Hussain Sarfraz

Dr. Kates

DATA-101-610-2021C

09/12/2021

Cleaning Data Homework

Here is the homework for Week 2, which gives you an opportunity to practice your `dplyr` skills some more.

1. In the code above, we frequently used `not_cancelled`, rather than flights as our data. How did this simplify our code? Think especially about the functions we used within `summarise()`.

   The object 'not_cancelled' is an object that contains a dataset which has a list of flights that have not been cancelled and have taken off.

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

   Figure 1: A image of the object 'non_cancelled' being defined. As you can see the departure delays and arrival delays are being filtered from the flight's dataset. This means we are asking R to give us the delayed flights only from the flight's dataset

   One might think about using the group_by() function to filter out the delayed flights. This could work but, there are multiple variables that need to be sub-grouped within the 'non_cancelled' dataset. Because of this it would be easier to create an object that stores all the data for non-cancelled flights (in this example, this is the non_cancelled object).

   Also, the mean() function was used within the summarize() function to calculate the mean of delayed flights per day. The mean for the delayed flights was computed based off the groups per day (these groups were created in the group_by() function). If the flight delay variables (dep_delay() and arr_delay()) were used in the group_by() function along with the year, month, and day then R would get confused since it would try to create 2 groups at the same time (one group for each day and another group for delay times). Our objective is to find the average delayed flights per day so it makes sense to only include the year, month, and day variables in the group_by() function and no other variables.

```
not_cancelled %>%
   group_by(.,year, month, day) %>%
   summarize(.,
      avg_delay1 = mean(arr_delay),
      avg_delay2 = mean(arr_delay[arr_delay > 0])
      )
```

Figure 2: A image of the object, 'non_cancelled' being used to find average delay times. The object 'not_cancelled' contains a dataset of delayed flights only (in other words, flights that were not cancelled). The variable avg_delay1 includes the average delays of flights that were delayed and were taken off early (positive and negative values). avg_delay2 calculates the mean of all delayed flights (positive values only)

2. You might suspect that there is a relationship between the average delay (on a given day) and the proportion of flights that are cancelled on that day. For example, if there is bad weather, many flights might start off delayed, but then end up cancelled. Let's test this intuition. First, find the average delay and proportion of flights cancelled each day. Second, plot them against one another and comment on the relationship. Did our intuition hold?

Flights that have longer delays have a higher chance of getting cancelled. I reached this answer by finding the average delay and proportion of flights cancelled. I then used geom_point() to plot the data points and to see a relationship.

NOTE: I used the 'dep_delay' variable to calculate the average flights delay because if a flight is cancelled it would not have a arrival delay so using the variable 'arr_delay' to calculate the flight delay average would not be useful.

I typed this in R:

```
flights %>%
  group_by(.,year, month, day) %>%
  mutate(cancelled = ifelse((is.na(dep_delay) & is.na(arr_delay)),1,0)) %>%
  summarize(.,
            avg_delay1 = mean(dep_delay,na.rm=TRUE),
            flights_proportion = 100*mean(cancelled)) %>%
  filter(flights_proportion < 10) %>%
  ggplot(mapping = aes(x = avg_delay1, y = flights_proportion)) +
  geom_point(alpha=1/10)
```

Figure 3: A image of the code I wrote to find the average delay and proportion for flights. The variable 'avg_delay1' consists of the average delay of all flights (positive and negative values) and na.rm=TRUE is used to remove any blank (NA) values from the calculation. The variable 'flights_proportion' is the percentage value of the flights that were cancelled.

