D213 Task 1 Time Series Modeling

Scott Babcock WGU - MS, Data Analytics

Created: June 05 2025 *Last Edited: September 10 2025*

R Packages Used in Analysis

```r
library(naniar) # missing values
library(tseries) # ADF & KPSS tests
library(TSstudio) # create TS train/test splits
library(forecast) # auto ARIMA function
library(astsa) # ARIMA modeling
library(dplyr) # data manipulation
library(Metrics) # RMSE for model evaluation
```

A1, Research Question

The research question the analysis will aim to answer is: Can patterns be identified and future revenues predicted using time-series analysis?

A2, Analysis Objectives

The primary objective of the analysis is to identify patterns within the data using time-series analysis and utilize them to predict future revenue for the company.

B1, Assumptions

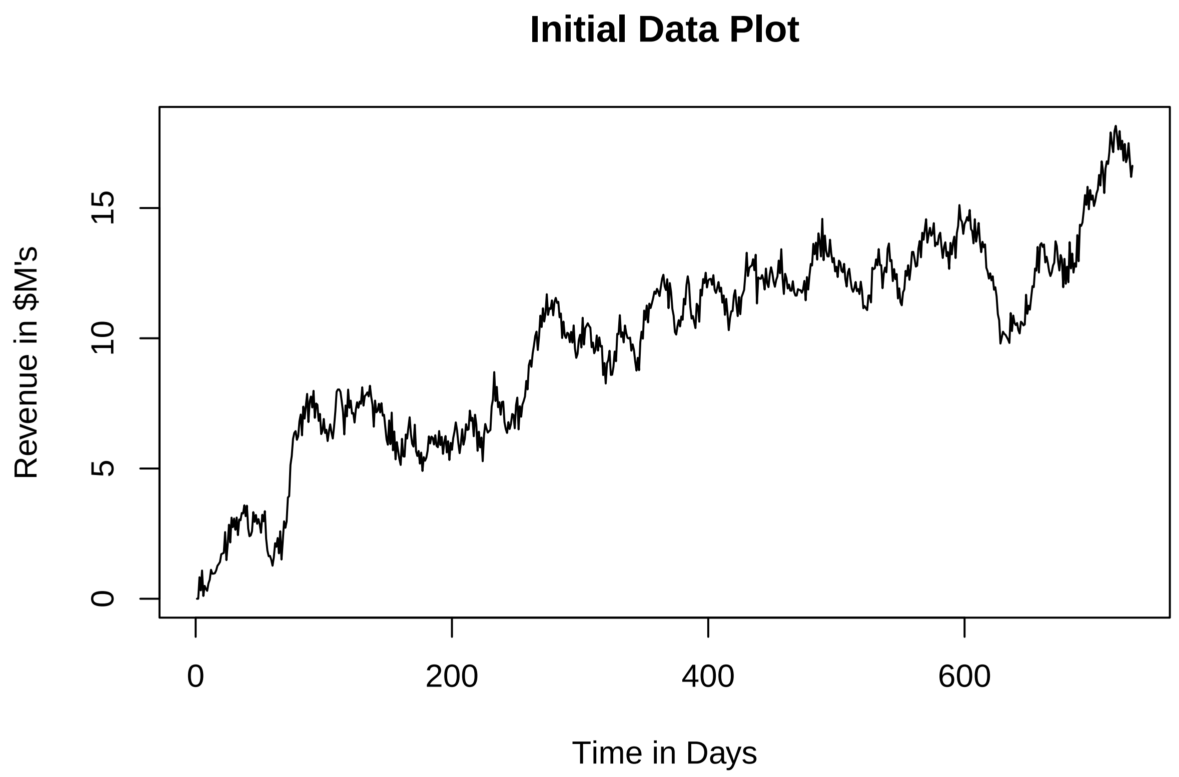
There are several key assumptions in time-series analysis. The first is the concept of **stationarity**. The data must be stationary to perform time-series analysis, meaning the mean and variance are consistent over time, and the trend is removed. To achieve this, the data must often be differenced. Stationarity can be tested using the Augmented Dickey-Fuller *(ADF)* and Kwiatkowski-Phillips-Schmidt-Shin *(KPSS)* functions. Another key assumption of time-series analysis is **autocorrelation**. In simple terms, autocorrelation refers to patterns in the data, such as whether past behavior can influence future behavior. Identifying autocorrelation is key for predicting future values. Higher autocorrelation will make future values more predictable. Finally, the data must be in **chronological order** to properly assess trends and patterns. Each observation must occur following the previous observation (Wainaina, 2023).

C1, Time Series Plot

Below is a line plot showing the initial data with revenue on the y-axis and day number on the x-axis.

```r
data <- read.csv("teleco_time_series.csv")

plot(data, type = "l",
     xlab = "Time in Days",
     ylab = "Revenue in $M's",
     main = "Initial Data Plot")
```



C2, Time Step Formatting

The following steps were applied to ensure the time-series is in the proper format.

Missing Values
The data was first checked for missing values. No missing values were found.

```
miss_var_summary(data)
```

```
## # A tibble: 2 x 3
##   variable n_miss pct_miss
##   <chr>     <int>    <num>
## 1 Day           0        0
## 2 Revenue       0        0
```

Duplication
The data was checked for duplicate records. There were no duplicate records identified.

```
sum(duplicated(data))
```

```
## [1] 0
```

Chronological Order
The data was checked to ensure day numbers coincided with row numbers, ensuring proper order. The data was deemed to be in chronological order. There are 731 days of revenue being tracked.

```r
head(data, 10)
```

```
##    Day     Revenue
## 1    1 0.000000000
## 2    2 0.000793191
## 3    3 0.825541786
## 4    4 0.320332280
## 5    5 1.082554085
## 6    6 0.107653784
## 7    7 0.493901361
## 8    8 0.376698452
## 9    9 0.304074848
## 10  10 0.591747849
```

```r
tail(data, 10)
```

```
##     Day  Revenue
## 722 722 17.25593
## 723 723 17.58363
## 724 724 16.82176
## 725 725 17.45906
## 726 726 16.76128
## 727 727 16.93156
## 728 728 17.49067
## 729 729 16.80364
## 730 730 16.19481
## 731 731 16.62080
```

