ASSIGNMENT 6: COLLABORATIVE PROJECT

ALY 6015_Intermediate Analytics

Spring 2019 CPS Quarter Term A

Instructor: Steward Huang

Yuyi Zhang

May 16, 2019

1 Introduction

In this report, we are going to go through an air traffic passenger counts dataset and apply regression and time series analysis to them. The dataset is on monthly basis from 2005 to the current year. We are also going to use a weather report as a dependent dataset in regression analysis.

2 Method

2.1 Descriptive and Regression

In this part, we are going to use hist() to create histogram, density() to create density plots. As well as boxplots, normal probability plots. Then conduct a regression analysis.

2.2 Time Series

This is for seasonal data analysis. We are going to decomposing the passenger counts dataset and select a candidate for ARIMA model.

3 Analysis

3.1 Part A: Descriptive and Regression

3.1.1 Data Preparation

In this phase, we are going load dataset “Air Traffic Passenger Statistics” and aggregate the passenger count column by monthly activity. After aggregation, filter passenger count to a new value called “ps” for further use.

```
Console  Terminal ×  Jobs ×
~/
> # Loading packages and file
> library(readr)
> library(dplyr)
> Air_Traffic_Passenger_Statistics <- read_csv("FCR/NEU/CPS/Analytics_2018/ALY 6015_Intermediate Analytics/Week 6_Collaborative Pro
ject/Air_Traffic_Passenger_Statistics.csv", head(TRUE))
Parsed with column specification:
cols(
  `Activity Period` = col_double(),
  `Operating Airline` = col_character(),
  `Operating Airline IATA Code` = col_character(),
  `Published Airline` = col_character(),
  `Published Airline IATA Code` = col_character(),
  `GEO Summary` = col_character(),
  `GEO Region` = col_character(),
  `Activity Type Code` = col_character(),
  `Price Category Code` = col_character(),
  Terminal = col_character(),
  `Boarding Area` = col_character(),
  `Passenger Count` = col_double()
)
>
>
> # Part A: Descriptive and Regression
> # Aggregate Data
> Passenger_Statistics <- Air_Traffic_Passenger_Statistics %>%
+   group_by(`Activity Period`) %>%
+   summarise(`Passenger Count` = sum(`Passenger Count`))
> summary(Passenger_Statistics)
 Activity Period  Passenger Count
 Min.   :200507   Min.   :2223024
 1st Qu.:200812   1st Qu.:3105958
 Median :201205   Median :3593364
 Mean   :201194   Mean   :3676154
 3rd Qu.:201510   3rd Qu.:4190367
 Max.   :201903   Max.   :5692572
> ps <- Passenger_Statistics$`Passenger Count`
> 
```

Figure 1. Data preparation

3.1.2 Create Histogram

Use hist() to create a histogram. Made few adjustments for clear chart.

```
Console  Terminal ×  Jobs ×
~/
> # Create Histogram
> hist(ps, breaks = 15, ylim = c(0,20), col = "#ffffe6", main = "Passenger Count by Month 2005-2019", xlab = "Passenger Counts")
> 
```

Figure 2. Create Histogram

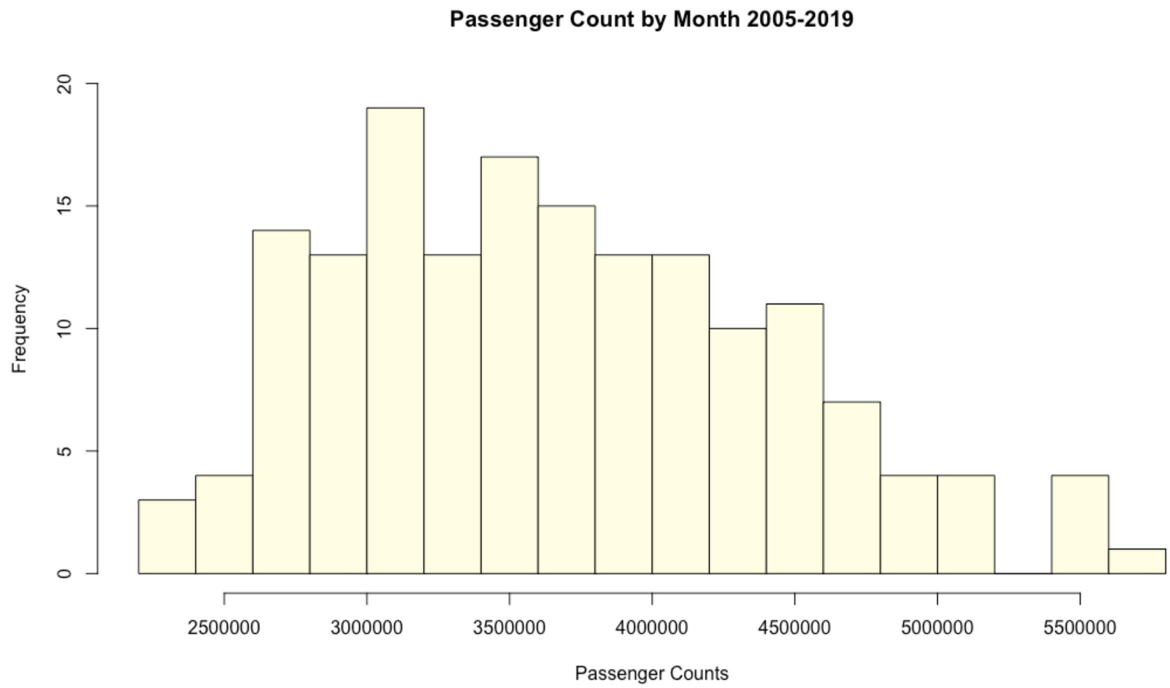
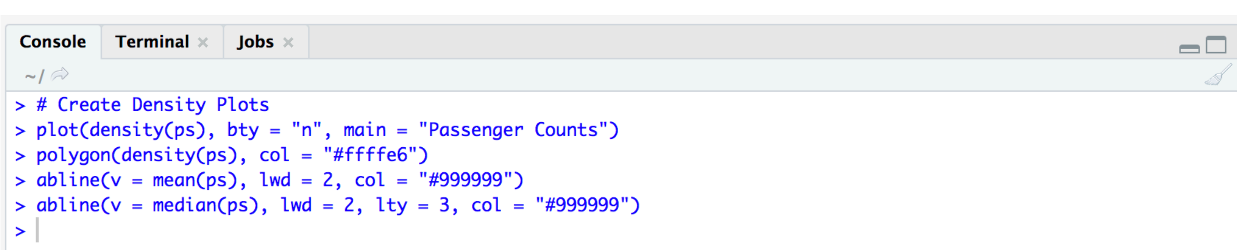


