Data624 Project 1

Group 4

2021-06-26

Table of Contents

Group Members

* Subhalaxmi Rout
* Kenan Sooklall
* Devin Teran
* Christian Thieme
* Leo Yi

## Introduction

The dataset for this project was provided as a de-identified excel spreadsheet that included six different *groups*. Each group had two variables to be forecasted 140 periods into the future using 1622 historical periods. This data had been provided as time series data but the time period of the observations were not given. In place of a timestamp, we were given a column called SeriesInd which contains unique numerical values that represents an undisclosed time period.

Using glimpse, we can view the attributes of our dataset:

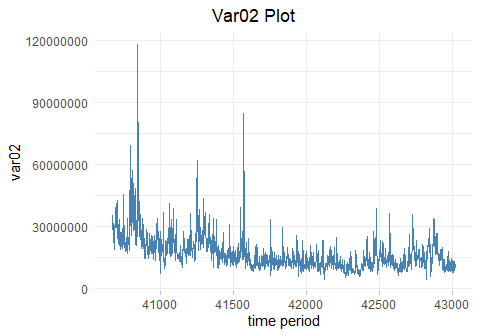
## Rows: 10,572  
## Columns: 8  
## $ row\_index <int> 40669, 40669, 40669, 40669, 40669, 40669, 40670, 40670, 4...  
## $ group <chr> "S03", "S02", "S01", "S06", "S05", "S04", "S03", "S02", "...  
## $ var01 <dbl> 30.64286, 10.28000, 26.61000, 27.48000, 69.26000, 17.2000...  
## $ var02 <int> 123432400, 60855800, 10369300, 39335700, 27809100, 165874...  
## $ var03 <dbl> 30.34000, 10.05000, 25.89000, 26.82000, 68.19000, 16.8800...  
## $ var05 <dbl> 30.49000, 10.17000, 26.20000, 27.02000, 68.72000, 16.9400...  
## $ var07 <dbl> 30.57286, 10.28000, 26.01000, 27.32000, 69.15000, 17.1000...  
## $ date <date> 2011-05-06, 2011-05-06, 2011-05-06, 2011-05-06, 2011-05-...

In our initial exploration, we noticed that the index SeriesInd was in sequential order, however, after every 5 observations there would be two values missing before the series continued again. We used count\_gaps() to analyze these implicit gaps within our data and came to the consensus that this was most likely daily data, showing data for Monday through Friday but excluding weekends and holidays. We move forward with this assumption.

#### Initial Exploration of the Data

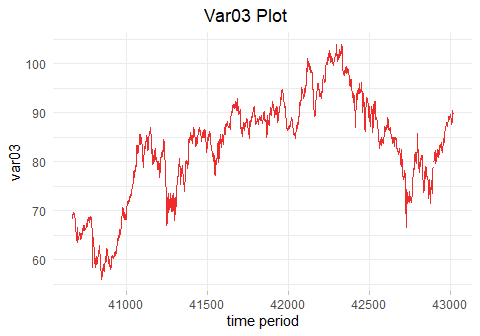
In order to forecast each variable within each group, we needed to break the larger dataset into its individual groups so we could perform our analysis and properly visualize our data.

We began by visualizing our data using a line-plot, this helped us to see initial trend, seasonality (or lack thereof), and cyclicity of our variables. In our analysis we identified two types of variables, one type seemed to have lower numerical values moderate fluctuations form day to day. The other type of variable had very large numerical values and had extreme fluctuations from day to day. We’ll visualize the two variables from group S05 below as an example:



Looking at the plot above, you can see the values on the Y-axis are extremely large and the variation from day to day is fairly significant. Additionally, we can see several large spikes throughout in the series. In looking at this plot, because there is so much movement, its hard to see if there is a trend or some type of seasonality. We determined additional visuals and tests would be needed to confirm.

