Richard Gresham

CPSC 375 Big Data

September 27, 2022

Homework #4

Prepare your answers as a **single PDF file**.

Group work: You may work in groups of 1-3. Include all group member names in the PDF file. Only one person in the group should submit to Canvas.

Due: check on Canvas.

Load the nycflights13 library (will have to install the nycflights13 package first) which contains flight arrival and departure data in a table called flights. Apply the tidyverse's data wrangling verbs to answer these questions. For each question, **give only the (one line as a data pipeline) code beginning with flights %>% ….**

1. List data only for flights that departed on February 12, 2013. Input: flights %>% filter(year == 2013, month == 2, day == 12)

   Output:

```
# A tibble: 893 × 19
    year month   day dep_time sched_d…¹
   <int> <int> <int>    <int>      <int>
 1  2013     2    12       17       2245
 2  2013     2    12      506        500
 3  2013     2    12      520        525
 4  2013     2    12      524        530
 5  2013     2    12      535        540
 6  2013     2    12      539        540
 7  2013     2    12      551        600
 8  2013     2    12      552        600
 9  2013     2    12      553        600
10  2013     2    12      555        600
# … with 883 more rows, 14 more
#   variables: dep_delay <dbl>,
#   arr_time <int>,
#   sched_arr_time <int>,
#   arr_delay <dbl>, carrier <chr>,
#   flight <int>, tailnum <chr>,
#   origin <chr>, dest <chr>, …
# ℹ Use `print(n = ...)` to see more rows, and `colnames()` to s
ee all variable names
```

2. List data only for flights that were delayed (both arrival and departure) by more than 2 hours.

   Input: flights %>% filter(dep_delay > 120, arr_delay >120)

   Output:

```
> flights %>% filter(dep_delay > 120, arr_delay >120)
# A tibble: 8,335 × 19
    year month   day dep_time sched_…¹ dep_d…² arr_t…³ sched…⁴ arr_d…⁵
   <int> <int> <int>    <int>    <int>   <dbl>   <int>   <int>   <dbl>
 1  2013     1     1      848     1835     853    1001    1950     851
 2  2013     1     1      957      733     144    1056     853     123
 3  2013     1     1     1114      900     134    1447    1222     145
 4  2013     1     1     1815     1325     290    2120    1542     338
 5  2013     1     1     1842     1422     260    1958    1535     263
 6  2013     1     1     1856     1645     131    2212    2005     127
 7  2013     1     1     1934     1725     129    2126    1855     151
 8  2013     1     1     1938     1703     155    2109    1823     166
 9  2013     1     1     1942     1705     157    2124    1830     174
10  2013     1     1     2006     1630     216    2230    1848     222
# … with 8,325 more rows, 10 more variables: carrier <chr>,
#   flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
#   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
#   time_hour <dttm>, and abbreviated variable names ¹sched_dep_time,
#   ²dep_delay, ³arr_time, ⁴sched_arr_time, ⁵arr_delay
# ℹ Use `print(n = ...)` to see more rows, and `colnames()` to see all va
riable names
```

3. List data only for flights that were delayed (either arrival or departure) by more than 2 hours.

Input: flights %>% filter(dep_delay > 120 | arr_delay >120)

Output:

```
# A tibble: 11,422 × 19
    year month   day dep_time sched_…¹ dep_d…² arr_t…³ sched…⁴ arr_d…⁵
   <int> <int> <int>    <int>    <int>   <dbl>   <int>   <int>   <dbl>
 1  2013     1     1      811      630     101    1047     830     137
 2  2013     1     1      848     1835     853    1001    1950     851
 3  2013     1     1      957      733     144    1056     853     123
 4  2013     1     1     1114      900     134    1447    1222     145
 5  2013     1     1     1505     1310     115    1638    1431     127
 6  2013     1     1     1525     1340     105    1831    1626     125
 7  2013     1     1     1540     1338     122    2020    1825     115
 8  2013     1     1     1549     1445      64    1912    1656     136
 9  2013     1     1     1558     1359     119    1718    1515     123
10  2013     1     1     1732     1630      62    2028    1825     123
# … with 11,412 more rows, 10 more variables: carrier <chr>,
#   flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
#   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
#   time_hour <dttm>, and abbreviated variable names 'sched_dep_time,
#   ²dep_delay, ³arr_time, ⁴sched_arr_time, ⁵arr_delay
# ℹ Use `print(n = ...)` to see more rows, and `colnames()` to see all va
riable names
~ |
```

As you can see above at times of early departures we see the data still because the arr_delay was greater than 2 hours.