Figure 3. Histogram

This distribution is unimodal with only one main cluster. There are 20 samples on horizontal axis, while middle 10 bars are higher than the rest. The highest relative frequency is 19. This chart is skewed to the right with only 2 outliers in it. In another word, the normal passenger load would be 3 million to 4.25 million per month. Occasionally, the passenger counts would be less than 2.5 million are higher than 5.25 million.

3.1.3 Create Density Plots

Use density() to create Kernel Density Plots.




```
> # Create Density Plots
> plot(density(ps), bty = "n", main = "Passenger Counts")
> polygon(density(ps), col = "#ffffe6")
> abline(v = mean(ps), lwd = 2, col = "#999999")
> abline(v = median(ps), lwd = 2, lty = 3, col = "#999999")
>
```

Figure 4. Create Density Plots

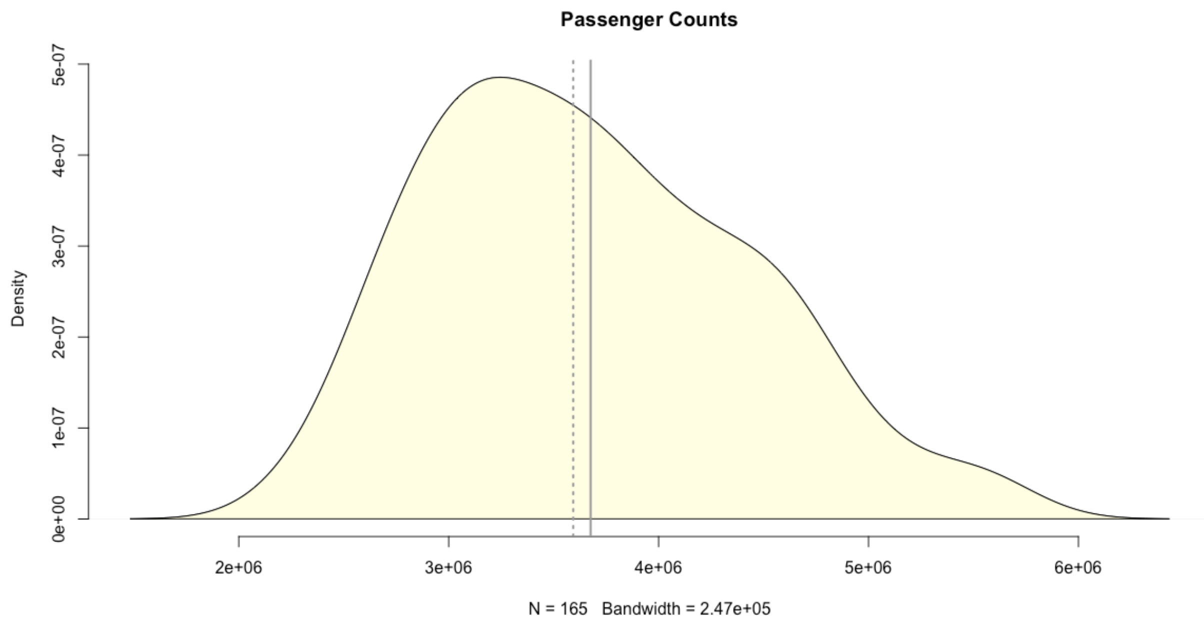


Figure 5. Density Plots

As we can see that density plots are much smoother than histogram. In these plots we also added mean bar (gray bar) and medium bar (dash dots) to get a better idea of the basic distribution. Because the mean bar is to the right of the median, so this distribution has a longer right-hand tail.

3.1.4 Create Box Plot

Use boxplot() to create a box plot in R. Then use rug() to make enhancement.

```
Console   Terminal ×   Jobs ×
~/
> # Boxplots
> boxplot(ps, col = "#ffffe6", main = "Passenger Count by Month 2005-2019", xlab = "Passenger Counts", frame.plot =
  TRUE, boxwex = 0.35, horizontal = TRUE)
> rug(ps, side = 1)
>
```

