**Summer DATA 624: Predictive Analysis Project 1**

**Group 4**

**June 27th 2020**

Layla Quinones

Sergio Ortega

Jack Russo

William Outcalt

Neil Shah

**Table of Contents**

Contents

[**Executive Summary** 3](#_heading=h.gjdgxs)

[**Overview** 4](#_heading=h.30j0zll)

[**Methodology** 4](#_heading=h.1fob9te)

[**Data Ingestion** 4](#_heading=h.3znysh7)

[**Exploratory Data Analysis** 4](#_heading=h.2et92p0)

[**Missing Data Imputation and Outliers** 5](#_heading=h.tyjcwt)

[**Model Forecasting** 8](#_heading=h.3dy6vkm)

[**Results** 8](#_heading=h.1t3h5sf)

[**Conclusion** 8](#_heading=h.4d34og8)

[**Appendix** 8](#_heading=h.2s8eyo1)

[**References** 16](#_heading=h.17dp8vu)

# **Preface**

This report summarizes Group 4’s step-by-step methodology in forecasting two variables in six unknown time-series. Step by step rationale will be provided and corroborated with visual and statistical evidence. The goal is to provide a complete narrative on the practical aspects of time-series forecasting, challenges encountered and lessons learned.

This document has two paths: a **“General Overview”** section with a cursory summary for casual readers, and **“Technical Detail”** for statisticians and practitioners.

# **Technical Summary**

Group 4 forecasted two-variables on six unknown time-series using a baseline Naive, Drift, Exponential and ARIMA model, using MAPE as an error metric. Time-series were processed for outliers replacement, seasonal-trend decomposition, missing value imputation and tested for stationarity. Models were trained via a 80/20 test-train split, residuals were tested for zero-mean and MAPE was compared to Naive model baseline. ARIMA models were found to produce lowest MAPE values on testing data.

# **Introduction**

Time series are datasets indexed by units of time. Quarterly sales, stock prices or number of passengers on a bus for a given day are all examples of time-series. Building accurate projections about the future behavior of time-series, or **forecasting**, is an area of active research, significance and many statistical models exist. However before one can haphazardly apply a forecasting model, practical concerns of the quality of data, the nature of the time-series and underlying assumptions in a model must be addressed.

# **Methodology**

Group 4 adopted the following step-by-step protocol to forecast given data.

1. **Data Extraction**: Extracting the raw data into the environment
2. **Exploratory Data Analysis & Data Processing**: Visualizing trends and structure of data. Followed by imputing missing data, processing outliers and performing transformations for model input.
3. **Model Development**: Applying forecasting models on processed data
4. **Analysis**: Choosing the best suited models based on statistics and error metrics.
5. **Results:** Process and present findings

## 

## **Data Extraction**

**General Overview:**

Data was uploaded online and loaded into computational software (R) . The data was composed of six series with 2 forecast variables, and was separated into individual series.

**Technical Detail:**

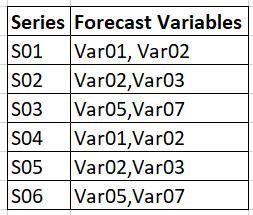
Data was presented as an Excel sheet [Figure 1], loaded into GitHub for ease, and featured six time-series groups *(S01,S02,S03,S04,S06)* each with variables (*Var01, Var02, Var03, Var05 and Var07)*, targeted for forecast. Raw data was loaded into R via the read.csv() function and subsequent groupswere manipulated via filter() and select().

A screenshot of a cell phone

Description automatically generated

**Figure 1: Excel screenshot of raw data**

Series with targeted variables to be forecasted were defined in the following figure



