Week 2 Exercises

Kaylynn Hiller

2026-01-20

Week 2

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.1     v stringr   1.6.0
v ggplot2   4.0.0     v tibble    3.3.1
v lubridate 1.9.4     v tidyr     1.3.2
v purrr     1.2.1
-- 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 becom
```

Exercise 3.2.5

1. In a single pipeline for each condition, find all flights that meet the condition:

- Had an arrival delay of two or more hours

```
# A tibble: 10,200 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      811            630       101     1047            830
 2  2013     1     1      848           1835       853     1001           1950
 3  2013     1     1      957            733       144     1056            853
 4  2013     1     1     1114            900       134     1447           1222
 5  2013     1     1     1505           1310       115     1638           1431
 6  2013     1     1     1525           1340       105     1831           1626
 7  2013     1     1     1549           1445        64     1912           1656
 8  2013     1     1     1558           1359       119     1718           1515
 9  2013     1     1     1732           1630        62     2028           1825
10  2013     1     1     1803           1620       103     2008           1750
# i 10,190 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>
```

- Flew to Houston (`IAH` or `HOU`)

```
# A tibble: 9,313 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      623            627        -4      933            932
 4  2013     1     1      728            732        -4     1041           1038
 5  2013     1     1      739            739         0     1104           1038
 6  2013     1     1      908            908         0     1228           1219
 7  2013     1     1     1028           1026         2     1350           1339
 8  2013     1     1     1044           1045        -1     1352           1351
 9  2013     1     1     1114            900       134     1447           1222
10  2013     1     1     1205           1200         5     1503           1505
# i 9,303 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>
```

- Were operated by United, American, or Delta

```
# A tibble: 139,504 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      554            600        -6      812            837
 5  2013     1     1      554            558        -4      740            728
 6  2013     1     1      558            600        -2      753            745
 7  2013     1     1      558            600        -2      924            917
 8  2013     1     1      558            600        -2      923            937
 9  2013     1     1      559            600        -1      941            910
10  2013     1     1      559            600        -1      854            902
```

```
# i 139,494 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>
```

- Departed in summer (July, August, and September)

```
# A tibble: 86,326 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     7     1        1           2029       212      236           2359
 2  2013     7     1        2           2359         3      344            344
 3  2013     7     1       29           2245       104      151              1
 4  2013     7     1       43           2130       193      322             14
 5  2013     7     1       44           2150       174      300            100
 6  2013     7     1       46           2051       235      304           2358
 7  2013     7     1       48           2001       287      308           2305
 8  2013     7     1       58           2155       183      335             43
 9  2013     7     1      100           2146       194      327             30
10  2013     7     1      100           2245       135      337            135
# i 86,316 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>
```

- Arrived more than two hours late but didn’t leave late

```
# A tibble: 29 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    27     1419           1420        -1     1754           1550
 2  2013    10     7     1350           1350         0     1736           1526
 3  2013    10     7     1357           1359        -2     1858           1654
 4  2013    10    16      657            700        -3     1258           1056
 5  2013    11     1      658            700        -2     1329           1015
 6  2013     3    18     1844           1847        -3       39           2219
 7  2013     4    17     1635           1640        -5     2049           1845
 8  2013     4    18      558            600        -2     1149            850
 9  2013     4    18      655            700        -5     1213            950
10  2013     5    22     1827           1830        -3     2217           2010
# i 19 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>
```

- Were delayed by at least an hour, but made up over 30 minutes in flight

```
# A tibble: 1,844 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     2205           1720       285       46           2040
 2  2013     1     1     2326           2130       116      131             18
 3  2013     1     3     1503           1221       162     1803           1555
 4  2013     1     3     1839           1700        99     2056           1950
 5  2013     1     3     1850           1745        65     2148           2120
 6  2013     1     3     1941           1759       102     2246           2139
 7  2013     1     3     1950           1845        65     2228           2227
 8  2013     1     3     2015           1915        60     2135           2111
 9  2013     1     3     2257           2000       177       45           2224
10  2013     1     4     1917           1700       137     2135           1950
# i 1,834 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. Sort flights to find the flights with the longest departure delays. Find the flights that left earliest in the morning.

```
# 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>
# 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    13        1           2249        72      108           2357
 2  2013     1    31        1           2100       181      124           2225
 3  2013    11    13        1           2359         2      442            440
 4  2013    12    16        1           2359         2      447            437
 5  2013    12    20        1           2359         2      430            440
 6  2013    12    26        1           2359         2      437            440
 7  2013    12    30        1           2359         2      441            437
 8  2013     2    11        1           2100       181      111           2225
 9  2013     2    24        1           2245        76      121           2354
10  2013     3     8        1           2355         6      431            440
# 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>
```

