Math 300 NTI Lesson 6

group_by, mutate, and arrange

Professor Bradley Warner

June, 2022

Contents

Objectives

1. Use the `group_by()` function to create aggregated data frames to use with other functions, in particular `summarize()`, to explore, explain, and visualize.
2. Use the `mutate()` function to create new variables in a data frame in order to explore, explain, and visualize.
3. Use the `arrange()` function to sort data frames to explore, explain, and visualize.

Reading

Chapter 3.4 - 3.6

Lesson

Remember that you will be running this more like a lab than a lecture. You want them using `R` and answering questions. Have them open the notes rmd and work through it together.

Work through the learning checks LC3.5 - LC3.12.

- We changed the scaffolded code method. We have `eval=FALSE` so that `R` does not try to evaluate the code chunk. They have to remove this and then complete the code.
- It is important to note that the `group_by()` function doesn't change data frames by itself. Rather it changes the meta-data, or data about the data, specifically the grouping structure. It is only after we apply the `summarize()` function that the data frame changes. The book does a good job explaining meta-data.

- The `group_by()` can be used on more than two variables but they must be in the same call to `group_by()`.
- Using `arrange()` is straight forward expect the use of `desc()` within the `arrange()` call to sort in decreasing order.
- As a rough rule of thumb, as long as you are not losing original information that you might need later, it's acceptable practice to overwrite existing data frames with updated ones.
- LC 3.6 is difficult. Let them explore and wrestle with this question. The warning can be ignored. We will not experiment with the `.groups` option.
- LC 3.12 is more difficult as we combined code. The default use of `geom_boxplot()` works for exploring data but we provided code on how to clean up the x-axis. Discuss this code if you want and have time.
- The use of `kable()` is only to have the output printed in a form that looks good. This is not something we need to present to the students.

Setup

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

LC 3.5 (Objective 1)

(LC3.5) Recall from Chapter 2 when we looked at plots of temperatures by months in NYC. What does the standard deviation column in the `summary_monthly_temp` data frame, which we need to create from the code at the section 3.4, tell us about temperatures in New York City throughout the year?

Solution:

```
# Code from the book
summary_temp_by_month <- weather %>%
  group_by(month) %>%
  summarize(
    mean = mean(temp, na.rm = TRUE),
    std_dev = sd(temp, na.rm = TRUE)
  )
```

```
# Output
summary_temp_by_month
```

```
## # A tibble: 12 x 3
##    month  mean std_dev
##    <int> <dbl>   <dbl>
##  1     1  35.6   10.2
##  2     2  34.3    6.98
##  3     3  39.9    6.25
##  4     4  51.7    8.79
##  5     5  61.8    9.68
##  6     6  72.2    7.55
##  7     7  80.1    7.12
##  8     8  74.5    5.19
##  9     9  67.4    8.47
## 10    10  60.1    8.85
## 11    11  45.0   10.4
## 12    12  38.4    9.98
```

The standard deviation is a quantification of **spread** and **variability**. We see that the period in November, December, and January has the most variation in weather, so you can expect very different temperatures on different days in those months.

LC 3.6 (Objective 1)

(LC3.6) What code would be required to get the mean and standard deviation temperature for each day in 2013 for NYC?