**Figure 2: Series IDs with variables**

## **Exploratory Data Analysis & Data Processing**

**General Overview:**

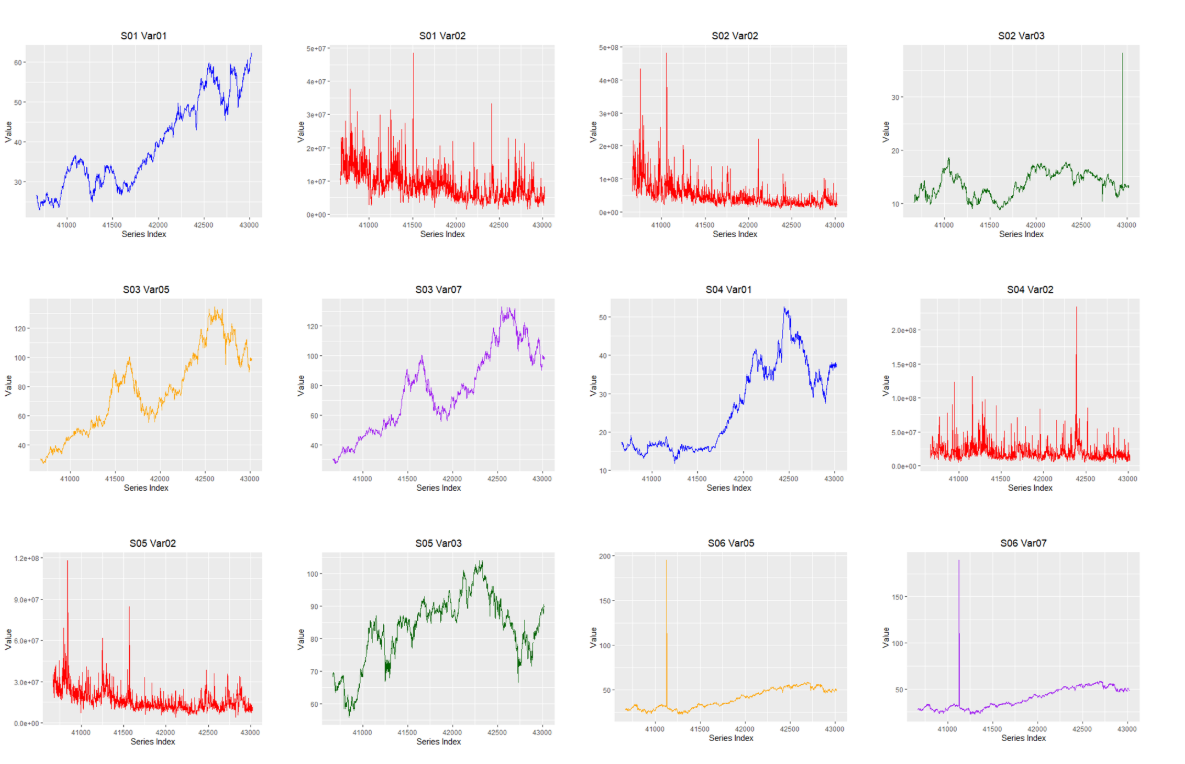
Exploratory data analysis (**EDA**) is the use of plots and statistical analysis to better understand the nature of the data. It’s a typical precursor to data-processing, in which the data is manipulated and “cleaned” to allow for better forecasting model performance. In our case, the individual time series were plotted to visualize their components specifically **seasona**l (recurring patterns) or **trend** (directional movement). No seasonality was found.

**Outliers**--or extreme values that can skew analysis--were identified via box-plots that break down percentile ranges of the data. Outliers were then replaced when possible with estimates from surrounding data. **Missing values** were **linearly interpolated**--estimated via a line between known points--rather than eliminated to ensure forecast integrity.

Data was checked for **stationarity**--that is their properties such as mean or variance--doesn’t change over time via a statistical test (**Augmented Dickey-Fuller Test** or **ADF**), and can be a requirement for modeling (ARIMA). Non-stationary time-series were **differenced, subtracted from a previous period to stabilize the mean by reducing trend and seasonality**  in order to try to produce stationarity and re-checked via ADF. This resulted in a “clean” training set that was ready forecast modeling.

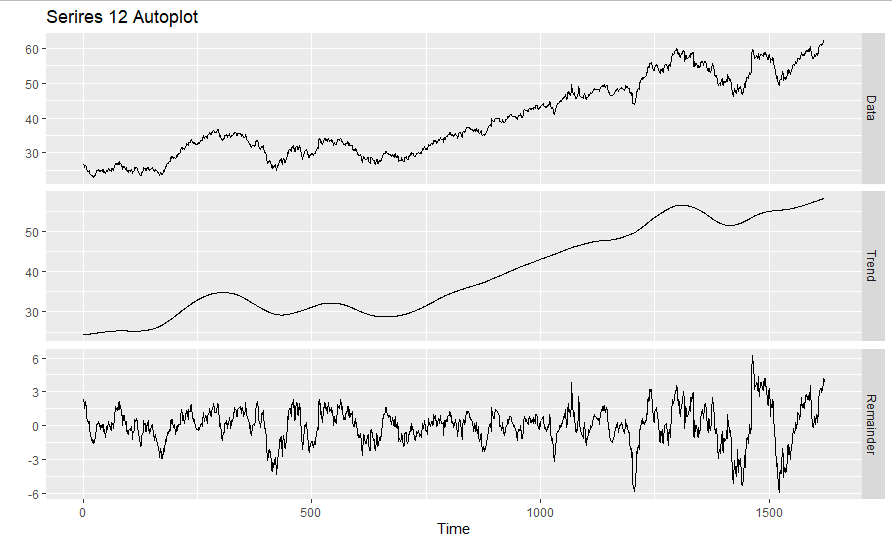
**Technical Detail:**

Plots of each series (Figure 3) were generated to identify seasonal and trend-cycle components, which would delineate which forecasting techniques would be appropriate, and to observe overall behavior. Obvious out-liers (the green spike in S02-03) were present in the data which will be explored via box-plots.