3. Sort flights to find the fastest flights. (Hint: Try including a math calculation inside of your function.)

```
# A tibble: 336,776 x 20
   speed  year month   day dep_time sched_dep_time dep_delay arr_time
   <dbl> <int> <int> <int>    <int>          <int>     <dbl>    <int>
 1 1132.  2013     5    25     1709           1700         9     1923
 2 1047.  2013     7     2     1558           1513        45     1745
 3 1043.  2013     5    13     2040           2025        15     2225
 4 1032.  2013     3    23     1914           1910         4     2045
 5  952.  2013     1    12     1559           1600        -1     1849
 6  908.  2013    11    17      650            655        -5     1059
 7  897.  2013     2    21     2355           2358        -3      412
 8  896.  2013    11    17      759            800        -1     1212
 9  892.  2013    11    16     2003           1925        38       17
10  892.  2013    11    16     2349           2359       -10      402
# 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>
```

4. Was there a flight on every day of 2013? Yes

```
# A tibble: 1 x 1
```

```
  num_days
     <int>
1      365
```

5. Which flights traveled the farthest distance? Which traveled the least distance?

```
# 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      857            900        -3     1516           1530
 2  2013     1     2      909            900         9     1525           1530
 3  2013     1     3      914            900        14     1504           1530
 4  2013     1     4      900            900         0     1516           1530
 5  2013     1     5      858            900        -2     1519           1530
 6  2013     1     6     1019            900        79     1558           1530
 7  2013     1     7     1042            900       102     1620           1530
 8  2013     1     8      901            900         1     1504           1530
 9  2013     1     9      641            900      1301     1242           1530
10  2013     1    10      859            900        -1     1449           1530
# 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>
```

```
# 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     7    27       NA            106        NA       NA            245
 2  2013     1     3     2127           2129        -2     2222           2224
 3  2013     1     4     1240           1200        40     1333           1306
 4  2013     1     4     1829           1615       134     1937           1721
 5  2013     1     4     2128           2129        -1     2218           2224
 6  2013     1     5     1155           1200        -5     1241           1306
 7  2013     1     6     2125           2129        -4     2224           2224
 8  2013     1     7     2124           2129        -5     2212           2224
 9  2013     1     8     2127           2130        -3     2304           2225
10  2013     1     9     2126           2129        -3     2217           2224
# 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>
```

6. Does it matter what order you used filter() and arrange() if you're using both? Why/why not? Think about the results and how much work the functions would have to do.

It does not matter. Filter pulls out rows that meet a condition but does not change the ordering. Arrange changes the ordering but leaves all rows in. Since they perform separate actions, the order does not matter.

Exercise 3.3.5

1. Compare dep_time, sched_dep_time, and dep_delay. How would you expect those three numbers to be related?

I would expect the scheduled departure time (sched_dep_time) plus the delay (dep_delay) to equal the actual departure time (dep_time).

```
# A tibble: 336,776 x 3
   dep_time sched_dep_time dep_delay
      <int>          <int>     <dbl>
 1      517            515         2
 2      533            529         4
 3      542            540         2
 4      544            545        -1
 5      554            600        -6
 6      554            558        -4
 7      555            600        -5
 8      557            600        -3
 9      557            600        -3
10      558            600        -2
# i 336,766 more rows
```