Figure 6. Code for Box Plots

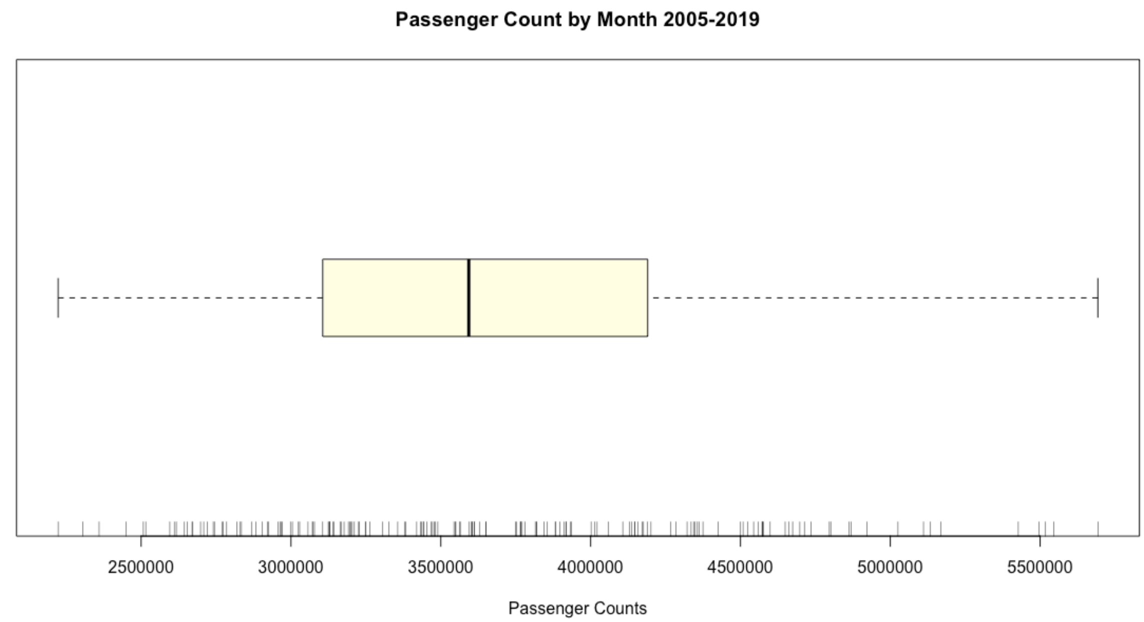
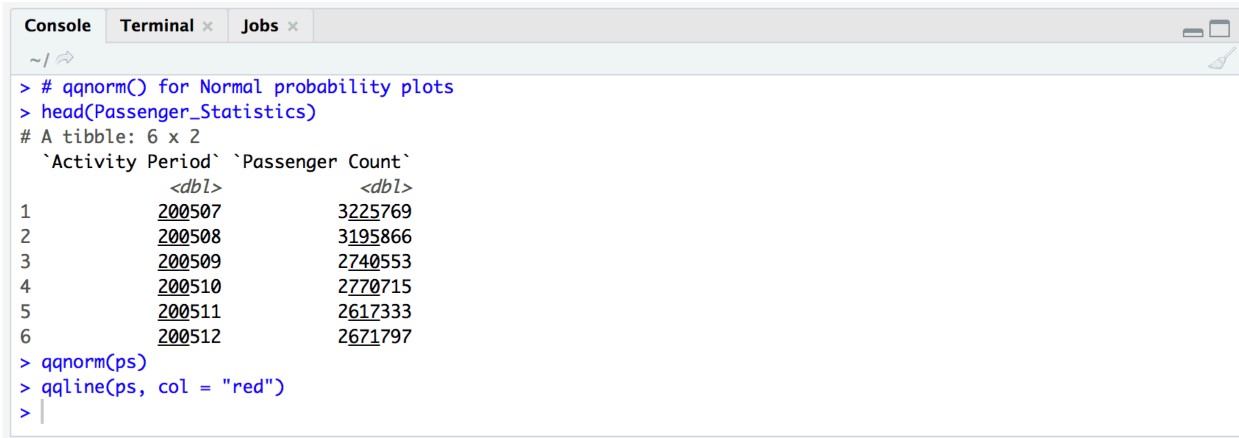


Figure 7. Box Plots

In the box plot we could get a better idea of where the median is. In the box we could see that interquartile range is about 3,200,000 to 4,250,000. As the idea that we've got from the first plots, the histogram plots that, this is a skewed to the right. Better than that, we could see a roughly data allocation throughout the rug in below. But if we want to see the frequency of each of each range, histogram would be the best chart that we are looking for.

3.1.5 Create Normal Probability Plots

Use qqnorm() to create normal probability plots. Quantile-Quantile plots are used to determine if data can be approximated by a statistical distribution. In this case, we want to know if the ps(passenger counts of passenger statistics) was normally distributed.



```
Console  Terminal ×  Jobs ×
~/
> # qqnorm() for Normal probability plots
> head(Passenger_Statistics)
# A tibble: 6 x 2
  `Activity Period` `Passenger Count`
              <dbl>             <dbl>
1            200507           3225769
2            200508           3195866
3            200509           2740553
4            200510           2770715
5            200511           2617333
6            200512           2671797
> qqnorm(ps)
> qqline(ps, col = "red")
>
```

Figure 8. Code for Normal Probability Plots (QQ plot)