4. List data only for flights that were operated by United, American, or Delta.

These need to be looked via their carrier codes United = “UA”, American = “AA” Delta = “DL”

Input: flights %>% filter(carrier == "UA" | carrier == "AA" | carrier == "DL")

Output:

```
# A tibble: 139,504 × 19
    year month   day dep_time sched_d…¹
   <int> <int> <int>    <int>     <int>
 1  2013     1     1      517       515
 2  2013     1     1      533       529
 3  2013     1     1      542       540
 4  2013     1     1      554       600
 5  2013     1     1      554       558
 6  2013     1     1      558       600
 7  2013     1     1      558       600
 8  2013     1     1      558       600
 9  2013     1     1      559       600
10  2013     1     1      559       600
# … with 139,494 more rows, 14 more
#   variables: dep_delay <dbl>,
#   arr_time <int>,
#   sched_arr_time <int>,
#   arr_delay <dbl>, carrier <chr>,
#   flight <int>, tailnum <chr>,
#   origin <chr>, dest <chr>, …
# ℹ Use `print(n = ...)` to see more rows, and `colnames()` to see all variable names
```

5. Sort data in order of fastest flights (air_time).

   Arrange function automatically does it in ascending order.

   Input: flights %>% arrange(-air_time)

   Output:

```
# A tibble: 336,776 × 19
    year month   day dep_time sched_…¹ dep_d…² arr_t…³ sched…⁴ arr_d…⁵
   <int> <int> <int>    <int>    <int>   <dbl>   <int>   <int>   <dbl>
 1  2013     3    17     1337     1335       2    1937    1836      61
 2  2013     2     6      853      900      -7    1542    1540       2
 3  2013     3    15     1001     1000       1    1551    1530      21
 4  2013     3    17     1006     1000       6    1607    1530      37
 5  2013     3    16     1001     1000       1    1544    1530      14
 6  2013     2     5      900      900       0    1555    1540      15
 7  2013    11    12      936      930       6    1630    1530      60
 8  2013     3    14      958     1000      -2    1542    1530      12
 9  2013    11    20     1006     1000       6    1639    1555      44
10  2013     3    15     1342     1335       7    1924    1836      48
# … with 336,766 more rows, 10 more variables: carrier <chr>,
#   flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
#   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
#   time_hour <dttm>, and abbreviated variable names ¹sched_dep_time,
#   ²dep_delay, ³arr_time, ⁴sched_arr_time, ⁵arr_delay
# ℹ Use `print(n = ...)` to see more rows, and `colnames()` to see all va
riable names
```

6. Sort data in order of longest duration flights (air_time).

   Longest to shortest is in descending order.

   Input: flights %>% arrange(desc(-air_time))

   Output:

```
> flights %>% arrange(desc(-air_time))
# A tibble: 336,776 × 19
    year month   day dep_time sched_…¹ dep_d…² arr_t…³ sched…⁴ arr_d…⁵
   <int> <int> <int>    <int>    <int>   <dbl>   <int>   <int>   <dbl>
 1  2013     1    16     1355     1315      40    1442    1411      31
 2  2013     4    13      537      527      10     622     628      -6
 3  2013    12     6      922      851      31    1021     954      27
 4  2013     2     3     2153     2129      24    2247    2224      23
 5  2013     2     5     1303     1315     -12    1342    1411     -29
 6  2013     2    12     2123     2130      -7    2211    2225     -14
 7  2013     3     2     1450     1500     -10    1547    1608     -21
 8  2013     3     8     2026     1935      51    2131    2056      35
 9  2013     3    18     1456     1329      87    1533    1426      67
10  2013     3    19     2226     2145      41    2305    2246      19
# … with 336,766 more rows, 10 more variables: carrier <chr>,
#   flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
#   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
#   time_hour <dttm>, and abbreviated variable names 'sched_dep_time,
#   ²dep_delay, ³arr_time, ⁴sched_arr_time, ⁵arr_delay
# ℹ Use `print(n = ...)` to see more rows, and `colnames()` to see all va
riable names
```

7. Show only the origin and destination of flights sorted by longest flights.

   In this one arrange must be used first as select will limit our observations to only columns dest or origin.