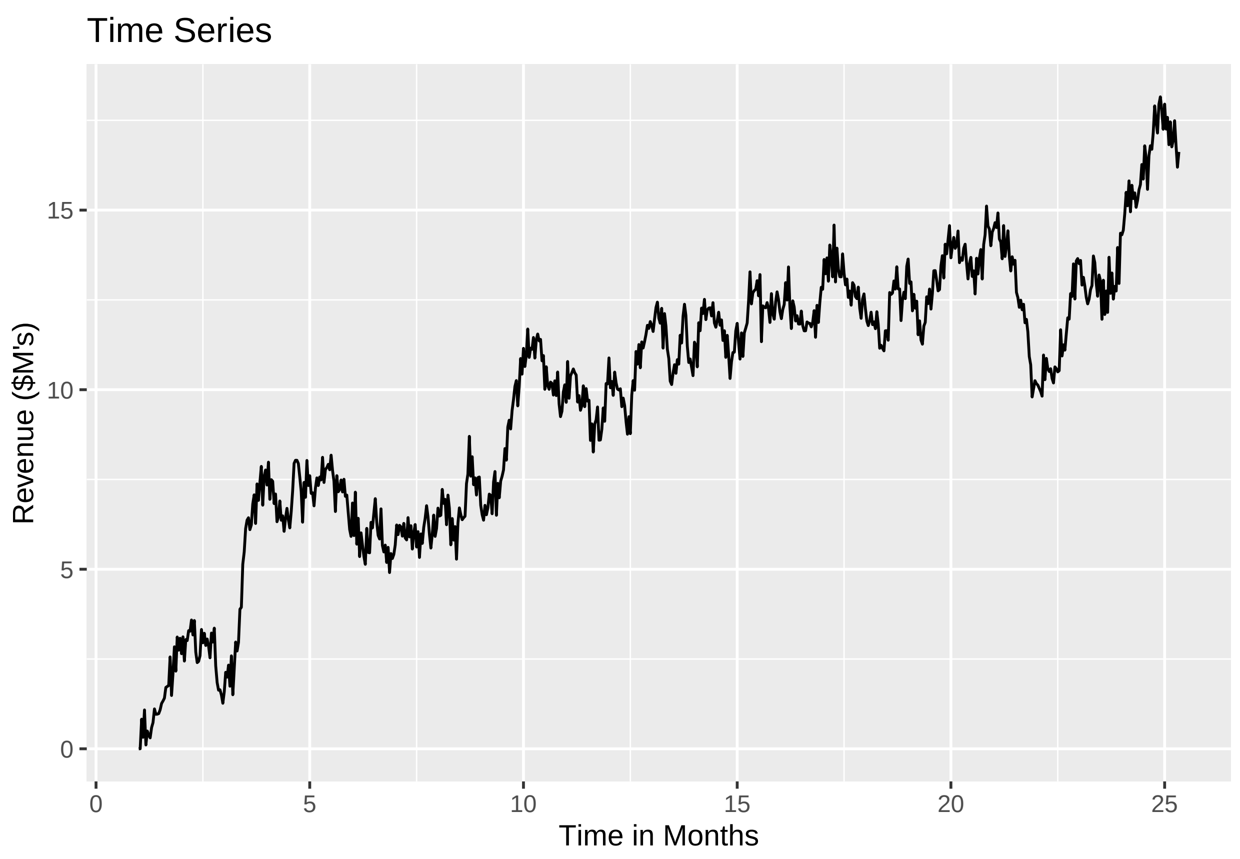
Time Series Format The data frame was converted to a time-series object with a frequency of 30, signifying that the data will be looked at in terms of monthly cycles.

```r
ts_data <- ts(data$Revenue, frequency = 30)

is.ts(ts_data) #confirm time series object
```

```
## [1] TRUE
```

```r
autoplot(ts_data,
         main = "Time Series",
         xlab = "Time in Months",
         ylab = "Revenue ($M's)")
```



C3, Evaluating Stationarity

The Augmented Dickey-Fuller *(ADF)* and Kwiatkowski-Phillips-Schmidt-Shin *(KPSS)* tests were used to evaluate whether the data was stationary.
In the ADF test, if the p-value is less than or equal to 0.05, the null hypothesis is rejected, and the data is deemed stationary. The KPSS test is the opposite. If the resulting p-value in the KPSS test is less than or equal to 0.05, the null hypothesis is rejected, but it signals that the data is not stationary (Vijay, 2025).

```
adf.test(ts_data)
```

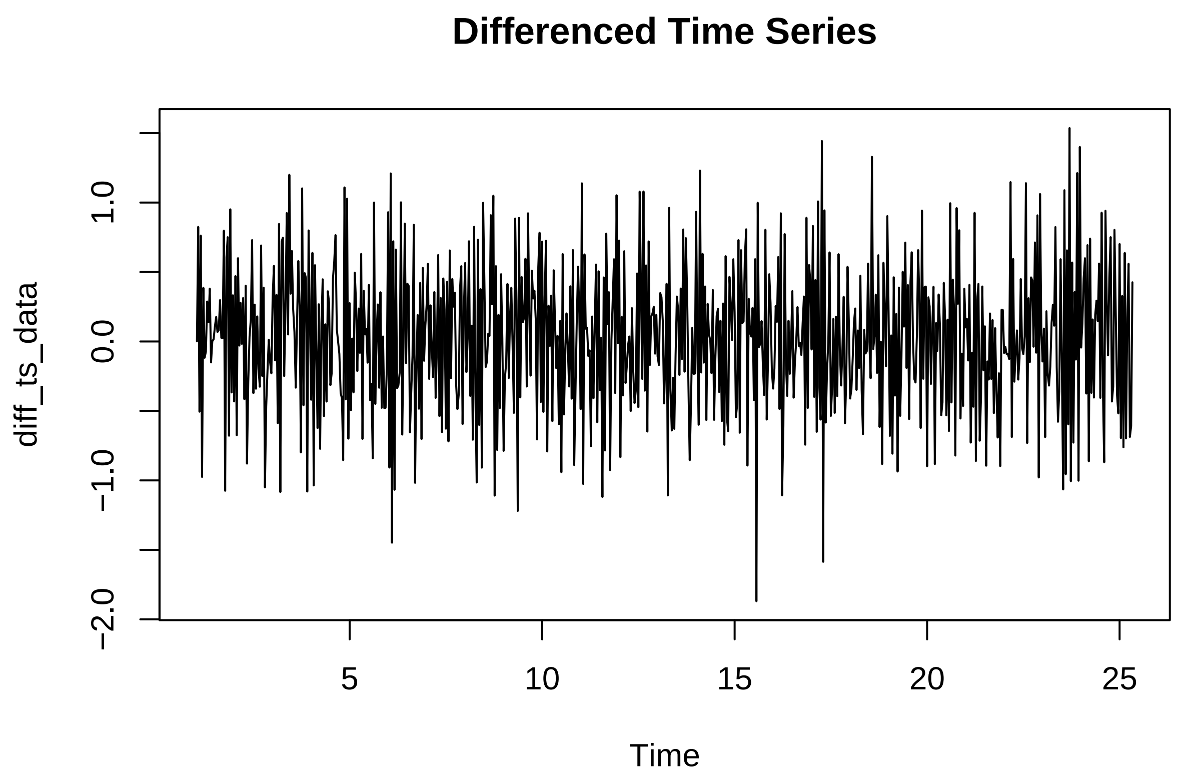
```
##
##  Augmented Dickey-Fuller Test
##
## data:  ts_data
## Dickey-Fuller = -3.6938, Lag order = 9, p-value = 0.02431
## alternative hypothesis: stationary
```

```
kpss.test(ts_data)
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  ts_data
## KPSS Level = 8.7827, Truncation lag parameter = 6, p-value = 0.01
```

