Assignment 5

Kyle Yeo

Assignment 5

Kyle yeo

Due Friday, October 2, 11:59pm CT

The purpose of this assignment is to give you practice using lubridate commands and to review dplyr, ggplot2, and basic exploratory data analysis skills.

Turn in an HTML file and this R Markdown file after you have edited it.

Data

The questions involve five data sets involving international flights arriving to Chicago's O'Hare airport from January 1, 2016 through June 30, 2020 with one separate file for each year.

Each data set is in five separate CSV files: `ORD-2016.csv`, `ORD-2017.csv`, `ORD-2018.csv`, `ORD-2019.csv`, and `ORD-2020.csv`.

Problems

1

> Read in the five data sets. If needed, change the date variable into date format. (The date is recorded inconsistently across the data sets.) Use `bind_rows()` to combine these data sets into a single data set. Add columns for *year*, *month* (character valued, Jan-Dec), *day* (day of the month), and *wday* (day of the week, character valued, Sun - Sat). Reorder the variables so that these new variables all appear directly after date. Remove the terminal variable. Rename *all_total* to *passengers*, *all_flights* to *flights*, and *all_booths* to *booths*. Arrange the rows by date and hour. Remove the data sets from each individual year (use `rm()`).

> After these changes, how many rows and columns are in the complete data set?

```
ord1 <- read_csv("C:/stat_240/data/ORD-2016.csv")
ord2 <- read_csv("C:/stat_240/data/ORD-2017.csv")
ord3 <- read_csv("C:/stat_240/data/ORD-2018.csv")
ord4 <- read_csv("C:/stat_240/data/ORD-2019.csv") #ymd
ord5 <- read_csv("C:/stat_240/data/ORD-2020.csv") #ymd

ord1 <- ord1 %>%
  mutate(date = mdy(date))
ord2 <- ord2 %>%
  mutate(date = mdy(date))
ord3 <- ord3 %>%
  mutate(date = mdy(date))
ord4 <- ord4 %>%
  mutate(date = ymd(date))
ord5 <- ord5 %>%
  mutate(date = ymd(date))

all_data <- bind_rows(ord1, ord2, ord3, ord4, ord5)



final_data <- all_data %>%
  mutate(year = year(date)) %>%
  mutate(month = month(date, label = TRUE)) %>%
  mutate(day = day(date)) %>%
  mutate(wday = wday(date, label = TRUE)) %>%
  select(airport, date, year, month, day, wday, everything()) %>%
  select(-terminal) %>%
  rename(passengers = all_total, flights = all_flights, booths = all_booths) %>%
  arrange(date, hour)


final_data
```

```
## # A tibble: 29,450 x 24
##    airport date        year month   day wday  hour        us_avg_wait us_max_wait
##    <chr>   <date>     <dbl> <ord> <int> <ord> <chr>             <dbl>       <dbl>
##  1 ORD     2016-01-01  2016 1         1 금    0000 - 01~            8          15
##  2 ORD     2016-01-01  2016 1         1 금    0400 - 05~           28          40
##  3 ORD     2016-01-01  2016 1         1 금    0500 - 06~           12          44
##  4 ORD     2016-01-01  2016 1         1 금    0700 - 08~            4          25
##  5 ORD     2016-01-01  2016 1         1 금    0800 - 09~            8          38
##  6 ORD     2016-01-01  2016 1         1 금    0900 - 10~            5          14
##  7 ORD     2016-01-01  2016 1         1 금    1000 - 11~           40          77
##  8 ORD     2016-01-01  2016 1         1 금    1100 - 12~            3          15
##  9 ORD     2016-01-01  2016 1         1 금    1200 - 13~            8          34
## 10 ORD     2016-01-01  2016 1         1 금    1300 - 14~           13          52
## # ... with 29,440 more rows, and 15 more variables: non_us_avg_wait <dbl>,
## #   non_us_max_wait <dbl>, all_avg_wait <dbl>, all_max_wait <dbl>,
## #   all_n_0..15 <dbl>, all_n_16..30 <dbl>, all_n_31..45 <dbl>,
## #   all_n_46..60 <dbl>, all_n_61..90 <dbl>, all_n_91..120 <dbl>,
## #   all_n_120.. <dbl>, all_excluded <dbl>, passengers <dbl>, flights <dbl>,
## #   booths <dbl>
```

```
rm(ord1, ord2, ord3, ord4, ord5)
```

29450 rows and 24cols

2

> Do any rows contain missing data? If so, how many? Are there any dates in the range from January 1, 2016 through June 30, 2020 that are missing? If so, which ones?

