Predicting number of failures

Name: Yukang Xu

Ⅰ. Introduction

As a data scientist in ProcessMiner, we need to predict the number of failures that will occur at certain time of a day. These predictions will be useful for various business optimization decisions to reduce failures and the costs.

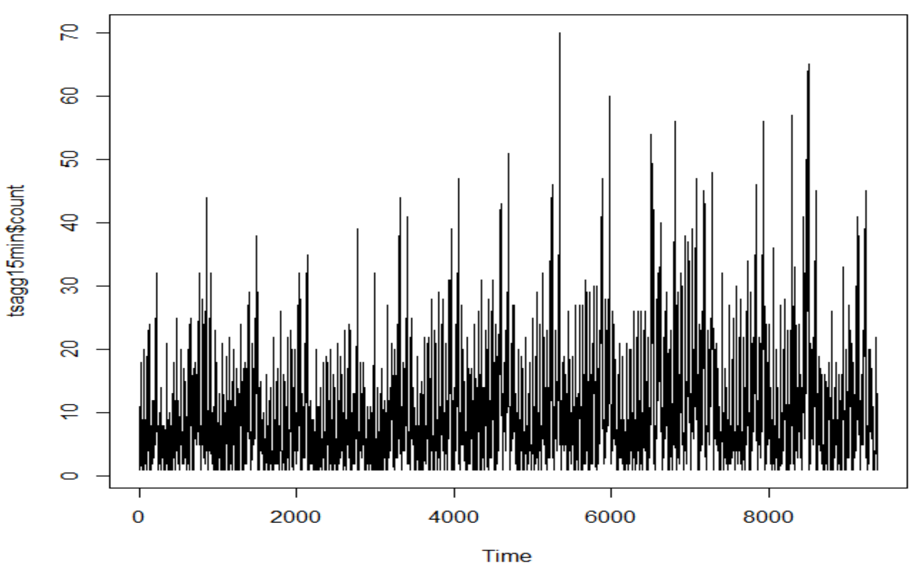
The data we have contains all the timestamps of failures. We need to implement a time-series forecasting method and visualization at the same 15-minute granularity over the next hour (4 periods ahead).

Ⅱ. Data Analysis

A. Clean the data

After import data into data frame, we cut the time line into multiple 15 minutes session. Then I count how many timestamps during one time period. The variance appears to be stable across the years observed, so there's no need for a transformation. There may be a trend, which is supported by the results of the ndiffs() function.

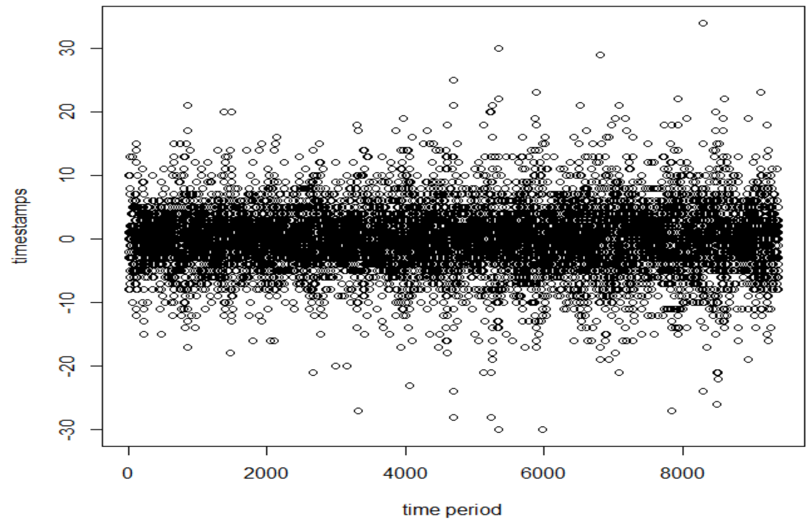
```
> head(tsagg15min)
                  cut count
1 1970-01-01 20:12:00     8
2 1970-01-01 20:27:00     8
3 1970-01-01 20:42:00     6
4 1970-01-01 20:57:00     3
5 1970-01-01 21:12:00     3
6 1970-01-01 21:27:00     1
```



B. Stationary Test

The differenced time series is plotted in the bottom and certainly looks more stationary. Applying the ADF test to the differenced series suggest that it's now stationary, so you can proceed to the next step.

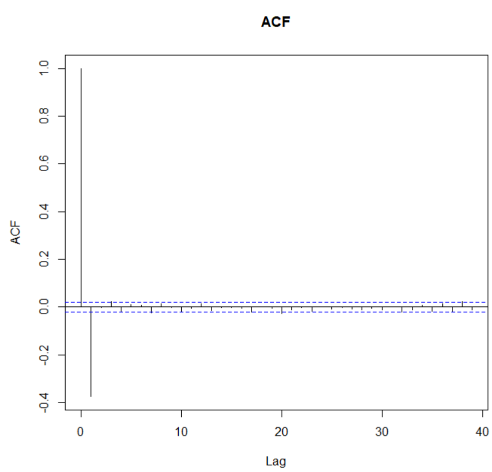
```
> adf.test(diff)

Augmented Dickey-Fuller Test

data:  diff
Dickey-Fuller = -27.432, Lag orde
r = 21, p-value = 0.01
alternative hypothesis: stationar
y
```

