L03 Data Transformation

Data Science I (STAT 301-1)

Shay Lebovitz

Contents

Overview

The goal of this lab is to start building the skills to transform data using the `dplyr` package in R (dplyr reference page). Students will also continue to develop the knowledge and skills needed to effectively access and utilize R documentation.

Datasets

This lab utilizes the `flights` dataset contained in the `nycflights13` package. Documentation/codebook can be accessed with `?fllights`, provided you have installed and loaded `nycflights13` to your current R session.

Exercises

Please complete the following exercises. Be sure your solutions are clearly indicated and that the document is neatly formatted.

Load Packages You should always begin by loading all necessary packages towards the beginning of your documents. Assume that that all necessary packages have been installed. User should be able to determine if a package needs to be installed either through knowing their R repository or an error message. **Your code should never have install commands.**

```
# Loading package(s)
library(nycflights13)
library(tidyverse)
```

Exercise 1 (Website: 4.4 Practice 1)

Why does this code not work?

```
my_variable <- 10
my_varıable
#> Error in eval(expr, envir, enclos): object 'my_varıable' not found
```

Look carefully! (This may seem like an exercise in pointlessness, but training your brain to notice even the tiniest difference will pay off when programming.)

Not an i in `my_varıable`, thus not the same spelling, thus not the same variable

Exercise 2 (Website: 4.4 Practice 2)

Tweak each of the following R commands so that they run correctly:

```
# Command 1
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))

# Command 2
fliter(mpg, cyl == 8)

# Command 3
filter(diamonds, carat > 3)
```

Exercise 3 (Website: 5.2.4 Ex. 1)

Find all flights that:

a. Had an arrival delay of two or more hours

```
flights %>%
  filter(arr_delay > 120) %>%
  select(year, month, day, origin, dest, arr_delay)
```

```
## # A tibble: 10,034 x 6
##     year month   day origin dest  arr_delay
##    <int> <int> <int> <chr>  <chr>     <dbl>
##  1  2013     1     1 LGA    CLT         137
##  2  2013     1     1 JFK    BWI         851
##  3  2013     1     1 EWR    BOS         123
##  4  2013     1     1 LGA    IAH         145
##  5  2013     1     1 EWR    RIC         127
##  6  2013     1     1 EWR    MCO         125
##  7  2013     1     1 EWR    MCI         136
##  8  2013     1     1 JFK    IAD         123
##  9  2013     1     1 EWR    DAY         123
## 10  2013     1     1 LGA    BNA         138
## # ... with 10,024 more rows
```

b. Flew to Houston (`IAH` or `HOU`)

```
flights %>%
  filter(dest %in% c("IAH", "HOU")) %>%
  select(year, month, day, origin, dest)
```

```
## # A tibble: 9,313 x 5
##     year month   day origin dest
##    <int> <int> <int> <chr>  <chr>
##  1  2013     1     1 EWR    IAH
##  2  2013     1     1 LGA    IAH
##  3  2013     1     1 LGA    IAH
##  4  2013     1     1 LGA    IAH
##  5  2013     1     1 EWR    IAH
##  6  2013     1     1 EWR    IAH
##  7  2013     1     1 LGA    IAH
##  8  2013     1     1 EWR    IAH
##  9  2013     1     1 LGA    IAH
## 10  2013     1     1 EWR    IAH
## # ... with 9,303 more rows
```

c. Were operated by United, American, or Delta

```
flights %>%
  filter(carrier %in% c('AA', 'UA', 'DL')) %>%
  select(year, month, day, origin, dest, carrier)
```

```
## # A tibble: 139,504 x 6
##     year month   day origin dest  carrier
##    <int> <int> <int> <chr>  <chr> <chr>
##  1  2013     1     1 EWR    IAH   UA
##  2  2013     1     1 LGA    IAH   UA
##  3  2013     1     1 JFK    MIA   AA
##  4  2013     1     1 LGA    ATL   DL
##  5  2013     1     1 EWR    ORD   UA
##  6  2013     1     1 LGA    ORD   AA
##  7  2013     1     1 JFK    LAX   UA
##  8  2013     1     1 EWR    SFO   UA
##  9  2013     1     1 LGA    DFW   AA
## 10  2013     1     1 EWR    LAS   UA
## # ... with 139,494 more rows
```