Solution

```
count_na <- function(x)
{
  return ( sum(is.na(x) ) )
}

final_data %>%
  summarise_all(count_na)
```

```
## # A tibble: 1 x 24
##   airport  date  year month   day  wday  hour us_avg_wait us_max_wait
##     <int> <int> <int> <int> <int> <int> <int>       <int>       <int>
## 1       0     0     0     0     0     0     0           0           0
## # ... with 15 more variables: non_us_avg_wait <int>, non_us_max_wait <int>,
## #   all_avg_wait <int>, all_max_wait <int>, all_n_0..15 <int>,
## #   all_n_16..30 <int>, all_n_31..45 <int>, all_n_46..60 <int>,
## #   all_n_61..90 <int>, all_n_91..120 <int>, all_n_120.. <int>,
## #   all_excluded <int>, passengers <int>, flights <int>, booths <int>
```

```
missing_dates <- final_data["date"] %>%
  distinct() %>%
  unlist() #vector


date_seq <- seq(ymd("2016-01-01"), ymd("2020-06-30"), 1)
date_seq[!date_seq %in% missing_dates]
```

```
## [1] "2016-03-07" "2016-03-08" "2016-03-09" "2018-05-08" "2019-10-28"
## [6] "2019-10-29" "2019-10-30" "2020-02-08"
```

There is no missing data and there are 8 missing dates in the range from January 1, 2016 through June 30, 2020, which are "2016-03-07", "2016-03-08", "2016-03-09", "2018-05-08", "2019-10-28", "2019-10-29", "2019-10-30", "2020-02-08."

3

> Calculate the total numbers of flights and passengers in each month and year and store this information in a table. Summarize this table to find the total number of passengers and flights in each year from 2016 - 2019. Which year has the most of each?

Solution

```
final_data %>%
  group_by(year, month) %>%
  summarize(passengers = sum(passengers),
            flights = sum(flights)) %>%
  select(year, month, passengers, flights) %>%
  arrange(year)
```

```
## # A tibble: 54 x 4
## # Groups:   year [5]
##     year month passengers flights
##    <dbl> <ord>      <dbl>   <dbl>
##  1  2016 1         382618    2200
##  2  2016 2         297269    2008
##  3  2016 3         371282    2146
##  4  2016 4         397915    2211
##  5  2016 5         419454    2350
##  6  2016 6         471971    2394
##  7  2016 7         535122    2514
##  8  2016 8         509441    2396
##  9  2016 9         416257    2103
## 10  2016 10        387634    2033
## # ... with 44 more rows
```

```
total_num_by_year <- final_data %>%
  group_by(year) %>%
  summarise(passengers = sum(passengers),
            flights = sum(flights)) %>%
  select(year, passengers, flights)
total_num_by_year
```

```
## # A tibble: 5 x 3
##    year passengers flights
##   <dbl>      <dbl>   <dbl>
## 1  2016    4873212   26519
## 2  2017    5189091   28040
## 3  2018    5601907   29529
## 4  2019    5684548   29059
## 5  2020    1056594    6621
```

2019 has the most number of passengers and 2018 has the most number of flights.

4

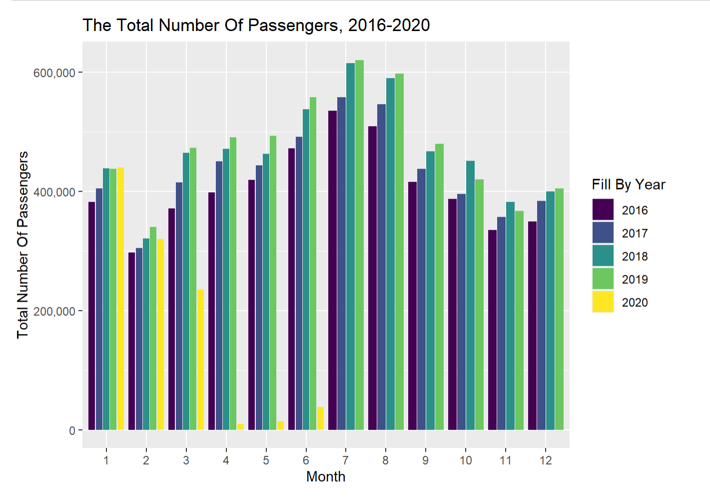
> Display the total number of passengers by month and year with a bar chart where month is the primary variable on the x-axis and there is a separate bar (not stacked, and filled with a different color) for each year. Add meaningful axis and legend labels and a title to this graph. (See the layer in the section below which uses the `scale_fill_discrete()` function to control the legend title. In addition, `guides()` can offer even finer control over legend characteristics.) Change the scale on the y axis so that values are printed as numbers with commas and not using scientific notation. (See the help for the **ggplot2** function `scale_y_continuous()` and the **scales** function `label_comma()`.) Describe any patterns or interesting trends that you see.