There are conflicting results after running the two tests on the initial time-series data. The ADF test returns a p-value of 0.024, which suggests that the data is stationary. However, the KPSS test also returns a p-value of 0.01, which indicates that the data is not stationary. This means that the data is partially stationary and should be differenced. A new time-series vector is created using the `diff()` function to difference the data. When the differenced data is plotted, the expected result would be to see the mean hovering around zero.

```
diff_ts_data <- diff(ts_data)
ts.plot(diff_ts_data,
        gpars = list(
          main = "Differenced Time Series"
        )
)
```



```
adf.test(diff_ts_data)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff_ts_data
## Dickey-Fuller = -8.6354, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

```
kpss.test(diff_ts_data)
```

```
##
```

```
##  KPSS Test for Level Stationarity
##
## data:  diff_ts_data
## KPSS Level = 0.069457, Truncation lag parameter = 6, p-value = 0.1
```

The resulting plot looks stationary since the mean of the data is hovering around zero. The ADF and KPSS tests back this up. The new p-value from the ADF test is 0.01, which means the data is stationary. The KPSS p-value is now 0.1, which means the data is stationary.

C4, Steps for Data Prep

Many steps were taken to prepare the data for analysis. In previous sections, the data was checked for missing values, duplicate records, and whether it was in chronological order. The data was then converted to a time-series object and evaluated for stationarity. The final step before creating the ARIMA model(s) is to split the data into training and test sets. No random sample was taken in this time-series analysis. The training set will be the first 671 days, and the test set will be the final 60 days or two months.

```r
ts_split <- ts_split(ts_data, sample.out = 60)
train <- ts_split$train
test <- ts_split$test

length(train)
```

```
## [1] 671
```

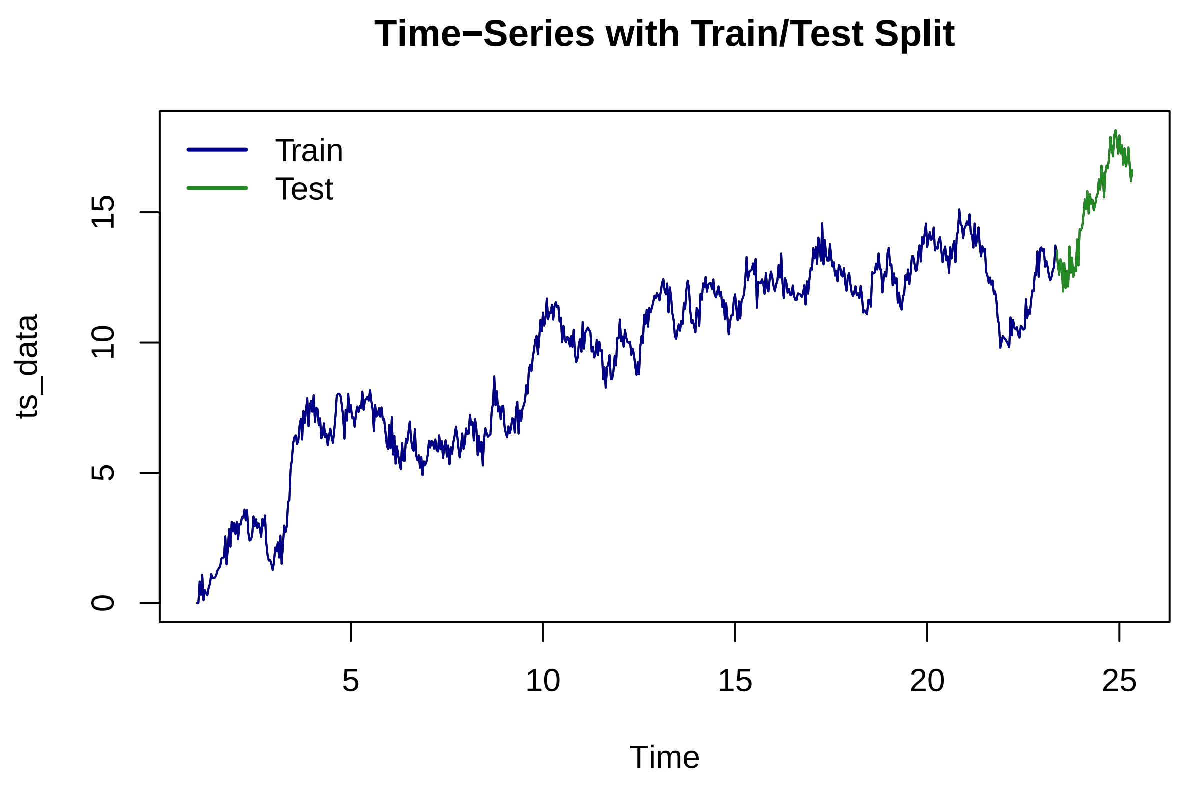
```r
length(test)
```