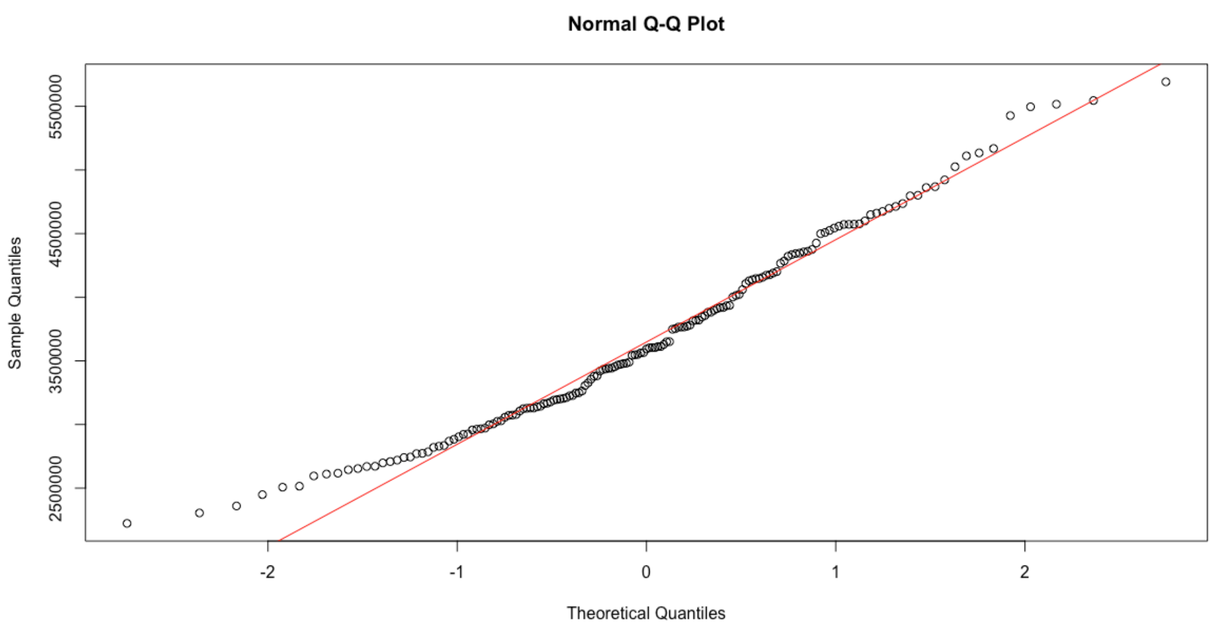


Figure 9. Normal Probability Plots

As we can see from the qq plots, the ps quantiles vs. standard normal distributed quantiles are not perfectly fit in a straight line. Therefore, ps is not very much normally distributed. But in from the 1 standard distribution part, it looks fit better than the ones besides.

3.1.6 Linear Regression

In this part, we are going to generate a linear regression test. First step, we are going to pull out the dependent data that we are going to use for the test. The first dataset would be monthly average temperature in San Francesco. Read the csv file and use the summary() to get a general idea of the data shown as below.

```
Console  Terminal ×  Jobs ×
~/
> # Regression Equation
> San_Francisco_Monthly_Average_Temperature <- read_csv("FCR/NEU/CPS/Analytics_2018/ALY 6015_Intermediate Analytics/Week 6_Collabora
tive Project/San Francisco Monthly Average Temperature.csv")
Parsed with column specification:
cols(
  Year = col_double(),
  Month = col_character(),
  Temperature = col_double()
)
> summary(San_Francisco_Monthly_Average_Temperature)
      Year          Month            Temperature
 Min.   :2005   Length:165         Min.   :55
 1st Qu.:2008   Class :character   1st Qu.:59
 Median :2012   Mode  :character   Median :63
 Mean   :2012                      Mean   :64
 3rd Qu.:2015                      3rd Qu.:70
 Max.   :2019                      Max.   :73
>
```

Figure 10. Loading file

The second dataset that we are going to use is actually in the original file that we've been using the whole time: the united passenger counts for each month. So we used filter() to aggregate what we want and use summary() to get a general idea. Code is shown as below.

```
Console  Terminal ×  Jobs ×
~/
> United_only <- Air_Traffic_Passenger_Statistics %>%
+   filter(`Operating Airline` == "United Airlines" | `Operating Airline` == "United Airlines - Pre 07/01/2013" ) %>%
+   group_by(`Activity Period`) %>%
+   summarise(`Passenger Count` = sum(`Passenger Count`))
> summary(United_only)
 Activity Period  Passenger Count
 Min.   :200507   Min.   : 915793
 1st Qu.:200812   1st Qu.:1225906
 Median :201205   Median :1359374
 Mean   :201194   Mean   :1421260
 3rd Qu.:201510   3rd Qu.:1589155
 Max.   :201903   Max.   :2272873
>
```

Figure 11. Data Preparation

After all data have been prepared, we use lm() to generate linear regression. We set the total passenger counts as independent variable, the monthly average temperature in San Francisco and United Airline passenger counts as dependent variables. Then we use summary() to get the summary report of the regression shown as below.

```
Console  Terminal ×  Jobs ×
~/
> Regression <- lm(Passenger_Statistics$`Passenger Count` ~ San_Francisco_Monthly_Average_Temperature$Temperature + United_only$`Pas
senger Count`)
> summary(Regression)

Call:
lm(formula = Passenger_Statistics$`Passenger Count` ~ San_Francisco_Monthly_Average_Temperature$Temperature +
    United_only$`Passenger Count`)

Residuals:
    Min      1Q  Median      3Q     Max
-584434 -134487   58617  183175  368988

Coefficients:
                                                       Estimate Std. Error t value Pr(>|t|)
(Intercept)                                           1.873e+05  2.115e+05   0.886    0.377
San_Francisco_Monthly_Average_Temperature$Temperature -4.202e+03  3.609e+03  -1.165    0.246
United_only$`Passenger Count`                         2.644e+00  7.618e-02  34.708   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 245700 on 162 degrees of freedom
Multiple R-squared:  0.8973,	Adjusted R-squared:  0.896
F-statistic: 707.6 on 2 and 162 DF,  p-value: < 2.2e-16

>
```

Figure 12. Linear Regression

In this report, the residual section shows all the residual quantile information (the distance from the data to the fitted line). Ideally, they should be symmetrically distributed around the line. That is to say, ideally the minimum value and the maximum value would be the same distance from zero, as well as 1Q and 3Q. But in our data, the absolute value of min and max are quite different.

In the coefficients section, we got the least-squares estimates for the fitted line. In this case, we've got our equation as:

$$ps = 187{,}300 - 4{,}202 \times temperature + 2.644 \times United\ passenger\ count + residuals$$

The std. error and t value are provided to show how the p-value were calculated. If the value is equal to zero, it means that the variable doesn't have much use in the model. But in this case, we've got a significant t-value of United Airline passenger counts, 34.708. Moreover, the p-value is way much smaller than 0.05, which is statistically significant.