2. Brainstorm as many ways as possible to select dep_time, dep_delay, arr_time, and arr_delay from flights.

```
# 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>

# A tibble: 336,776 x 4
   dep_time dep_delay arr_time arr_delay
      <int>     <dbl>    <int>     <dbl>
 1      517         2      830        11
 2      533         4      850        20
 3      542         2      923        33
 4      544        -1     1004       -18
 5      554        -6      812       -25
 6      554        -4      740        12
 7      555        -5      913        19
 8      557        -3      709       -14
 9      557        -3      838        -8
10      558        -2      753         8
# i 336,766 more rows

# A tibble: 336,776 x 6
   dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay
      <int>          <int>     <dbl>    <int>          <int>     <dbl>
 1      517            515         2      830            819        11
 2      533            529         4      850            830        20
 3      542            540         2      923            850        33
 4      544            545        -1     1004           1022       -18
 5      554            600        -6      812            837       -25
 6      554            558        -4      740            728        12
 7      555            600        -5      913            854        19
 8      557            600        -3      709            723       -14
 9      557            600        -3      838            846        -8
10      558            600        -2      753            745         8
# i 336,766 more rows

# A tibble: 336,776 x 4
   dep_time dep_delay arr_time arr_delay
      <int>     <dbl>    <int>     <dbl>
 1      517         2      830        11
 2      533         4      850        20
 3      542         2      923        33
 4      544        -1     1004       -18
 5      554        -6      812       -25
 6      554        -4      740        12
 7      555        -5      913        19
 8      557        -3      709       -14
 9      557        -3      838        -8
10      558        -2      753         8
# i 336,766 more rows
```

```
# A tibble: 336,776 x 4
   dep_time dep_delay arr_time arr_delay
      <int>     <dbl>    <int>     <dbl>
 1      517         2      830        11
 2      533         4      850        20
 3      542         2      923        33
 4      544        -1     1004       -18
 5      554        -6      812       -25
 6      554        -4      740        12
 7      555        -5      913        19
 8      557        -3      709       -14
 9      557        -3      838        -8
10      558        -2      753         8
# i 336,766 more rows
```

3. What happens if you specify the name of the same variable multiple times in a select() call?

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

You just get the variable once.

4. What does the any_of() function do? Why might it be helpful in conjunction with this vector?

```
# A tibble: 336,776 x 5
    year month   day dep_delay arr_delay
   <int> <int> <int>     <dbl>     <dbl>
 1  2013     1     1         2        11
 2  2013     1     1         4        20
 3  2013     1     1         2        33
 4  2013     1     1        -1       -18
 5  2013     1     1        -6       -25
 6  2013     1     1        -4        12
 7  2013     1     1        -5        19
 8  2013     1     1        -3       -14
 9  2013     1     1        -3        -8
10  2013     1     1        -2         8
# i 336,766 more rows
```

The any_of() function in conjuction with the select function selects any variables listed. If you set specific variables to select outside the pipeline, the code is cleaner and easier to change.

5. Does the result of running the following code surprise you? How do the select helpers deal with upper and lower case by default? How can you change that default?

```
# A tibble: 336,776 x 6
   dep_time sched_dep_time arr_time sched_arr_time air_time time_hour
      <int>          <int>    <int>          <int>    <dbl> <dttm>
 1      517            515      830            819      227 2013-01-01 05:00:00
 2      533            529      850            830      227 2013-01-01 05:00:00
 3      542            540      923            850      160 2013-01-01 05:00:00
 4      544            545     1004           1022      183 2013-01-01 05:00:00
 5      554            600      812            837      116 2013-01-01 06:00:00
 6      554            558      740            728      150 2013-01-01 05:00:00
 7      555            600      913            854      158 2013-01-01 06:00:00
 8      557            600      709            723       53 2013-01-01 06:00:00
 9      557            600      838            846      140 2013-01-01 06:00:00
10      558            600      753            745      138 2013-01-01 06:00:00
# i 336,766 more rows
```

The result did not surprise me. By default, select helpers treat upper and lower case the same. You can change this by specifying with ignore.case = FALSE.

6. Rename air_time to air_time_min to indicate units of measurement and move it to the beginning of the data frame.

```
# A tibble: 336,776 x 19
   air_time_min  year month   day dep_time sched_dep_time dep_delay arr_time
          <dbl> <int> <int> <int>    <int>          <int>     <dbl>    <int>
 1          227  2013     1     1      517            515         2      830
 2          227  2013     1     1      533            529         4      850
 3          160  2013     1     1      542            540         2      923
 4          183  2013     1     1      544            545        -1     1004
 5          116  2013     1     1      554            600        -6      812
 6          150  2013     1     1      554            558        -4      740
 7          158  2013     1     1      555            600        -5      913
 8           53  2013     1     1      557            600        -3      709
 9          140  2013     1     1      557            600        -3      838
10          138  2013     1     1      558            600        -2      753
# i 336,766 more rows
# i 11 more variables: sched_arr_time <int>, arr_delay <dbl>, carrier <chr>,
#   flight <int>, tailnum <chr>, origin <chr>, dest <chr>, distance <dbl>,
#   hour <dbl>, minute <dbl>, time_hour <dttm>
```

7. Why doesn't the following work, and what does the error mean?

We did not select the arr_delay variable in the select function so it is not available to arrange. Select keeps only the columns listed.

