Solutions for dplyr/tidyverse URPP tutorial.Rmd

Stefan Wyder

June 4, 2018

```
library(tidyverse)
library(nycflights13)
```

Exercises 2

Find all flights that

1. Had an arrival delay of two or more hours

```
filter(flights, arr_delay >= 120)
```

```
## # A tibble: 10,200 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1      811            630      101.     1047
##  2  2013     1     1      848           1835      853.     1001
##  3  2013     1     1      957            733      144.     1056
##  4  2013     1     1     1114            900      134.     1447
##  5  2013     1     1     1505           1310      115.     1638
##  6  2013     1     1     1525           1340      105.     1831
##  7  2013     1     1     1549           1445       64.     1912
##  8  2013     1     1     1558           1359      119.     1718
##  9  2013     1     1     1732           1630       62.     2028
## 10  2013     1     1     1803           1620      103.     2008
## # ... with 10,190 more rows, and 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>
```

2. Flew to Houston (IAH or HOU)

```
filter(flights, dest %in% c("IAH", "HOU"))
```

```
## # A tibble: 9,313 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1      517            515        2.      830
##  2  2013     1     1      533            529        4.      850
##  3  2013     1     1      623            627       -4.      933
##  4  2013     1     1      728            732       -4.     1041
##  5  2013     1     1      739            739        0.     1104
##  6  2013     1     1      908            908        0.     1228
##  7  2013     1     1     1028           1026        2.     1350
##  8  2013     1     1     1044           1045       -1.     1352
##  9  2013     1     1     1114            900      134.     1447
## 10  2013     1     1     1205           1200        5.     1503
## # ... with 9,303 more rows, and 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>
```

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

```
filter(flights, month >= 7, month <= 9)
```

```
## # A tibble: 86,326 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     7     1        1           2029      212.      236
##  2  2013     7     1        2           2359        3.      344
##  3  2013     7     1       29           2245      104.      151
##  4  2013     7     1       43           2130      193.      322
##  5  2013     7     1       44           2150      174.      300
##  6  2013     7     1       46           2051      235.      304
##  7  2013     7     1       48           2001      287.      308
##  8  2013     7     1       58           2155      183.      335
##  9  2013     7     1      100           2146      194.      327
## 10  2013     7     1      100           2245      135.      337
## # ... with 86,316 more rows, and 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>
```

```
# also correct:
# filter(flights, between(month, 7, 9))
# filter(flights, month in c(7,8,9))
```

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

```
filter(flights, dep_delay <= 0, arr_delay > 120)
```

```
## # A tibble: 29 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1    27     1419           1420       -1.     1754
##  2  2013    10     7     1350           1350        0.     1736
##  3  2013    10     7     1357           1359       -2.     1858
##  4  2013    10    16      657            700       -3.     1258
##  5  2013    11     1      658            700       -2.     1329
##  6  2013     3    18     1844           1847       -3.       39
##  7  2013     4    17     1635           1640       -5.     2049
##  8  2013     4    18      558            600       -2.     1149
##  9  2013     4    18      655            700       -5.     1213
## 10  2013     5    22     1827           1830       -3.     2217
## # ... with 19 more rows, and 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>
```

```
# these are also correct (the !is.na condition is more explicit but redundant):
# filter(flights, !is.na(dep_delay), dep_delay <= 0, arr_delay > 120)
# filter(flights, !is.na(dep_delay) & dep_delay <= 0 & arr_delay > 120)
```

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

```
filter(flights, dep_delay >= 60, dep_delay - arr_delay > 30)
```

```
## # A tibble: 1,844 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1     2205           1720      285.       46
##  2  2013     1     1     2326           2130      116.      131
##  3  2013     1     3     1503           1221      162.     1803
##  4  2013     1     3     1839           1700       99.     2056
##  5  2013     1     3     1850           1745       65.     2148
##  6  2013     1     3     1941           1759      102.     2246
##  7  2013     1     3     1950           1845       65.     2228
##  8  2013     1     3     2015           1915       60.     2135
##  9  2013     1     3     2257           2000      177.       45
## 10  2013     1     4     1917           1700      137.     2135
## # ... with 1,834 more rows, and 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>
```

```
# explicit omission of NAs is redundant but might increase readibility
# filter(flights, !is.na(dep_delay),
#        dep_delay >= 60, dep_delay - arr_delay > 30)
```

