BASA – Data Exploration and Visualization

The Borealian Air Security Agency (BASA) has provided you with two cleaned and processed datasets, detailing the passenger scans at the pre-board screening (PBS) at the gate and some flight-level information for departures from the Borealian airfield of Auckland (AUC) for the months of September, October, November and December 20X6.

In Phase I of this project you have cleaned and processed the original dataset. In Phase III, you will be predicting the wait times at this PBS gate. In Phase IV, you will be clustering the flights leaving AUC during the period of interest.

In Phase II, you will gather a preliminary understanding of the datasets and the system it represents *via* simple exploration and visualization. No sophisticated analysis is expected at this stage; the focus is on your ability to provide answers to the questions using summarization and visualization methods. In many instances, the answers will be approximate and qualitative, to some extent.

The (clean) **passenger-level** dataset is found in `dat_P_sub_c.csv`; its data dictionary is the same as the original set, except for the following items:

- `WT_flag`: 1 (missing value), 0 (not missing)
- `S2_Sch_Flag`: 1 (S2>Scheduled Departure Time), 0 (not the case)
- `S2_Act_Flag`: 1 (S2>Actual Departure Time), 0 (not the case)
- `Sch_Act_Flag`: 1 (Scheduled Departure Time>Actual Departure Time), 0 (not the case)
- `Flight_ID`: flight identifier (integer)
- `Delay_in_Seconds`: (Actual Departure - Scheduled Departure) in seconds

The (clean) **flight-level** dataset is found in `dat_F_sub.csv`; its data dictionary is as follows:

- `Airfield`: Originating airfield
- `Flight_ID`: Unique identifier for flights
- `Sch_Departure`: Scheduled departure time
- `Act_Departure`: Actual departure time
- `Time_of_Day`: Night, Morning, Afternoon, Evening, for Act_Departure
- `Period_of_Week`: Weekday and Weekend, for Actual Departure
- `Day_of_Week`: Monday, ... , Sunday, for Actual Departure
- `Month`: January, ... , December, for Actual Departure
- `Season`: Winter, Spring, Summer, Autumn, for Actual Departure
- `Year`: Year for Actual Departure
- `tot_pass`: Total number of passengers on a flight boarding at AUC
- `N`: Number of passengers with Wait Time information
- `min`: min(Wait Time) for a flight
- `mean`: average(Wait Time) for a flight
- `median`: median(Wait Time) for a flight
- `max`: max(Wait Time) for a flight
- `mean_WTL`: proportion of passengers with Wait Time > 4 hours
- `mean_City_Flag`: proportion of passengers with BFO Dest City information
- `mode_BFO_Dest_City`: mode(BFO Dest City) for a flight
- `sum_city_mode`: number of passengers with final destination mode city
- `N_of_Dest_City`: number of distinct BFO Dest Cities for a flight
- `mode_BFO_Dest_Country_Code`: mode(BFO Dest Country Code) for a flight
- `sum_city_mode`: number of passengers boarding at AUC with final destination mode country
- `N_of_Dest_Country`: number of distinct BFO Dest Country Codes for a flight
- `Delay_in_Seconds`: (Act Departure - Sch Departure) in seconds

Note: `N_of_Dest_City` and `N_of_Dest_Country` includes the `mode_BFO_Dest_City/Country`.

`1.` Start by loading the datasets and make any modification required to the field formats. Provide a summary of the dataset and its structure, and various univariate plots.

PASSENGER-LEVEL DATASET

2. (**demo**) Explore the change in traffic volume throughout the day. Can you infer when the airfield is in operation? When do passengers arrive at the airfield? Are these patterns similar for weekdays and weekends? Are they distinct on a month-to-month basis?

Steps:

i. To study the change in traffic volume throughout the day, count the number of S2 scans per hour, and visualize them using a histogram.

ii. To study monthly trend, further break up the hourly S2 scans by month and produce four additional histograms.

iii. Follow similar steps for the weekday/weekend patterns.

iv. The above graphs can be used to infer the operation hours of AUC, as well as (typical) arrival time.