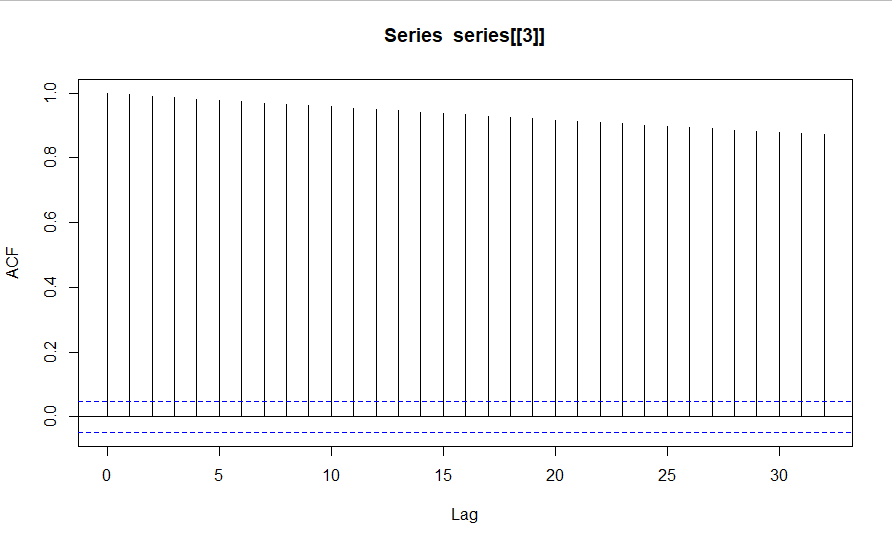


**Figure 3: Plots of data series prior to processing**

Series S01-02,S02-02,S02-03,S04-02 and S05-02 had differing frequencies from the other sets. At first it was thought to be seasonal behavior but upon use of mstl(), an automated STL decompositions [Figure 4] showing no seasonality component, as well as lack of periodic lags in ACF plots [Figure 5], we concluded it was evidence of cycle-trend and not seasonality.



**Figure 4: STL decomp of Series[12] revealing no seasonality**

****

**Figure 5: ACF of S02-Var-02 showing cycle-trend but no seasonality**

High variability in Var-02 and applied log() transformation [Figure 6] to stabilize variance--which also will improve model performance.

*# Calculate log for series 2,3,8,9*  
log\_indexes <- **c**(2,3,8,9)   
log\_series <- **list**()

*# Iterate through each series*  
**for** (i **in** **seq**(1,12,1)){  
 **ifelse**(i **%in%** log\_indexes,

log\_series[[i]] <- **log**(series[[i]]), *#apply log to those listed*

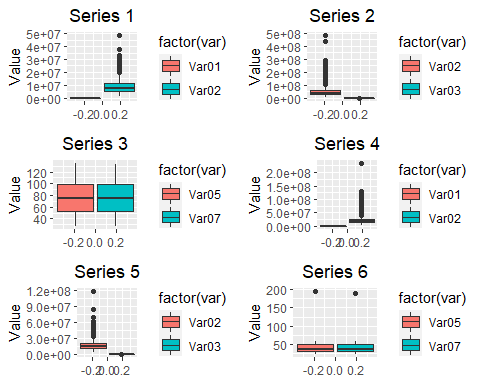
log\_series[[i]] <- series[[i]])  
}

**Figure 6 : Log transformations**

Box plot profiles [Figure 7] were used to explore the distribution of values across the time -series to identify outliers in the data that could affect our analysis. A cursory view shows significant outlier presence for the following Series-variables:

S01V02, S02V02, S04V02, S05V02, S02V03, S05V03, S06V05, S06V07.

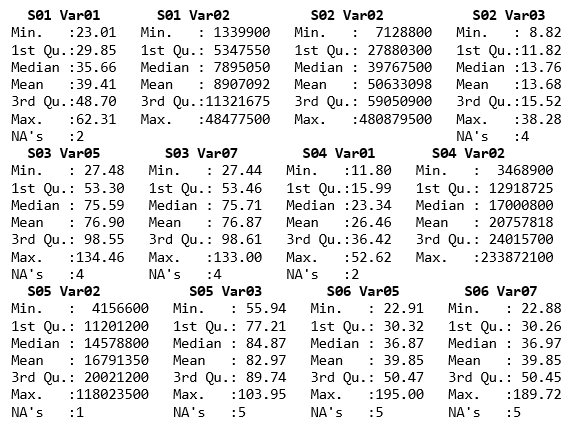
All these outliers need to be processed before we can proceed to forecast modelling.



**Figure 7: Box-plot analysis of time-series for outliers**

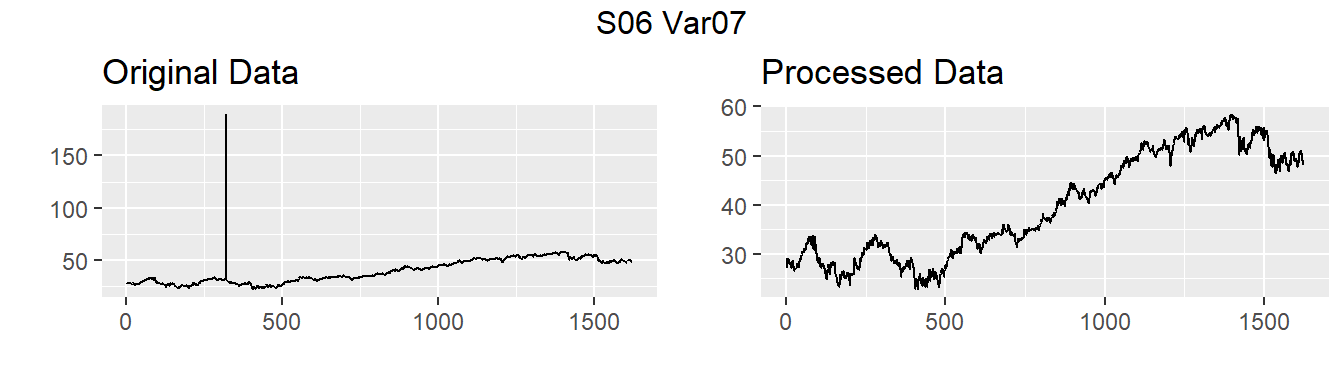
## **Missing Data Imputation and Outliers**

Missing data or NA values were found in 9 of the series and summarized in the last row for each set in Figure 8.