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

```
filter(flights, is.na(dep_time))
```

```
## # A tibble: 8,255 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1       NA           1630        NA       NA
##  2  2013     1     1       NA           1935        NA       NA
##  3  2013     1     1       NA           1500        NA       NA
##  4  2013     1     1       NA            600        NA       NA
##  5  2013     1     2       NA           1540        NA       NA
##  6  2013     1     2       NA           1620        NA       NA
##  7  2013     1     2       NA           1355        NA       NA
##  8  2013     1     2       NA           1420        NA       NA
##  9  2013     1     2       NA           1321        NA       NA
## 10  2013     1     2       NA           1545        NA       NA
## # ... with 8,245 more rows, and 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>
```

Since arr_time is also missing, these are canceled flights.

```
# this is not the tidyverse way:
table(filter(flights, is.na(dep_time))$arr_time, useNA = "always")
```

```
## 
## <NA> 
## 8255
```

```r
# tidyverse way (which we will learn only later in the course)
filter(flights, is.na(dep_time)) %>% count(arr_time)
```

```
## # A tibble: 1 x 2
##   arr_time     n
##      <int> <int>
## 1       NA  8255
```

Exercises 3

1. Sort flights to find the most delayed flights. Find the flights that left earliest

```r
# The most delayed flights are found by sorting by dep_delay in descending order.
# There was a flight delayed more than 21 hours.
arrange(flights, desc(dep_delay))
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     9      641            900     1301.     1242
##  2  2013     6    15     1432           1935     1137.     1607
##  3  2013     1    10     1121           1635     1126.     1239
##  4  2013     9    20     1139           1845     1014.     1457
##  5  2013     7    22      845           1600     1005.     1044
##  6  2013     4    10     1100           1900      960.     1342
##  7  2013     3    17     2321            810      911.      135
##  8  2013     6    27      959           1900      899.     1236
##  9  2013     7    22     2257            759      898.      121
## 10  2013    12     5      756           1700      896.     1058
## # ... with 336,766 more rows, and 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>
```

Find the flights that left earliest

```r
# If we sort dep_delay in ascending order, we get those that left earliest.
# There was a flight that left 43 minutes early.
arrange(flights, dep_delay)
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013    12     7     2040           2123      -43.       40
##  2  2013     2     3     2022           2055      -33.     2240
##  3  2013    11    10     1408           1440      -32.     1549
##  4  2013     1    11     1900           1930      -30.     2233
##  5  2013     1    29     1703           1730      -27.     1947
##  6  2013     8     9      729            755      -26.     1002
##  7  2013    10    23     1907           1932      -25.     2143
##  8  2013     3    30     2030           2055      -25.     2213
##  9  2013     3     2     1431           1455      -24.     1601
## 10  2013     5     5      934            958      -24.     1225
## # ... with 336,766 more rows, and 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>
```

2. How could you use arrange() to sort all missing values to the start? (Hint: use `is.na()`)

```
arrange(flights, desc(is.na(dep_time)), dep_time)
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1       NA           1630        NA       NA
##  2  2013     1     1       NA           1935        NA       NA
##  3  2013     1     1       NA           1500        NA       NA
##  4  2013     1     1       NA            600        NA       NA
##  5  2013     1     2       NA           1540        NA       NA
##  6  2013     1     2       NA           1620        NA       NA
##  7  2013     1     2       NA           1355        NA       NA
##  8  2013     1     2       NA           1420        NA       NA
##  9  2013     1     2       NA           1321        NA       NA
## 10  2013     1     2       NA           1545        NA       NA
## # ... with 336,766 more rows, and 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>
```

3. Which flights traveled the longest? Which traveled the shortest?

```
# The longest flights are the Hawaii Air (HA 51) between JFK and HNL (Honolulu) at 4,983 miles.
arrange(flights, desc(distance))
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1      857            900       -3.     1516
##  2  2013     1     2      909            900        9.     1525
##  3  2013     1     3      914            900       14.     1504
##  4  2013     1     4      900            900        0.     1516
##  5  2013     1     5      858            900       -2.     1519
##  6  2013     1     6     1019            900       79.     1558
##  7  2013     1     7     1042            900      102.     1620
##  8  2013     1     8      901            900        1.     1504
##  9  2013     1     9      641            900     1301.     1242
## 10  2013     1    10      859            900       -1.     1449
## # ... with 336,766 more rows, and 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>
```