d. Departed in summer (July, August, and September)

```
flights %>%
  filter(month %in% c(7, 8, 9))
```

```
## # 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
## # ... with 86,316 more rows, and 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>
```

e. Arrived more than two hours late, but didn't leave late

```
flights %>%
  filter(arr_delay > 120 & dep_delay <= 0) %>%
  select(year, month, day, origin, dest, arr_delay, dep_delay)
```

```
## # A tibble: 29 x 7
##     year month   day origin dest  arr_delay dep_delay
##    <int> <int> <int> <chr>  <chr>     <dbl>     <dbl>
##  1  2013     1    27 EWR    ORD         124        -1
##  2  2013    10     7 LGA    MSN         130         0
##  3  2013    10     7 LGA    DFW         124        -2
##  4  2013    10    16 JFK    SJU         122        -3
##  5  2013    11     1 JFK    LAX         194        -2
##  6  2013     3    18 JFK    SFO         140        -3
##  7  2013     4    17 LGA    DTW         124        -5
##  8  2013     4    18 LGA    DFW         179        -2
##  9  2013     4    18 EWR    DFW         143        -5
## 10  2013     5    22 LGA    CLE         127        -3
## # ... with 19 more rows
```

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

```
flights %>%
  filter(dep_delay > 60 & arr_delay < (dep_delay - 30)) %>%
  select(year, month, day, origin, dest, arr_delay, dep_delay)
```

```
## # A tibble: 1,819 x 7
##     year month   day origin dest  arr_delay dep_delay
##    <int> <int> <int> <chr>  <chr>     <dbl>     <dbl>
##  1  2013     1     1 EWR    MIA         246       285
##  2  2013     1     1 JFK    LAS          73       116
##  3  2013     1     3 EWR    SFO         128       162
##  4  2013     1     3 JFK    EGE          66        99
##  5  2013     1     3 JFK    SFO          28        65
##  6  2013     1     3 EWR    PHX          67       102
##  7  2013     1     3 JFK    OAK           1        65
##  8  2013     1     3 JFK    CVG         141       177
##  9  2013     1     4 JFK    EGE         105       137
## 10  2013     1     4 JFK    SFO          97       145
## # ... with 1,809 more rows
```

g. Departed between midnight and 6am (inclusive)

```
flights %>%
  filter(dep_time%/%100 %in% c(0,1,2,3,4,5) | dep_time == 600) %>%
  select(year, month, day, origin, dest, dep_time)
```

```
## # A tibble: 9,344 x 6
##     year month   day origin dest  dep_time
##    <int> <int> <int> <chr>  <chr>    <int>
##  1  2013     1     1 EWR    IAH        517
```

```
##  2  2013     1     1 LGA    IAH        533
##  3  2013     1     1 JFK    MIA        542
##  4  2013     1     1 JFK    BQN        544
##  5  2013     1     1 LGA    ATL        554
##  6  2013     1     1 EWR    ORD        554
##  7  2013     1     1 EWR    FLL        555
##  8  2013     1     1 LGA    IAD        557
##  9  2013     1     1 JFK    MCO        557
## 10  2013     1     1 LGA    ORD        558
## # ... with 9,334 more rows
```

Exercise 4 (Website: 5.2.4 Ex. 2)

A useful `dplyr` filtering helper is `between()`. Rewrite your code to Exercise 3 (d) using `between()`? If you did use it, then indicate that you did.

```
filter(flights, between(month,7,9))
```

```
## # 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
## # ... with 86,316 more rows, and 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>
```

Used `between`

Exercise 5 (Website: 5.2.4 Ex. 3)

How many flights have a missing `dep_time`? What other variables are missing? What might these rows represent?

```
no_dep_time <- flights %>%
  filter(is.na(dep_time))
dim(no_dep_time)
```

```
## [1] 8255   19
```

There are 8255 flights with no departure time. These flights are also missing `dep_delay`, `arr_time`, `arr_delay`, and `air_time`. These are cancelled flights

Exercise 6 (Website: 5.3.1 Ex. 1)