   Input: flights %>% arrange(desc(-air_time)) %>% select(origin, dest)

   Output:

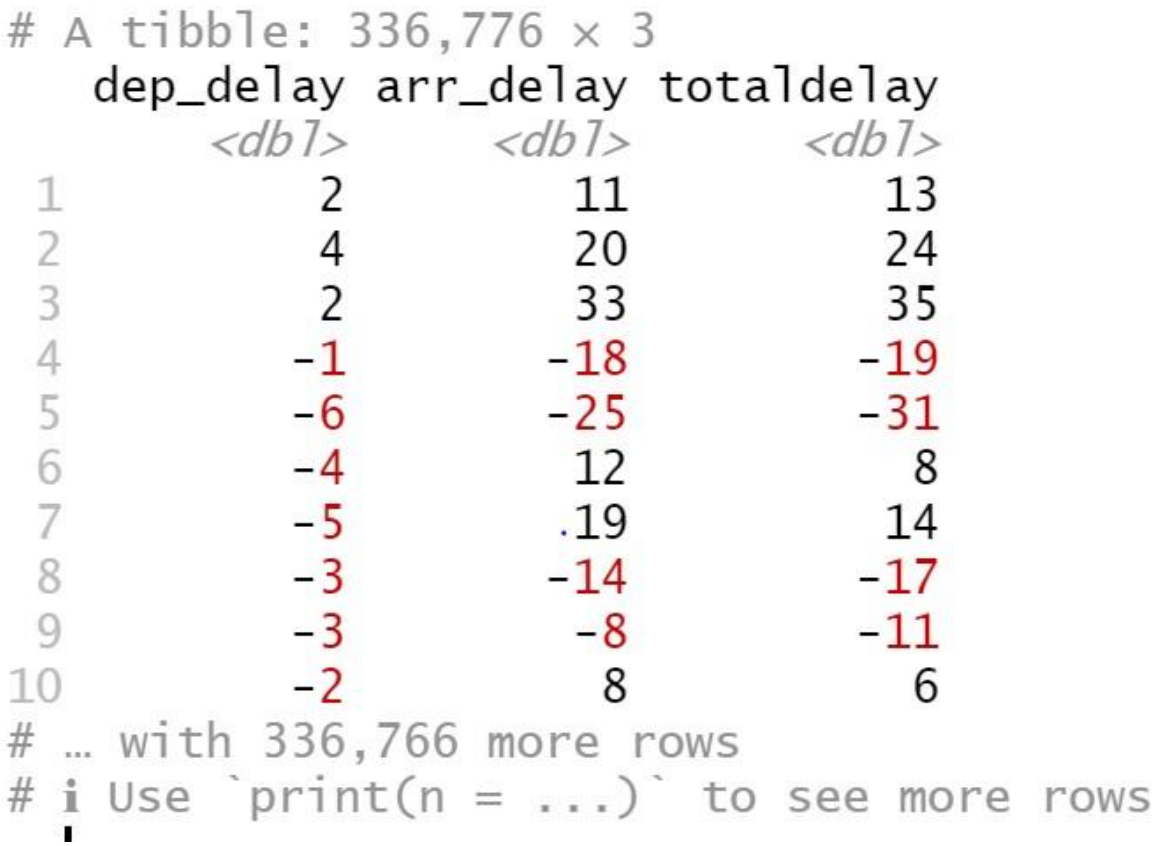
```
# A tibble: 336,776 × 2
   origin dest
   <chr>  <chr>
 1 EWR    BDL
 2 EWR    BDL
 3 EWR    BDL
 4 EWR    PHL
 5 EWR    BDL
 6 EWR    PHL
 7 LGA    BOS
 8 JFK    PHL
 9 EWR    BDL
10 EWR    BDL
# … with 336,766 more rows
# ℹ Use `print(n = ...)` to see more rows
```

8. Add a new variable that indicates the total delay (both departure and arrival delay).

   Input: flights %>% mutate(totaldelay = dep_delay + arr_delay)

   Output: to show any real output I will use an example where I output totaldelay, dep_delay, and arr_delay.

   flights %>% mutate(totaldelay = dep_delay + arr_delay) %>% select(dep_delay, arr_delay, totaldelay)



```
# A tibble: 336,776 × 3
   dep_delay arr_delay totaldelay
       <dbl>     <dbl>      <dbl>
 1         2        11         13
 2         4        20         24
 3         2        33         35
 4        -1       -18        -19
 5        -6       -25        -31
 6        -4        12          8
 7        -5        19         14
 8        -3       -14        -17
 9        -3        -8        -11
10        -2         8          6
# … with 336,766 more rows
# ℹ Use `print(n = ...)` to see more rows
```

To permanently keep our new total delay we must save the output back into flights.