This is the result I got:

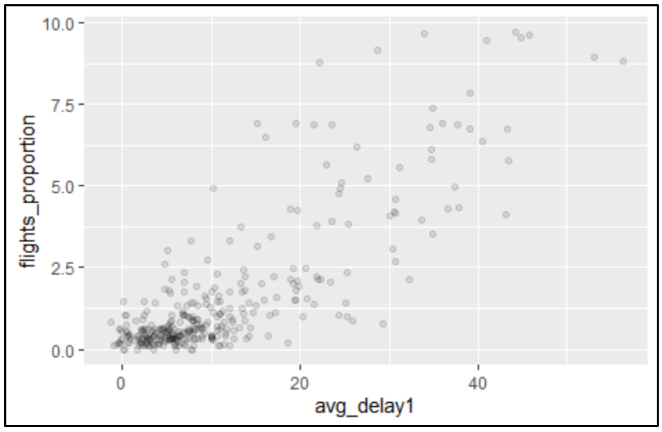

Figure 4: A scatterplot of the relationship between the variables 'avg_delay1' and 'flights_proportion'. As you can see, flights which have a longer delay time have a greater percentage of being cancelled. There are some outliers, such as the flight that had a 14-minute delay. But in general, the flights that have a longer day time have a higher proportion/'probability of getting cancelled' than the other flights.

In figure 3 I used the filter() function to remove some extreme outliers of data that I saw. I didn't want those points to mess up my observations which is why I added the filter function.

This is what the graph looked like without the filter function.

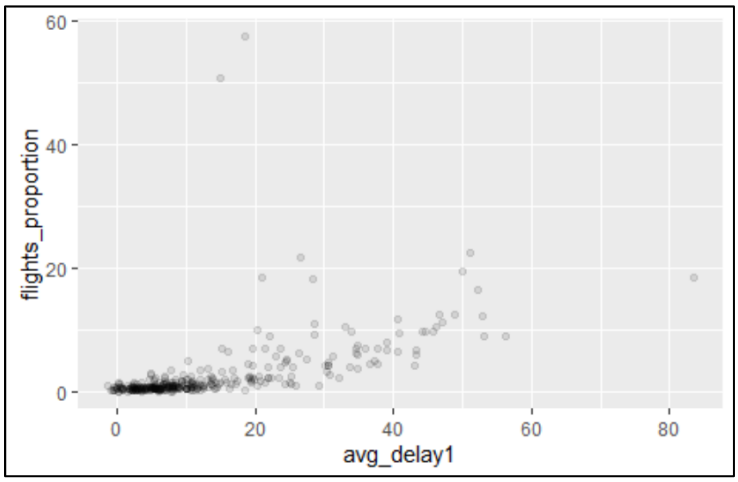

Figure 5: A scatterplot of the relationship between the variables 'avg_delay1' and 'flights_proportion'. This scatter plot does not have the filter function which is why the points in the graph look clustered which makes it a bit hard to read. Therefore, I put the filter() function in the code and set the specific requirement to (flights_proportion < 10). I wanted to remove the outlier points on the top of the graph and wanted to see the points at the bottom of the graph in greater detail, as shown in figure 4.

So, to summarize, flights that have longer delay times have a higher proportion/percentage to be cancelled. This pattern can be seen in figure 4 which shows that as a flights average delay time increases, then the proportion/percentage of the flight being cancelled increases.

3. No one likes to be delayed when flying. To try and avoid this, you might wonder what hour of the day is least likely to have a departure delay. What hour is it? Also, compute the percentage of flights that leave on time or early in each hour (i.e., the flights you want to find!). What hour of the day are you most likely to find these flights?

Flights that have the least delays would occur around 6-7 A.M. and I reached this conclusion from the data points that I plotted in the scatter plot in figure 7. I used this code in R to reach this conclusion and to make the graph.

The code I wrote in R:

```
not_cancelled %>%
  group_by(.,hour) %>%
  mutate(flight_early = ifelse((dep_delay <= 0),1,0)) %>%
  summarize(.,
            ontime_proportion = 100*mean(flight_early)) %>%
  ggplot(mapping = aes(x = hour, y = ontime_proportion)) +
  geom_point()
```

Figure 6: The code I used to create the scatter plot that displayed the relationship with on-time flights and their likelihood of occurrence in each hour

I used the 'not_cancelled' object because it only has the dataset for non-cancelled flights. This way, I do not need to worry about any blank (NA) values that appear when I am filtering the data further.