How could you use `arrange()` to sort the dataset so that flights missing `arr_delay` are at the start/top of the tibble/table? (Hint: use `is.na()`).?

```
flights %>%
  arrange(!is.na(arr_delay)) %>%
  select(year, month, day, origin, dest, arr_delay)
```

```
## # A tibble: 336,776 x 6
##     year month   day origin dest  arr_delay
##    <int> <int> <int> <chr>  <chr>     <dbl>
##  1  2013     1     1 LGA    XNA          NA
##  2  2013     1     1 EWR    STL          NA
##  3  2013     1     1 LGA    XNA          NA
##  4  2013     1     1 EWR    SAN          NA
##  5  2013     1     1 JFK    DFW          NA
##  6  2013     1     1 EWR    TUL          NA
##  7  2013     1     1 EWR    OKC          NA
##  8  2013     1     1 EWR    RDU          NA
##  9  2013     1     1 LGA    DFW          NA
## 10  2013     1     1 LGA    MIA          NA
## # ... with 336,766 more rows
```

Exercise 7 (Website: 5.3.1 Ex. 2)

Sort `flights` to find the most delayed flights. Find the flights that left earliest. Name the flight at top of the list (e.g. New York City JFK to Chicago ORD).

```
flights %>%
  arrange(dep_delay, dep_time) %>%
  select(year, month, day, origin, dest, dep_delay)
```

```
## # A tibble: 336,776 x 6
##     year month   day origin dest  dep_delay
##    <int> <int> <int> <chr>  <chr>     <dbl>
##  1  2013    12     7 JFK    DEN         -43
##  2  2013     2     3 LGA    MSY         -33
##  3  2013    11    10 LGA    IAD         -32
##  4  2013     1    11 LGA    TPA         -30
##  5  2013     1    29 LGA    DEN         -27
##  6  2013     8     9 LGA    DTW         -26
##  7  2013    10    23 EWR    TYS         -25
##  8  2013     3    30 LGA    DTW         -25
##  9  2013     5    14 LGA    SDF         -24
## 10  2013     5     5 LGA    FLL         -24
## # ... with 336,766 more rows
```

JFK to DEN was the most delayed flight, based on departure delay

Exercise 8 (Website: 5.3.1 Ex. 3)

Sort flights to find the fastest flights. Name the flight at top of the list (e.g. New York City JFK to Chicago ORD).

```
flights %>%
  arrange(air_time) %>%
  select(year, month, day, origin, dest, air_time)
```

```
## # A tibble: 336,776 x 6
##     year month   day origin dest  air_time
##    <int> <int> <int> <chr>  <chr>    <dbl>
##  1  2013     1    16 EWR    BDL         20
##  2  2013     4    13 EWR    BDL         20
##  3  2013    12     6 EWR    BDL         21
##  4  2013     2     3 EWR    PHL         21
##  5  2013     2     5 EWR    BDL         21
##  6  2013     2    12 EWR    PHL         21
##  7  2013     3     2 LGA    BOS         21
##  8  2013     3     8 JFK    PHL         21
##  9  2013     3    18 EWR    BDL         21
## 10  2013     3    19 EWR    BDL         21
## # ... with 336,766 more rows
```

EWR to BDL was the fastest flight, with an air time of just 20 minutes

Exercise 9 (Website: 5.3.1 Ex. 4)