```
## [1] 60
```

```r
length(train)+length(test)
```

```
## [1] 731
```

```r
plot(ts_data, main = "Time-Series with Train/Test Split")
lines(train, col = "darkblue")
lines(test, col = "forestgreen")
legend("topleft",
       legend = c("Train","Test"),
       col = c("darkblue","forestgreen"),
       lty = 1,
       lwd = 2,
       bty = "n")
```



C5, Cleaned Data Set

The cleaned data, along with training and test sets were exported to CSV and included in the submission.

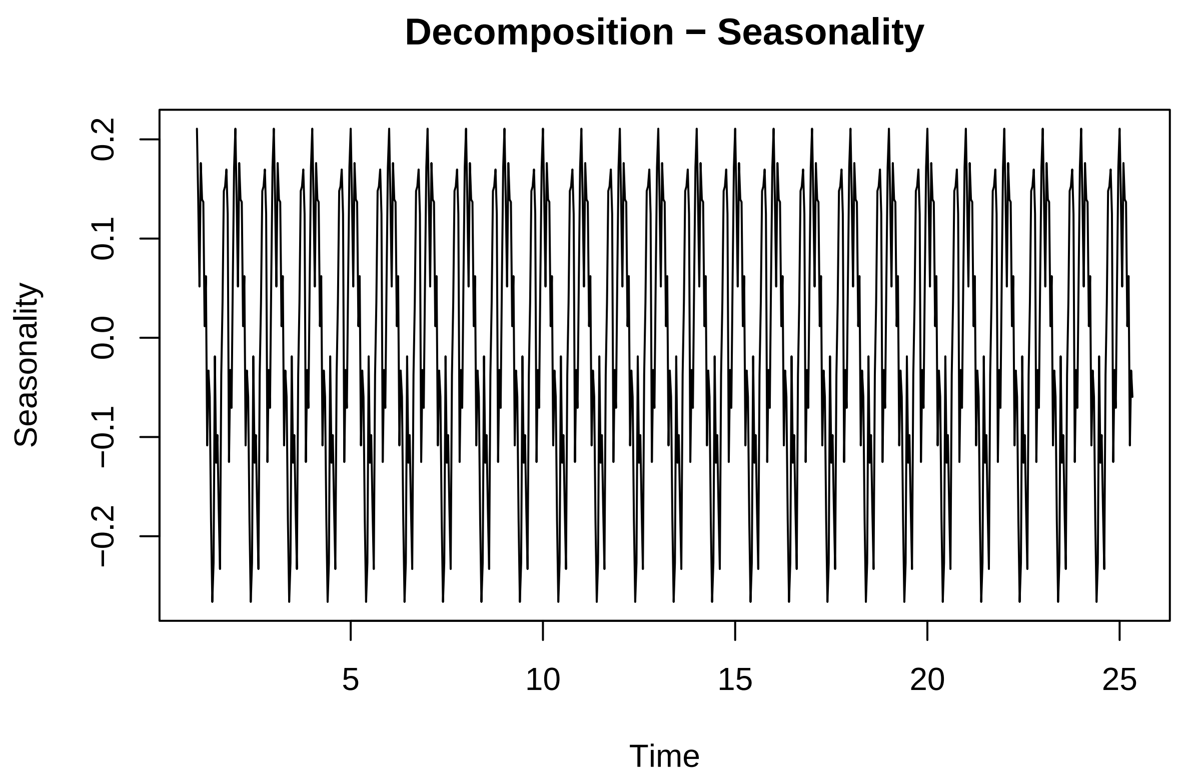
```
write.csv(ts_data,
          "d213_task1_babcock_cleaned_data.csv",
          row.names = FALSE)
write.csv(train,
          "d213_task1_babcock_training_data.csv",
          row.names = FALSE)
write.csv(test,
          "d213_task1_babcock_test_data.csv",
          row.names = FALSE)
```

D1, Findings and Visualizations

Seasonal Component
The seasonality of the data was checked as part of the time-series decomposition. The data appears to have monthly patterns, given the peaks and valleys at each interval.

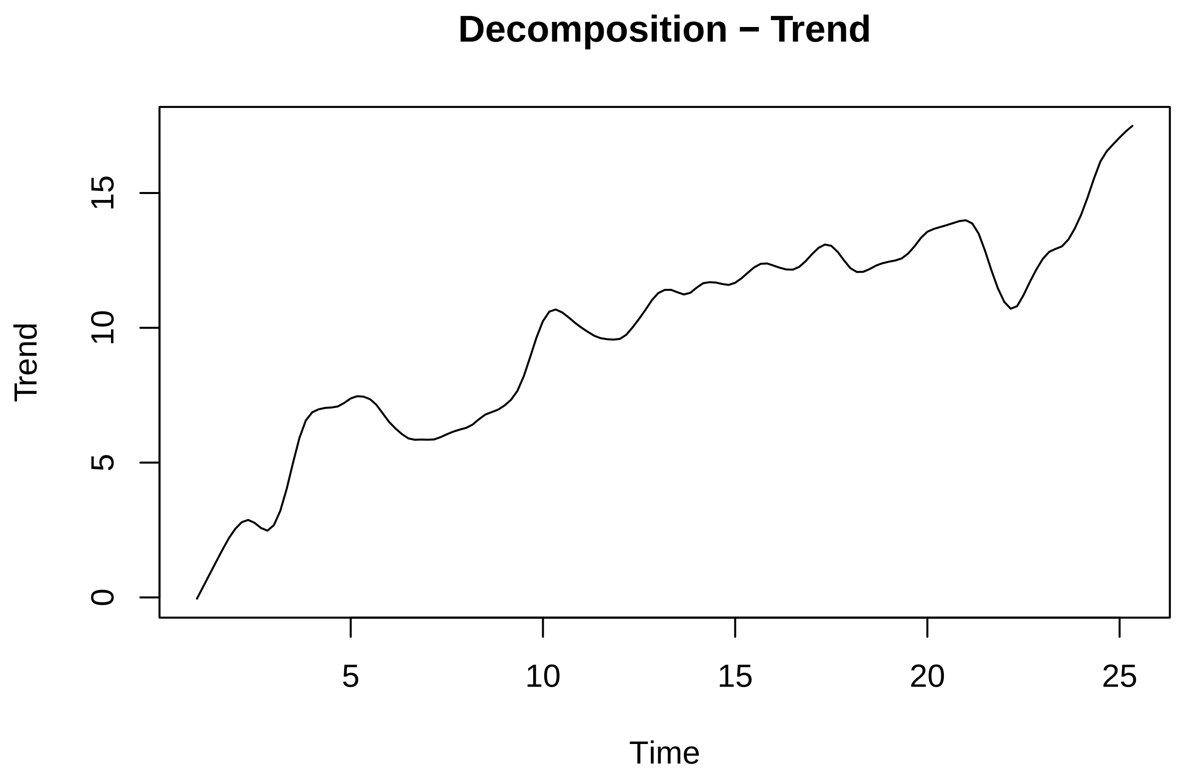
```
decomp <- stl(ts_data,s.window = "periodic")

