# Assignment D

```
library(nycflights13)
library(dplyr)
```

```

Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
library(ggplot2)
```

```
#1 Had an arrival delay of two or more hours
flights %>%
  filter(arr_delay >= 120)
```

```
# A tibble: 10,200 × 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>
```

```
#2 Flew to Houston (IAH or HOU)
flights %>%
  filter(dest %in% c("IAH", "HOU"))
```

```
# A tibble: 9,313 × 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>
```

```
#3 Were operated by United, American, or Delta
flights %>%
  filter(carrier %in% c("UA", "AA", "DL"))
```

```
# A tibble: 139,504 × 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>
```

```
#4 Departed in summer (July, August, and September)
flights %>%
  filter(month %in% c(7, 8, 9))
```

```
# A tibble: 86,326 × 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>
```

```
#5 Arrived more than two hours late but didn’t leave late
flights %>%
  filter(arr_delay > 120, dep_delay <= 0)
```

```
# A tibble: 29 × 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>
```

```
#6 Were delayed by at least an hour, but made up over 30 minutes in flight
flights %>%
  filter(dep_delay >= 60, (dep_delay - arr_delay) > 30)
```

```
# A tibble: 1,844 × 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>
```

## problem 4

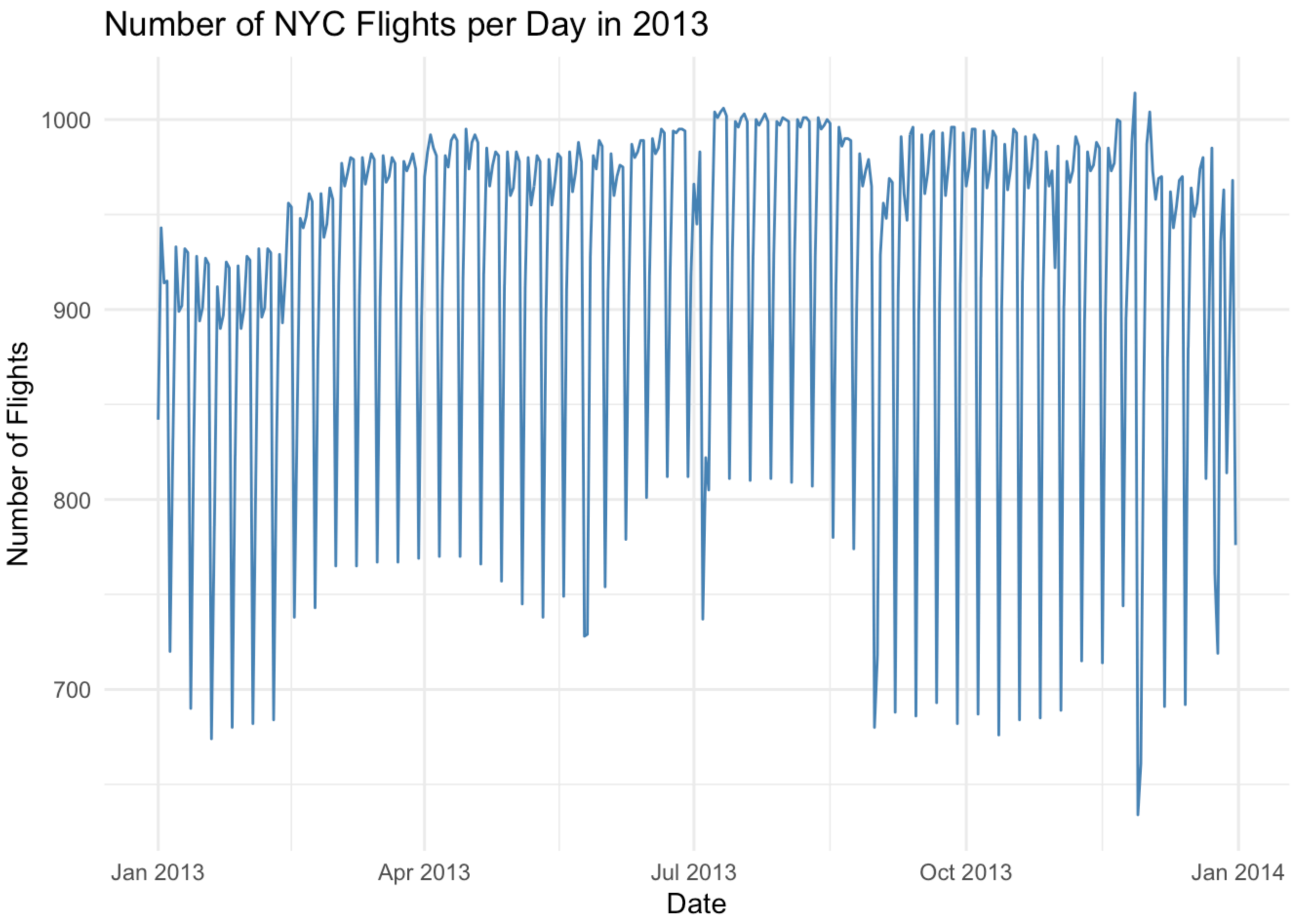
```r
        library(nycflights13)
        library(dplyr)
        library(ggplot2)

        # Count the number of flights per day
        daily_flights <- flights %>%
          group_by(year, month, day) %>%
          summarise(n = n(), .groups = "drop")

        # Check if there are any days with zero flights
        any_zero_days <- any(daily_flights$n == 0)
        any_zero_days  # FALSE means there was at least one flight every day
```

```
[1] FALSE
```

```r
        # List the days with zero flights (if any)
        daily_flights %>%
          filter(n == 0)
```

```
# A tibble: 0 × 4
# ℹ 4 variables: year <int>, month <int>, day <int>, n <int>
```

```r
        # Visualization: number of flights per day in 2013
        ggplot(daily_flights, aes(x = as.Date(sprintf("%d-%02d-%02d", year, month, day)),
          geom_line(color = "steelblue") +
          labs(title = "Number of NYC Flights per Day in 2013",
               x = "Date",
               y = "Number of Flights") +
          theme_minimal()
```