Which flights travelled the longest? Which travelled the shortest? Name the flight at top of the list (e.g. New York City JFK to Chicago ORD).

```
flights %>%
  select(year, month, day, origin, dest, distance) %>%
  arrange(desc(distance))
```

```
## # A tibble: 336,776 x 6
##     year month   day origin dest  distance
##    <int> <int> <int> <chr>  <chr>    <dbl>
##  1  2013     1     1 JFK    HNL       4983
##  2  2013     1     2 JFK    HNL       4983
##  3  2013     1     3 JFK    HNL       4983
##  4  2013     1     4 JFK    HNL       4983
##  5  2013     1     5 JFK    HNL       4983
##  6  2013     1     6 JFK    HNL       4983
##  7  2013     1     7 JFK    HNL       4983
##  8  2013     1     8 JFK    HNL       4983
##  9  2013     1     9 JFK    HNL       4983
## 10  2013     1    10 JFK    HNL       4983
## # ... with 336,766 more rows
```

```
flights %>%
  select(year, month, day, origin, dest, distance) %>%
  arrange(distance)
```

```
## # A tibble: 336,776 x 6
##     year month   day origin dest  distance
##    <int> <int> <int> <chr>  <chr>    <dbl>
##  1  2013     7    27 EWR    LGA         17
##  2  2013     1     3 EWR    PHL         80
##  3  2013     1     4 EWR    PHL         80
```

```
##  4  2013     1     4 EWR    PHL         80
##  5  2013     1     4 EWR    PHL         80
##  6  2013     1     5 EWR    PHL         80
##  7  2013     1     6 EWR    PHL         80
##  8  2013     1     7 EWR    PHL         80
##  9  2013     1     8 EWR    PHL         80
## 10  2013     1     9 EWR    PHL         80
## # ... with 336,766 more rows
```

The longest flight was JFK to HNL. The shortest flight was EWR to LGA

Exercise 10 (Website: 5.4.1 Ex. 1)

Brainstorm at least 3 ways to select `dep_time`, `dep_delay`, `arr_time`, and `arr_delay` from `flights`. *Hint: Use helper functions*

```
select(flights, dep_time, dep_delay, arr_time, arr_delay)
```

```
## # 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
## # ... with 336,766 more rows
```

```
select(flights, dep_time, dep_delay:arr_time, arr_delay)
```

```
## # 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
## # ... with 336,766 more rows
```

```
select(flights, starts_with('dep') | starts_with('arr'))
```

```
## # 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
## # ... with 336,766 more rows
```

Exercise 11 (Website: 5.4.1 Ex. 3)

What does the `one_of()` function do? Why might it be helpful in conjunction with this vector?

```
vars <- c("year", "month", "day", "dep_delay", "arr_delay")
select(flights, one_of(vars))
```

```
## # 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
## # ... with 336,766 more rows
```

The `one_of()` function allows you to select from a pre-formed list of variable names

Exercise 12 (Website: 5.4.1 Ex. 4)

Does the result of running the code below surprise you? Which default setting for `contains()` causes this to happen? What should happen if you run the code, but with the default setting changed?

```
select(flights, contains("TIME"))
```

This selects all varibales that have the word "TIME" in them. The default setting is `fixed = FALSE`. If you ran it with the default setting changed to `FALSE`, the method would no longer require an exact match of strings.

Exercise 13 (Website: 5.5.2 Ex. 1)

Currently `dep_time` and `sched_dep_time` are convenient to look at, but hard to compute with because they're not really continuous numbers. Convert them to a more convenient representation of number of minutes since midnight.