plot(decomp$time.series[,"seasonal"],
     main = "Decomposition - Seasonality",
     xlab = "Time",
     ylab = "Seasonality")
```



Trends
The trend is also analyzed as part of the time-series decomposition. There is a general upward trend as time goes on. It seems to be growing at a steady pace overall.

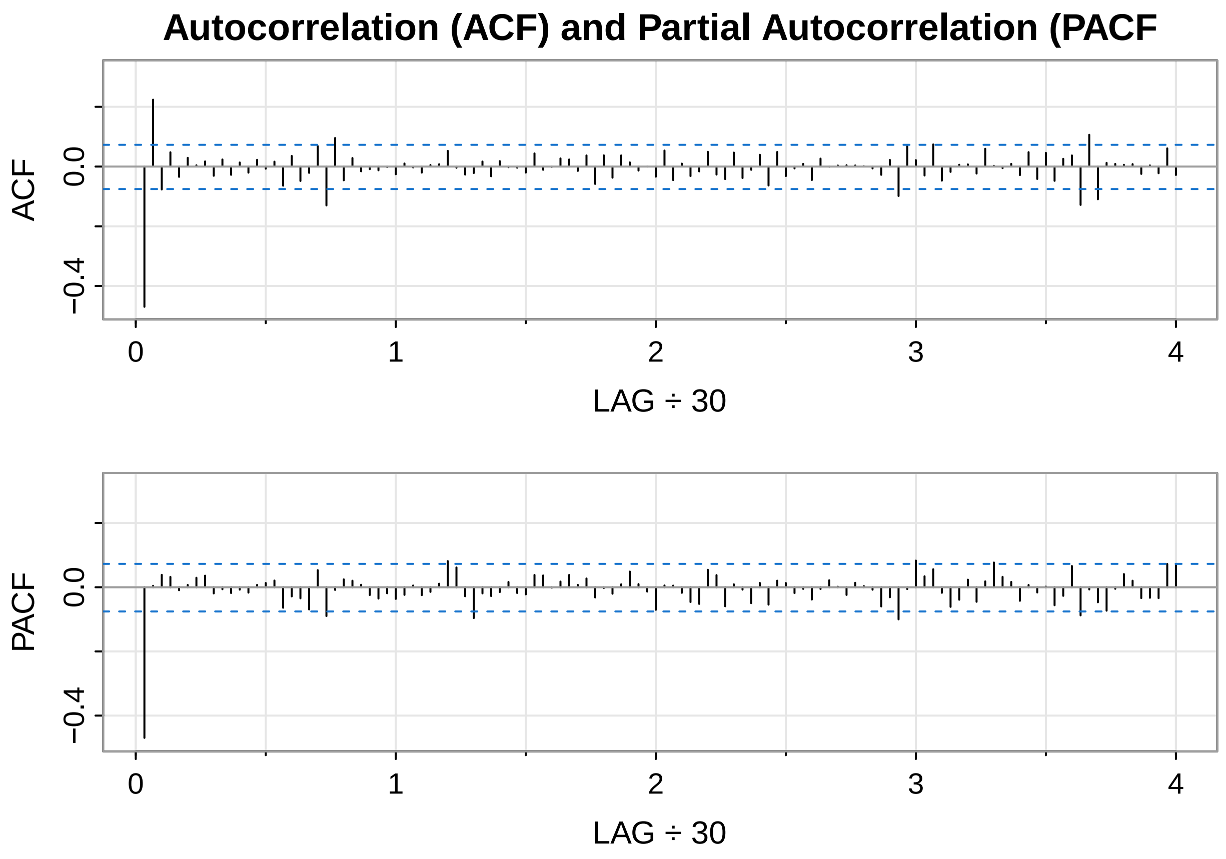
```
plot(decomp$time.series[,"trend"],
     main = "Decomposition - Trend",
     xlab = "Time",
     ylab = "Trend")
```



Autocorrelation Function

Autocorrelation (ACF) and partial autocorrelation (PACF) were examined after differencing the data. The ACF plot has a spike at lag one and a tails off thereafter. The plot indicates an AR(1) component. If the plot were interpreted as cutting off instead of tailing off, it would indicate an MA(1) component (Stoffer, n.d.). In the PACF plot, there is again a spike at lag 1 with all remaining values within the confidence bounds. The plot suggests a potential AR(1) component (Stoffer, n.d.). After looking at these plots, two models that should be considered are ARIMA(0,1,1) and ARIMA(1,1,0).

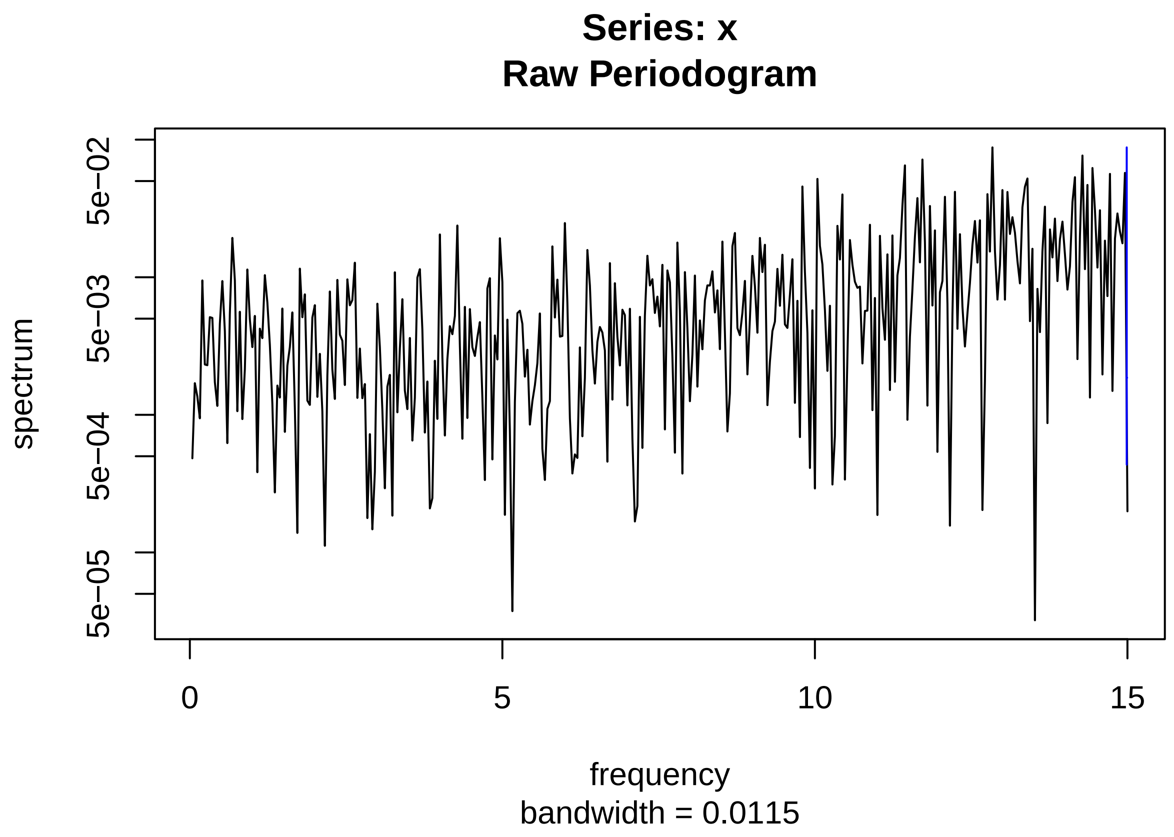
```
acf2(diff_ts_data,
     main = "Autocorrelation (ACF) and Partial Autocorrelation (PACF")
```