## problem 5

```
# Flight(s) with the farthest distance
farthest <- flights %>%
  filter(distance == max(distance, na.rm = TRUE))

# Flight(s) with the shortest distance
shortest <- flights %>%
  filter(distance == min(distance, na.rm = TRUE))

farthest
```

```
# A tibble: 342 × 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 332 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>
```

```
shortest
```

```
# A tibble: 1 × 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
# 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>
```

# Exercise3.3.5

## problem 1

```
hm_to_min <- function(x) {
  hour <- x %/% 100
  minute <- x %% 100
  hour * 60 + minute
}

flights_check <- flights %>%
  filter(!is.na(dep_time), !is.na(sched_dep_time)) %>%
  mutate(
    dep_time_min = hm_to_min(dep_time),
    sched_dep_time_min = hm_to_min(sched_dep_time),
    recomputed_delay = dep_time_min - sched_dep_time_min,
    # fix overnight issue: if recomputed < -1000, add 1440 minutes
    recomputed_delay = if_else(recomputed_delay < -1000,
                               recomputed_delay + 1440,
                               recomputed_delay),
    diff = recomputed_delay - dep_delay
  )

summary(flights_check$diff)
```

```
      Min.    1st Qu.     Median       Mean    3rd Qu.       Max. 
-1440.0000     0.0000     0.0000    -0.1972     0.0000     0.0000 
```

## problem 4

```
        variables <- c("year", "month", "day", "dep_delay", "arr_delay")

        # Example 1: using any_of() - safe, skips missing vars
        flights %>%
          select(any_of(variables)) %>%
          head()
```

```
# A tibble: 6 × 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
```

```
        # Example 2: what happens if a variable name is wrong
        bad_vars <- c("year", "month", "day", "arr_dlay")  # typo: arr_delay -> arr_dlay

        # any_of(): will quietly skip the missing column
        flights %>%
          select(any_of(bad_vars)) %>%
          head()
```

```
# A tibble: 6 × 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
```

```
        # all_of(): will throw an error if any column is missing
```