In the last section we see the r-squared value is 0.8973. This means that United airline passenger counts explain 89.73% of the variation in total passenger counts in San Francisco airport. The p-value of R-squared is 2.2e-16. This means, again, the United Airline passenger counts give out a reliable estimate for total passenger count in San Francisco airport.

3.2 Part B: Time Series

3.2.1 Data preparation

In part B, we are going to conduct time series analysis. First of all, convert ps data to time series format.

```
> # Part B: Time Series
> # Time Series Preparation
> pstimeseries <- ts(Passenger_Statistics$`Passenger Count`, frequency=12, start=c(2005,7))
> pstimeseries
         Jan     Feb     Mar     Apr     May     Jun     Jul     Aug     Sep     Oct     Nov     Dec
2005                                                 3225769 3195866 2740553 2770715 2617333 2671797
2006 2448889 2223024 2708778 2773293 2829000 3071396 3227605 3143839 2720100 2834959 2653887 2698200
2007 2507430 2304990 2820085 2869247 3056934 3263621 3382382 3436417 2957530 3129309 2922500 2903637
2008 2670053 2595676 3127387 3029021 3305954 3453751 3603946 3612297 3004720 3124451 2744485 2962937
2009 2644539 2359800 2925918 3024973 3177100 3419595 3649702 3650668 3191526 3249428 2971484 3074209
2010 2785466 2515361 3105958 3139059 3380355 3612886 3765824 3771842 3356365 3490100 3163659 3167124
2011 2883810 2610667 3129205 3200527 3547804 3766323 3935589 3917884 3564970 3602455 3326859 3441693
2012 3211600 2998119 3472440 3563007 3820570 4107195 4284443 4356216 3819379 3844987 3478890 3443039
2013 3204637 2966477 3593364 3604104 3933016 4146797 4176486 4347059 3781168 3910790 3466878 3814984
2014 3432625 3078405 3765504 3881893 4147096 4321833 4499221 4524918 3919072 4059443 3628786 3855835
2015 3550084 3248144 4001521 4021677 4361140 4558511 4801148 4796653 4201394 4374749 4013814 4129052
2016 3748529 3543639 4137679 4172512 4573996 4922125 5168724 5110638 4543759 4571997 4266481 4343369
2017 3897685 3481405 4335287 4425920 4698067 5134110 5496516 5516837 4736005 4868674 4572702 4660504
2018 4190367 3882181 4674035 4713183 5025595 5427144 5692572 5545859 4649100 4861782 4508606 4576449
2019 4156821 3752763 4599189
> plot.ts(pstimeseries)
> 
```

Figure 13. Code for Data Preparation

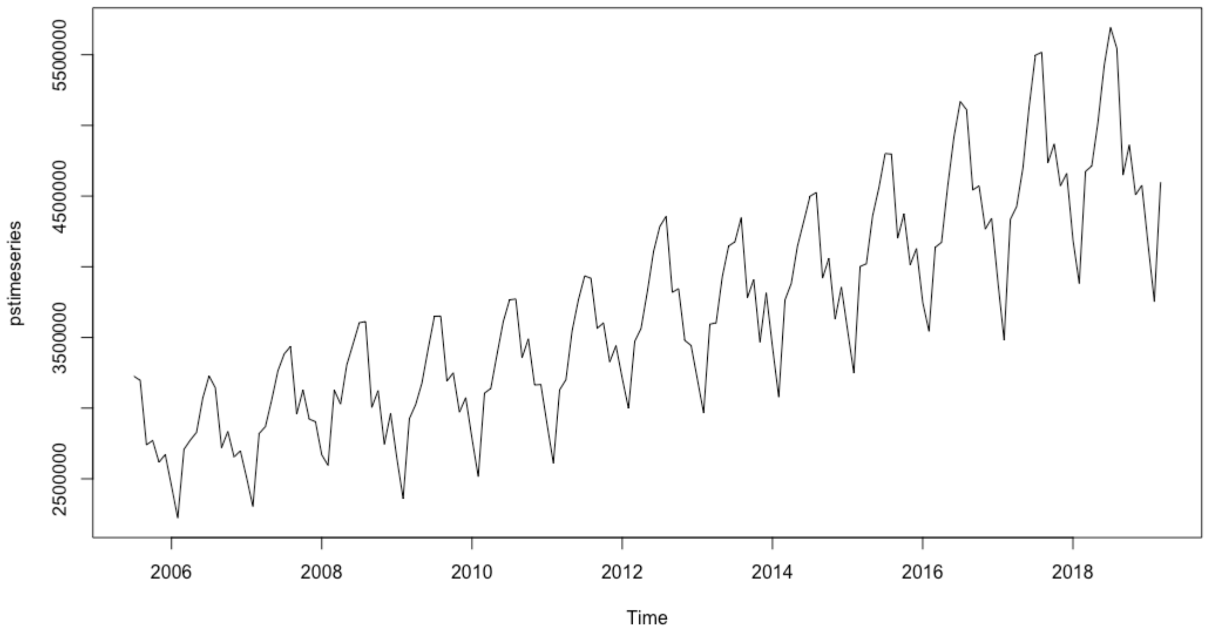


Figure 14. Original Data Plot

3.2.2 Decomposing

Use decompose() and then draw plots to get a visual illustration of data decomposition. This dataset is a seasonal time series consists of a trend component, a seasonal component and an irregular component. So in this process, we are going to separate the time series into three components: trend, seasonal and irregular.