The other variables in our dataset looked very similar to Var03 from S05 shown below:



We can see variation within this type of variable is considerably less than in the the other type. Additionally, it looks like there is no apparent seasonality. In many of the variables of this type of data, we did see long trends (either upward or downward) and cyclicity. In looking at all the plots for every variable, we determined that further analysis was necessary to determine trend and seasonal components of the data.

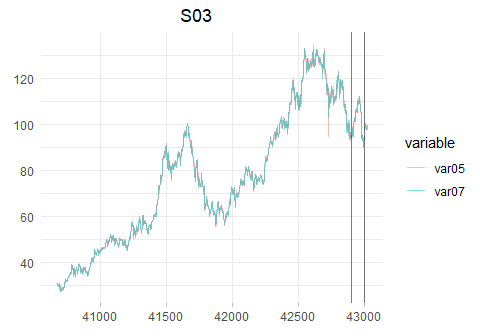
## Data Preparation

Many of the time series algorithms, both for forecasting and visualization, require that there be no missing values in the data. With this requirement, we deemed it necessary to fill nulls early in our data prep process.

#### Imputing Missing Values

In data preparation, imputation of missing values is a critical step. Each of the variables provided had missing values within the data and many approaches for filling them seemed appropriate. The approaches we took differed depending on if we were imputing for the first or second type of data. For the data that had large variations from day to day (that looked almost like white noise) we deemed taking an average appropriate. For the more stable datasets, we decided to use tidyr::fill() to fill the missing value with the previous value in the dataset. This seemed to make sense as each point in this datasets never seemed to be very far away from the previous one.

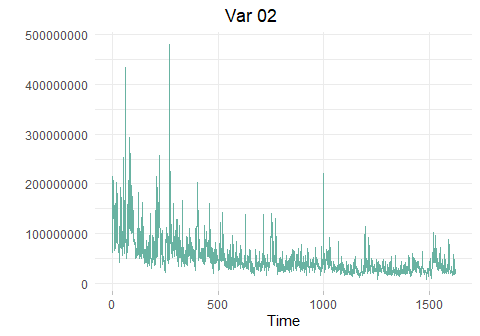
Below is a plot of Var05 and Var07 from S03. The vertical lines represent the points where there is missing data. In this case, we used tidyr::fill() to fill the missing value with the previous value in the dataset.You can see that by choosing this method, we haven’t interfered with the natural trend of the data.



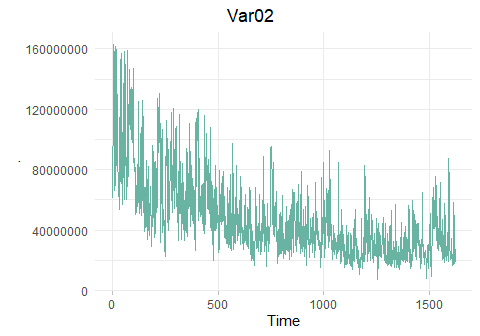
#### Outliers

In several of the datasets, there are clear outliers that would strongly influence the models we built. Our approach was to use the forecast::tsclean function to resolve this issue. We use tsclean function to return a cleaned version of a time series with outliers and missing values replaced by estimated values. This function internally calls na\_interpolation and replaces the outliers with the estimated values, if we select the parameter replace.missing = True then it interpolates missing values as well.

Looking at Var02 from S03 in the plot below, we can see that there are some extreme spikes within the data. While these observations may not be *erroneous* data, they certainly could have a significant impact on our modeling:



