Stat. 650 - Midterm Solution

Eric A. Suess

2020-09-16

To run this notebook, in your Midterm R Project run the R script first to create the lyftbaywheels dataframe for 2019 and 2020..

Note that the script saves the dataframe to an .Rds, .csv, .feather file in the *data_final* subdirectory. This can be read into the Global Environment where you answer the questions from the Midterm.

Read the data into the Global Environment so the R Notebook will Preview and Knit.

```
library(pacman)

p_load(tidyverse, tictoc, arrow, naniar, DataExplorer)
```

Try running your data wrangling R script. Or try using the Source > Source as Local Job . . .

What is the advantage of using the source() function to run your R script separately?

Only need to run this once. So after you have created a .Rds file you do not need to run this again. So note the eval=FALSE., this prevents the knit from running this R code chunck when knitting your R Notebook

```
source('lyftbaywheels01-update2020-ver03.R')
```

Test out some different ways to read a dataframe to the harddrive. The winner is feather!

```
tic()
lyftbaywheels1 <- readRDS("data_final/lyftbaywheels_final.Rds")
toc()
```

```
## 3.742 sec elapsed
```

```
tic()
lyftbaywheels2 <- read_feather("data_final/lyftbaywheels_final.feather")
toc()
```

```
## 1.359 sec elapsed
```

Your solution to the Midterm should show you have “come up to speed” on the current project of working with and analyzing the LyftBayWheels data.

Answers to the questions:

1. Explain what the GBFS is?

Answer: GBFS stands for General Bikeshare Feed Specification, a standardized data feed for bike share system availability. It is maintained by the North American Bike Share Association.

2. Explain any difficulties you encountered getting the code to work.

Answer: The answer to this question varies.

I have had many difficulties working with the FordGoBike data. My difficulties started with the errors I was receiving the the start_id and end_id variables not being read in correctly with the red_csv(). Figuring out the type conflict took some effort to find this problem. Learning to convert type of columns was a learning experience. Dealing with the day of the week variable was another difficulty. Getting the day of the week was possible with the day() function, but then realizing this is the number for the calendar month, this was not useful. Then discovering the wday() function was very useful, but the label option was necessary to convert to M, Tues, W, Th, F values.

Other difficulties:
a. Running out of memory on their computer. The main way to deal with this is to reduce the amount of data imported. So maybe working with on May, June, July 2018 data, rather than working with all of the data. b. Dealing with the missing data. Wlyftbaywheels1hen replacing the age values greater than 100 may change the type of variable to char and then it needs to be returned to integer. c. There were some mistakes in the code. For example, when fixing the type problems with the June, July, August 2018 data files there was a mistake with the end_station_id be assigned the start_station_id. d. libcurl on Windows.

3. The analysis is to work with the data since Lyft BayWheels started, start with the data since May 2019. Modify the code to download the data to be analyzed. How many bike rentals have there been before the COVID-19 lockdown in CA? How many bike rentals were there been after the lockdown? How many bike rentals have there been since the beginning of Lyft BayWheels?

Answer: See the R script that contains the data wrangling steps.

Before COVID-19	?
After COVID-19	3,229,177

```r
lyftbaywheels1 %>% count()
```

```
## # A tibble: 1 x 1
##         n
##     <int>
## 1 3229177
```

4. There is a part of the code that uses the as.integer() function for some reason. Explain what this function is being used for in the code.

Answer: The as.integer() function is used in the fordgobike01.Rmd code to change the type of variable, for start_station_id and end_station_id, in the June, July, August 2018 dataframes from *char* to *integer* values, which are what they are in the earlier dataframes.

Note that the read_bulk function from the readbulk R package does not have this problem.

5. In 2020, what month had the highest number of riders? What month had the lowest number of riders? Interpret any seasonal patterns.

Answer: The Age variable is created using this code:

highest	?
lowest	?