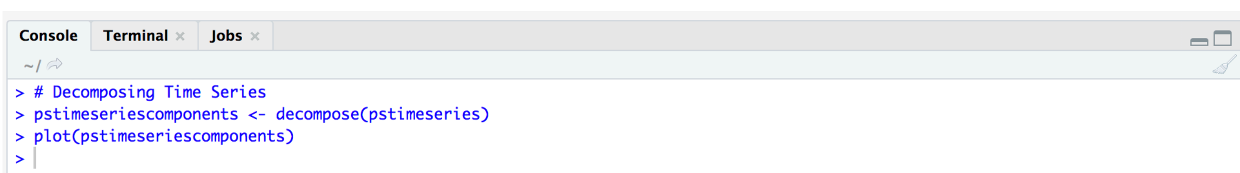


Figure 15. Code for Decomposing

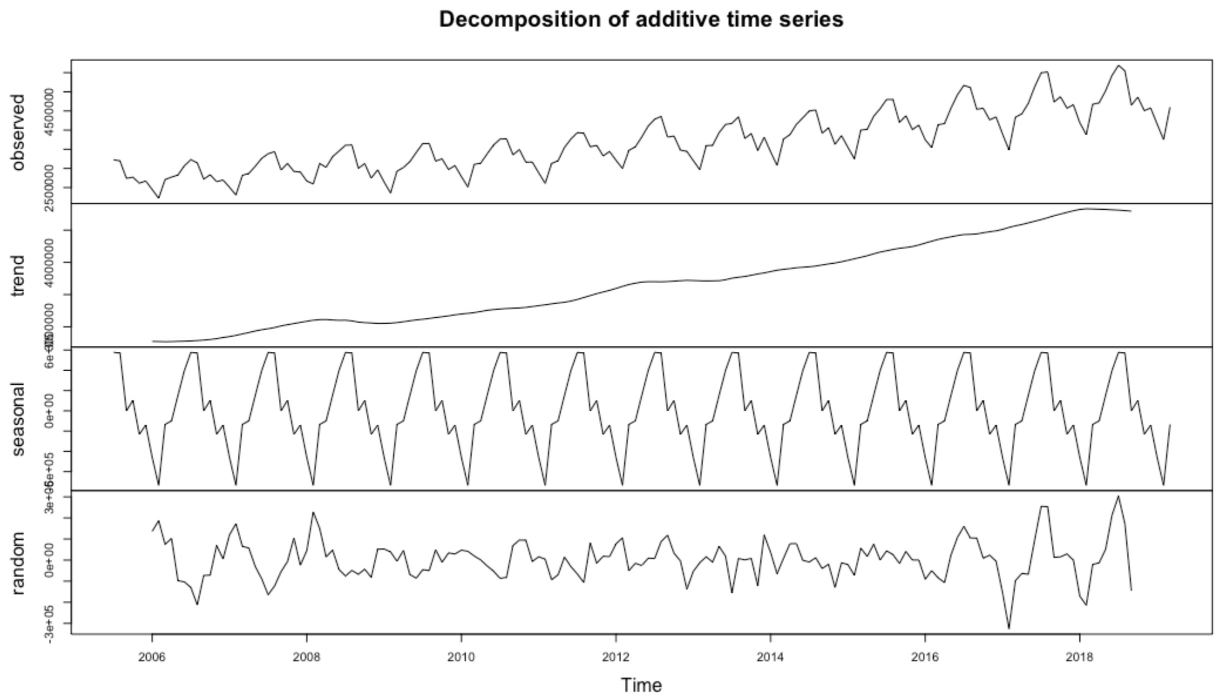


Figure 16. Decomposition Plots

This plot shows the original time series and three components we mentioned before. As expected, the trend shows overall increase. There is a highly consistent seasonal cycle as shown in the third plot from top. Irregular noise shown in the bottom.

3.2.3 Differencing

Containing of trends and seasonality define a time series data as being non-stationary. Stationary datasets are those that have a stable mean and variance and are turn much easier to be modeled. Differencing is a method to transform for making time series data stationary.