Hint: the following R code shows you one way to do this.

```
#install.packages("ggplot2")
library(ggplot2)

# Read in the dataset (at passenger level)
dat.P.Sub=read.csv("dat_P_sub_c.csv")
attach(dat.P.Sub)


#-------------------------------------------------------------------------------#

# Description: Create new variables that breakdown S2 into Year, Month, Day, Hour,
# and  Minute
# Aim: Create variables to help better categorize observations

test=as.POSIXlt(S2)
S2.Year=test$year+1900
S2.Month=test$mon+1
S2.Day=test$mday
S2.Hour=test$hour
S2.Min=test$min
S2.dat=cbind(S2.Year, S2.Month, S2.Day, S2.Hour, S2.Min)
S2.dat=data.frame(S2.dat)
S2.dat.SEP=S2.dat[S2.Month==9,]
S2.dat.OCT=S2.dat[S2.Month==10,]
S2.dat.NOV=S2.dat[S2.Month==11,]
S2.dat.DEC=S2.dat[S2.Month==12,]
S2.dat.WKD=S2.dat[which(Period_of_Week=="1 - WEEKDAY"),]
S2.dat.WKE=S2.dat[-which(Period_of_Week=="1 - WEEKDAY"),]

#-------------------------------------------------------------------------------#

# Description: Visualize Y=Number of S2 count vs. X=Hour using histogram
# Aim: Produce Histogram for traffic volume per hour
# NOTE: At the end of this section, table() is used to tabulate the histogram

# Setting graphic parameters
plot.main.S2="S2 count by hour - Overall"
plot.main.S2.SEP="S2 count by hour - September"
plot.main.S2.OCT="S2 count by hour - October"
plot.main.S2.NOV="S2 count by hour - November"
plot.main.S2.DEC="S2 count by hour - December"
plot.main.S2.WKD="S2 count by hour - Weekday"
```

```
plot.main.S2.WKE="S2 count by hour - Weekend"
xlabel="Hour"
ylabel="count"
ylimit=c(0, 1.6e4) # For overall S2 scan per hour
ylimit2=c(0, 4.5e3) # Per month S2 scan per hour
Height=480
Width=2*Height

# Histograms of S2, spanning four months
png("Histogram_of_S2_(Overall).png", width=Width, height=Height)
ggplot(data=S2.dat, aes(S2.dat$S2.Hour)) +
  geom_histogram(breaks=seq(0.5,23.5, by=1),
                 col="grey",
                 aes(fill=..count..)) +
                 ylim(ylimit) +
                 labs(title=plot.main.S2) +
                 labs(x=xlabel, y=ylabel)
dev.off()

#---------------------------#

# Histograms of S2 (September)
png("Histogram_of_S2_in_9.png", width=Width, height=Height)
ggplot(data=S2.dat.SEP, aes(S2.dat.SEP$S2.Hour)) +
  geom_histogram(breaks=seq(0.5,23.5, by=1),
                 col="grey",
                 aes(fill=..count..)) +
                 ylim(ylimit2) +
                 labs(title=plot.main.S2.SEP) +
                 labs(x=xlabel, y=ylabel)
dev.off()

#---------------------------#

# Histograms of S2 (October)
png("Histogram_of_S2_in_10.png", width=Width, height=Height)
ggplot(data=S2.dat.OCT, aes(S2.dat.OCT$S2.Hour)) +
  geom_histogram(breaks=seq(0.5,23.5, by=1),
                 col="grey",
                 aes(fill=..count..)) +
                 ylim(ylimit2) +
                 labs(title=plot.main.S2.OCT) +
                 labs(x=xlabel, y=ylabel)
dev.off()

#---------------------------#

# Histograms of S2 (November)
png("Histogram_of_S2_in_11.png", width=Width, height=Height)
ggplot(data=S2.dat.NOV, aes(S2.dat.NOV$S2.Hour)) +
  geom_histogram(breaks=seq(0.5,23.5, by=1),
                 col="grey",
                 aes(fill=..count..)) +
                 ylim(ylimit2) +
                 labs(title=plot.main.S2.NOV) +
                 labs(x=xlabel, y=ylabel)
dev.off()

#---------------------------#
```

```
# Histograms of S2 (December)
png("Histogram_of_S2_in_12.png", width=Width, height=Height)
ggplot(data=S2.dat.DEC, aes(S2.dat.DEC$S2.Hour)) +
  geom_histogram(breaks=seq(0.5,23.5, by=1),
                 col="grey",
                 aes(fill=..count..)) +
                 ylim(ylimit2) +
                 labs(title=plot.main.S2.DEC) +
                 labs(x=xlabel, y=ylabel)
dev.off()

#---------------------------#

# Histograms of S2 (Weekdays)
png("Histogram_of_S2_in_weekdays.png", width=Width, height=Height)
ggplot(data=S2.dat.WKD, aes(S2.dat.WKD$S2.Hour)) +
  geom_histogram(breaks=seq(0.5,23.5, by=1),
                 col="grey",
                 aes(fill=..count..)) +
                 labs(title=plot.main.S2.WKD) +
                 labs(x=xlabel, y=ylabel)
dev.off()

#---------------------------#

# Histograms of S2 (Weekends)
png("Histogram_of_S2_in_weekends.png", width=Width, height=Height)
ggplot(data=S2.dat.WKE, aes(S2.dat.WKE$S2.Hour)) +
  geom_histogram(breaks=seq(0.5,23.5, by=1),
                 col="grey",
                 aes(fill=..count..)) +
                 labs(title=plot.main.S2.WKE) +
                 labs(x=xlabel, y=ylabel)
dev.off()

#---------------------------#

# Also, we can summarize these information using table()
table.S2.All=table(c(S2.dat$S2.Hour, 6:22))-1
table.S2.SEP=table(c(S2.dat.SEP$S2.Hour, 6:22))-1
table.S2.OCT=table(c(S2.dat.OCT$S2.Hour, 6:22))-1
table.S2.NOV=table(c(S2.dat.NOV$S2.Hour, 6:22))-1
table.S2.DEC=table(c(S2.dat.DEC$S2.Hour, 6:22))-1

table.S2.matrix=cbind(table.S2.All,table.S2.SEP,table.S2.OCT,table.S2.NOV,table.S2.DEC)
colnames(table.S2.matrix)=c("Total", "SEP", "OCT", "NOV", "DEC")
table.S2.matrix # Tabulated version of above histograms

#-------------------------------------------------------------------------------#
```