**Figure 8: Data summaries by time-series prior to pre-processing.**

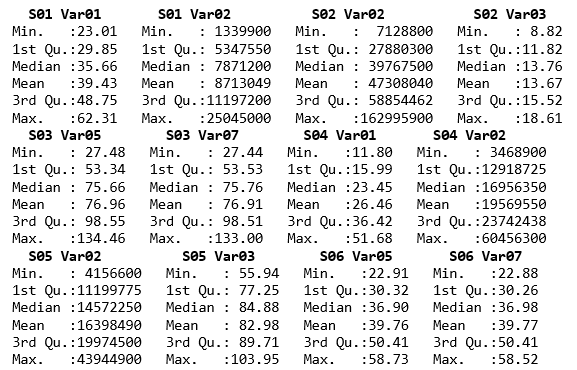
The missing data points for the time series consisted of less than 1% of overall data but were still processed for continuity across models. Group 4 used the tsclean() function which identifies outliers through percentile binning and replacing them via interpolation, as seen below in Figure 9.



**Figure 9: Example of outlier removal on series 06**

The tsclean() function also will linearly interpolate missing data as demonstrated below in Figure 10.

.

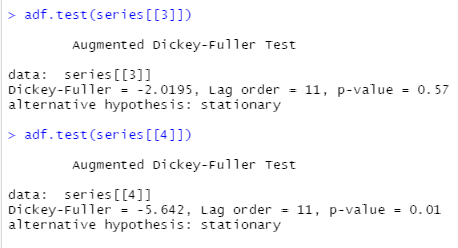


**Figure 10: Data summaries post outlier and NA imputation**

## 

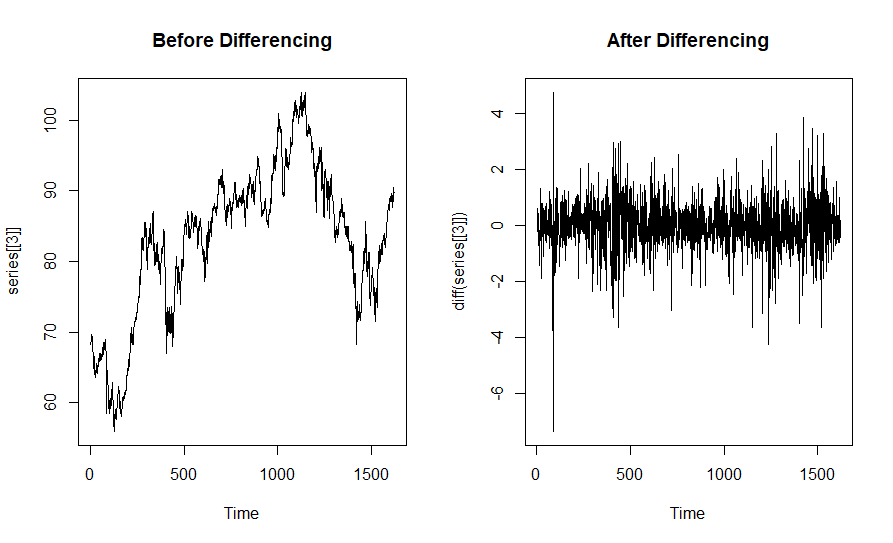
## **Stationary Analysis**

Stationary time series is a requirement for many forecasting techniques (i.e.ARIMA) and can improve model performance. Time series were hypothesis tested via the Augmented Dickey Fuller test (ADF) in Figure 11 via the R function adf.test() , using a critical p=.05 and the null hypothesis as non-stationary unit root.

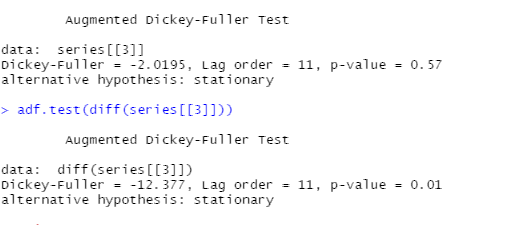


**Figure 11: ADF test with critical p=.05; Insufficient evidence to reject null for Series 3**

ADF-test revealed that Series 4, Series 5, Series 10 and Series 11 were stationary. Differencing was applied to non-stationary time series, and re-tested via ADF for stationarity as seen in Figure 12 and 13.



**Figure 12: Example of impact differencing on making Series 3 stationary**



**Figure 13: Example retesting of differenced time series for stationarity; reject null for Series 3 indicating stationarity**

## 

## 

## 

## **Model Forecasting**

**General Overview:**

After data was cleaned and imputed, a portion of data was marked as **training**, or a set that would be used to train the models, and the remaining reserved as **testing**, which would be used to evaluate model performance. Training and test splits are common in practice and help prevent **overfitting**, which happens when a model learns the specific nuances of a particular data set but will not perform on unseen or new data. Training set accuracy from each model was measured using the mean average percent error metric (**MAPE**). The optimal model for each time series was chosen based on the lowest MAPE metric versus a baseline model. **Residuals**--or the remaining information left over from the model-- were tested to ensure models best statistically fit the data.