I then used the mutate() function to add a column that converted the occurrence of early flights and late flights on a 1-0 scale (this would be helpful when finding the proportion of early flights). I used the ifelse() function to find flights that arrived early on time. I did this by setting the condition (dep_delay <= 0) to pull out the negative dep_delay values (the negative values represent an early departure).

The summarize() function was used to calculate the proportion of early flight occurrences for each hour group that I made in the group_by() function.

I then created another variable called 'ontime_proportion' to get the percentage of the early flight occurrences and displayed this in the table below as the y-axis.

NOTE: since the question only asked for departure delays, I only used the dep_delay variable and not the arr_delay variable

My output:

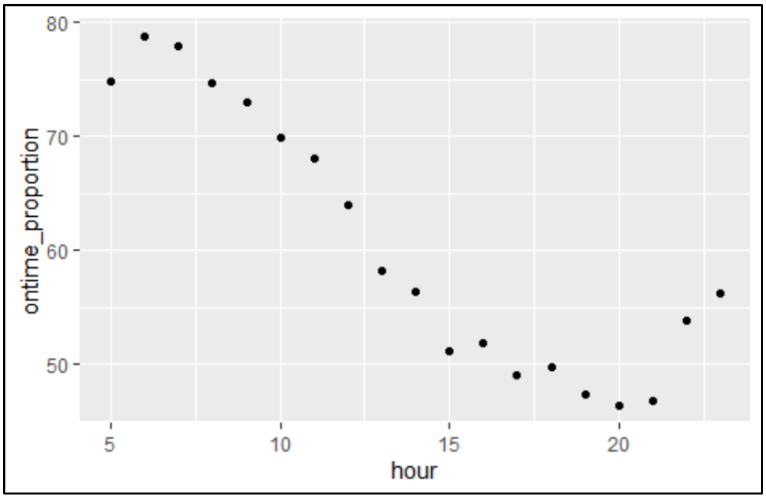

Figure 7: A scatter plot of the proportion of early and on-time flights that occur every hour. As you can see around 5-8 AM is the best time to book a flight since there is a high chance that the flights going around those times will depart early or on time.

4. Which carriers are most likely to have a departure delay of at least 30 minutes? *Hint:* using the `ifelse()` function may be helpful

ExpressJet (EV) is the airline that will most likely have a departure delay for 30 minutes or more with a proportion of roughly 26%. Mesa Airlines (YV) comes in second with a proportion of roughly 23%. I reached this conclusion from the scatterplot in figure 9. I used the code below to reach this conclusion. I also displayed this data in a boxplot to display the data in another format visualization.

What I wrote in R:

```
not_cancelled %>%
  group_by(.,carrier) %>%
  mutate(flight_30minlate = ifelse((dep_delay >= 30),1,0)) %>%
  summarize(.,
            f_30minlate_proportion = 100*mean(flight_30minlate)) %>%
  ggplot(mapping = aes(x = carrier, y = f_30minlate_proportion)) +
  geom_point()
```

Figure 8: The code used to calculate and graph the proportion of 30 or 30+ minute delayed flights. I then sorted this by each carrier.

I used the 'not_cancelled' object because it only has the dataset for non-cancelled flights. This way, I do not need to worry about any blank (NA) values that appear when I am filtering the data further.

I then used the mutate() function to add a column that converted the flights that occurred 30 minutes or late on a 1-0 scale (this would be helpful when finding the proportion of 30+ minute late flights). I used the ifelse() function to find flights that arrived 30 or 30+ minutes late. I did this by setting the condition (dep_delay >= 0) to pull out the 30 or 30+ departure delay values.

The summarize() function was used to calculate the proportion of 30 or 30+ departure delay occurrences for each carrier group that I made in the group_by() function.

I then created another variable called 'f_30minlate_proportion' to get the percentage of the 30 or 30+ flight delay occurrences and displayed this in the table below as the y-axis.