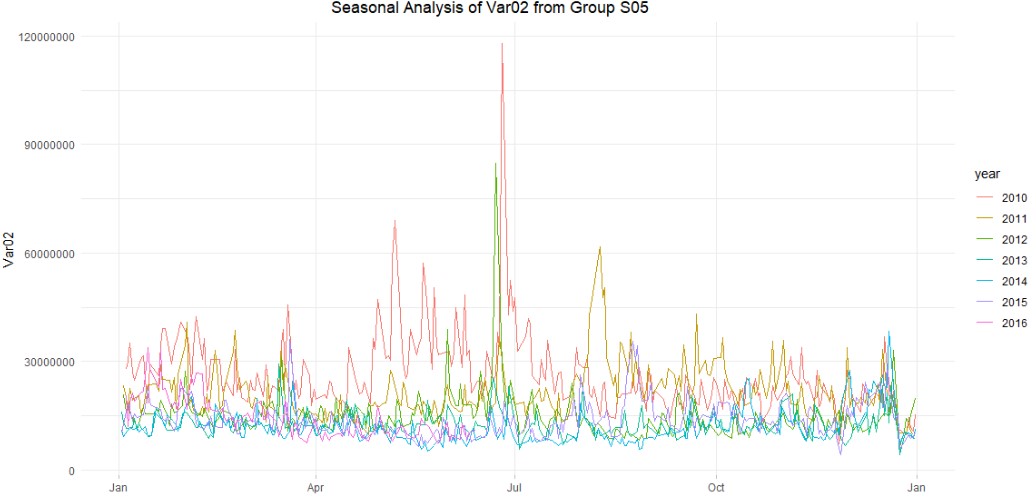
After using forecast::tsclean, you can see the spikes have been muted and the data looks cleaner.



## Time Series Analysis

#### Seasonality

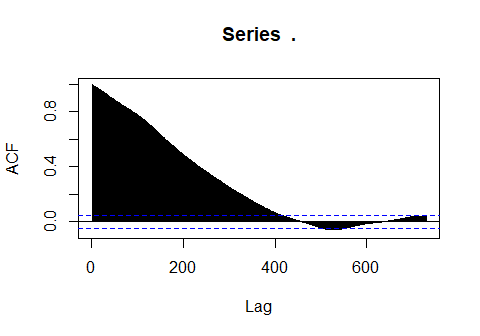
Understanding seasonality in this data was extremely difficult since we didn’t know what the time period of our observations were. A key component to the time series objects in R is understanding the frequency of your data. All further analysis hinges on the values you place in your frequency parameter. Based on our previous discussion, we made the assumption that this was daily data and used both 1 and 365 as frequency values in our time series objects. With our assumption in mind, we had several approaches to determining seasonality. One method was to construct plots and to visually examine them for seasonality. One such plot is the seasonal plot created by using gg\_season. This plot allows us to overlay different years of a variable on top of each other so seasonality can be inspected:



In reviewing the seasonal plot above, we can see certain time periods in every year that have peaks and valleys indicating a possibility of seasonality.

Another method was to plot autocorrelations with a large lag and to look for spikes in order to determine the length of the period. Finally, we also used the frequency function from the stats package to assess the data’s inherent frequency.

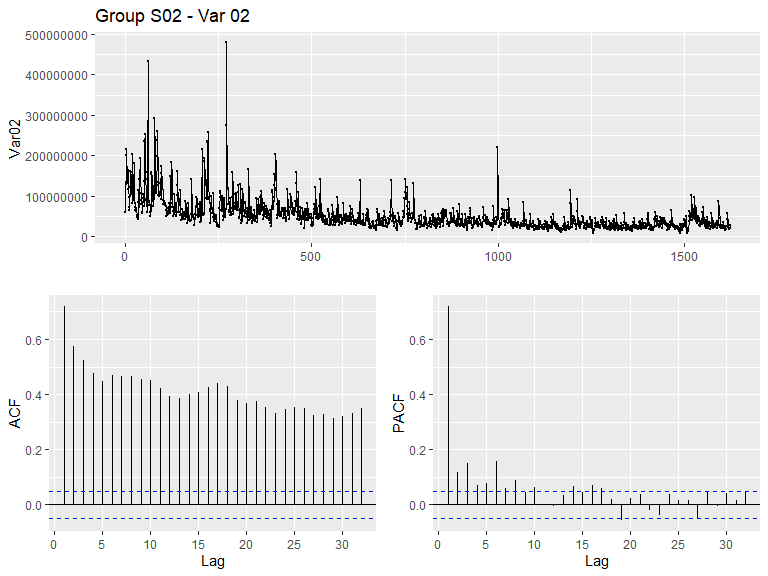
Below is an autocorrelation plot for Var05 from group S03 lagging 720 days. This lag period was chosen to include approximately two years worth of observations. This plot shows a steady decline. If seasonality existed within this data, we would observe peaks, which we could then use to set the frequency for the time series.