**Technical Detail:**

An 80/20 training testing split was created via index partitioning.

The following forecasting methods were chosen for due to underlying assumptions and simplicity.

* **Naïve Method:** forecasts for all future values were set to the value of the last observation in historical data (used as baseline)
  + R function used: naive()
* **Drift Method:** forecasts for future values fall on a “line” drawn from the first and last values extrapolated into the future.
  + R function used: rwf()
* **Exponential smoothing:** forecasts for all future values were based on a weighted-average in which more recent observations have more significance than earlier values.
  + R function used: ses()
* **ARIMA:** a forecasting model with a combination of **AutoRegressive** model, in which values internally regressed upon each other, and a **MovingAverage** model in which a smoothed average is used for predictions. Tuning parameters include p, d and q.
  + R function used: auto.arima() and arima()

**Note:** Functions from the forecast() library were used for forecasting

Optimum models were based on the lowest training set MAPE (**Mean Absolute Percent Error**) comparison against the baseline Naive model. MAPE--as defined in Figure 14below--is a measure of the sum of the difference between fitted values and actual values divided by the number of data points and expressed as a percent.



**Figure 14: MAPE metric [10]**

The model that contained the lowest MAPE value for all series was the ARIMA models. Previous mentioned log transformation of Var 02 significantly reduced variability in it’s MAPE value and corroborated our processing step.

Residuals were analyzed to ensure zero mean and resemblance of white noise through ACF plots (= 0.05) and Ljung box model fit test, ideally indicating models captured all behavior of data. An initial Ljung Box-test discovered that *S01 Var02*, *S03 Var03*, and *S06 Var05* failed the Ljung Box-Test for the auto.arima() function.

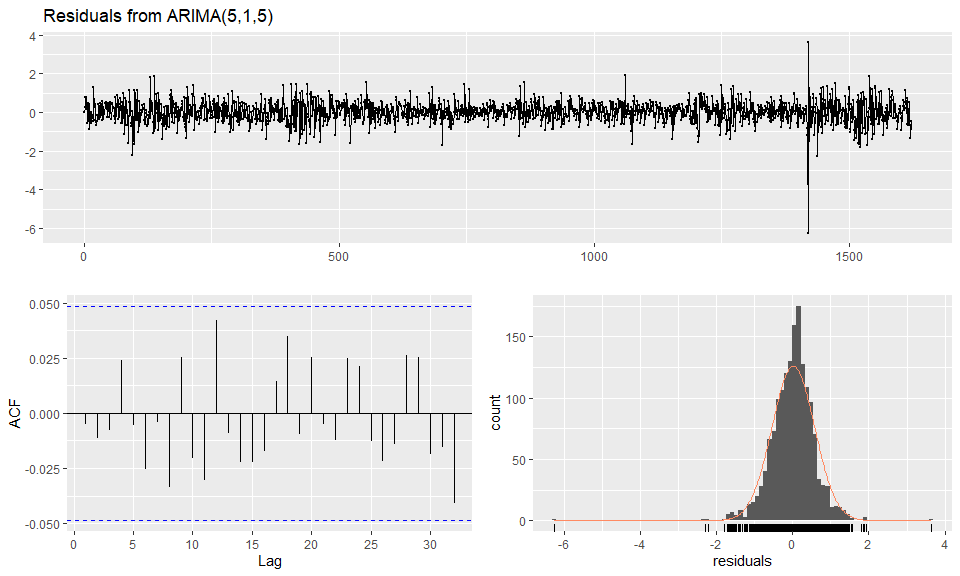
We developed a function that allowed a function that found optimal values for tuning parameters p,d and q for ARIMA models as seen in the code snippet in Figure 15 below:

*#Series 2, 4, 11 all failed Ljung-Box test*  
*# Lets manually find optimal parameters*  
optim\_AIC <- **function**(series, series\_index){  
 df <- **data.frame**()  
 count <- **c**(0,1,2,3,4,5)  
 **for** (i **in** series\_index){  
 **for**(p **in** count){  
 **for**(d **in** count){  
 **for**(q **in** count){  
 params <- **c**(p,d,q)  
 **try**(aic\_values <- **AIC**(**arima**(log\_series[[i]], order=params)))  
 **try**(series <- i)  
 df <- **rbind**(df,**c**(aic\_values,series,params))  
 }  
 }  
 }  
 }  
 **colnames**(df) <- **c**("AIC", "Series", "P","D","Q")  
 **return**(df)  
}