Solution:

```
summary(weather)
```

```
##     origin               year          month             day       
##  Length:26115       Min.   :2013   Min.   : 1.000   Min.   : 1.00  
##  Class :character   1st Qu.:2013   1st Qu.: 4.000   1st Qu.: 8.00  
##  Mode  :character   Median :2013   Median : 7.000   Median :16.00  
##                     Mean   :2013   Mean   : 6.504   Mean   :15.68  
##                     3rd Qu.:2013   3rd Qu.: 9.000   3rd Qu.:23.00  
##                     Max.   :2013   Max.   :12.000   Max.   :31.00  
##                                                                    
##       hour            temp             dewp           humid       
##  Min.   : 0.00   Min.   : 10.94   Min.   :-9.94   Min.   : 12.74  
##  1st Qu.: 6.00   1st Qu.: 39.92   1st Qu.:26.06   1st Qu.: 47.05  
##  Median :11.00   Median : 55.40   Median :42.08   Median : 61.79  
##  Mean   :11.49   Mean   : 55.26   Mean   :41.44   Mean   : 62.53  
##  3rd Qu.:17.00   3rd Qu.: 69.98   3rd Qu.:57.92   3rd Qu.: 78.79  
##  Max.   :23.00   Max.   :100.04   Max.   :78.08   Max.   :100.00  
##                  NA's   :1        NA's   :1       NA's   :1       
##     wind_dir       wind_speed         wind_gust         precip        
##  Min.   :  0.0   Min.   :   0.000   Min.   :16.11   Min.   :0.000000  
##  1st Qu.:120.0   1st Qu.:   6.905   1st Qu.:20.71   1st Qu.:0.000000  
##  Median :220.0   Median :  10.357   Median :24.17   Median :0.000000  
##  Mean   :199.8   Mean   :  10.518   Mean   :25.49   Mean   :0.004469  
##  3rd Qu.:290.0   3rd Qu.:  13.809   3rd Qu.:28.77   3rd Qu.:0.000000  
##  Max.   :360.0   Max.   :1048.361   Max.   :66.75   Max.   :1.210000  
##  NA's   :460     NA's   :4          NA's   :20778                     
##     pressure          visib          time_hour                     
##  Min.   : 983.8   Min.   : 0.000   Min.   :2013-01-01 01:00:00  
##  1st Qu.:1012.9   1st Qu.:10.000   1st Qu.:2013-04-01 21:30:00  
##  Median :1017.6   Median :10.000   Median :2013-07-01 14:00:00  
##  Mean   :1017.9   Mean   : 9.255   Mean   :2013-07-01 18:26:37  
##  3rd Qu.:1023.0   3rd Qu.:10.000   3rd Qu.:2013-09-30 13:00:00  
##  Max.   :1042.1   Max.   :10.000   Max.   :2013-12-30 18:00:00  
##  NA's   :2729                                                   
```

There is only one year 2013 so we don't need to group by it, we could but it would not change anything.

```
summary_temp_by_day <- weather %>%
  group_by(month, day) %>%
  summarize(
    mean = mean(temp, na.rm = TRUE),
    std_dev = sd(temp, na.rm = TRUE)
  )
```

```
## ‘summarise()‘ has grouped output by ’month’. You can override using the
## ‘.groups‘ argument.
```

```
head(summary_temp_by_day)
```

```
## # A tibble: 6 x 4
## # Groups:   month [1]
##   month   day  mean std_dev
##   <int> <int> <dbl>   <dbl>
## 1     1     1  37.0    4.00
## 2     1     2  28.7    3.45
## 3     1     3  30.0    2.58
## 4     1     4  34.9    2.45
## 5     1     5  37.2    4.01
## 6     1     6  40.1    4.40
```

Note: `group_by(day)` is not enough, because `day` is a value between 1-31. We need to `group_by(year, month, day)` or `group_by(month, day)`.

LC 3.7 (Objective 1)

(LC3.7) Recreate `by_monthly_origin`, but instead of grouping via `group_by(origin, month)`, group variables in a different order `group_by(month, origin)`. What differs in the resulting dataset?

Solution:

```
by_origin_monthly <- flights %>%
  group_by(origin, month) %>%
  summarize(count = n())
```

```
## ‘summarise()‘ has grouped output by ’origin’. You can override using the
## ‘.groups‘ argument.
```

```
head(by_origin_monthly)
```

```
## # A tibble: 6 x 3
## # Groups:   origin [1]
##   origin month count
##   <chr>  <int> <int>
## 1 EWR        1  9893
## 2 EWR        2  9107
## 3 EWR        3 10420
## 4 EWR        4 10531
## 5 EWR        5 10592
## 6 EWR        6 10175
```

```
by_monthly_origin <- flights %>%
  group_by(month, origin) %>%
  summarize(count = n())
```

```
## ‘summarise()‘ has grouped output by ’month’. You can override using the
## ‘.groups‘ argument.
```

```
head(by_monthly_origin)
```

```
## # A tibble: 6 x 3
## # Groups:   month [2]
##   month origin count
##   <int> <chr>  <int>
## 1     1 EWR     9893
## 2     1 JFK     9161
## 3     1 LGA     7950
## 4     2 EWR     9107
## 5     2 JFK     8421
## 6     2 LGA     7423
```

In `by_monthly_origin` the `month` column is now first and the rows are sorted by `month` instead of origin. If you compare the values of `count` in `by_origin_monthly` and `by_monthly_origin` using the `View()` function, you'll see that the values are actually the same, just presented in a different order.