We will use later how to use the pipe which is handy for plausibility checking and printing all columns: We can use the `head()` function to take the first row and `glimpse()` to print all columns:

```
 arrange(flights, desc(distance)) %>% head(n=1) %>% glimpse()
```

```
## Observations: 1
## Variables: 19
## $ year           <int> 2013
## $ month          <int> 1
## $ day            <int> 1
## $ dep_time       <int> 857
## $ sched_dep_time <int> 900
## $ dep_delay      <dbl> -3
## $ arr_time       <int> 1516
## $ sched_arr_time <int> 1530
## $ arr_delay      <dbl> -14
## $ carrier        <chr> "HA"
## $ flight         <int> 51
## $ tailnum        <chr> "N380HA"
## $ origin         <chr> "JFK"
## $ dest           <chr> "HNL"
## $ air_time       <dbl> 659
## $ distance       <dbl> 4983
## $ hour           <dbl> 9
## $ minute         <dbl> 0
## $ time_hour      <dttm> 2013-01-01 09:00:00
```

Alternatively, we can use the `print()` function (`width = Inf` to show all columns)

```
arrange(flights, desc(distance)) %>% print(., n=1, width = Inf)
```

```
## # A tibble: 336,776 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>
## 1  2013     1     1      857            900        -3.     1516
##   sched_arr_time arr_delay carrier flight tailnum origin dest  air_time
##            <int>     <dbl> <chr>    <int> <chr>   <chr>  <chr>     <dbl>
## 1           1530      -14. HA           51 N380HA  JFK    HNL       659.
##   distance  hour minute time_hour
##      <dbl> <dbl>  <dbl> <dttm>
## 1    4983.    9.      0. 2013-01-01 09:00:00
## # ... with 3.368e+05 more rows
```

Which traveled the shortest?

```
# Apart from an EWR to LGA flight that was canceled, the shortest flights are
# the Envoy Air Flights between EWR and PHL at 80 miles.
arrange(flights, distance)
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     7    27       NA            106        NA       NA
##  2  2013     1     3     2127           2129        -2.     2222
##  3  2013     1     4     1240           1200        40.     1333
##  4  2013     1     4     1829           1615       134.     1937
##  5  2013     1     4     2128           2129        -1.     2218
##  6  2013     1     5     1155           1200        -5.     1241
##  7  2013     1     6     2125           2129        -4.     2224
##  8  2013     1     7     2124           2129        -5.     2212
##  9  2013     1     8     2127           2130        -3.     2304
## 10  2013     1     9     2126           2129        -3.     2217
## # ... with 336,766 more rows, and 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>
```

Exercises 4

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

```
select(flights, dep_time, dep_delay, arr_time, arr_delay)
select(flights, starts_with("dep_"), starts_with("arr_"))
select(flights, matches("^(dep|arr)_(time|delay)$"))
# using ends_with() doesn't work well since it would return both sched_arr_time and sched_dep_time.
# also the base R subsetting with square brackets works:
flights[, c("dep_time", "dep_delay", "arr_time", "arr_delay")]
```

2. What happens if you include the name of a variable multiple times in a `select()` call?

```
select(flights, year, month, day, year, year)
```

```
## # A tibble: 336,776 x 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
##   7  2013     1     1
##   8  2013     1     1
##   9  2013     1     1
##  10  2013     1     1
## # ... with 336,766 more rows
```

It ignores the duplicates, and that variable is only included once. No error, warning, or message is emitted.

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

```
# The default behavior for contains() is to ignore case.
select(flights, contains("TIME"))
```

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

```
select(flights, contains("TIME", ignore.case = FALSE))
```

```
## # A tibble: 336,776 x 0
```

Exercises 5

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.

   Actual departure and arrival times, local time zone. It seems they are not minutes as values between 60 and 99 are missing (but the variable description could be better).