**Figure 15 : Tuning algorithm**

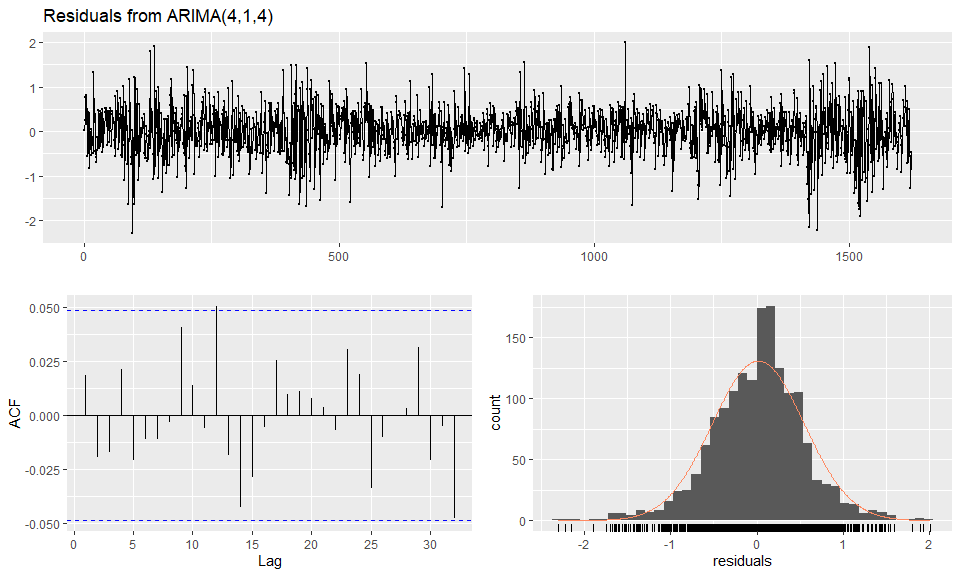
The algorithm in Figure 15 accepts a time series and iterates through different *p*,*d* and *q* parameters and computes AIC values for corresponding time series.The parameters that correspond with the smallest AIC values are selected to achieve an optimal ARIMA model, and this model is then interpreted once again using the Ljung-Box test.

After applying the new model with specific tuning parameters, S06 Var05 still failed the Ljung Box test. After inspection of residuals, we noticed that the residuals for this model contained an obvious outlier as seen in the residual plots in Figure 16below

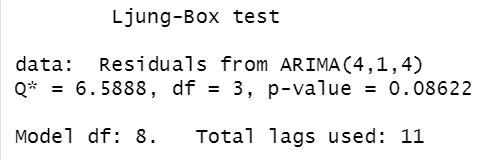


**Figure 16: Residual, Residual ACF and Residual histogram of S06 Var05 with outlier**

After imputing this specific data point with the mean value of the data set, it passed the Ljung Box Test. Figure 17 is a visualization of the residual plots and Ljung Box-Test output using the checkresiduals().