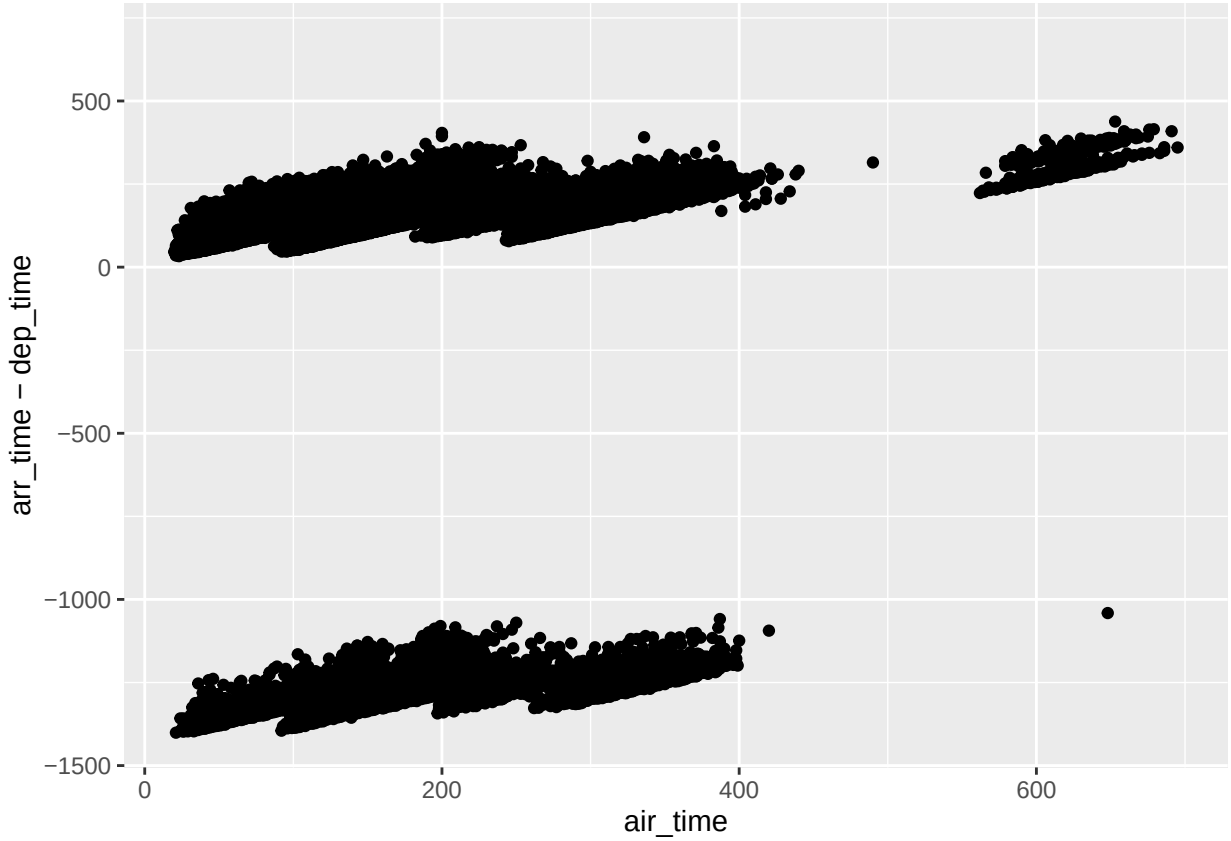
```
flights1 <- flights %>%
  mutate(
```

```
    dep_time = 60 * (dep_time %/% 100) + (dep_time %% 100),
    sched_dep_time = 60 * (sched_dep_time %/% 100) + (sched_dep_time %% 100)
  )
```

Exercise 14 (Website: 5.5.2 Ex. 2)

Compare `air_time` with `arr_time - dep_time`. What do you expect to see? What do you see? What do you need to do to fix it?

```
flights %>%
  transmute(
    dep_time =  60 * (dep_time %/% 100) + (dep_time %% 100),
    arr_time = 60 * (arr_time %/% 100) + (arr_time %% 100),
    air_time = air_time
  ) %>%
  ggplot() +
    geom_point(mapping = aes(x = air_time, y = arr_time - dep_time))
```