Exercise 3.5.7

1. Which carrier has the worst average delays? Challenge: can you disentangle the effects of bad airports vs. bad carriers? Why/why not? (Hint: think about...)

```
# A tibble: 16 x 2
   carrier avg_dep_del
   <chr>         <dbl>
 1 F9            20.2
 2 EV            20.0
 3 YV            19.0
 4 FL            18.7
 5 WN            17.7
 6 9E            16.7
 7 B6            13.0
 8 VX            12.9
 9 OO            12.6
```

```
10 UA            12.1
11 MQ            10.6
12 DL             9.26
13 AA             8.59
14 AS             5.80
15 HA             4.90
16 US             3.78
```

```
# A tibble: 1 x 4
  carrier dest  origin     n
  <chr>   <chr> <chr>  <int>
1 F9      DEN   LGA      685
```

```
# A tibble: 5 x 3
  carrier dest      n
  <chr>   <chr> <int>
1 B6      DEN     338
2 DL      DEN    1043
3 F9      DEN     685
4 UA      DEN    3796
5 WN      DEN    1404
```

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

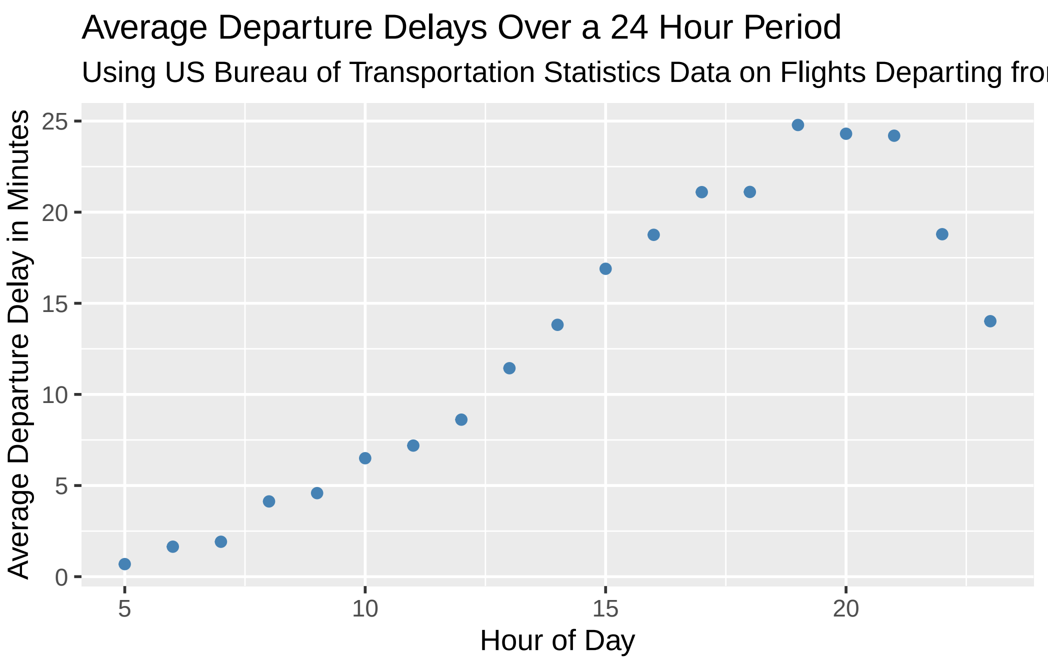
```
# A tibble: 8 x 5
# Groups:   carrier, dest [5]
  carrier dest  origin     n avg_dd
  <chr>   <chr> <chr>  <int>  <dbl>
1 DL      DEN   LGA      678   9.82
2 UA      DEN   LGA     1626  10.6
3 UA      DEN   EWR     2170  14.1
4 WN      DEN   LGA      715  15.4
5 DL      DEN   JFK      365  16.6
6 F9      DEN   LGA      685  20.2
7 B6      DEN   JFK      338  23.9
8 WN      DEN   EWR      689  24.4
```

The carrier with the worst average delays is Frontier airlines. However, they are headquartered in Denver and this data set only includes flights from LGA to DEN. This means there's no variance in F9 across airports since there is only one flight path being recorded. We cannot disentangle the effects of bad airports from this data set and would have to make a lot of assumptions in order to do so.