9. Show only the origin and destination of flights sorted by descending order of total delay.

   Input: flights %>% mutate(totaldelay = dep_delay + arr_delay) %>%

   arrange(desc(totaldelay)) %>% select(origin, dest)

   Output:

```
# A tibble: 336,776 × 2
   origin dest
   <chr>  <chr>
 1 EWR    SFO
 2 JFK    SFO
 3 LGA    MSY
 4 JFK    PHX
 5 JFK    MKE
 6 EWR    SFO
 7 EWR    SEA
 8 JFK    MCI
 9 LGA    DEN
10 LGA    DSM
# … with 336,766 more rows
# ℹ Use `print(n = ...)` to see more rows
```

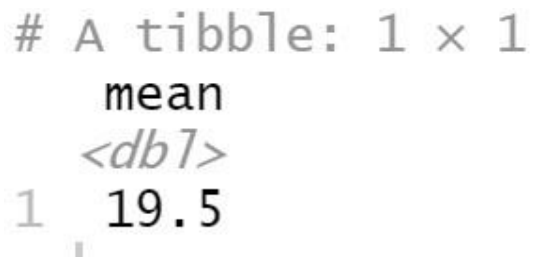
10. Show only the origin and destination of 10 most delayed flights [Hint: there are multiple ways of solving this. Some additional functions that you will find useful are head(), slice(), min_rank().]

    Input: flights %>% mutate(totaldelay = dep_delay + arr_delay) %>% arrange(desc(totaldelay)) %>% select(origin, dest) %>% slice(1:10) Output:

```
# A tibble: 10 × 2
   origin dest
   <chr>  <chr>
 1 EWR    SFO
 2 JFK    SFO
 3 LGA    MSY
 4 JFK    PHX
 5 JFK    MKE
 6 EWR    SFO
 7 EWR    SEA
 8 JFK    MCI
 9 LGA    DEN
10 LGA    DSM
```

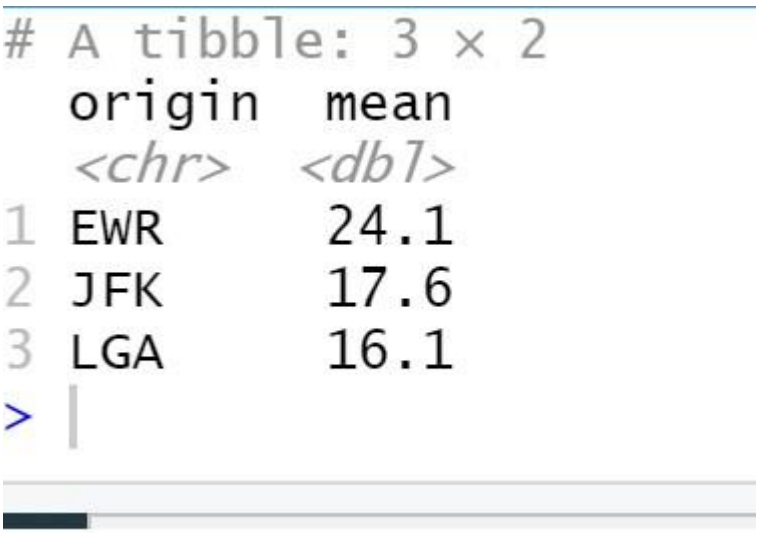
11. Show the average total delay for all flights.

Input: flights %>% mutate(totaldelay = dep_delay + arr_delay) %>% summarize(mean = mean(totaldelay, na.rm = TRUE)) Output:



12. Show the average total delay for every departure city.

    Input: flights %>% mutate(totaldelay = dep_delay + arr_delay) %>% group_by(origin) %>% summarize(mean = mean(totaldelay, na.rm = TRUE)) Output:



13. Show the average total delay for every departure-arrival city pair.

    Input: flights %>% mutate(totaldelay = dep_delay + arr_delay) %>% group_by(origin, dest) %>% summarize(mean = mean(totaldelay, na.rm = TRUE))

    Output:

```
# A tibble: 224 × 3
# Groups:   origin [3]
   origin dest   mean
   <chr>  <chr> <dbl>
 1 EWR    ALB    37.8
 2 EWR    ANC    10.4
 3 EWR    ATL    28.6
 4 EWR    AUS    11
 5 EWR    AVL    17.4
 6 EWR    BDL    24.8
 7 EWR    BNA    30.3
 8 EWR    BOS    17.3
 9 EWR    BQN    34.5
10 EWR    BTV    30.0
# … with 214 more rows
# ℹ Use `print(n = ...)` to see more rows
```