Solution

```
data <- final_data %>%

  group_by(year, month) %>%
  summarize(passengers = sum(passengers)) %>%
  select(year, month, passengers)

ggplot(data, aes(x=month, y =passengers, fill = year)) +
  geom_col(position = "dodge2") +
  xlab("Month") +
  ylab("Total Number Of Passengers") +
  ggtitle("The Total Number Of Passengers, 2016-2020") +
  guides(fill = guide_legend(title = "Fill By Year")) +
  scale_y_continuous(labels = comma)
```



In summer, especially July and August, there are the most number of passengers and I think this is because of the summer vacation and the influx of the international students.

5

> Add a `weekend` column to the combined data set which is TRUE for Saturdays and Sundays and FALSE for other days. Make a scatter plot with the average time for US passengers on the x-axis and the average time for non-US passengers on the y-axis. Use different colors for weekend and weekdays. Add a line to the plot that passes through the origin with a slope of one (explore `geom_abline()` to do this). Add straight regression lines to the plot, separately for weekends and weekdays (`geom_smooth()` with `method = "lm"`). Plot the data from different years in different facets. Change the color legend so that TRUE displays as "Weekend" and FALSE displays as "Weekday". (Use `scale_color_discrete()` and experiment with the `name` and `labels` arguments.)

> Are there any noteworthy patterns to the data? What are the primary differences between domestic and international flyers and are the patterns different on weekdays versus weekends?

```
weekend_data <- final_data %>%
  mutate(weekend = ifelse(wday == "Sat" | wday == "Sun", "TRUE", "FALSE")) %>%
  select(year, wday, weekend, us_avg_wait, non_us_avg_wait)

ggplot(weekend_data, aes(x = us_avg_wait, y = non_us_avg_wait, color = weekend)) +
  geom_point() +
  geom_smooth(method = "lm") +
  geom_abline(intercept = 0, slope = 1) +
  facet_wrap(~year) +
  scale_color_discrete(name = "Weekend & Weekdays", labels = c("TRUE" = "Weekend","FALSE" = "Weekday"))
```

The average waiting time for foreign passengers is much greater than the average waiting time for Americans regardless of weekday. In the year of 2016, 2017, 2018 and 2019, the average waiting time on weekend is less than it is on weekday. However, in 2020, there is little difference between weekday and weekend.

6

> Calculate separately for each year, the fraction of cases (a case is a single hour on a single date) for which the average time to get through passport control is greater for non US passport holders than it is for passport holders. Comment on how these values relate to the graphs in the previous problem.

```
non_us <-final_data %>%

  group_by(year) %>%
  rename(us = us_avg_wait, non_us = non_us_avg_wait) %>%
  mutate(fraction = us/non_us) %>%

  filter(fraction < 1) %>%


  summarise(non_us_case = n()) %>%
  select(year, non_us_case)

us <- final_data %>%
  group_by(year) %>%
  rename(us = us_avg_wait, non_us = non_us_avg_wait) %>%
  mutate(fraction = non_us/us) %>%
  filter(fraction <1) %>%
  summarise(us_case = n()) %>%
  select(year, us_case)

merge(non_us, us)
```

```
##   year non_us_case us_case
## 1 2016        6075     314
## 2 2017        5959     440
## 3 2018        6438     288
## 4 2019        6493     198
## 5 2020        1886     140
```

The above table represents the number of cases for which the average time to get through passport control is greater for non US passport holders than it is for passport holders, and this trend applies to all of the years from 2016 to 2020. This table has similarity with the above graphs in terms of the pattern that average time to get through passport control is greater for non US passport holders than it is for passport holders. ### 7