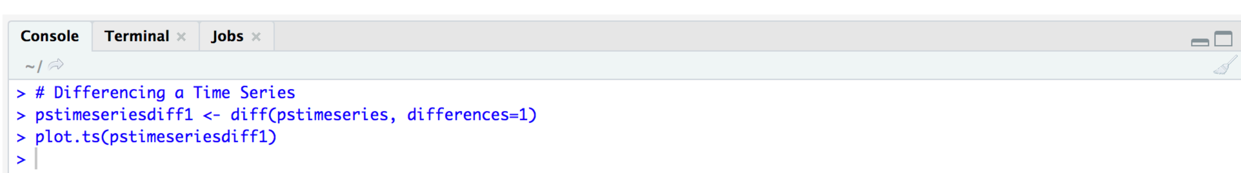


Figure 17. Code for differencing

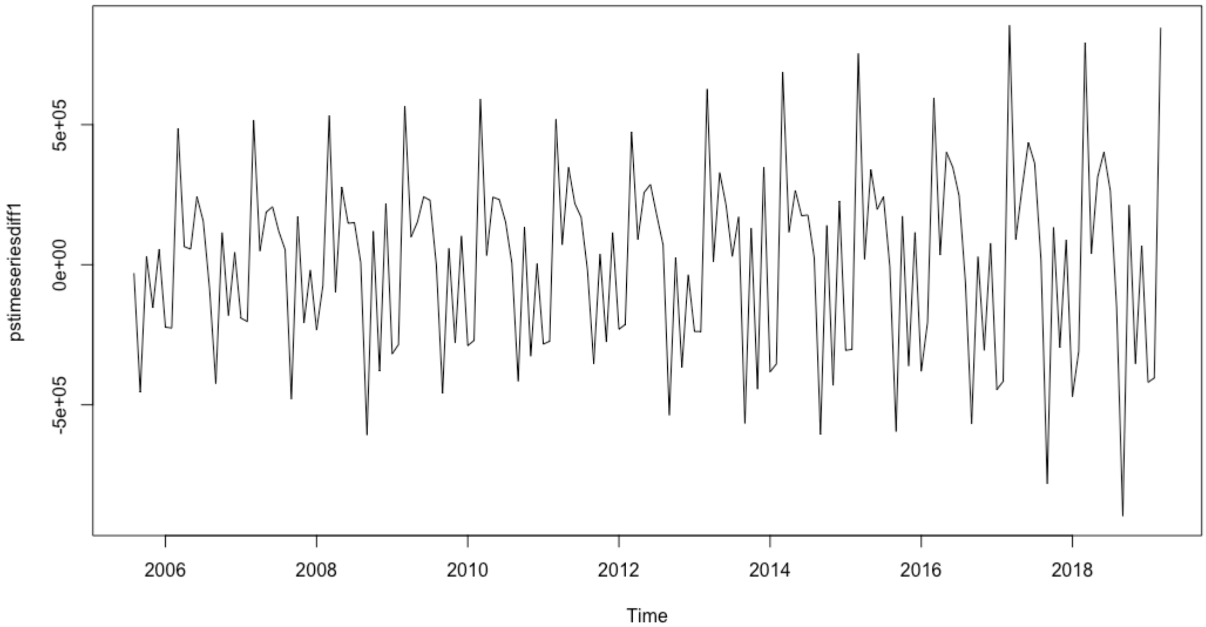


Figure 18. Differencing plot

In this case, the difference order would be 1, which means performing a lag-1 differencing operation that would transform to remove a trend.

3.2.4 Selecting a Candidate ARIMA Model manually

To select candidates for ARIMA model, we are going to use two ways: manually and automatically. We need to conduct residual analysis and find the appropriate values of p, d, q representing the AR order, the degrees of differencing and the MA order, respectively. After the previous step, we know that the d value is 1. To manually selective p and q values, we need to do ACF and PACF analysis.

```
Console   Terminal ×   Jobs ×
~/
> # Selecting a Candidate ARIMA Model
> acf(pstimeseriesdiff1, lag.max=20)
> acf(pstimeseriesdiff1, lag.max=20, plot = FALSE)

Autocorrelations of series 'pstimeseriesdiff1', by lag

0.0000 0.0833 0.1667 0.2500 0.3333 0.4167 0.5000 0.5833 0.6667 0.7500 0.8333 0.9167 1.0000 1.0833 1.1667 1.2500 1.3333
 1.000 -0.085  0.116  0.016 -0.189  0.040 -0.737  0.032 -0.196  0.019  0.126 -0.062  0.873 -0.077  0.120  0.012 -0.182
1.4167 1.5000 1.5833 1.6667
 0.031 -0.667  0.030 -0.185
>
```

Figure 19. Code for ACF

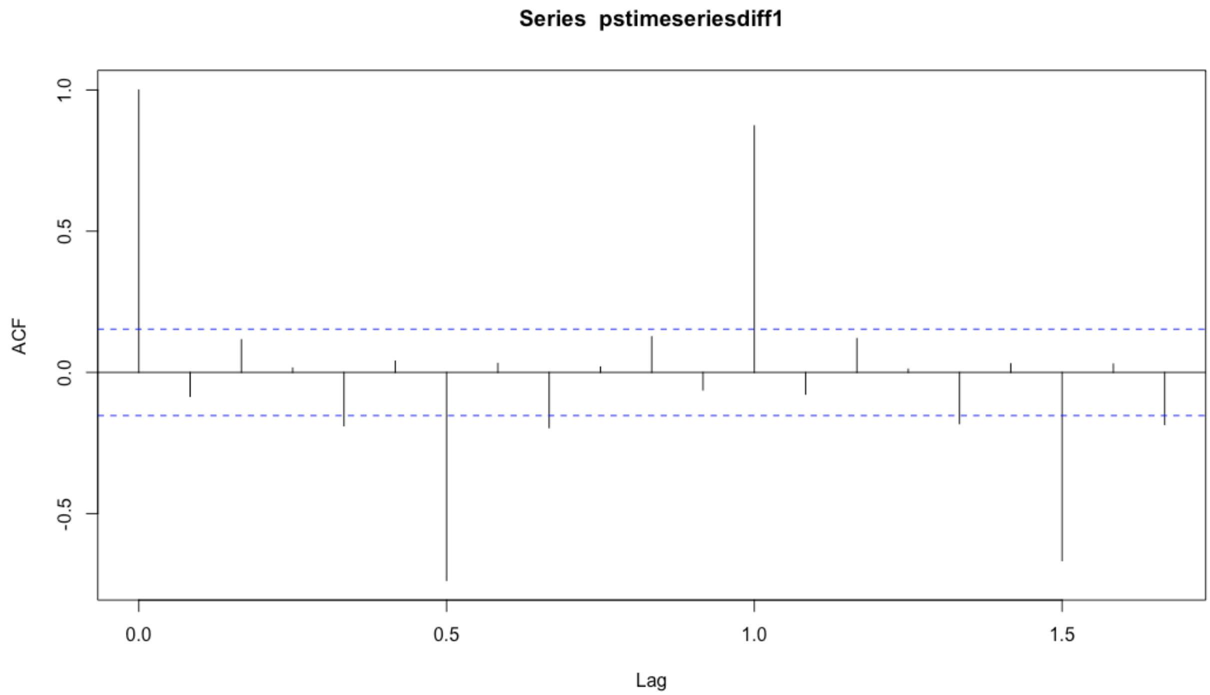


Figure 20. ACF plot

The blue dash lines indicate bounds for statistical significance. The scale is from -1 to 1 because it is the correlation coefficient. As we can see from the ACF plot, lag 0 exceed the significance bounds. But lag 1 is a negative value. Therefore, the q value would be 1.

```
Console   Terminal ×   Jobs ×
~/
> pacf(pstimeseriesdiff1, lag.max=20)
> pacf(pstimeseriesdiff1, lag.max=20, plot = FALSE)

Partial autocorrelations of series ‘pstimeseriesdiff1’, by lag

0.0833 0.1667 0.2500 0.3333 0.4167 0.5000 0.5833 0.6667 0.7500 0.8333 0.9167 1.0000 1.0833 1.1667 1.2500 1.3333 1.4167
-0.085  0.109  0.034 -0.202  0.005 -0.729 -0.151 -0.349 -0.039 -0.320 -0.286  0.601  0.014 -0.054 -0.044 -0.069 -0.094
1.5000 1.5833 1.6667
 0.146  0.021  0.036
>
```

