Stripe: Data Analyst, Written Project Assessment

Myles Thomas

2/8/2021

This assignment is part of the interviewing process with Stripe for a data analyst role. Using over 1 million transactions on the Stripe database from the year of 2018, this time series data is to be used to make predictions on the future.

Using the data provided, please provide an estimate for the amount of money we should expect to be paid out on Jan. 1, 2019 (the day after the last day in the dataset).

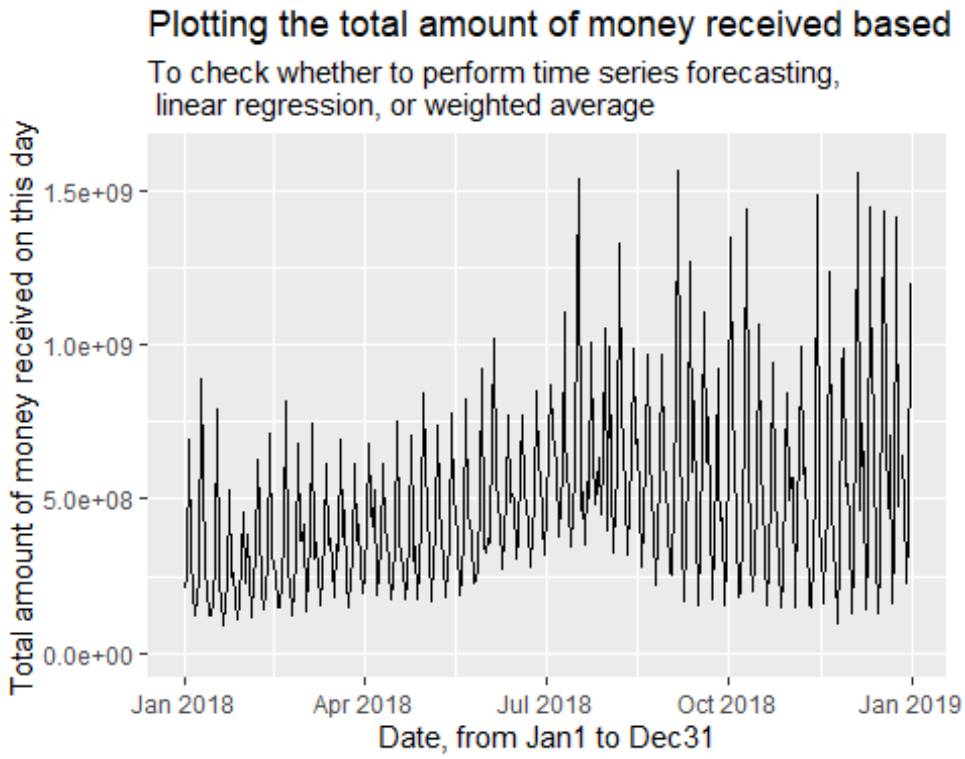
The dataset payouts.csv has information on the date, number of transactions, and amount of the transaction for various merchant id's.

```
# Load datasets
payouts <- read.csv("payouts.csv")

# Show the top 3
head(payouts[ , 3:5], 3)
```

```
##                          recipient_id count amount
## 1 id_23d90ec275370c686dedd7dc1c5e93b3     1   9786
## 2 id_72f05535ba5e6e5b141db6b5c1f1b13b     1   3750
## 3 id_d70e8046fe5583e1154b2e077133e27c     1   2258
```

Add a total amount column and plot the time series by total amount of money received by day.



Here is the same graph, but an interactive version to see which dates had which dollar amount. (Only visible on HTML output)

```
data <- data.frame(date = df_datenew$DATE, value = df_datenew$DailyReceived)

# Usual area chart
p <- data %>%
  ggplot( aes(x=date, y=value)) +
    geom_area(fill="#69b3a2", alpha=0.5) +
    geom_line(color="#69b3a2") +
    ylab("Money Received on this Day") +
    xlab("Date, spanning all of 2018") +
    theme_ipsum()

# Turn it interactive with ggplotly
#p <- ggplotly(p)
#p
```

In general, it appears that there is a direct positive correlation between days passing and amount of money received, which suggests that Stripe is getting more and more popular and used by more merchants. A weighted average would not do the increased popularity its justice, since it would be taking into account the beginning of the year when less merchants were using Stripe. I do not see any trends of regular time series data, either.