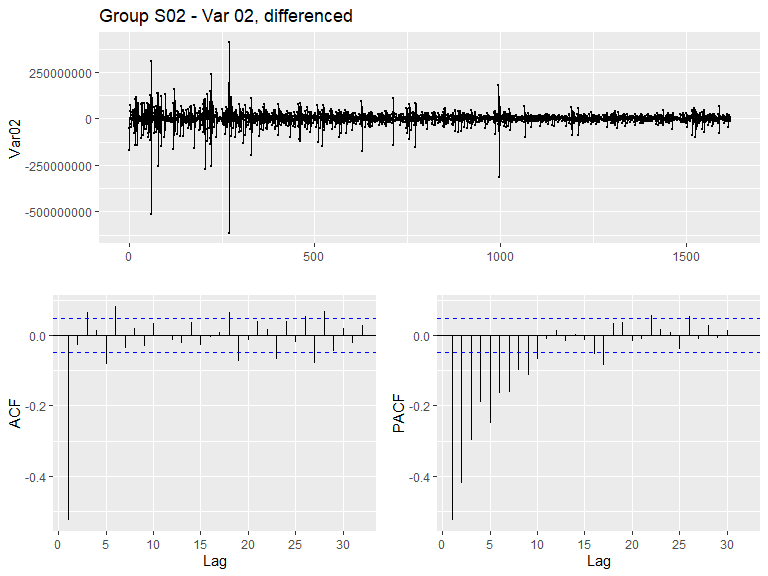
#### Differencing

Having stationary data is critical to time series forecasting. Differencing allows us to transform the time series into stationary data. To determine whether a series was stationary or not, we plotted the series using ggtsdisplay and observed the plot, as well as running Kwiatkowski-Phillips-Schmidt-Shin (KPSS) tests. If the data was non-stationary, we made use of the diff function to difference our data. If the data still showed signs on non-stationarity, we differenced it again. The differenced results were plotted each time until stationarity was reached.

We show this process below for Var02 from Group S02:



Looking at the plot above we can see that this data is not stationary. The top plot does not look like white noise and the ACF and PACF plot both have many spikes above the critical threshold. This tells us that we need to difference our data.



Reviewing our results above, we can see that the top plot looks much more like white noise, additionally we see that we have fewer spikes crossing the critical threshold. This data is now stationary.

## Modeling

Having prepared our data for modeling, we split each time series into training and testing sets, using about 80% of the first 1,622 rows as the training data.

We created multiple models using the training data and compared the accuracy against the test data. The models with the minimum errors, using RMSE and MAPE, were selected and then the best models were used to generate the forecast for the next 140 periods.

Depending on the dataset, we ran auto.arima and ets models and selected the method that was most accurate. Accuracy between modeling approaches differed by variable.

We show below our comparison of model diagnostics between auto.arima and ets for Var05 from group S03:

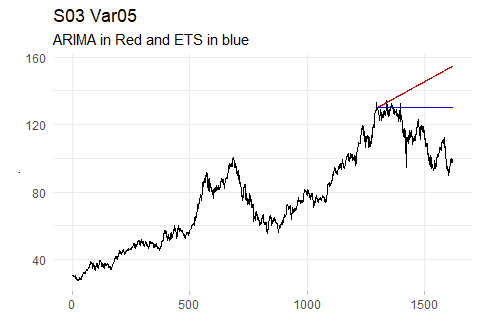
**auto.arima() accuracy:**

## ME RMSE MAE MPE MAPE MASE  
## Training set -0.000129212 1.248385 0.8778607 -0.03489299 1.314015 0.98769  
## Test set -29.049859982 34.509005 29.0577636 -27.65330257 27.659181 32.69319  
## ACF1 Theil's U  
## Training set -0.0002637416 NA  
## Test set 0.9856681602 16.27615