---

3. (**step-by-step**) When do passengers get scanned at S1? Is there a relationship between the frequency of missed S1 scans and the time of the day and/or the traffic level?

Before proceeding to the analysis, it is important to consider "why people get scanned at S1?" S1 scans are required if we want to study queueing information. Intuitively, we expect passengers to potentially not be scanned when the queue is empty (in direct violation of regulations). At the same time, it might be reasonable to assume that only certain passengers are scanned when the traffic volume is very high (in order to avoid the scanning procedure at S1 to directly or indirectly affect the queue wait time). To test the above conjecture (i.e., to test whether the traffic volume affects the frequency of S1 scans), you will provide a preliminary analysis using a scatterplot.

Steps:

i. What should the x y variables represent? (x should denote the traffic volume and y can be either the frequency or proportion of S1 scans per observed x).

ii. Obtain x and y (x can be obtained using question 2, and y can be calculated using `WT_flag`).

iii. Use `scatterplot()` to visualize and interpret the result. Do you think the traffic volume affects the frequency of S1 scans?

---

4. (**open-ended**) Does the airfield provide a sufficient number of servers to ensure a "reasonable" wait time experience?

---

5. (**conceptual**) Do passengers arrive at the airfield well before the scheduled departure time? Did anyone miss their flight? Is there any pattern there?

FLIGHT-LEVEL DATASET

6. (**step-by-step**) What can you say about the size (# of passengers getting on at AUC) of each flight? What are typical flight sizes? Are there particular time of day or days where larger flights leave?

Steps:

i. Visualize the number of passengers in each flight, using `hist()`, `boxplot()`, and `summary()` (you do not need to include all three in your report; however, you should justify which is/are most effective).

ii. Use the above information to talk about the "typical" size of a flight.

iii. Define what is considered a "larger" flight.

iv. Create a series of boxplots of the number of passengers in each flight for different times of day (or hour, whichever is more suitable).

v. Create a series of boxplots of the number of passengers in each flight for Weekdays/Weekends (or day of the week, whichever is more suitable).

---

7. (**open-ended**) Are international flights larger (# of passengers getting on at AUC) compared to domestic flights?

---

8. (**conceptual**) How often do flights get delayed? Is there any pattern?