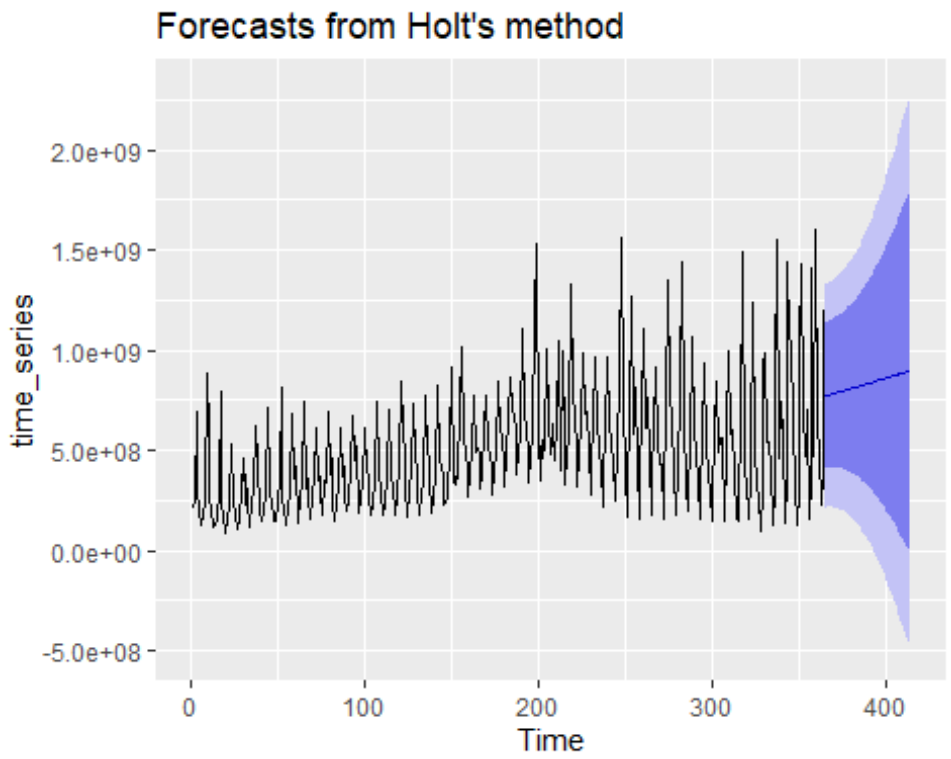
I will proceed with 'irregular time series' techniques. To make predictions using data WITH a trend but WITHOUT seasonality, double exponential smoothing forecasts – also known as

Holt's linear method – can be used for series with trend (but no seasonality). As seen in the plots above, the overall trend is upwards, but there is not a seasonality to the data.

Note: Within holt you can manually set the alpha/beta parameters; however, if you leave those parameters as NULL, the holt function will actually identify the optimal model parameters. (It does this by minimizing AIC and BIC values)

```
# create the time series
my.ts <- xts(data$value, data$date)
time_series <- as.ts(my.ts)

# plot the time_series
holt.ts <- holt(time_series, h = 50)
autoplot(holt.ts)
```



Here is the same time series, along the forecast of the next 50 days plotted and the resulting 80% and 95% confidence intervals. (It was only necessary to forecast 1 day ahead for January 1st, but the plot looks better with more days forecasted)

```
##                      ME      RMSE       MAE       MPE     MAPE      MASE
## Training set -8590355 283313554 209710402 -36.40126 59.33651 0.8731903
##                    ACF1
## Training set 0.2936343
```

Due to the volatility of this irregular data, and the large measurments of daily money received, our numbers for RMSE and some of the accuracy measuring statistics are very high (not good!). With only 365 observations to train the data on and a huge variance, our

forecast for the first day of 2019 will be best served to be a 95% confidence interval instead of just a point estimate, since our point estimate has a large probability of being very far from the truth.

Here is the point estimate, along with the resulting 95% confidence interval, for January 1 2019 using the double smoothing method for this time series data.

```
# grab the 2 bounds
df.preds <- as.data.frame(holt.ts)
lb<-df.preds[1, 4]
ub<-df.preds[1, 5]
pe<-df.preds[1, 1]

# print the point estimate
pe

## [1] 772210615

# interval 95% ci
interval95 <- c(lb, ub)
interval95

## [1]  213858360 1330562869
```