NOTE: since the question only asked for departure delays, I only used the dep_delay variable and not the arr_delay variable

My Output:

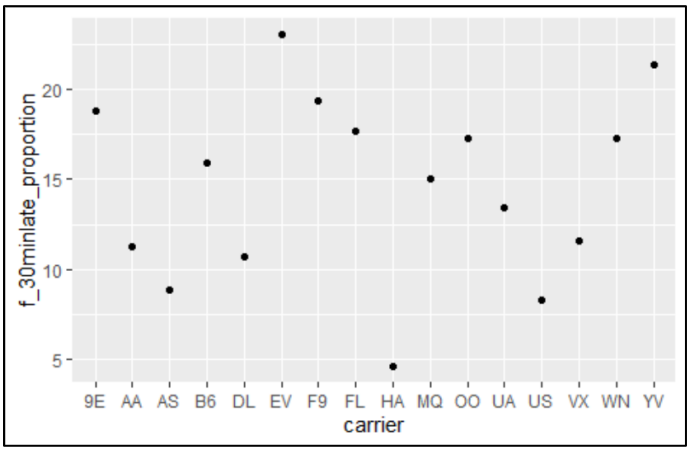

Figure 9: A scatter plot of the proportion of flights 30+ minutes late, sorted by each airline carrier.

I also decided to display this data through a bar chart to show that there is another way to display the data.

```
not_cancelled %>%
  group_by(.,carrier) %>%
  mutate(flight_30minlate = ifelse((dep_delay >= 30),1,0)) %>%
  summarize(.,
            f_30minlate_proportion = 100*mean(flight_30minlate)) %>%
  ggplot(mapping = aes(x = carrier, y = f_30minlate_proportion)) +
  geom_bar(stat='identity')
```

Figure 18: The code I used to display the data that was on figure 9 on a bar chart. I used stat='identity' to tell R to not use an automatic calculation for the y-axis and to use the calculations made in the variable 'f_30minlate_proportion'

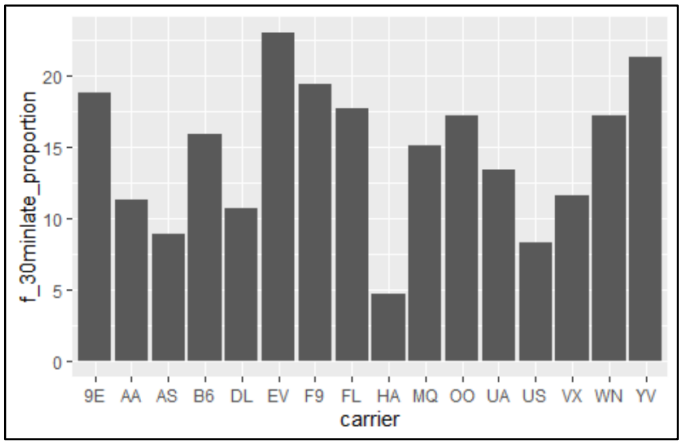

Figure 19: A bar chart of the proportion of flights 30+ minutes late, sorted by each airline carrier. As you can see the graph displays the same data shown in Figure 9

5. What destination has the smallest average arrival delay?

Lexington (LEX) has the smallest average arrival delay with an average of -22. I got this answer from the code I wrote which I talk about further below.

For the code I wrote, I grouped everything by destination since the question wanted the lowest average arrival delay for each destination. I then got the mean for all arrival delays for each destination through the summarize() function. I stored that value in the variable 'arrivaldelay_average'.