To get the departure times in the number of minutes, (integer) divide dep_time by 100 to get the hours since midnight and multiply by 60 and add the remainder of dep_time divided by 100.

```
mutate(flights,
       dep_time_mins = dep_time %/% 100 * 60 + dep_time %% 100,
       sched_dep_time_mins = sched_dep_time %/% 100 * 60 + sched_dep_time %% 100) %>%
  select(dep_time, dep_time_mins, sched_dep_time, sched_dep_time_mins)
```

```
## # A tibble: 336,776 x 4
##    dep_time dep_time_mins sched_dep_time sched_dep_time_mins
##       <int>         <dbl>          <int>               <dbl>
##  1      517          317.            515                315.
##  2      533          333.            529                329.
##  3      542          342.            540                340.
##  4      544          344.            545                345.
##  5      554          354.            600                360.
##  6      554          354.            558                358.
##  7      555          355.            600                360.
##  8      557          357.            600                360.
##  9      557          357.            600                360.
## 10      558          358.            600                360.
## # ... with 336,766 more rows
```

The clean way is to define a function first and then to reuse it.

```
time2mins <- function(x) {
  x %/% 100 * 60 + x %% 100
}
mutate(flights,
       dep_time_mins = time2mins(dep_time),
       sched_dep_time_mins = time2mins(sched_dep_time)) %>%
  select(dep_time, dep_time_mins, sched_dep_time, sched_dep_time_mins)
```

```
## # A tibble: 336,776 x 4
##    dep_time dep_time_mins sched_dep_time sched_dep_time_mins
##       <int>         <dbl>          <int>               <dbl>
##  1      517          317.            515                315.
##  2      533          333.            529                329.
##  3      542          342.            540                340.
##  4      544          344.            545                345.
##  5      554          354.            600                360.
##  6      554          354.            558                358.
##  7      555          355.            600                360.
##  8      557          357.            600                360.
##  9      557          357.            600                360.
## 10      558          358.            600                360.
## # ... with 336,766 more rows
```

2. Find the 10 most delayed flights using a ranking function. How do you want to handle ties? Carefully read the documentation for `min_rank()`.

```
# We want to handle ties by taking the minimum of tied values.
# If three flights are have the same value and are the most delayed,
# we would say they are tied for first, not tied for third or second.
mutate(flights,
       dep_delay_rank = min_rank(-dep_delay)) %>%
filter(dep_delay_rank <= 10) %>%
arrange(dep_delay_rank) %>%
select(dep_delay_rank, everything())
```

```
## # A tibble: 10 x 20
##    dep_delay_rank  year month   day dep_time sched_dep_time dep_delay
##             <int> <int> <int> <int>    <int>          <int>     <dbl>
##  1              1  2013     1     9      641            900     1301.
##  2              2  2013     6    15     1432           1935     1137.
##  3              3  2013     1    10     1121           1635     1126.
##  4              4  2013     9    20     1139           1845     1014.
##  5              5  2013     7    22      845           1600     1005.
##  6              6  2013     4    10     1100           1900      960.
##  7              7  2013     3    17     2321            810      911.
##  8              8  2013     6    27      959           1900      899.
##  9              9  2013     7    22     2257            759      898.
## 10             10  2013    12     5      756           1700      896.
## # ... with 13 more variables: arr_time <int>, 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>
```