```
##       [,1] [,2]  [,3] [,4]  [,5] [,6] [,7] [,8]  [,9] [,10] [,11] [,12] [,13]
## ACF  -0.47 0.22 -0.08 0.05 -0.03 0.03 0.00 0.02 -0.03  0.02 -0.03  0.01 -0.02
## PACF -0.47 0.00  0.04 0.03 -0.01 0.01 0.03 0.04 -0.02 -0.01 -0.02 -0.01 -0.02
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25]
## ACF   0.02 -0.01  0.02 -0.06  0.04 -0.05 -0.02  0.07 -0.13  0.10 -0.05  0.03
## PACF  0.01  0.01  0.02 -0.06 -0.03 -0.03 -0.07  0.05 -0.09 -0.01  0.02  0.02
##      [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36] [,37]
## ACF  -0.02 -0.01 -0.01  0.00 -0.03  0.01  0.00 -0.02  0.01  0.01  0.05  0.00
## PACF  0.01 -0.02 -0.04 -0.02 -0.04 -0.02  0.01 -0.02 -0.01  0.01  0.08  0.06
##      [,38] [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46] [,47] [,48] [,49]
## ACF  -0.03 -0.02  0.02 -0.03  0.02  0.00  0.00 -0.02  0.04 -0.01     0  0.03
## PACF -0.03 -0.10 -0.02 -0.03 -0.02  0.02 -0.02 -0.02  0.04  0.04     0  0.02
##      [,50] [,51] [,52] [,53] [,54] [,55] [,56] [,57] [,58] [,59] [,60] [,61]
## ACF   0.02 -0.01  0.04 -0.06  0.04 -0.04  0.04  0.01 -0.01  0.00 -0.03  0.05
## PACF  0.04  0.01  0.03 -0.03  0.00 -0.02  0.01  0.05  0.01 -0.01 -0.07  0.01
##      [,62] [,63] [,64] [,65] [,66] [,67] [,68] [,69] [,70] [,71] [,72] [,73]
## ACF  -0.05  0.01 -0.03 -0.02  0.05 -0.03 -0.04  0.05 -0.04 -0.01  0.04 -0.06
## PACF  0.01 -0.02 -0.05 -0.05  0.05  0.04 -0.06  0.01 -0.01 -0.05  0.01 -0.05
##      [,74] [,75] [,76] [,77] [,78] [,79] [,80] [,81] [,82] [,83] [,84] [,85]
## ACF   0.05 -0.03 -0.01  0.01 -0.05  0.03  0.00     0  0.00  0.00     0 -0.01
## PACF  0.02  0.01 -0.02 -0.01 -0.04 -0.01  0.02     0 -0.02  0.01     0 -0.01
##      [,86] [,87] [,88] [,89] [,90] [,91] [,92] [,93] [,94] [,95] [,96] [,97]
## ACF  -0.03  0.02  -0.1  0.07  0.02 -0.03  0.07 -0.05 -0.02  0.01  0.01 -0.02
## PACF -0.06 -0.03  -0.1 -0.01  0.08  0.03  0.06 -0.02 -0.06 -0.04  0.02 -0.05
##      [,98] [,99] [,100] [,101] [,102] [,103] [,104] [,105] [,106] [,107] [,108]
## ACF   0.06  0.00  -0.01   0.01  -0.03   0.05  -0.04   0.05  -0.05   0.03   0.04
## PACF  0.02  0.08   0.03   0.02  -0.04   0.01  -0.02   0.00  -0.06  -0.03   0.07
##      [,109] [,110] [,111] [,112] [,113] [,114] [,115] [,116] [,117] [,118]
## ACF   -0.13   0.11  -0.11   0.01   0.01   0.01   0.01  -0.02   0.00  -0.02
## PACF  -0.09  -0.01  -0.05  -0.07   0.00   0.04   0.02  -0.03  -0.03  -0.03
##      [,119] [,120]
## ACF    0.06  -0.03
## PACF   0.07   0.07
```

Spectral Density
The spectral density plot of the differenced data is relatively flat across all frequencies. This plot indicates that the data is stationary and has no strong seasonal patterns. When testing ARIMA models, seasonal components can be excluded (D213 ARIMA II, n.d.).

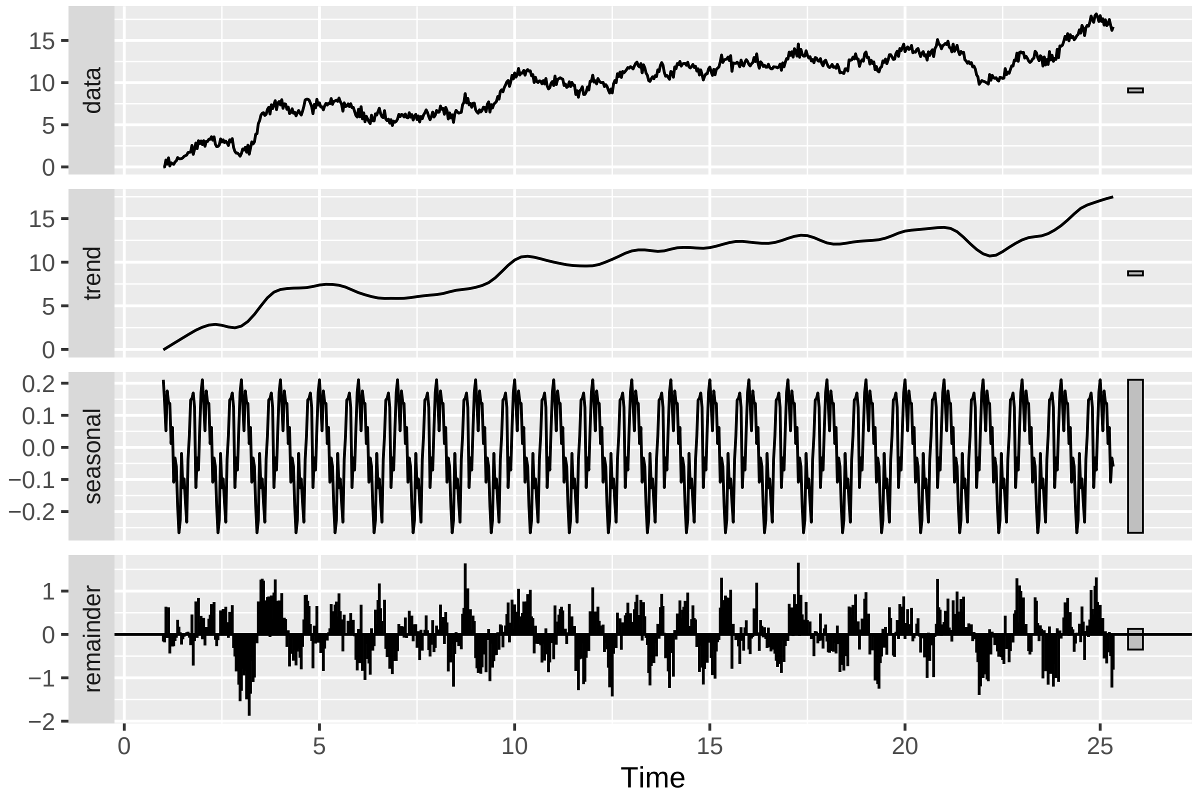
```
spectrum(diff_ts_data)
```



Decomposed Time Series
The entirety of the decomposed time-series is shown below. As previously discussed, the data shows an upward trend and monthly patterns appear. The residuals appear to be dispersed randomly around zero.

```
autoplot(decomp,
         main = "Decomposed Time Series")
```

Decomposed Time Series



D2, ARIMA Model

Based on the ACF and PACF plots, two possible models to consider were ARIMA(1,1,0) and ARIMA(0,1,1). The spectral density of the differenced data suggested that there was no strong seasonality in the data, so the seasonal component will not be considered in the analysis. Three additional models will be considered: ARIMA(2,1,0), ARIMA(0,1,2), and ARIMA(1,1,1). The training data will be used to fit the model. AIC scores will be used to determine the best model fit. The AIC scores will be saved to a data frame for comparison.