NOTE: since the question only asked for arrival delays, I only used the 'arr_delay' variable and not the 'dep_delay' variable

The code I wrote in R:

```
not_cancelled %>%
  group_by(.,dest) %>%
  summarize(.,
            arrivaldelay_average = mean(arr_delay)) %>%
  ggplot(mapping = aes(x = dest, y = arrivaldelay_average)) +
  geom_point()
```

Figure 10: The code I initially wrote to graph the arrival average delays for all destinations

The output:

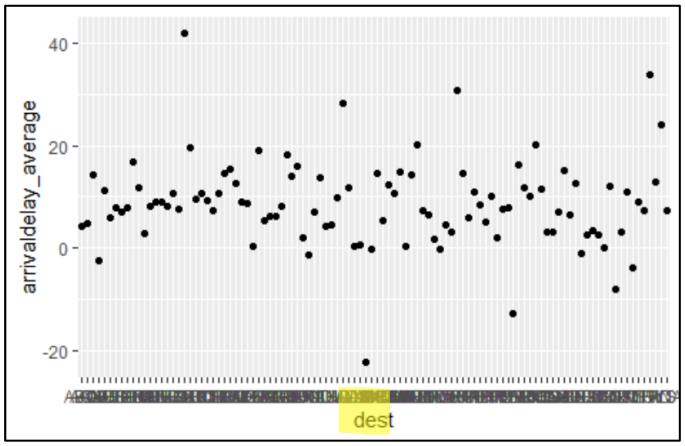

Figure 11: The output I got when I ran the code written in figure 9. I realized that I could not read the destinations since there were too many. So, I decided to make my code more readable by adding the filter() function. ***NOTE****: I highlighted the point in yellow since that was the point, I needed the location of. The question asked for lowest arrival average which is why I highlighted that point.*

The second piece of code that I entered (with the filter function):

```
not_cancelled %>%
  group_by(.,dest) %>%
  summarize(.,
            arrivaldelay_average = mean(arr_delay)) %>%
  filter(arrivaldelay_average < -20) %>%
  ggplot(mapping = aes(x = dest, y = arrivaldelay_average)) +
  geom_point()
```

Figure 12: The code that I wrote again this time with the filter() function. The filter function is asking R to only show the arrival delay averages which are less than -20. I got this number from my observations looking at the graph in figure 11.

The output:

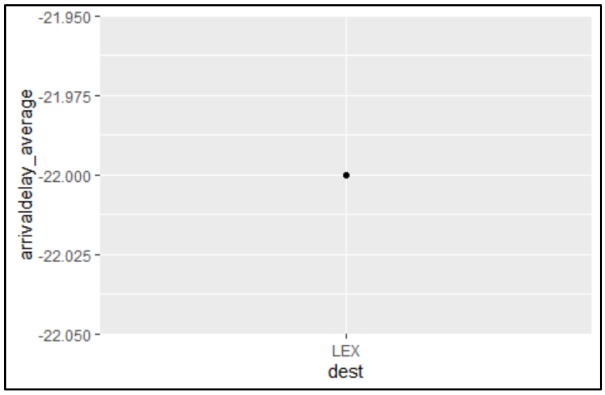

Figure 13: The second graph that I got after running the code in figure 12. This graph that is clearer than the one in figure 11 and gives me the data that I am looking for which is the destination that has the smallest average.

6. **BONUS:** Load the Lahman() library, which contains data on baseball players and their batting averages. First, convert it to a tibble (the tidyverse data structure we'll cover in a future lecture) by calling: `batting <- as_tibble(Lahman::Batting)`. Remember that with a built-in R data like `Batting`, you can write `?Batting` in the R Console to display the help file, which will explain what the variables mean.

   a. Then find the players with the best or worst batting averages (batting average is simply the number of hits a player has, divided by the number of at bats they have). Why would this lead you astray?

When graphing this data, it would be impossible to see which players have the best and worst batting averages for 2 main reasons:

1. All the players have many batting averages, and this fills up the scatterplot making it impossible to even filter out the best or worst batting average.
2. There would be many players in the x-axis so it would be hard to see the name of all players. This can easily be fixed by adding a filter function to only show the graph of the best and worst players.