LC 3.8 (Objective 1)

(LC3.8) How could we identify how many flights left each of the three airports for each `carrier`?

Solution: We could summarize the count from each airport and carrier using the `n()` function, which *counts rows*.

```
count_flights_by_airport <- flights %>%
  group_by(origin, carrier) %>%
  summarize(count = n())
```

```
head(count_flights_by_airport,n=10)
```

```
## # A tibble: 10 x 3
## # Groups:   origin [1]
##    origin carrier count
##    <chr>  <chr>   <int>
##  1 EWR    9E       1268
##  2 EWR    AA       3487
##  3 EWR    AS        714
##  4 EWR    B6       6557
##  5 EWR    DL       4342
##  6 EWR    EV      43939
##  7 EWR    MQ       2276
##  8 EWR    OO          6
##  9 EWR    UA      46087
## 10 EWR    US       4405
```

Note: the `n()` function counts rows, whereas the `sum(VARIABLE_NAME)` function sums all values of a certain numerical variable `VARIABLE_NAME`.

LC 3.9 (Objective 1)

(LC3.9) How does the `filter` operation differ from a `group_by` followed by a `summarize`?

Solution:

- `filter` picks out rows from the original dataset without modifying them, whereas
- `group_by %>% summarize` computes summaries of numerical variables, and hence reports new values.

LC 3.10 (Objective 2)

(LC3.10) What do positive values of the `gain` variable in `flights` correspond to? What about negative values? And what about a zero value?

Solution:

- Say a flight departed 20 minutes late, i.e. `dep_delay = 20`
- Then arrived 10 minutes late, i.e. `arr_delay = 10`.
- Then `gain = dep_delay - arr_delay = 20 - 10  = 10` is positive, so it "made up/gained time in the air."
- 0 means the departure and arrival delay times were the same, so no time was made up in the air. We see in most cases that the `gain` is near 0 minutes.

LC 3.11 (Objective 2)

(LC3.11) Could we create the `dep_delay` and `arr_delay` columns by simply subtracting `dep_time` from `sched_dep_time` and similarly for arrivals? Try the code out and explain any differences between the result and what actually appears in `flights`.

Solution: No because you can't do direct arithmetic on times. The difference in time between 12:03 and 11:59 is 4 minutes, but `1203-1159 = 44`. Plus there are time zones, departure and arrival times are in the local timezone, which cause problems with simple subtraction.

```
LC3.11<- flights %>%
  mutate(time_gain=dep_time-arr_time,gain = dep_delay - arr_delay) %>%
  select(air_time,dep_time,arr_time,time_gain,dep_delay,arr_delay, gain)
```

```
head(LC3.11)
```

```
## # A tibble: 6 x 7
##   air_time dep_time arr_time time_gain dep_delay arr_delay  gain
##      <dbl>    <int>    <int>     <int>     <dbl>     <dbl> <dbl>
## 1      227      517      830      -313         2        11    -9
## 2      227      533      850      -317         4        20   -16
## 3      160      542      923      -381         2        33   -31
## 4      183      544     1004      -460        -1       -18    17
## 5      116      554      812      -258        -6       -25    19
## 6      150      554      740      -186        -4        12   -16
```

LC 3.12 (Objective 2)

(LC3.12) What can we say about the distribution of `gain`? Describe it in a few sentences using a boxplot and the `gain_summary` data frame values.