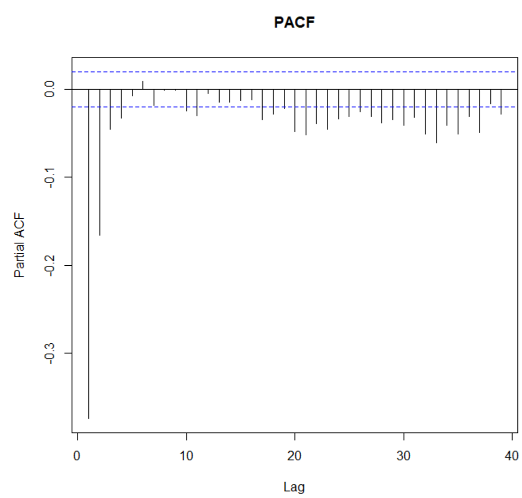


C. Model Selection

Possible models are selected based on the traditional ACF and PACF plots. We can also use auto.arima() function to verify our guessing. I finally choose Arima model arima(0,1,1).

```
> auto.arima(count)

Series: count
ARIMA(0,1,1)

Coefficients:
          ma1
      -0.4403
s.e.   0.0091
```





D. Model Evaluation

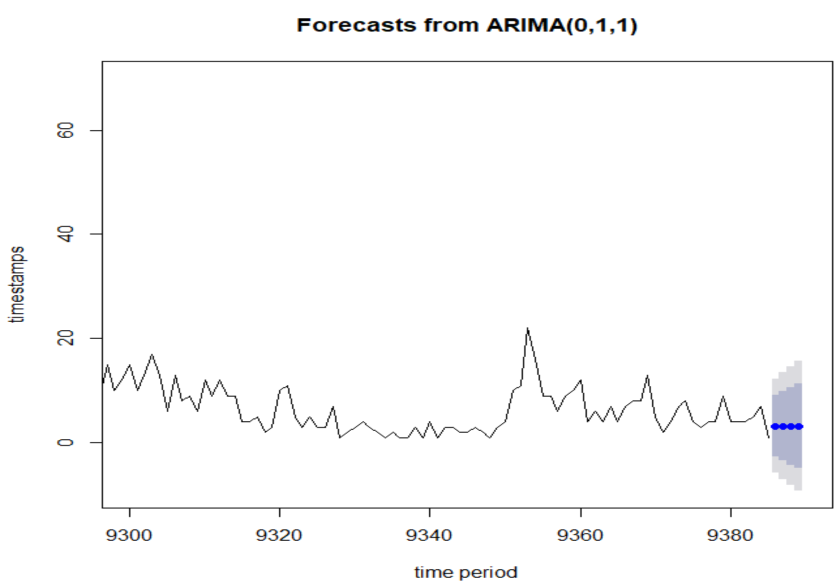
The Box.test() function provides a test that the autocorrelations are all zero. The results aren't significant, suggesting that the autocorrelations don't differ from zero. This ARIMA model appears to fit the data well.

```
> Box.test(fit$residuals,type="Ljung-Box")

        Box-Ljung test

data:   fit$residuals
X-squared = 0.31629, df = 1, p-value = 0.5738
```

Ⅲ. Forecasting

Once a final model has been chosen, it can be used to make predictions of future values. In the next listing, the forecast() function from the forecast package is used to predict four time periods ahead. Point estimates are given by the blue dots, and 80% and 95% confidence bands are represented by dark and light bands, respectively.

```
> forecast
     Point Forecast     Lo 80     Hi 80     Lo 95    Hi 95
9386       3.189239 -2.669985  9.048463 -5.771670 12.15015
9387       3.189239 -3.525210  9.903688 -7.079625 13.45810
9388       3.189239 -4.283188 10.661666 -8.238851 14.61733
9389       3.189239 -4.971061 11.349539 -9.290862 15.66934
```



From what we analyzed above, We can draw a conclusion that over the next hour, there will be three timestamps in every single time period.

R Code:

```r
##########################
# Load the package
install.packages("RJSONIO")
install.packages("highfrequency")
library("RJSONIO")
library("highfrequency")
library(xts)
library('stats')
library(matrixStats)
library(forecast)
library(tseries)

# Import data
json_file = 'C:/Users/xuyuk/Documents/failuretime.json'
json_file = RJSONIO::fromJSON(json_file)
data = as.data.frame(json_file)
data1='null'
data1$time=as.POSIXlt(data$time)
data1$count=seq(1,1,length.out=93142)
data1=as.data.frame(data1)
data1=data1[,-1]
str(data1)

# Visualize our data
data1$cut=cut(data1$time, breaks="15 mins")
tsagg15min = aggregate(count~cut,FUN='sum',data = data1);
head(tsagg15min)
plot.ts(tsagg15min$count)

# Stationary test
count=tsagg15min$count
ndiffs(count)
diff=diff(count)
plot(diff,xlab="time period",ylab="timestamps")
adf.test(diff)

# Model selection
acf(diff,main='ACF')
pacf(diff,main='PACF')
auto.arima(count)

# Fitting the model
fit = arima(count, order = c(0, 1, 1))

# Model evaluation
qqnorm(fit$residuals)
qqline(fit$residuals)
Box.test(fit$residuals,type="Ljung-Box")
accuracy(fit)

# Prediction
forecast=forecast(fit,4)
plot(forecast(fit,4),xlab="time period",ylab="timestamps")
plot(forecast(fit,4),xlim=c(9300,9390),xlab="time period",ylab="timestamps")
###################
```