# Exercise3.3.7

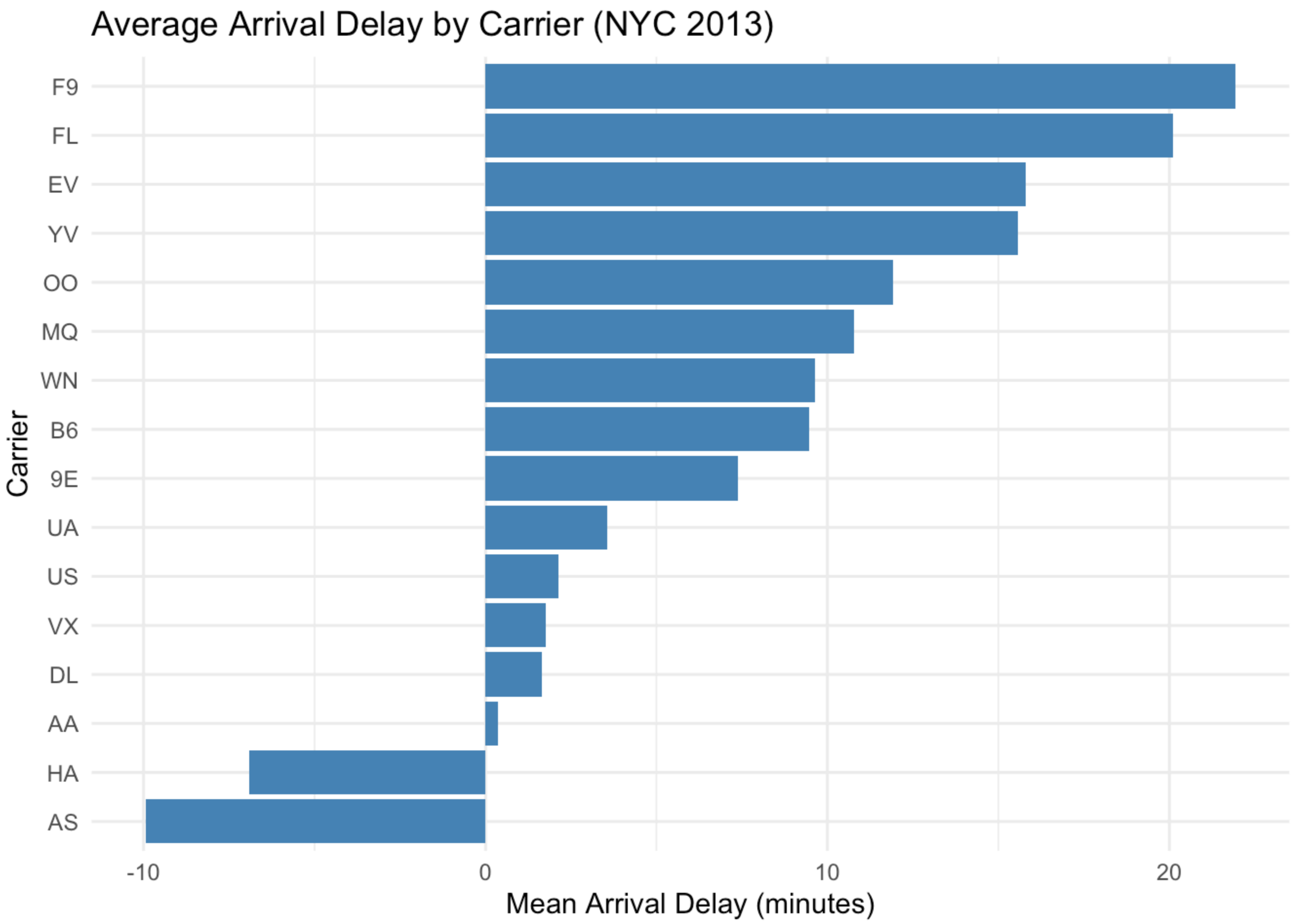
## problem 1

```
        library(nycflights13)
        library(dplyr)
```

```r
library(ggplot2)

# 1. Average arrival delay by carrier
avg_delay <- flights %>%
  group_by(carrier) %>%
  summarise(
    mean_arr_delay = mean(arr_delay, na.rm = TRUE),
    n = n(),
    .groups = "drop"
  ) %>%
  arrange(desc(mean_arr_delay))

print(avg_delay)
```

```
# A tibble: 16 × 3
   carrier mean_arr_delay     n
   <chr>            <dbl> <int>
 1 F9              21.9     685
 2 FL              20.1    3260
 3 EV              15.8   54173
 4 YV              15.6     601
 5 OO              11.9      32
 6 MQ              10.8   26397
 7 WN               9.65  12275
 8 B6               9.46  54635
 9 9E               7.38  18460
10 UA               3.56  58665
11 US               2.13  20536
12 VX               1.76   5162
13 DL               1.64  48110
14 AA               0.364 32729
15 HA              -6.92    342
16 AS              -9.93    714
```

```r
# Plot: average delay by carrier
ggplot(avg_delay, aes(x = reorder(carrier, mean_arr_delay), y = mean_arr_delay))
  geom_col(fill = "steelblue") +
  coord_flip() +
  labs(title = "Average Arrival Delay by Carrier (NYC 2013)",
       x = "Carrier",
       y = "Mean Arrival Delay (minutes)") +
  theme_minimal()
```



```
# 2. Carrier × destination delays
carrier_dest_delay <- flights %>%
  group_by(carrier, dest) %>%
  summarise(
    mean_delay = mean(arr_delay, na.rm = TRUE),
    n = n(),
    .groups = "drop"
  )

# Weighted average (accounting for # of flights per destination)
weighted_delays <- carrier_dest_delay %>%
  group_by(carrier) %>%
  summarise(weighted_mean_delay = weighted.mean(mean_delay, n), .groups = "drop")
  arrange(desc(weighted_mean_delay))

print(weighted_delays)
```