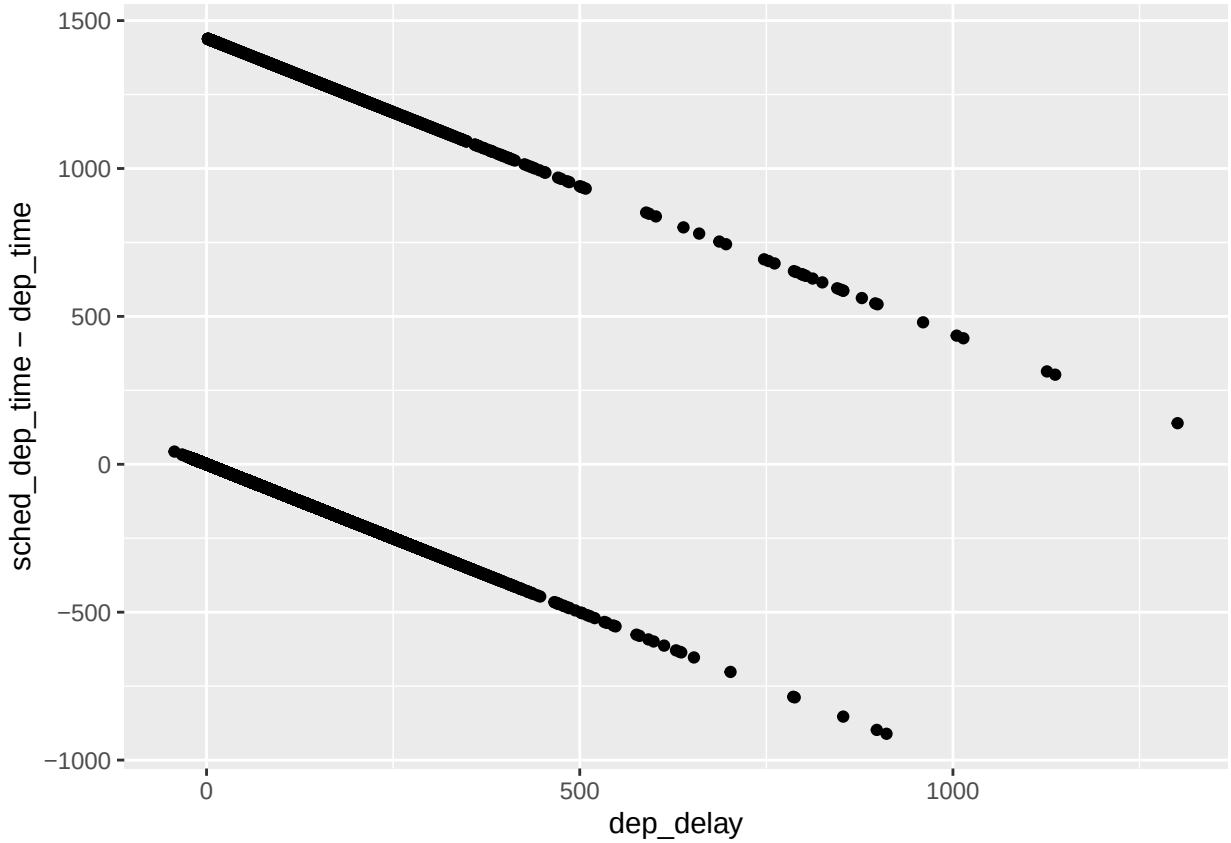
For the most part, `air_time` matches up well with `arr_time - dep_time`, with the former usually being slightly lower due to taxi times. However, there are some times when they don't relate because of overnight flights, >24 hour flights, and time zone issues.

Exercise 15 (Website: 5.5.2 Ex. 3)

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

```
flights %>%
  transmute(
    dep_time = 60 * (dep_time %/% 100) + (dep_time %% 100),
    sched_dep_time = 60 * (sched_dep_time %/% 100) + (sched_dep_time %% 100),
    dep_delay = dep_delay,
  ) %>%
  group_by(dep_time, sched_dep_time, dep_delay) %>%
  summarise(
    expected_vs_actual = dep_delay - (dep_time - sched_dep_time)
  ) %>%
  ggplot() +
    geom_point(mapping = aes(x = dep_delay, y = sched_dep_time - dep_time))
```

```
## `summarise()` regrouping output by 'dep_time', 'sched_dep_time', 'dep_delay' (override with `.groups`
```



It looks like it's an exact match, where `dep_delay` = `dep_time - sched_dep_time`. It gets messed up when the flight is delayed to a new day, say from 11:50 pm to 12:10 am.

Exercise 16 (Website: 5.5.2 Ex. 4)

Find the 10 most delayed flights using the `min_rank()` function.

```
flights %>%
  mutate (
    rank = min_rank(dep_delay)
  ) %>%
  arrange(desc(rank)) %>%
```

```
  select(year, month, day, origin, dest, dep_delay)
```

```
## # A tibble: 336,776 x 6
##     year month   day origin dest  dep_delay
##    <int> <int> <int> <chr>  <chr>     <dbl>
##  1  2013     1     9 JFK    HNL        1301
##  2  2013     6    15 JFK    CMH        1137
##  3  2013     1    10 EWR    ORD        1126
##  4  2013     9    20 JFK    SFO        1014
##  5  2013     7    22 JFK    CVG        1005
##  6  2013     4    10 JFK    TPA         960
##  7  2013     3    17 LGA    MSP         911
##  8  2013     6    27 JFK    PDX         899
##  9  2013     7    22 LGA    ATL         898
## 10  2013    12     5 EWR    MIA         896
## # ... with 336,766 more rows
```