```
arima_110 <- sarima(train,
                    1,1,0,
                    details = FALSE)
```

```
## <><><><><><><><><><><><><><>
##
## Coefficients:
##          Estimate     SE  t.value p.value
## ar1       -0.4540 0.0345 -13.1725  0.0000
## constant   0.0201 0.0124   1.6242  0.1048
##
## sigma^2 estimated as 0.2171175 on 668 degrees of freedom
##
## AIC = 1.31986  AICc = 1.319887  BIC = 1.340042
##
```

```
arima_011 <- sarima(train,
                        0,1,1,
                        details = FALSE)
```

```
## <><><><><><><><><><><><><><>
##
## Coefficients:
##          Estimate     SE  t.value p.value
## ma1       -0.3767 0.0304 -12.3836  0.0000
## constant   0.0199 0.0115   1.7282  0.0844
##
## sigma^2 estimated as 0.227476 on 668 degrees of freedom
##
## AIC = 1.36635  AICc = 1.366377  BIC = 1.386532
##
```

```
arima_210 <- sarima(train,
                    2,1,0,
                    details = FALSE)
```

```
## <><><><><><><><><><><><><><>
##
## Coefficients:
##          Estimate     SE  t.value p.value
## ar1       -0.4474 0.0387 -11.5545  0.0000
## ar2        0.0144 0.0388   0.3725  0.7096
## constant   0.0201 0.0126   1.6029  0.1094
##
## sigma^2 estimated as 0.2170724 on 667 degrees of freedom
##
## AIC = 1.322638  AICc = 1.322692  BIC = 1.349547
##
```

```
arima_012 <- sarima(train,
                        0,1,2,
                        details = FALSE)
```

```
## <><><><><><><><><><><><><><>
##
## Coefficients:
##          Estimate     SE  t.value p.value
## ma1       -0.4339 0.0374 -11.6021  0.0000
## ma2        0.2060 0.0367   5.6207  0.0000
## constant   0.0204 0.0139   1.4646  0.1435
##
## sigma^2 estimated as 0.2174992 on 667 degrees of freedom
##
## AIC = 1.324594  AICc = 1.324647  BIC = 1.351503
##
```

```
arima_111 <- sarima(train,
                    1,1,1,
                    details = FALSE)
```

```
## <><><><><><><><><><><><><><>
##
## Coefficients:
##          Estimate     SE t.value p.value
## ar1       -0.4745 0.0687 -6.9023  0.0000
## ma1        0.0258 0.0762  0.3385  0.7351
## constant   0.0201 0.0125  1.6076  0.1084
##
## sigma^2 estimated as 0.2170815 on 667 degrees of freedom
##
## AIC = 1.32268  AICc = 1.322734  BIC = 1.349589
##
```

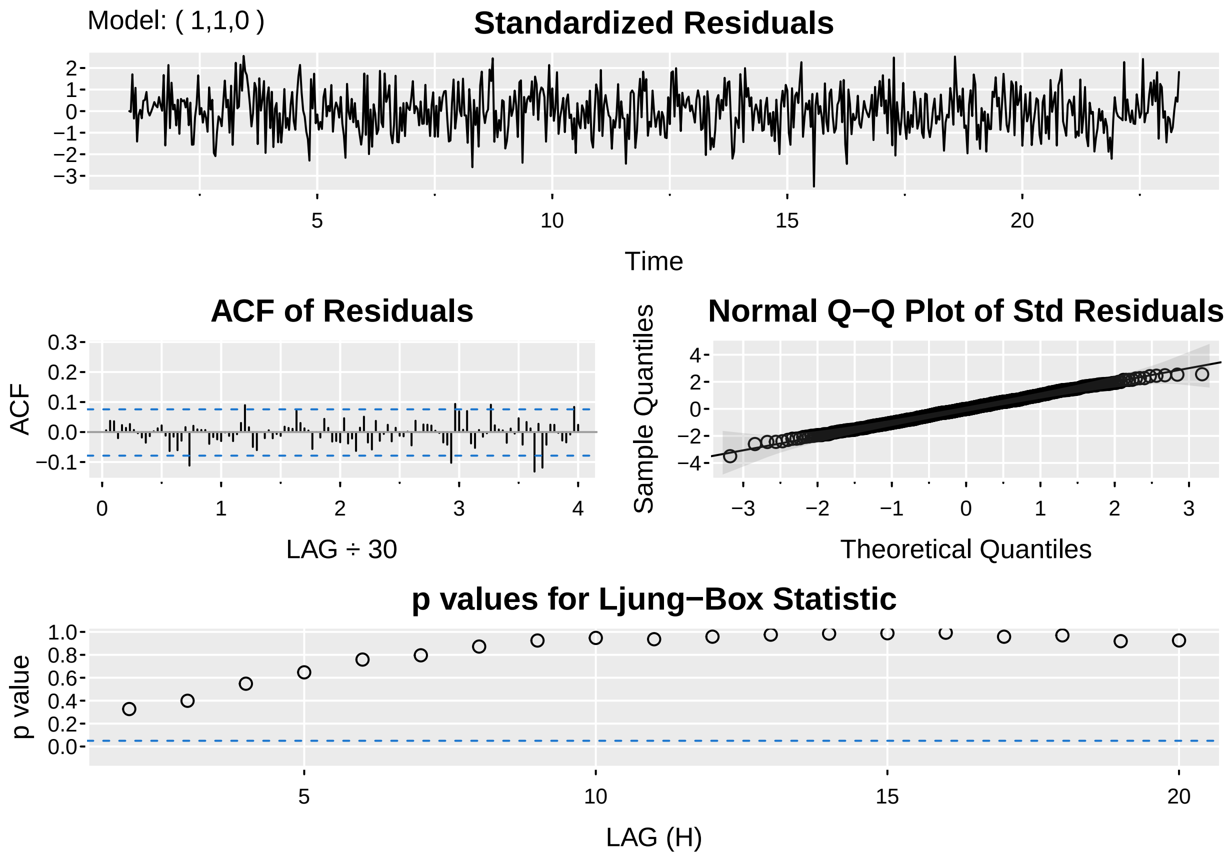
```
AIC_scores <- data.frame(
  Model = c("ARIMA (1,1,0)", "ARIMA (0,1,1)", "ARIMA (2,1,0)", "ARIMA (0,1,2)", "ARIMA (1,1,1)"),
  AIC = c(arima_110$ICs["AIC"], arima_011$ICs["AIC"], arima_210$ICs["AIC"], arima_012$ICs["AIC"],arima_
)
print(AIC_scores)
```

```
##           Model      AIC
## 1 ARIMA (1,1,0) 1.319860
## 2 ARIMA (0,1,1) 1.366350
## 3 ARIMA (2,1,0) 1.322638
## 4 ARIMA (0,1,2) 1.324594
## 5 ARIMA (1,1,1) 1.322680
```

The ARIMA(1,1,0) model produced the lowest AIC score. The `sarima()` function from the `astsa` package provides nice model summary visuals, which can be seen below. The residuals appear random and are hovered around zero. The ACF of the residuals is primarily within bounds. There is a tight fit with no significant deviations on the Q-Q plot. All of the lags have p-values above 0.05 in the Ljung-Box plot. These are all indicators that the model is a good fit (Stoffer, n.d.).