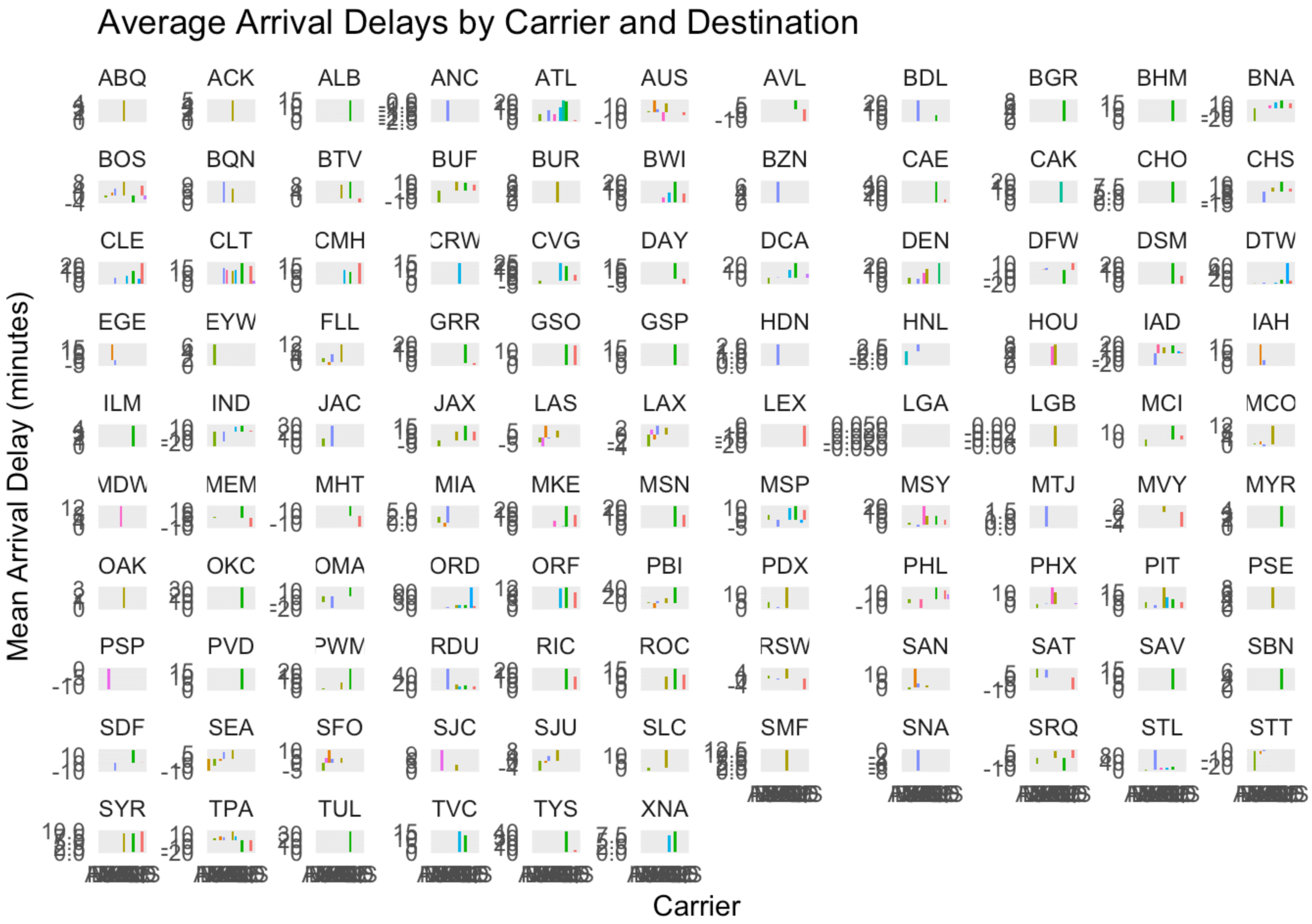
```
# A tibble: 16 × 2
   carrier weighted_mean_delay
   <chr>                 <dbl>
 1 F9                     21.9
 2 FL                     20.1
 3 EV                     15.8
 4 YV                   15.6 
 5 OO                   11.3 
 6 MQ                   10.8 
 7 WN                    9.65
 8 B6                    9.46
 9 UA                    3.57
10 VX                    1.77
11 DL                    1.65
12 AA                    0.359
13 HA                   -6.92
14 AS                   -9.93
15 9E                 NaN   
16 US                 NaN   
```

```r
# 3. Compare carriers at the same destination
# This shows whether a "bad carrier" is still worse at the same airport
dest_carrier_compare <- flights %>%
  group_by(dest, carrier) %>%
  summarise(mean_delay = mean(arr_delay, na.rm = TRUE),
            n = n(),
            .groups = "drop") %>%
  arrange(dest, desc(mean_delay))

head(dest_carrier_compare, 20)  # show top few
```

```
# A tibble: 20 × 4
   dest  carrier mean_delay     n
   <chr> <chr>        <dbl> <int>
 1 ABQ   B6           4.38    254
 2 ACK   B6           4.85    265
 3 ALB   EV          14.4     439
 4 ANC   UA          -2.5       8
 5 ATL   FL          20.7    2337
 6 ATL   EV          19.6    1764
 7 ATL   MQ          14.0    2322
 8 ATL   UA          10.5     103
 9 ATL   DL           7.42  10571
10 ATL   WN           6.90     59
11 ATL   9E           0.857    59
12 AUS   AA          16.2     365
13 AUS   B6          11.7     747
14 AUS   UA           4.28    670
15 AUS   DL           1.41    357
16 AUS   9E          -3.5       2
17 AUS   WN         -11.2     298
18 AVL   EV           8.80    265
19 AVL   9E         -12.1      10
20 BDL   UA          22.6       8
```

```r
# 4. Visualization: mean delay by carrier, faceted by destination
ggplot(dest_carrier_compare, aes(x = reorder(carrier, mean_delay), y = mean_delay
  geom_col() +
  facet_wrap(~ dest, scales = "free_y") +
  theme_minimal() +
  theme(legend.position = "none") +
  labs(title = "Average Arrival Delays by Carrier and Destination",
       x = "Carrier",
       y = "Mean Arrival Delay (minutes)")
```

```
Warning: Removed 2 rows containing missing values or values outside the scale range
(`geom_col()`).
```