Figure 21. Code for PACF

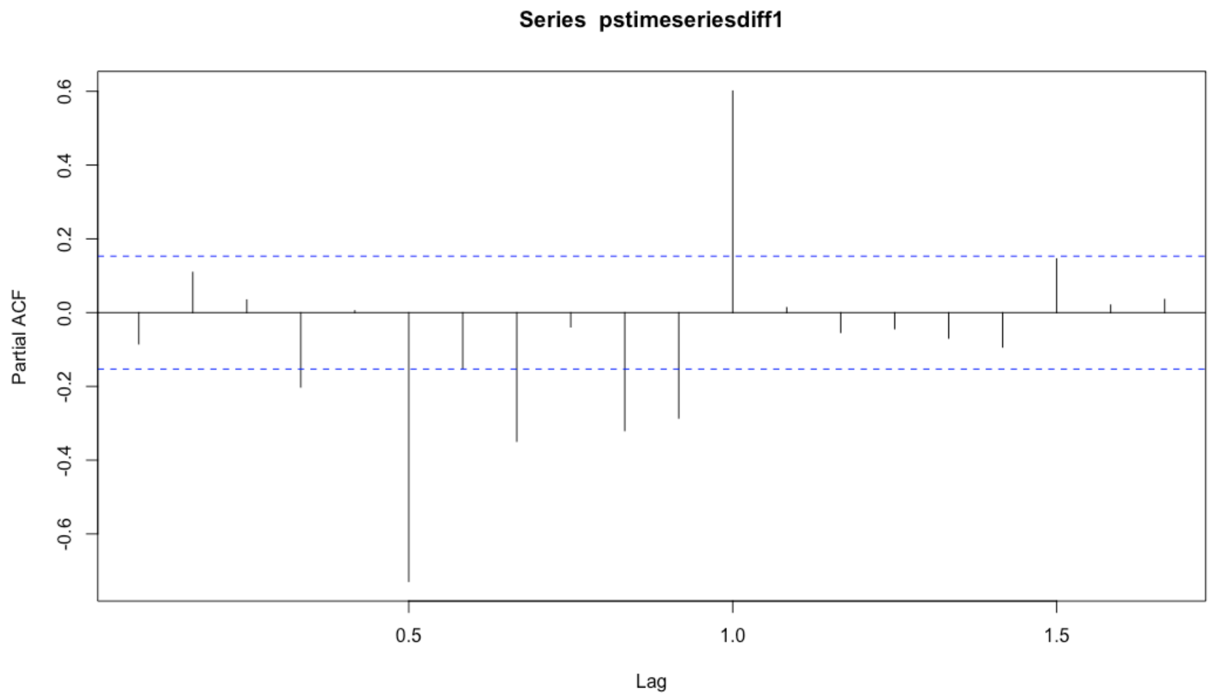


Figure 22. PACF plot

In this step, we could see that the partial autocorrelation at lag 5 is way exceed the significance bounds. Therefore the p value of ARIMA model is 5. The way to final decide the most appropriate values for ARIMA model is to consider the fewest parameter is the best. In this case, ARIMA (5,1,1) wins.

3.2.5 Selecting a Candidate ARIMA Model automatically

Use auto.arima() to calculate q, d, p values for the model.

```
Console  Terminal ×  Jobs ×
~/
> library("forecast")
> auto.arima(Passenger_Statistics$`Passenger Count`)
Series: Passenger_Statistics$`Passenger Count`
ARIMA(5,1,1) with drift

Coefficients:
         ar1     ar2      ar3      ar4      ar5      ma1      drift
      0.4669  0.1919  -0.0132  -0.2573  -0.2554  -0.9450  13518.065
s.e.  0.0778  0.0842   0.0853   0.0838   0.0797   0.0256   1452.513

sigma^2 estimated as 6.81e+10:  log likelihood=-2276.49
AIC=4568.98   AICc=4569.91   BIC=4593.78
```

Figure 22. Code for auto.arima()

As shown above, we’ve got ARIMA(5,1,1) with drift model. Therefore if we apply Arima() function, we are going to set include.drift = TRUE to alow drift in ARIMA model.

4 Conclusion

In this report we conduct two analysis: regression analysis and time series analysis. In part A, we’ve learned that the dataset is almost normally distributed and skewed to the right. After the regression we known that the United Airline passenger counts explain 89.73% of the variation in total passenger counts in San Francisco airport.

In the ARIMA analysis, we’ve conducted decomposing and differencing process. Then we selected candidates of the model as (5, 1, 1) both by manually and automatically.

Reference

1. City of San Francisco (May 10, 2019). *Air Traffic Passenger Statistics*. Retrieved from https://catalog.data.gov/dataset/air-traffic-passenger-statistics
2. Brownlee, J. (2017). *How to remove Trends and Seasonality with a Difference Transform in Python*. Retrieved from https://machinelearningmastery.com/remove-trends-seasonality-difference-transform-python/
3. Elprince, N. (2014). *Air Passengers Forecast*. Retrieved from https://rpubs.com/nohaelprince/47545