2. Find the flights that are most delayed upon departure to each destination.

```
# A tibble: 105 x 19
# Groups:   dest [105]
   dest  carrier dep_delay  year month   day dep_time sched_dep_time arr_time
   <chr> <chr>       <dbl> <int> <int> <int>    <int>          <int>    <int>
 1 HNL   HA           1301  2013     1     9      641            900     1242
 2 CMH   MQ           1137  2013     6    15     1432           1935     1607
 3 ORD   MQ           1126  2013     1    10     1121           1635     1239
 4 SFO   AA           1014  2013     9    20     1139           1845     1457
 5 CVG   MQ           1005  2013     7    22      845           1600     1044
 6 TPA   DL            960  2013     4    10     1100           1900     1342
 7 MSP   DL            911  2013     3    17     2321            810      135
 8 PDX   DL            899  2013     6    27      959           1900     1236
 9 ATL   DL            898  2013     7    22     2257            759      121
10 MIA   AA            896  2013    12     5      756           1700     1058
# i 95 more rows
# i 10 more variables: sched_arr_time <int>, arr_delay <dbl>, flight <int>,
#   tailnum <chr>, origin <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
#   minute <dbl>, time_hour <dttm>
```

3. How do delays vary over the course of the day? Illustrate your answer with a plot.



Delays do vary.

4. What happens if you supply a negative n to slice_min() and friends?

```
# 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
# 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>
```

```
# 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
# 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>
```

The result is listing the rows based on the smallest to largest dep_delay value. This result is the same as arranging the values by dep_delay.

5. Explain what count() does in terms of the dplyr verbs you just learned. What does the sort argument to count() do?

```
# A tibble: 3 x 2
  origin      n
  <chr>   <int>
1 EWR    120835
2 JFK    111279
3 LGA    104662
```

```
# A tibble: 3 x 2
  origin      n
  <chr>   <int>
1 EWR    120835
2 JFK    111279
3 LGA    104662
```

Count allows you to count the unique values of one or more variables similar to group_by and summarize. The sort options causes the results to be presented in descending order.

6. Suppose we have the following tiny data frame:

a. Write down what you think the output will look like, then check if you were correct, and describe what group_by() does.

```
# A tibble: 5 x 3
# Groups:   y [2]
      x y     z
  <int> <chr> <chr>
1     1 a     K
2     2 b     K
3     3 a     L
4     4 a     L
5     5 b     K
```

This command should group by the two groups in y.

b. Write down what you think the output will look like, then check if you were correct, and describe what arrange() does. Also, comment on how it's different from the group_by() in part (a).

```
# A tibble: 5 x 3
      x y     z
  <int> <chr> <chr>
1     1 a     K
2     3 a     L
3     4 a     L
4     2 b     K
5     5 b     K
```

This command should arrange the results by the y column alphabetically. This means it should be list the results as a,a,a,b,b. The row order changed in using arrange() and not group_by().

c. Write down what you think the output will look like, then check if you were correct, and describe what the pipeline does.

```
# A tibble: 2 x 2
  y     mean_x
  <chr>  <dbl>
1 a       2.67
2 b       3.5
```

The result should be a summary of the mean of x based on the two groups of y. The pipeline passes the group_by() function along into the summarize() function so the means are groups by y values.

d. Write down what you think the output will look like, then check if you were correct, and describe what the pipeline does. Then, comment on what the message says.

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

```
# A tibble: 3 x 3
# Groups:   y [2]
  y     z     mean_x
  <chr> <chr>  <dbl>
1 a     K        1
2 a     L        3.5
3 b     K        3.5
```

The result should be a summary of the mean of x based on the three groups of y and z (a K, a L, a K). This pipeline passes the group_by() function along into the summarize() function so the means are groups by y and z values. The message means that the results are grouped by y values first and the grouping can be overriden using the .groups argument.

e. Write down what you think the output will look like, then check if you were correct, and describe what the pipeline does. How is the output different from the one in part (d)?

```
# A tibble: 3 x 3
  y     z     mean_x
  <chr> <chr>  <dbl>
1 a     K        1
2 a     L        3.5
3 b     K        3.5
```

The answer should be the same except the results are not grouped.

f. Write down what you think the outputs will look like, then check if you were correct, and describe what each pipeline does. How are the outputs of the two pipelines different?

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

```
# A tibble: 3 x 3
# Groups:   y [2]
  y     z     mean_x
  <chr> <chr>  <dbl>
```

```
1 a     K        1
2 a     L        3.5
3 b     K        3.5

# A tibble: 5 x 4
# Groups:   y, z [3]
      x y     z     mean_x
  <int> <chr> <chr>  <dbl>
1     1 a     K        1
2     2 b     K        3.5
3     3 a     L        3.5
4     4 a     L        3.5
5     5 b     K        3.5
```

The first pipeline calculated the means for the three combinations of the y and z column values. The second pipeline creates a new column where these means are stored. The output of the first pipeline is three rows and five rows for the second.