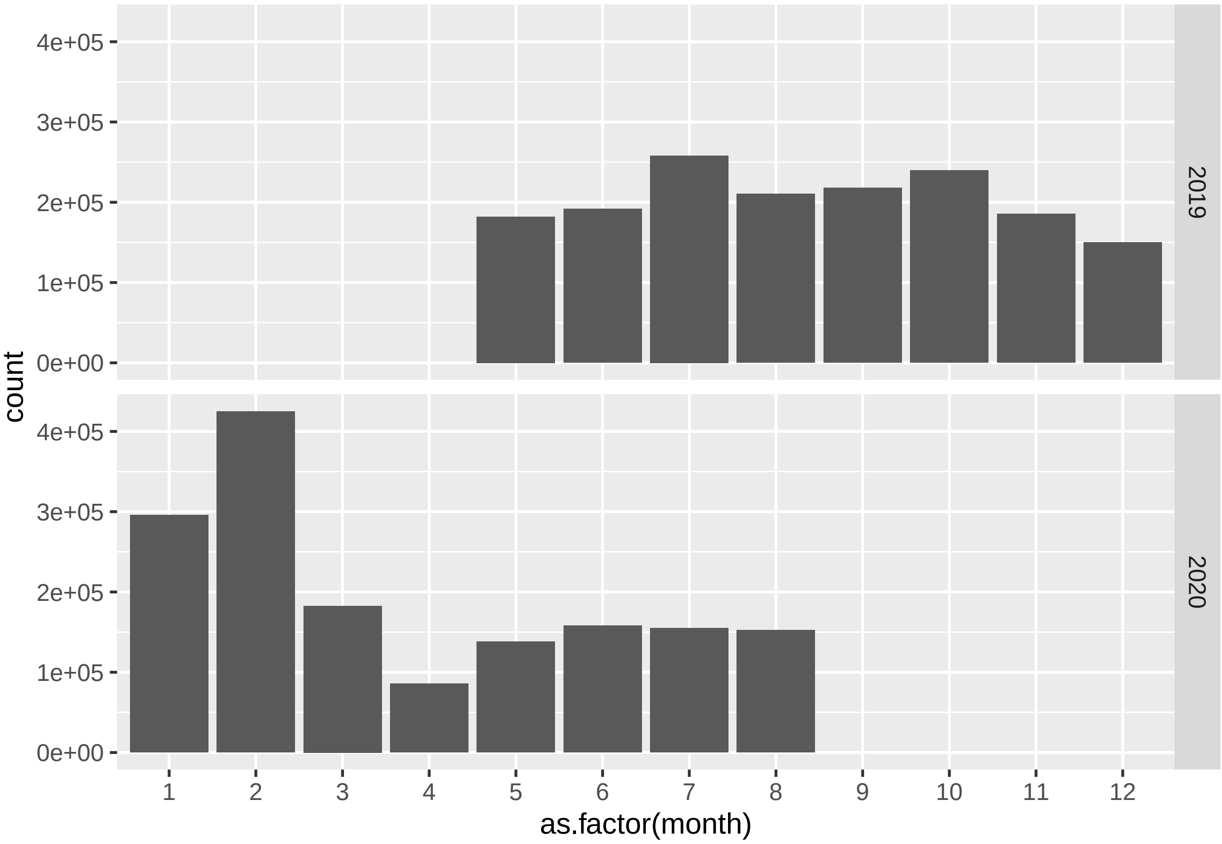
```
lyftbaywheels1 %>% head()
```

```
## # A tibble: 6 x 22
##   duration_sec start_time          end_time            start_station_id
##          <dbl> <dttm>              <dttm>                         <int>
## 1        48877 2019-05-31 20:34:56 2019-06-01 10:09:34              321
## 2        47050 2019-05-31 19:43:56 2019-06-01 08:48:06              246
## 3         5912 2019-05-31 23:54:24 2019-06-01 01:32:56              149
## 4         2629 2019-05-31 23:59:57 2019-06-01 00:43:46              186
## 5         4235 2019-05-31 23:05:48 2019-06-01 00:16:24               34
## 6         2954 2019-05-31 23:26:52 2019-06-01 00:16:07               50
## # ... with 18 more variables: start_station_name <chr>,
## #   start_station_latitude <dbl>, start_station_longitude <dbl>,
## #   end_station_id <int>, end_station_name <chr>, end_station_latitude <dbl>,
## #   end_station_longitude <dbl>, bike_id <int>, user_type <chr>,
## #   bike_share_for_all_trip <chr>, rental_access_method <chr>, ride_id <chr>,
## #   rideable_type <chr>, is_equity <lgl>, year <int>, month <int>, day <int>,
## #   week_day <fct>
```