## problem 2

```r
# Find the most delayed departure for each destination
worst_dep_by_dest <- flights %>%
  filter(!is.na(dep_delay)) %>%
  group_by(dest) %>%
  slice_max(order_by = dep_delay, n = 1, with_ties = FALSE) %>%
  ungroup() %>%
  select(year, month, day, carrier, flight, origin, dest,
         sched_dep_time, dep_time, dep_delay, arr_delay)

print(worst_dep_by_dest)
```

```
# A tibble: 104 × 11
    year month   day carrier flight origin dest  sched_dep_time dep_time
   <int> <int> <int> <chr>    <int> <chr>  <chr>          <int>    <int>
 1  2013    12    14 B6          65 JFK    ABQ             2001     2223
 2  2013     7    23 B6        1491 JFK    ACK              800     1139
 3  2013     1    25 EV        4309 EWR    ALB             2000      123
 4  2013     8    17 UA         887 EWR    ANC             1625     1740
 5  2013     7    22 DL        2047 LGA    ATL              759     2257
 6  2013     7    10 UA         503 EWR    AUS             1505     2056
 7  2013     6    14 EV        4519 EWR    AVL              816     1158
 8  2013     2    21 EV        4103 EWR    BDL             1316     1728
 9  2013    12     1 EV        5309 LGA    BGR             1056     1504
10  2013     4    10 EV        5038 LGA    BHM             1900       25
# ℹ 94 more rows
# ℹ 2 more variables: dep_delay <dbl>, arr_delay <dbl>
```