**Figure 17: Residual, Residual ACF and Residual histogram of S06 Var05 without outlier**

****

**Figure 18: Output for Ljung box test**

**Results**

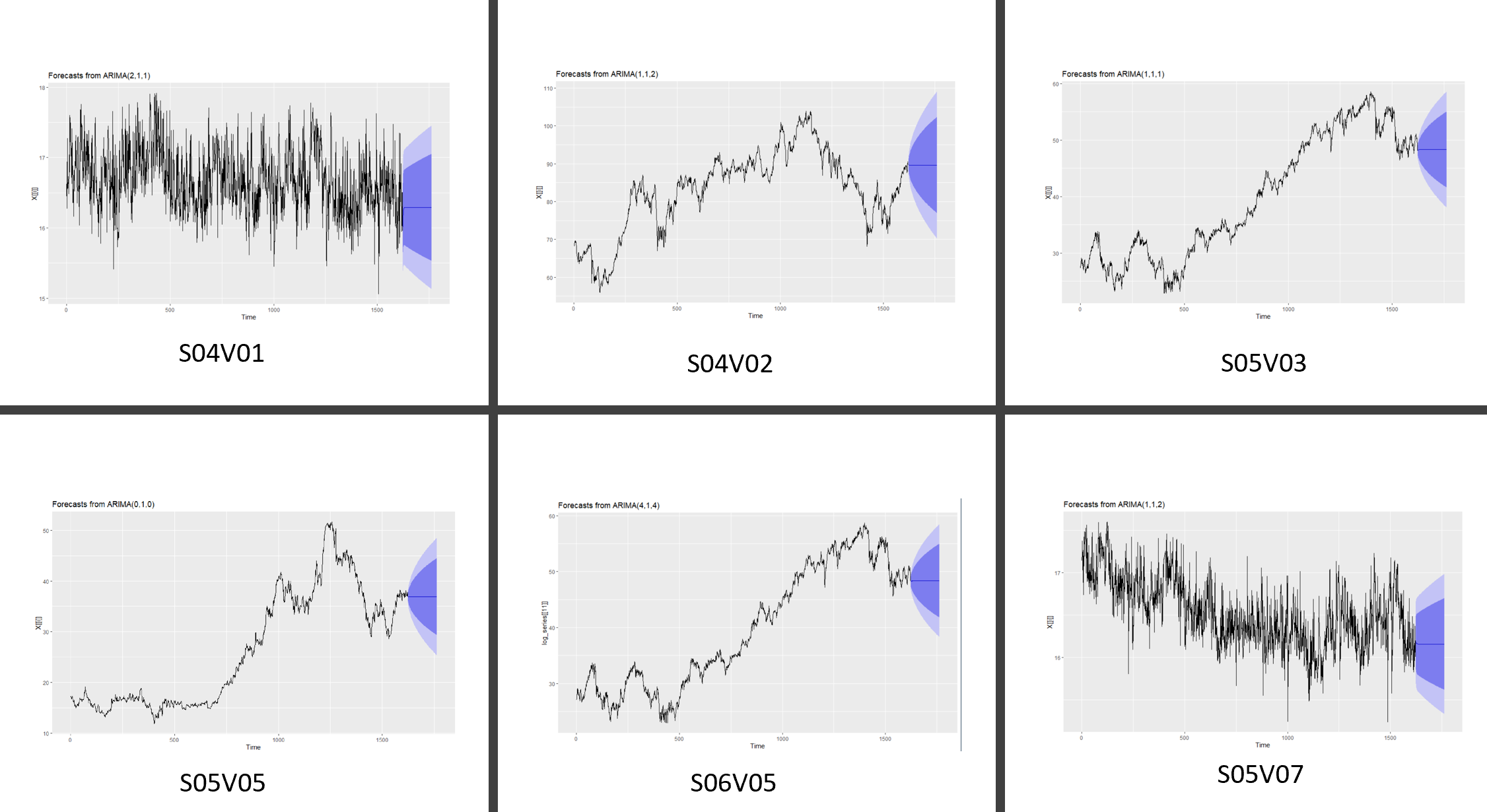
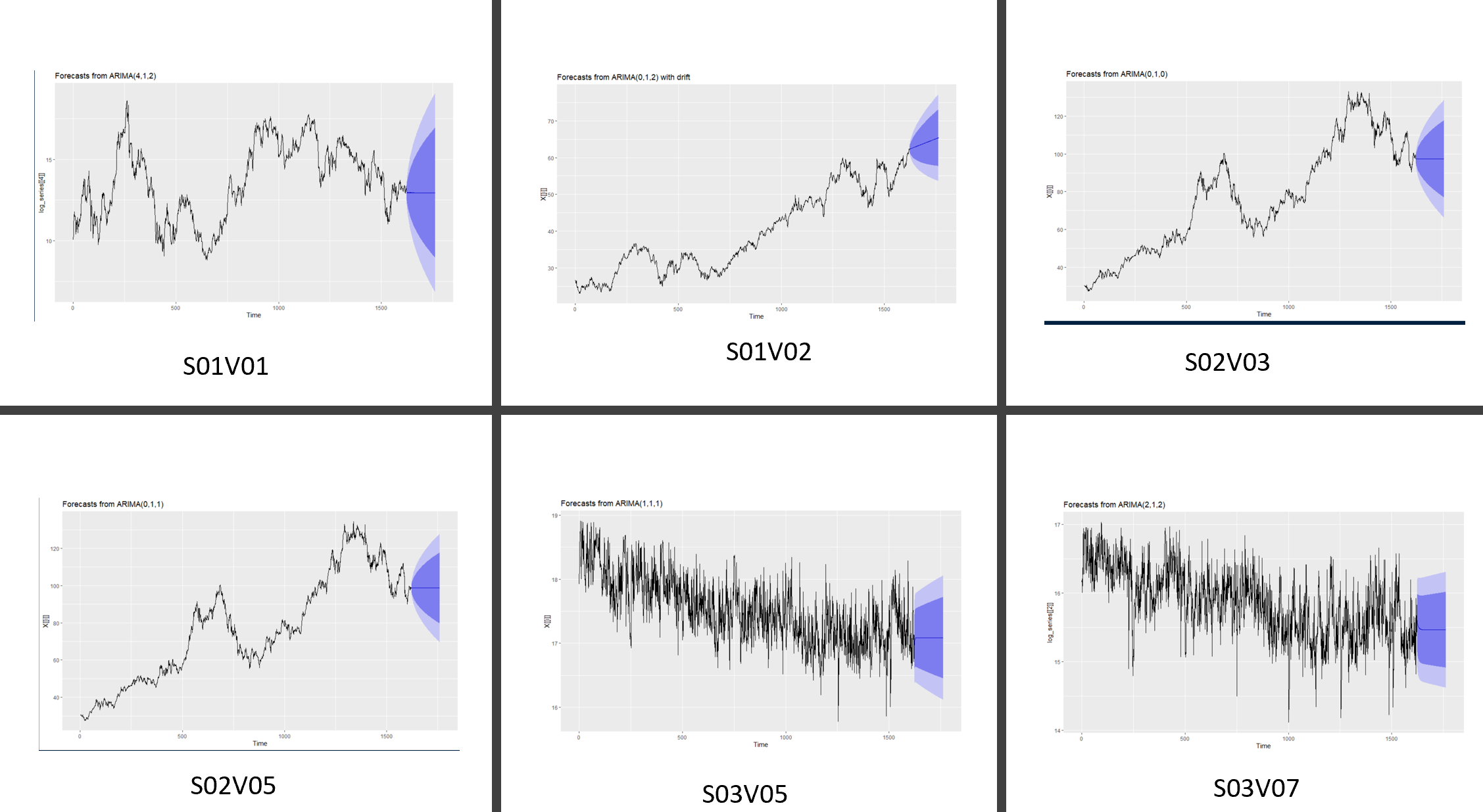
Forecasts were performed on 140 equally spaced time periods for each series using ARIMA models with optimal tuning parameters (Figure 15). The selected models provided errors which can be seen in Figure 19, and sufficient p-values to pass the Ljung Box test proving statistical evidence that any error from the model occured from white noise. It is important to note that the historical data provided for forecasting does not specify the metric of time (days, years, etc) therefore periods for seasonality were difficult to identify. In order to maintain the models’ accuracy and precision only trend and cyclic behaviors were taken into account when forecasting future periods. This resulted in modest projections as seen in Figure 19.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Series** | **S01 Var01** | **S01 Var02** | **S02 Var02** | **S02 Var03** |
| **MAPE** | 0.9037286 | 1.4400239 | 1.3109412 | 1.3263648 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Series** | **S03 Var05** | **S03 Var07** | **S04 Var01** | **S04 Var02** |
| **MAPE** | 1.3068679 | 1.2252978 | 1.2063118 | 1.5993922 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Series** | **S05 Var02** | **S05 Var03** | **S06 Var05** | **S06 Var07** |
| **MAPE** | 1.0557078 | 0.7947361 | 1.1115736 | 1.1379374 |