What I entered in R:

```
Batting %>%
  group_by(.,playerID) %>%
  summarize(.,
            batting_average = H/AB) %>%
  ggplot(mapping = aes(x = playerID, y = batting_average)) +
  geom_point()
```

Figure 14: The code I entered in R to see the best and worst batting averages for each baseball player

I used the 'group_by' function to separate each player through their ID. This way I can calculate the batting average for each player. I then use the summarize function to define a variable 'batting_average' which gets the number of hits a player has (H) and divides that by the number of bates they have (AB).

Since the variable 'player ID' is a categorical variable and 'batting_average' is a continuous variable, I am using a scatter plot to display my data.

My Output:

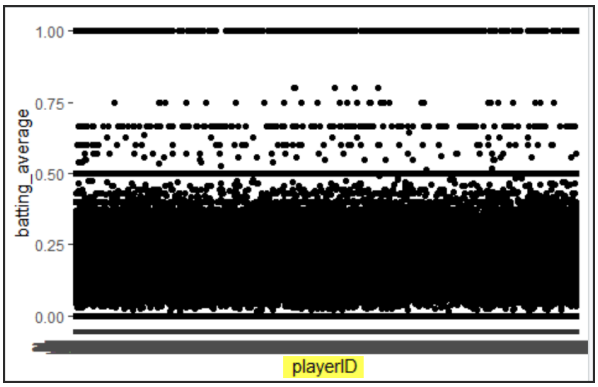

Figure 15: The graph I got that shows each players batting average. As you can see each player has so many batting averages that it fills up the graph and makes it hard to read or analyze. The x-axis is also hard to read since there are many players.

b. Now condition on players who had at least 500 at bats. How would you answer change?

Now I can see some points in my scatterplot, although it is still filled, I can see more points clearly. I would still need to set further conditions if I would like to see the data more clearly or even make different objects that represent different datasets.

```
Batting %>%
  group_by(.,playerID) %>%
  filter(AB >= 500) %>%
  summarize(.,
            batting_average = H/AB) %>%
  ggplot(mapping = aes(x = playerID, y = batting_average)) +
  geom_point()
```

Figure 16: The code used to graph the batting averages of player who had at least 500 bats

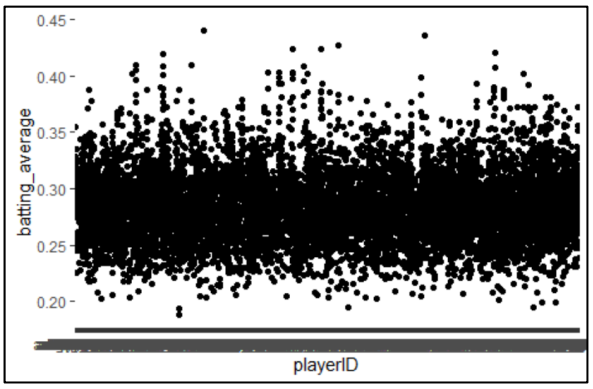

Figure 17: The graph that displays the batting averages of player who have at least 500 bats. The x-axis still can't be read because there are a lot of players, but the points can be seen a bit more clearly as compared to the scatterplot in figure 15.

Feedback

1. You got most of the pieces here! The entire flights dataset includes a bunch of missing values (NA) which you filter out with !is.na(dep_delay). This helps us because NAs in our dataset create problems when summarising later on. 0.5/1
2. Awesome job here. It's great to notice and filter out the outliers, so nice job finding those! Exactly right. 3/3
3. Nice job here! I liked being able to see this all laid out on a graph. You could also look at a table and sort by the proportion and get the answer too. Exactly right on your answers here. 3/3
4. Perfect! 2/2
5. 1/1
6. You were super close here. In our dataset, there were players who only batted once or twice, and got a hit each time (or never hit at all), so their perfect averages are skewing our data. We want to filter to players with larger at-bat sample sizes, so we can see true averages that give us more information. Don't be afraid to look at the raw tables and move your data around that way for problems like these. Visualization can be a great tool, but in situations like these where you noted how many sheer observations there are, it isn't always our most useful path. 1/2