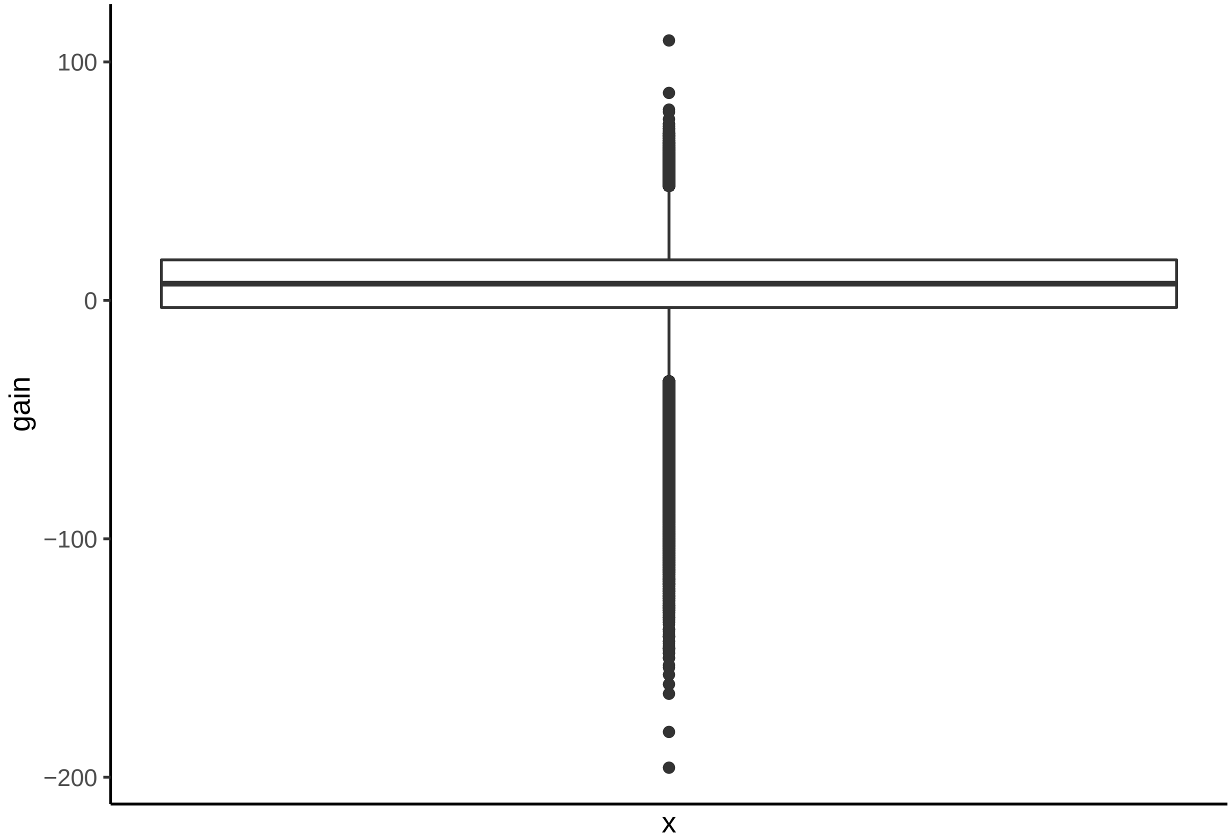
Solution: We must create the data frame from the notes. We copied the code from the book, we had to copy and combine two chunks of code.

```
gain_summary <- flights %>%
  mutate(gain = dep_delay - arr_delay) %>%
  summarize(
    min = min(gain, na.rm = TRUE),
    q1 = quantile(gain, 0.25, na.rm = TRUE),
    median = quantile(gain, 0.5, na.rm = TRUE),
    q3 = quantile(gain, 0.75, na.rm = TRUE),
    max = max(gain, na.rm = TRUE),
    mean = mean(gain, na.rm = TRUE),
    sd = sd(gain, na.rm = TRUE),
    missing = sum(is.na(gain))
  )
```

```
gain_summary
```

```
## # A tibble: 1 x 8
##     min    q1 median    q3   max  mean    sd missing
##   <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>   <int>
## 1  -196    -3      7    17   109  5.66  18.0    9430
```

```
flights %>%
  mutate(gain = dep_delay - arr_delay) %>%
  ggplot(aes(x=1,y=gain)) +
  geom_boxplot() +
  scale_x_continuous(breaks = NULL) +
  theme(axis.title.x = element_blank()) +
  theme_classic()
```



Most of the time the gain is a little above zero (the median is 7, meaning gain is above 0 at least 50% of the time) and between -50 and 50 minutes. There are some extreme cases however!

Documenting software

- File creation date: 2022-06-16
- R version 4.1.3 (2022-03-10)
- `ggplot2` package version: 3.3.6
- `dplyr` package version: 1.0.9
- `nycflights13` package version: 1.0.2