STAT 29000 Project 12

Topics: tidyverse, data.table part 4

Motivation: The becoming adept at using `tidyverse` and/or `data.table` will prove to be useful in data wrangling tasks, and to strengthen your resume for landing an internship or job. For this reason, we are going to continue to use these packages and solve problems that will expose you to more features.

Context: We now have experience in using the `tidyverse` suite of packages and `data.table` to solve problems. We are going to continue to learn to use these tools in different ways to solve more problems.

Scope: `data.table`, `tidyverse`, and the base R packages all provide utility to the R ecosystem. It is useful to deepen our understanding of what each of these contributes to our ability to analyze data.

You can read more about `data.table` in the official documentation:

https://cran.r-project.org/web/packages/data.table/data.table.pdf

You can read more about `tidyverse` in the official documentation

https://cran.r-project.org/web/packages/tidyverse/tidyverse.pdf

and on the `tidyverse` website:

https://www.tidyverse.org/

Don't forget the very useful documentation shortcut `?`. To use, simply type `?` in the console, followed by the name of the function you are interested in.

You can also look for package documentation by using `help(package=PACKAGENAME)`.

Question 1: maps

1a. We want to take a look at flights during the years 2012-2019, which have origin or destination (or both) in the state of Florida. We've explored how doing some preprocessing using a tool like `awk` can greatly reduce the time spent getting at the data you'd actually like. `data.table`'s `fread()` function has a cool feature where it lets you use a shell command to preprocess a file being read in. Use `fread()` and its useful `cmd` feature to read in all flight data from the years 2012-2019, where either the origin or the destination (or both) is Florida.

Hint: The result should be a `data.table` with 6760402 rows and 110 columns.

Hint: Try grepping multiple files. Notice how the output will be prepended with the filename. This could cause issues. To avoid this, look at the manual by running `man grep`.

Hint: If you have a different amount of rows, run `flight_data[,table(YEAR)]` on your new `data.table`. If there are any unexpected values, take note and remedy the situation.

Hint: Convenience features of fread are discussed here:

https://github.com/Rdatatable/data.table/wiki/Convenience-features-of-fread

1b. Use `ggmap` and `gridExtra` to create a 1x2 or 2x1 grid of maps. The first/top map should show a `geom_point` for each origin city, scaled by the square root of the number of flights from that origin to Florida. The second/bottom map should be zoomed in on Florida, and show a `geom_point` for each destination city, scaled by number of flights to that destination in Florida.

The resulting map is depicted at:

https://datamine.purdue.edu/seminars/fall2019/stat29000project12question1b.jpg

Hint: Since `ggmap` uses Google's API, you will need to recall your Google API key from Project 2.

Hint: Use the option `zoom` from the `get_map()` function from `ggmap()`. A good starting place to test the zoom values is somewhere around 4 and 9.

Hint: The function `geocode()` is a useful tools to get latitude and longitude of locations/cities given their names. If you use `geocode`, remember that you need to set 2 columns: one for longitude and one for latitude. Make sure to pay attention to the order that is returned (longitude first, then latitude).

Hint: Depending on your zoom, when plotting the origin cities, a few cities may be automatically excluded because they are outside your map. That is ok.

Hint: To scale the points, you may want to look at the option `size` in your point's aesthetics. Giving your points some transparency (`alpha`) may also be useful.

Hint: ggmap examples are discussed here:

https://blog.dominodatalab.com/geographic-visualization-with-rs-ggmaps/

Question 2: flying to Florida

We've provided you with pre-wrangled set of data from project 11. Load this dataset from the provided

`/class/datamine/data/examples/stat29000project12.RData`

like this:

```
load("/class/datamine/data/examples/stat29000project12.RData")
```

2a. Create two new columns (one called `flights_in`, one called `flights_out`) in our disney `data.table`, named `dat`. (Note that `dat` gets imported automatically with the `stat29000project12.RDate` file.) These columns should show the number of flights into one of the three closest airports to Disney World, and out of one of the three closest airports to Disney World, respectively, each day (`FL_DATE` column). The three closest airports to Disney World are: Orlando (MCO), Sanford (SFB), and Tampa (TPA).

Hint: Make sure to be pay attention to date formats when combining the datasets using the date. You will need to make some modifications to ensure they are using the same date format. Package `lubridate`, and functions `format`, `ymd`, `mdy` may be useful.

2b. Show the median number of flights coming into our select airports for each month-and-year pair.

Hint: Be careful. Our `flights_in` and `flights_out` columns show the totals for the day. Our newly supplemented Disney dataset has replicated values, because it samples wait times multiple times per day.

2c. We can calculate correlation between two variables using `cor()`. For instance, you can try: `cor(mtcars$hp, mtcars$mpg)`. As you can see, a cars horsepower and miles per gallon are correlated.

Within each month-and-year pair, calculate the correlation between the average posted waiting times and the number of flights into our select airports.

2d. There is some anecdotal evidence that suggests that the posted wait times for rides towards the end of the day are inflated compared to the actual wait times. Let's say 5PM or later qualifies as "towards the end of the day". Create a series of side-by-side bar plots (1 plot per day of the week), showing the difference in average posted vs average actual wait times for "towards the end of the day" and for all other hours. Make sure you make your graphic more glamorous by adding a theme and/or better labels.

The resulting map is depicted at:

https://datamine.purdue.edu/seminars/fall2019/stat29000project12question2d.jpg

Hint: To unstack a bar plot, look at the `position_dodge()` function, and the `position` argument.

Question 3: for the first time in forever

3a. Find the first day and time the posted wait time for Disney's Flight of Passage ride was less than or equal to 60 minutes between the times of 9AM-5PM.

Hint: You may want to look at the behavior of functions `which.min()` and `which.max()` for a vector of 0 and 1's.

3b. When was the first time that Splash Mountain had a higher maximum posted wait time in a day than each of the other rides? (Use column `date`; this is not `day_of_week`.) Get each such date, for each ride, compared to Splash Mountain.

Project Submission:

Submit your solutions for the project at this URL: https://classroom.github.com/a/z7BzVZ_I using the instructions found in the GitHub Classroom instructions folder on Blackboard.