### 6. What start station had the highest number of rides? That is, which start station was used most to start rides?

Answer: July had the highest number of riders. **January** had the lowest. There seems to be more riders in the summer than in the winter. This is from the number of records/observations/rows. See plot below.

```
lyftbaywheels1 %>% ggplot(aes(x=as.factor(month))) + geom_bar() + facet_grid(year ~ .)
```



7. Using the Amelia R package and the missmap() function determine the rate of missing data in the month of June 2020. Or try the visdat package and the vis_miss() function. Or check out the the naniar R package. (This might not work on your computer if you have too little RAM.) If you cannot get your code to run, sample the data first.

Answer: Write your answer.

```
lyftbaywheels1 %>% select(start_station_id, start_station_name) %>%
  count(start_station_id, start_station_name) %>%
  arrange(desc(n))
```

```
## # A tibble: 502 x 3
##    start_station_id start_station_name                                        n
##               <int> <chr>                                                 <int>
##  1               NA ""                                                   805784
##  2               58 "Market St at 10th St"                                41721
##  3               81 "Berry St at 4th St"                                  39100
##  4               30 "San Francisco Caltrain (Townsend St at 4th St)"      37885
##  5               15 "San Francisco Ferry Building (Harry Bridges Plaza)"  32424
##  6                3 "Powell St BART Station (Market St at 4th St)"        30788
##  7               21 "Montgomery St BART Station (Market St at 2nd St)"    29729
##  8               22 "Howard St at Beale St"                               27524
##  9                5 "Powell St BART Station (Market St at 5th St)"        24687
## 10                6 "The Embarcadero at Sansome St"                       23096
## # ... with 492 more rows
```

8. What Type of rider uses the Lyft BayWheels more? Subscribers or Customers?

Answer: Calculations are after the tables.

Age:

Min	Max	Mean
18	99	37.13

Mean age by gender:

Female	Male	Other
35.7	37.58	37.11

```r
lyftbaywheels1 %>% head()
```

```
## # A tibble: 6 x 22
##   duration_sec start_time          end_time            start_station_id
##          <dbl> <dttm>              <dttm>                         <int>
## 1        48877 2019-05-31 20:34:56 2019-06-01 10:09:34              321
## 2        47050 2019-05-31 19:43:56 2019-06-01 08:48:06              246
## 3         5912 2019-05-31 23:54:24 2019-06-01 01:32:56              149
## 4         2629 2019-05-31 23:59:57 2019-06-01 00:43:46              186
## 5         4235 2019-05-31 23:05:48 2019-06-01 00:16:24               34
## 6         2954 2019-05-31 23:26:52 2019-06-01 00:16:07               50
## # ... with 18 more variables: start_station_name <chr>,
## #   start_station_latitude <dbl>, start_station_longitude <dbl>,
## #   end_station_id <int>, end_station_name <chr>, end_station_latitude <dbl>,
## #   end_station_longitude <dbl>, bike_id <int>, user_type <chr>,
## #   bike_share_for_all_trip <chr>, rental_access_method <chr>, ride_id <chr>,
## #   rideable_type <chr>, is_equity <lgl>, year <int>, month <int>, day <int>,
## #   week_day <fct>
```