dest  flight_count avg_dist avg_delay
##    <chr>        <int>    <dbl>     <dbl>
##  1 ABQ            254    1826       4.38
##  2 ACK            264     199       4.85
##  3 ALB            418     143      14.4
##  4 ATL          16837     757.     11.3
##  5 AUS           2411    1514.      6.02
##  6 AVL            261     584.      8.00
##  7 BDL            412     116       7.05
##  8 BGR            358     378       8.03
##  9 BHM            269     866.     16.9
## 10 BNA           6084     758.     11.8
## # i 91 more rows
```

Exercise #3

Which carrier had the worst delays over the year?

First, filter out all canceled flights; Next, group by carrier; Then, calculate the average arrival delay per carrier; Last, rank your results.

Try to use the pipe |> to answer this question.

```r
flights |>
  filter(!is.na(dep_delay), !is.na(arr_delay)) |>
  group_by(carrier) |>
  summarize(avg_arr_delay = mean(arr_delay), .groups = "drop") |>
  arrange(desc(avg_arr_delay))
```

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