Exercise 17 (Website: 5.6.7 Ex. 2)

Come up with another approach that will give you the same output as `not_cancelled %>% count(dest)` and `not_cancelled %>% count(tailnum, wt = distance)` (without using `count()`). Dataset `not_cancelled` was created in this section of the book, but for your convenience the code is provided.

```
not_cancelled <- flights %>%
  filter(!is.na(dep_delay), !is.na(arr_delay))

not_cancelled %>%
  group_by(dest) %>%
  summarise(num = n())
```

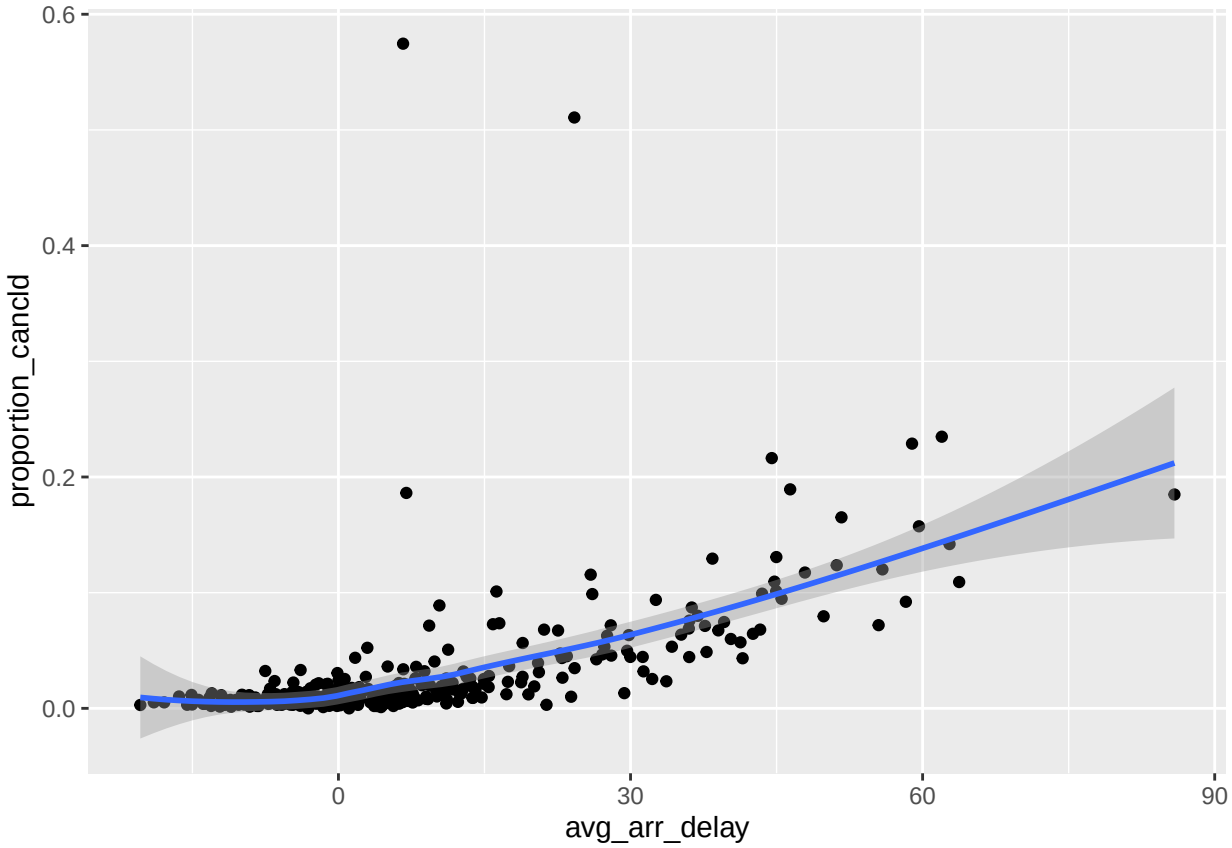
Exercise 18 (Website: 5.6.7 Ex. 4)