**ets() accuracy:**

## ME RMSE MAE MPE MAPE MASE  
## Training set 0.08263014 1.255415 0.8847694 0.1032869 1.322841 0.9954631  
## Test set -16.44346086 20.334135 16.5771789 -15.8117042 15.912734 18.6511526  
## ACF1 Theil's U  
## Training set 0.000387123 NA  
## Test set 0.976861010 9.672666

Additionally, we plot the results of our model run below:

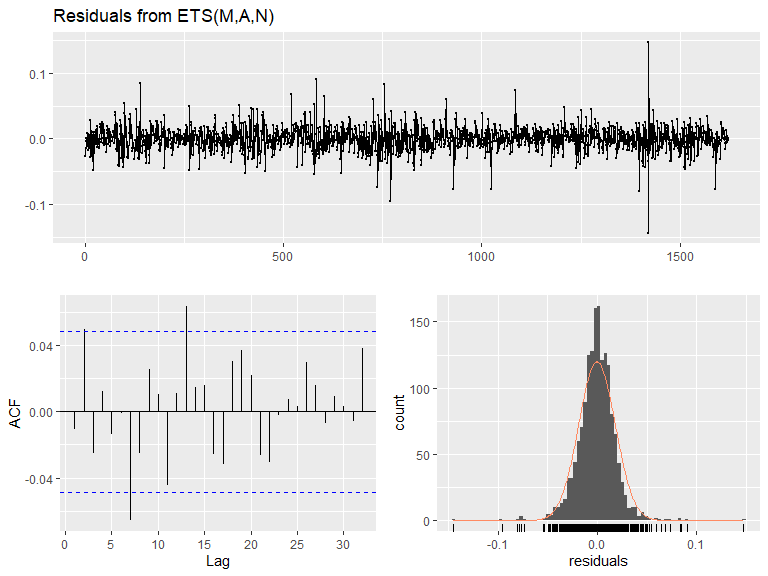


The black line is the actual data, the red line is the ARIMA model, and the blue line is the ETS model. Based on these results, it looks like exponential smoothing model performed the best on the test data.

In this case, we would choose the ETS model as it is more accurate.

#### Check Residuals

After fitting the models with the best performer on the test data, the final step before making the predictions was to check the residuals.



##   
## Ljung-Box test  
##   
## data: Residuals from ETS(M,A,N)  
## Q\* = 15.087, df = 6, p-value = 0.01959  
##   
## Model df: 4. Total lags used: 10

In this case, our residuals look like white noise and have a fairly normal distribution. Additionally the ACF plot has very few spikes over the critical boundary. This gives us confidence that our confidence intervals from this forecast would be reliable. As our objective was to generate the most accurate forecasts, and not necessarily to have perfect confidence intervals, some of our models don’t have picture perfect residuals, but generate more accurate forecasts than other models.

### Forecast

Finally, once we had chosen our models, we predicted the next 140 periods for each variable in all six groups, generating a total of 12 forecasts. We show the first several forecasted values from Var05 of S03.

fit\_v5 %>%  
 forecast(h=140) %>%  
 data.frame() %>%  
 select(Point.Forecast) %>%  
 head()

## Point.Forecast  
## 1623 98.74952  
## 1624 98.81825  
## 1625 98.88698  
## 1626 98.95571  
## 1627 99.02444  
## 1628 99.09317

## Conclusion

In this project we walked through a full time series analysis to understand a de-identified dataset and ultimately to generate forecasts for 12 individual variables. To dig deeper into each group, please see the following reports:

* [S01](https://rpubs.com/devinteran/Data624-Project1)
* [S02 & S06](https://rpubs.com/Subhalaxmi/784391)
* [S03](https://rpubs.com/data_consumer_101/784448)
* [S04](https://rpubs.com/ksooklall/785164)
* [S05](https://rpubs.com/christianthieme/785258)