r4ds Ex 5.7.1

MW

2019/05/29

5.7.1

1

> Refer back to the lists of useful mutate and filtering functions. Describe how each operation changes when you combine it with grouping.

skip…

2

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

```
flights %>% group_by(tailnum) %>%
    summarize(arr_delay = mean(arr_delay)) %>%
    filter(min_rank(desc(arr_delay)) == 1)
```

```
## # A tibble: 1 x 2
##   tailnum arr_delay
##   <chr>       <dbl>
## 1 N844MH        320
```

`N844MH` is worst on time record.

3

> What time of day should you fly if you want to avoid delays as much as possible?

```
flights %>% group_by(hour, minute) %>%
    summarize(arr_delay = mean(arr_delay, na.rm = TRUE)) %>%
    arrange(arr_delay)
```

```
## # A tibble: 1,021 x 3
## # Groups:   hour [20]
##     hour minute arr_delay
##    <dbl>  <dbl>     <dbl>
##  1     7     12     -35.4
##  2     6     26     -30
##  3     5      5     -26.5
##  4    22      8     -26
##  5     5     16     -25.8
##  6     5     55     -25
##  7     5     57     -23.7
##  8     7     26     -21.2
##  9    14     24     -19.5
## 10    23     45     -19
## # ... with 1,011 more rows
```

If you want not to get on delayed airplane, you should use an airplane from an early morning.

4

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

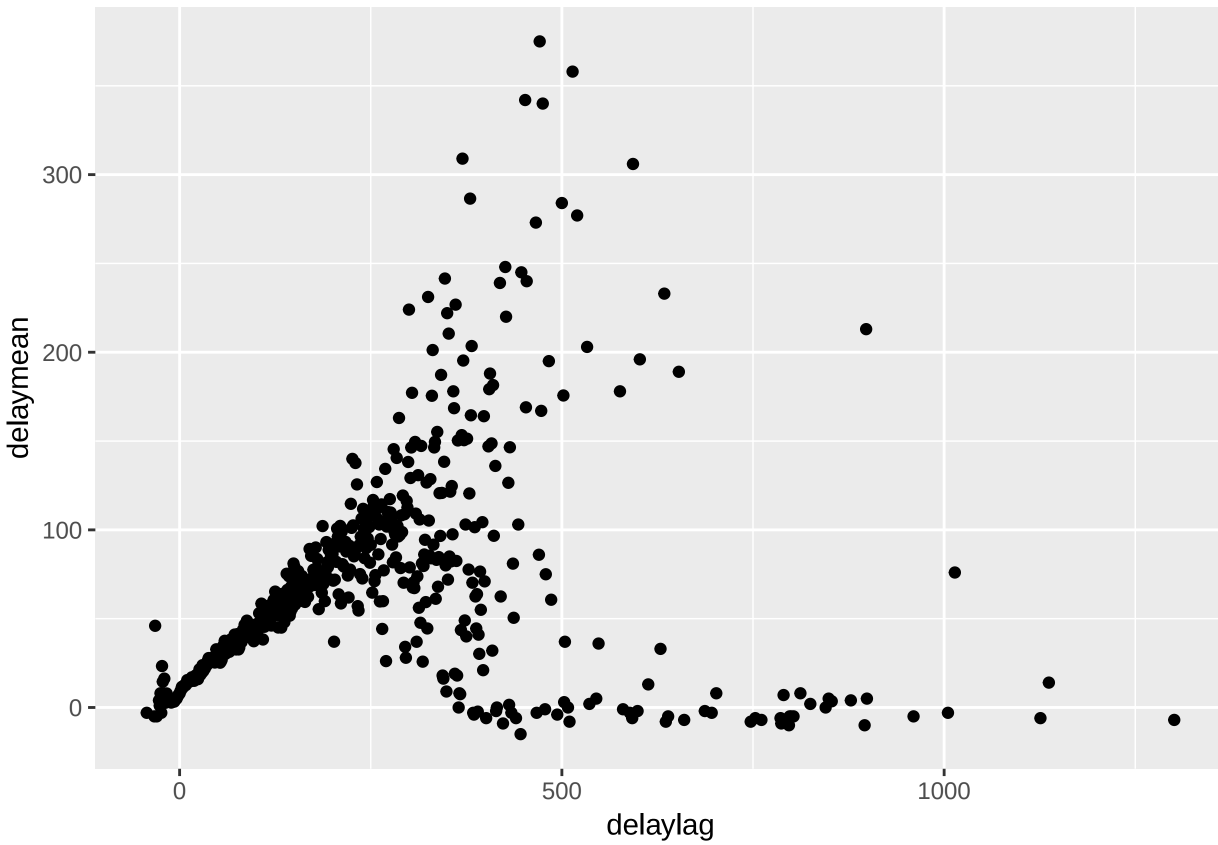
```
flights %>% group_by(dest) %>%
    mutate(sum_delay=sum(arr_delay, na.rm=TRUE), prop_delay=arr_delay/sum_delay)
```

```
## # A tibble: 336,776 x 21
## # Groups:   dest [105]
##     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      542            540         2      923
##  4  2013     1     1      544            545        -1     1004
##  5  2013     1     1      554            600        -6      812
##  6  2013     1     1      554            558        -4      740
##  7  2013     1     1      555            600        -5      913
##  8  2013     1     1      557            600        -3      709
##  9  2013     1     1      557            600        -3      838
## 10  2013     1     1      558            600        -2      753
## # ... with 336,766 more rows, and 14 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>, sum_delay <dbl>, prop_delay <dbl>
```

5

Delays are typically temporally correlated: even once the problem that caused the initial delay has been resolved, later flights are delayed to allow earlier flights to leave. Using `lag()` explore how the delay of a flight is related to the delay of the immediately preceding flight.

```
flights %>% group_by(origin) %>%
    mutate(delaylag = lag(dep_delay)) %>%
    filter(!is.na(dep_delay), !is.na(delaylag)) %>%
    group_by(delaylag) %>%
    summarize(delaymean=mean(dep_delay)) %>%
    ggplot(aes(x=delaylag, y=delaymean))+
        geom_point()
```



6

Look at each destination. Can you find flights that are suspiciously fast? (i.e. flights that represent a potential data entry error). Compute the air time of a flight relative to the shortest flight to that destination. Which flights were most delayed in the air?

7

Find all destinations that are flown by at least two carriers. Use that information to rank the carriers.

8

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