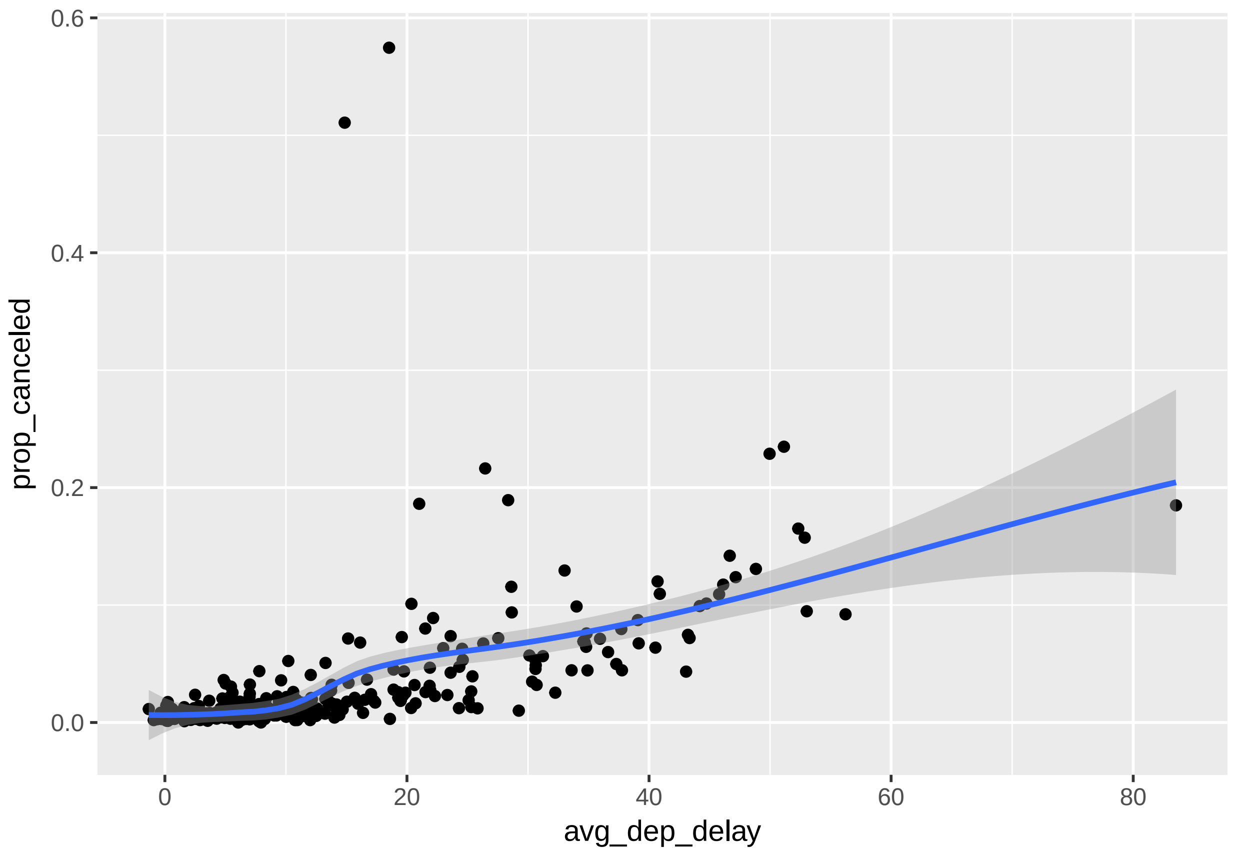
Exercises 6

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

```
canceled_delayed <-
  flights %>%
  mutate(canceled = (is.na(arr_delay) | is.na(dep_delay))) %>%
  group_by(year, month, day) %>%
  summarise(prop_canceled = mean(canceled),
            avg_dep_delay = mean(dep_delay, na.rm = TRUE))

ggplot(canceled_delayed, aes(x = avg_dep_delay, prop_canceled)) +
  geom_point() +
  geom_smooth()
```

```
## `geom_smooth()` using method = 'loess'
```



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

2. Which carrier has the worst delays? Challenge: can you disentangle the effects of bad airports vs. bad carriers? Why/why not? (Hint: think about flights %>% group_by(carrier, dest) %>% summarise(n()))

```
flights %>%
  group_by(carrier) %>%
  summarise(arr_delay = mean(arr_delay, na.rm = TRUE)) %>%
  arrange(desc(arr_delay))
```

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

```
filter(airlines, carrier == "F9")
```

```
## # A tibble: 1 x 2
##   carrier name
##   <chr>   <chr>
## 1 F9      Frontier Airlines Inc.
```

Frontier Airlines (FL) has the worst delays.

You can get part of the way to disentangling the effects of airports vs. carriers by comparing each flight's delay to the average delay of destination airport. However, you'd really want to compare it to the average delay of the destination airport, after removing other flights from the same airline. But this is beyond the scope of this tutorial.

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

```
flights %>%
  arrange(tailnum, year, month, day) %>%
  group_by(tailnum) %>%
  mutate(delay_gt1hr = dep_delay > 60) %>%
  mutate(before_delay = cumsum(delay_gt1hr)) %>%
  filter(before_delay < 1) %>%
  count(sort = TRUE)
```

```
## # A tibble: 3,755 x 2
## # Groups:   tailnum [3,755]
##    tailnum     n
##    <chr>   <int>
##  1 N954UW    206
##  2 N952UW    163
##  3 N957UW    142
##  4 N5FAAA    117
##  5 N38727     99
##  6 N3742C     98
##  7 N5EWAA     98
##  8 N705TW     97
##  9 N765US     97
## 10 N635JB     94
## # ... with 3,745 more rows
```