> Add a column named `booth_rate` to the data set which estimates the average number of passengers per booth per hour. For example, if 1000 passengers arrive between 05:00 and 06:00, the average wait time is 40 minutes, and there are 10 booths open, then an estimate of the total number of passengers per booth per hour could be computed like this: 1000/10 = 100 passengers per booth; (40 minutes per passenger * 1 hour per 60 minutes) = 2/3 hours per passenger; booth rate = 100 / (2/3) = (1000 * 60) / (10 * 40) = 150 passengers per booth per hour. This is an estimate because it assumes available booths change on the hour and it ignores how rates change when passenger wait durations stretch into the next time period. Add another column called `time_of_day` which takes the value "overnight" from 1am to 5am, "early morning" from 5am to 8am, "morning" from 8am to noon, "afternoon" from noon to 5pm, and "early evening" from 5pm to 8pm, and "late evening" from 8pm to 1am. Use `reorder()` to put the `time_of_day` variable in this order.

> After calculating this statistic, filter out cases where there are fewer than 200 total passengers, the average wait time is zero, or the booth rate is over 500. Make side_by_side boxplots of the booth rate versus the day of the week using different colors for each day of the week, different facets for each time of day, and fill color white if it is a weekday and gray if it is on the weekend. **Hints:** Use `case_when()` to set values of the `time_of_day`. Use `scale_fill_manual()` to set the fill values to white or gray.

> Which time of day has the lowest booth rate? Do booth rates tend to be higher on the weekend or on weekdays during each time of day? Is this effect large or small relative to variation in the booth rate within a day of week and time of day?

```
q7_data <- final_data %>%
  mutate(booth_rate = (passengers/booths)/(all_avg_wait/60)) %>%
  mutate(time_of_day = case_when(hour == "0100 - 0200" ~ 'overnight',
                                 hour == "0200 - 0300" ~ 'overnight',
                                 hour == "0300 - 0400" ~ 'overnight',
                                 hour == "0400 - 0500" ~ 'overnight',
                                 hour == "0500 - 0600" ~ 'early morning',
                                 hour == "0600 - 0700" ~ 'early morning',
                                 hour == "0700 - 0800" ~ 'early morning',
                                 hour == "0800 - 0900" ~ 'morning',
                                 hour == "0900 - 1000" ~ 'morning',
                                 hour == "1000 - 1100" ~ 'morning',
                                 hour == "1100 - 1200" ~ 'morning',
                                 hour == "1200 - 1300" ~ 'afternoon',
                                 hour == "1300 - 1400" ~ "afternoon",
                                 hour == "1400 - 1500" ~ "afternoon",
                                 hour == "1500 - 1600" ~ "afternoon",
                                 hour == "1600 - 1700" ~ "afternoon",
                                 hour == "1700 - 1800" ~ "early evening",
                                 hour == "1800 - 1900" ~ "early evening",
                                 hour == "1900 - 2000" ~ "early evening",
                                 hour == "2000 - 2100" ~ "late evening",
                                 hour == "2100 - 2200" ~ "late evening",
                                 hour == "2200 - 2300" ~ "late evening",
                                 hour == "2300 - 0000" ~ "late evening",
                                 hour == "0000 - 0100" ~ "late evening")) %>%
  mutate(time_of_day = as.factor(time_of_day)) %>%
  mutate(index = case_when(time_of_day == "overnight" ~1,
                           time_of_day == "early morning" ~2,
                           time_of_day == "morning" ~3,
                           time_of_day == "afternoon" ~4,
                           time_of_day == "early evening" ~5,
                           time_of_day == "late evening" ~6)) %>%
  mutate(time_of_day = reorder(time_of_day, index)) %>%
  mutate(weekend = ifelse(wday == "Sat" | wday == "Sun", "Weekend", "Weekday")) %>%
  select(time_of_day, everything()) %>%
  filter(passengers >= 200 | all_avg_wait !=0 | booth_rate <= 500)

ggplot(q7_data, aes(x = wday, y = booth_rate, colour = wday)) +
  geom_boxplot(aes(fill = weekend), position = "dodge") +
  facet_wrap(~time_of_day) +
  scale_fill_manual(values = c("white", "gray")) +
  scale_y_log10()
```

```
## Warning: Removed 11 rows containing non-finite values (stat_boxplot).
```

morning has the lowest booth rate and booth rates tend to be higher on the weekend during each time of day. This effect is small relative to variation in the booth rate within a day of week and time of day.