**Figure 19: MAPE values for all 12 forecasts (training data).**

****

**Figure 20: Plot of cleaned data and 140 Forecasted Periods**

# **Conclusions**

Extract data, explore and understand your data, is there outliers or missing data?, take care of it, perform necessary tests for some of the methods you plan to apply, be sure to split your data into test/train sets so data doesn’t overfit, pick the optima method that perform the best with the minimal error, perform your forecast and make it useful as a decision making tool for your business.

Forecasting modelling is as much an art as science, the methodology described demonstrates the different steps required in order to forecast time series. Hopefully what you notice is that each of them required several nuances to be identified and handled. Sometimes the answer is not clear cut. Ultimately the analyst selects the approach, and as such any result presented here is subject to“analyst bias”. The final user or final consumer like yourself will need to make a decision based on their domain expertise as well as the use of the provided information or data.

# 

# **References**

1. Ortega Cruz, Sergio et all. Group 4 project Github. <https://github.com/sortega7878/DATA624/tree/master/PROJECT1>
2. Kuhn, Maxwell and Johnson Kjell. Applied Predictive Modeling. Springer 2013
3. Hyndman, Rob and Athanasopoulus, George. Forecasting: Principle and Practices: 2nd Edition. OTexts: Melbourne, Australia
4. Hyndman, Rob. MSTL. <https://www.rdocumentation.org/packages/forecast/versions/8.12/topics/mstl>
5. Hyndman, Rob TSoutlier

<https://www.rdocumentation.org/packages/forecast/versions/8.12/topics/tsoutliers>.

1. Hyyndman, Rob. “Measuring Time Series Characteristics”. <https://robjhyndman.com/hyndsight/tscharacteristics/>
2. Burk, Scott. Basic Timeseries in R and RStudio. YouTube. <https://www.youtube.com/playlist?list=PLX-TyAzMwGs-I3i5uiCin37VFMSy4c50F>
3. Buuren, Stef. Flexible Imputation of Missing Data: Second Edition- <https://stefvanbuuren.name/fimd/sec-pmm.html>
4. Predictive Mean Matching Imputation. Statistics Globe <https://statisticsglobe.com/predictive-mean-matching-imputation-method/>
5. Tomáš Cipra, José Trujillo and Asunción Rubio *Management Science* Vol. 41, No. 1 (Jan., 1995), pp. 174-178. <https://www.jstor.org/stable/2632910?seq=1>
6. Stack Overflow: Storing ggplot objects in a loop in R:. <https://stackoverflow.com/questions/31993704/storing-ggplot-objects-in-a-list-from-within-loop-in-r>
7. Stack Overflow: R error “could not find function multiplot” using Cookbook example. <https://stackoverflow.com/questions/24387376/r-error-could-not-find-function-multiplot-using-cookbook-example>
8. Stack Overflow: How to correct for outliers in time-series? <https://stats.stackexchange.com/questions/69874/how-to-correct-outliers-once-detected-for-time-series-data-forecasting>
9. Mean Actual Percent Error. Wikipedia. <https://en.wikipedia.org/wiki/Mean_absolute_percentage_error>

**Codebase Appendix (Github URL)**

The following is the complete code base used for this project which is available online in its entirety at:

<https://github.com/sortega7878/DATA624/blob/master/PROJECT1/project1_final_group4.Rmd>

For those interested in ARIMA optimization algorithm:

*#Series 2, 4, 11 all failed Ljung-Box test*  
*# Lets manually find optimal parameters*  
optim\_AIC <- **function**(series, series\_index){  
 df <- **data.frame**()  
 count <- **c**(0,1,2,3,4,5)  
 **for** (i **in** series\_index){  
 **for**(p **in** count){  
 **for**(d **in** count){  
 **for**(q **in** count){  
 params <- **c**(p,d,q)  
 **try**(aic\_values <- **AIC**(**arima**(log\_series[[i]], order=params)))  
 **try**(series <- i)  
 df <- **rbind**(df,**c**(aic\_values,series,params))  
 }  
 }  
 }  
 }  
 **colnames**(df) <- **c**("AIC", "Series", "P","D","Q")  
 **return**(df)  
}

The above algorithm iterates through different *p*,*d* and *q* parameters, each time recording the corresponding AIC value. The output is a list of parameters and AIC values for each time-series passed through the function. The parameters that correspond with the smallest AIC values are selected to recreate the ARIMA model, and this model is then interpreted once more using the Ljung-Box test.