Look at the number of cancelled flights per day. Is there a pattern? Is the proportion of cancelled flights related to the average delay?

```
flights %>%
  mutate(cancelled = is.na(arr_delay) | is.na(dep_delay)) %>%
  group_by(year, month, day) %>%
  summarise(
    num_cancelled = sum(cancelled),
    proportion_cancld = mean(cancelled),
    avg_arr_delay = mean(arr_delay, na.rm = TRUE)
  ) %>%
  ggplot(mapping = aes(x = avg_arr_delay, y = proportion_cancld)) +
    geom_point() +
    geom_smooth()
```

```
## `summarise()` regrouping output by 'year', 'month' (override with `.groups` argument)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



It does look like there's a positive correlation between the proportion of flights cancelled and the average delay per flight

Exercise 19 (Website: 5.7.1 Ex. 2)

Which plane (`tailnum`) has the worst on-time record?

```
flights %>%
  group_by (tailnum) %>%
  mutate(on_time = (arr_delay <= 0), num_flights = n()) %>%
  filter(num_flights > 10) %>%
  summarise(
    prptn_on_time = mean(on_time, na.rm = TRUE)
  ) %>%
  arrange(prptn_on_time)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 3,401 x 2
##    tailnum prptn_on_time
##    <chr>           <dbl>
##  1 N168AT         0.0588
##  2 N337AT         0.0769
##  3 N169AT         0.0909
##  4 N290AT         0.125
##  5 N273AT         0.154
##  6 N326AT         0.176
##  7 N913JB         0.188
```

```
##  8 N988AT         0.2
##  9 N518LR         0.231
## 10 N338AT         0.25
## # ... with 3,391 more rows
```

N168AT has a 5.88% on-time rate, the lowest for all tailnums with > 10 flights.

Exercise 20 (Website: 5.7.1 Ex. 4)

For each destination, compute the total minutes of delay. For each flight, compute the proportion of the total delay for its destination.

```
flights %>%
  group_by(dest) %>%
  mutate(
    total_del_mins = sum(dep_delay, na.rm = TRUE),
    prptn_tot_del = dep_delay / total_del_mins
  ) %>%
  select(year, month, day, origin, dest, dep_delay, total_del_mins, prptn_tot_del)
```

```
## # A tibble: 336,776 x 8
## # Groups:   dest [105]
##     year month   day origin dest  dep_delay total_del_mins prptn_tot_del
##    <int> <int> <int> <chr>  <chr>     <dbl>          <dbl>         <dbl>
##  1  2013     1     1 EWR    IAH           2          77012    0.0000260
##  2  2013     1     1 LGA    IAH           4          77012    0.0000519
##  3  2013     1     1 JFK    MIA           2         103261    0.0000194
##  4  2013     1     1 JFK    BQN          -1          11032   -0.0000906
##  5  2013     1     1 LGA    ATL          -6         211391   -0.0000284
##  6  2013     1     1 EWR    ORD          -4         225840   -0.0000177
##  7  2013     1     1 EWR    FLL          -5         151933   -0.0000329
##  8  2013     1     1 LGA    IAD          -3          91555   -0.0000328
##  9  2013     1     1 JFK    MCO          -3         157661   -0.0000190
## 10  2013     1     1 LGA    ORD          -2         225840   -0.00000886
## # ... with 336,766 more rows
```

Exercise 21 (Website: 5.7.1 Ex. 8)

For each plane, count the number of flights before the first delay of greater than 1 hour.

```
flights %>%
  group_by(tailnum) %>%
  filter(!is.na(dep_time)) %>%
  select(tailnum, year, month, day, dep_time, dep_delay) %>%
  arrange(year, month, day, dep_time) %>%
  mutate(
    flag = cumsum(dep_delay > 60)
  ) %>%
  summarise(
    num_flights_before_delay = sum(flag == 0)
  )
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 4,037 x 2
```

```
##    tailnum num_flights_before_delay
##    <chr>                      <int>
##  1 D942DN                         0
##  2 N0EGMQ                        53
##  3 N10156                         9
##  4 N102UW                        25
##  5 N103US                        46
##  6 N104UW                         3
##  7 N10575                         0
##  8 N105UW                        22
##  9 N107US                        20
## 10 N108UW                        36
## # ... with 4,027 more rows
```