## problem4

```
df <- tibble(x = c(5, 2, 8, 1, 9))

# 1. Positive n with slice_min(): pick the 2 smallest values
df %>% slice_min(x, n = 2)
```

```
# A tibble: 2 × 1
      x
  <dbl>
1     1
2     2
```

```
# Expected: rows with x = 1, 2

# 2. Negative n with slice_min(): drop the 2 smallest values
df %>% slice_min(x, n = -2)
```

```
# A tibble: 3 × 1
      x
  <dbl>
1     1
2     2
3     5
```

```
# Expected: rows with x = 5, 8, 9
```

```r
# 3. Positive n with slice_max(): pick the 2 largest values
df %>% slice_max(x, n = 2)
```

```
# A tibble: 2 × 1
      x
  <dbl>
1     9
2     8
```

```r
# Expected: rows with x = 9, 8

# 4. Negative n with slice_max(): drop the 2 largest values
df %>% slice_max(x, n = -2)
```

```
# A tibble: 3 × 1
      x
  <dbl>
1     9
2     8
3     5
```

```r
# Expected: rows with x = 5, 2, 1


# Positive n: find the 5 most delayed departures
flights %>%
  filter(!is.na(dep_delay)) %>%
  slice_max(dep_delay, n = 5) %>%
  select(year, month, day, carrier, flight, origin, dest,
         sched_dep_time, dep_time, dep_delay)
```

```
# A tibble: 5 × 10
   year month   day carrier flight origin dest  sched_dep_time dep_time
  <int> <int> <int> <chr>    <int> <chr>  <chr>          <int>    <int>
1  2013     1     9 HA          51 JFK    HNL              900      641
2  2013     6    15 MQ        3535 JFK    CMH             1935     1432
3  2013     1    10 MQ        3695 EWR    ORD             1635     1121
4  2013     9    20 AA         177 JFK    SFO             1845     1139
5  2013     7    22 MQ        3075 JFK    CVG             1600      845
# ℹ 1 more variable: dep_delay <dbl>
```

```r
# Negative n: drop the 5 most delayed departures, keep the rest
flights %>%
  filter(!is.na(dep_delay)) %>%
  slice_max(dep_delay, n = -5) %>%
  summarise(total_remaining = n())
```

```
# A tibble: 1 × 1
  total_remaining
```

```
            <int>
1          328516
```

## Problem 6

### a

```
df <- tibble(
  x = 1:5,
  y = c("a", "b", "a", "a", "b"),
  z = c("K", "K", "L", "L", "K")
)
df
```

```
# A tibble: 5 × 3
      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
```

```
df |>
  group_by(y)
```

```
# A tibble: 5 × 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
```

so the data will be the same, but grouping information is added.

### b

```
df <- tibble(
  x = 1:5,
  y = c("a", "b", "a", "a", "b"),
  z = c("K", "K", "L", "L", "K")
)

df
```

```
# A tibble: 5 × 3
      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    
```

```
df |> arrange(y)
```

```
# A tibble: 5 × 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    
```

Arrange() reorders the rows of the data frame according to the values of one or more columns.

### C

We expect to see a 2x2 table.

```
df |>
  group_by(y) |>
  summarize(mean_x = mean(x))
```

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

group_by(y) tells R to treat rows with the same value of y as belonging to the same group, and summarize(mean_x = mean(x)) then calculates the mean of x within each group. The result is a collapsed summary table that contains one row per group along with the group labels and their corresponding summary statistics.

### D

```
df |>
  group_by(y, z) |>
  summarize(mean_x = mean(x))
```

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

# A tibble: 3 × 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
```

group_by(y, z) divides the data into groups defined by each unique combination of y and z, and summarize(mean_x = mean(x)) then calculates the mean of x within each group. The result is a summary tibble where each row corresponds to a unique (y, z) pair along with its calculated group statistic.

## E

```
        df |>
          group_by(y, z) |>
          summarize(mean_x = mean(x), .groups = "drop")
```

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

group_by(y, z) creates subgroups based on each unique combination of y and z, and summarize(mean_x = mean(x), .groups = “drop”) computes the mean of x within each subgroup while removing all grouping information from the result. The final output is a simple tibble with one row per (y, z) pair and no residual grouping.

## F

```
        df |>
          group_by(y, z) |>
          mutate(mean_x = mean(x))
```

```
# A tibble: 5 × 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 summarize() pipeline collapses each group into a single row, producing a smaller tibble with one row per (y, z) combination, while the mutate() pipeline keeps the original number of rows and simply adds a new column containing the group’s mean repeated across all rows in that group. In other words, summarize() reduces the data, whereas mutate() augments it by attaching group statistics without changing the row count.