```
final_arima <- sarima(train,
                      1,1,0,
                      gg=TRUE)
```

```
## initial  value -0.647595
## iter   2 value -0.762911
## iter   3 value -0.762912
## iter   4 value -0.762913
## iter   5 value -0.762913
## iter   5 value -0.762913
## iter   5 value -0.762913
## final  value -0.762913
## converged
## initial  value -0.763486
## iter   2 value -0.763486
## iter   3 value -0.763486
## iter   3 value -0.763486
## iter   3 value -0.763486
## final  value -0.763486
## converged
## <><><><><><><><><><><><><><>
##
## Coefficients:
##          Estimate     SE  t.value p.value
## ar1       -0.4540 0.0345 -13.1725  0.0000
## constant   0.0201 0.0124   1.6242  0.1048
##
## sigma^2 estimated as 0.2171175 on 668 degrees of freedom
##
## AIC = 1.31986  AICc = 1.319887  BIC = 1.340042
##
```



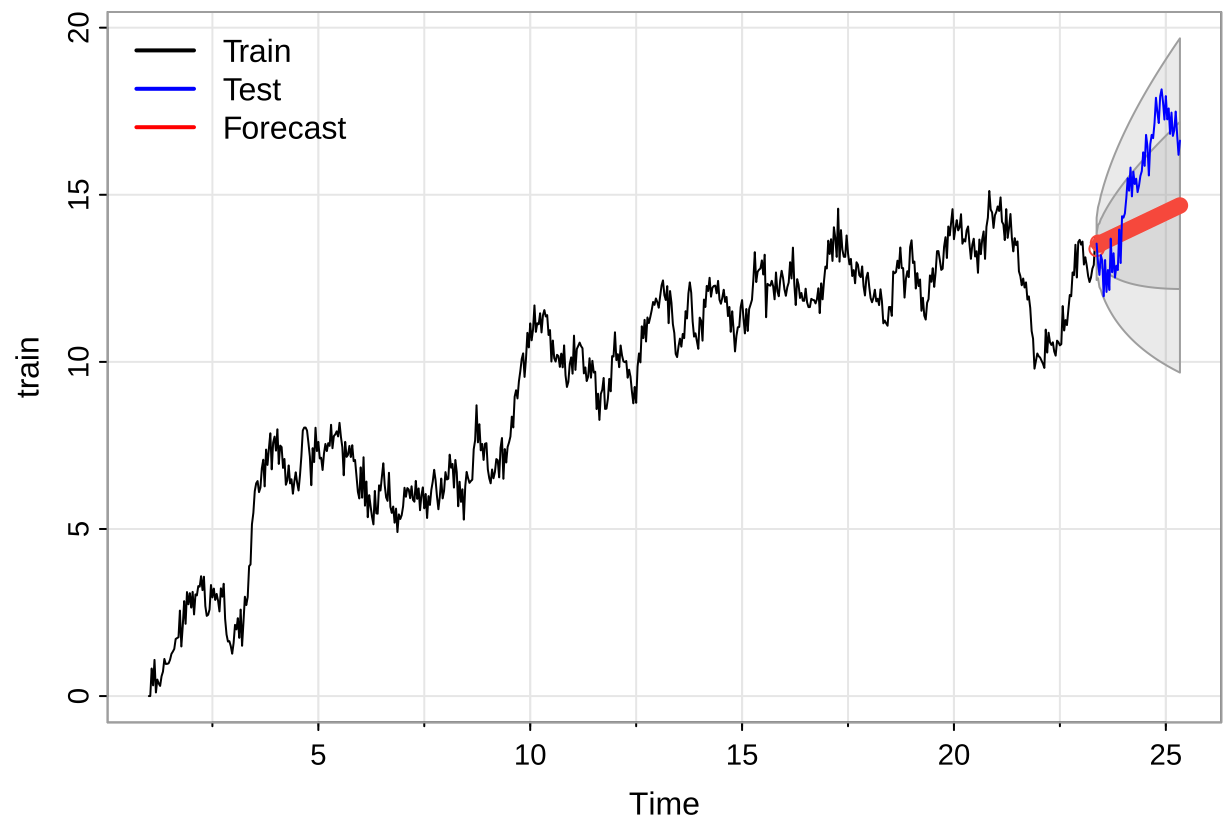
D3, Forecasting

Now that the ARIMA(1,1,0) model has been fit on the training data, predictions will be made 60 days out to compare to the unseen test data.

```
# forecast 60 days using the train data and compare to test set
sarima.for(train,
           n.ahead = 60,
           1,1,0,
           plot.all = TRUE)
```

```
## $pred
## Time Series:
## Start = c(23, 12)
## End = c(25, 11)
## Frequency = 30
##  [1] 13.37954 13.56526 13.51020 13.56445 13.56907 13.59623 13.61315 13.63472
##  [9] 13.65418 13.67460 13.69458 13.71476 13.73485 13.75499 13.77510 13.79522
## [17] 13.81534 13.83546 13.85558 13.87570 13.89582 13.91593 13.93605 13.95617
## [25] 13.97629 13.99641 14.01653 14.03665 14.05677 14.07689 14.09701 14.11713
## [33] 14.13724 14.15736 14.17748 14.19760 14.21772 14.23784 14.25796 14.27808
## [41] 14.29820 14.31832 14.33843 14.35855 14.37867 14.39879 14.41891 14.43903
## [49] 14.45915 14.47927 14.49939 14.51951 14.53962 14.55974 14.57986 14.59998
## [57] 14.62010 14.64022 14.66034 14.68046
##
## $se
## Time Series:
## Start = c(23, 12)
## End = c(25, 11)
## Frequency = 30
##  [1] 0.4659587 0.5308976 0.6361384 0.7062844 0.7781646 0.8405084 0.8999868
##  [8] 0.9551492 1.0075626 1.0572654 1.1047846 1.1503205 1.1941308 1.2363856
## [15] 1.2772450 1.3168365 1.3552722 1.3926473 1.4290454 1.4645391 1.4991927
## [22] 1.5330633 1.5662015 1.5986529 1.6304586 1.6616556 1.6922776 1.7223552
## [29] 1.7519165 1.7809872 1.8095910 1.8377496 1.8654833 1.8928106 1.9197489
## [36] 1.9463145 1.9725223 1.9983864 2.0239200 2.0491355 2.0740445 2.0986578
## [43] 2.1229858 2.1470381 2.1708240 2.1943520 2.2176305 2.2406671 2.2634692
## [50] 2.2860440 2.3083980 2.3305375 2.3524688 2.3741974 2.3957290 2.4170688
## [57] 2.4382218 2.4591929 2.4799867 2.5006075
```

```
lines(test, col = "blue")
legend("topleft",
       legend = c("Train","Test","Forecast"),
       col = c("black","blue","red"),
       lty = 1,
       lwd = 2,
       bty = "n")
```



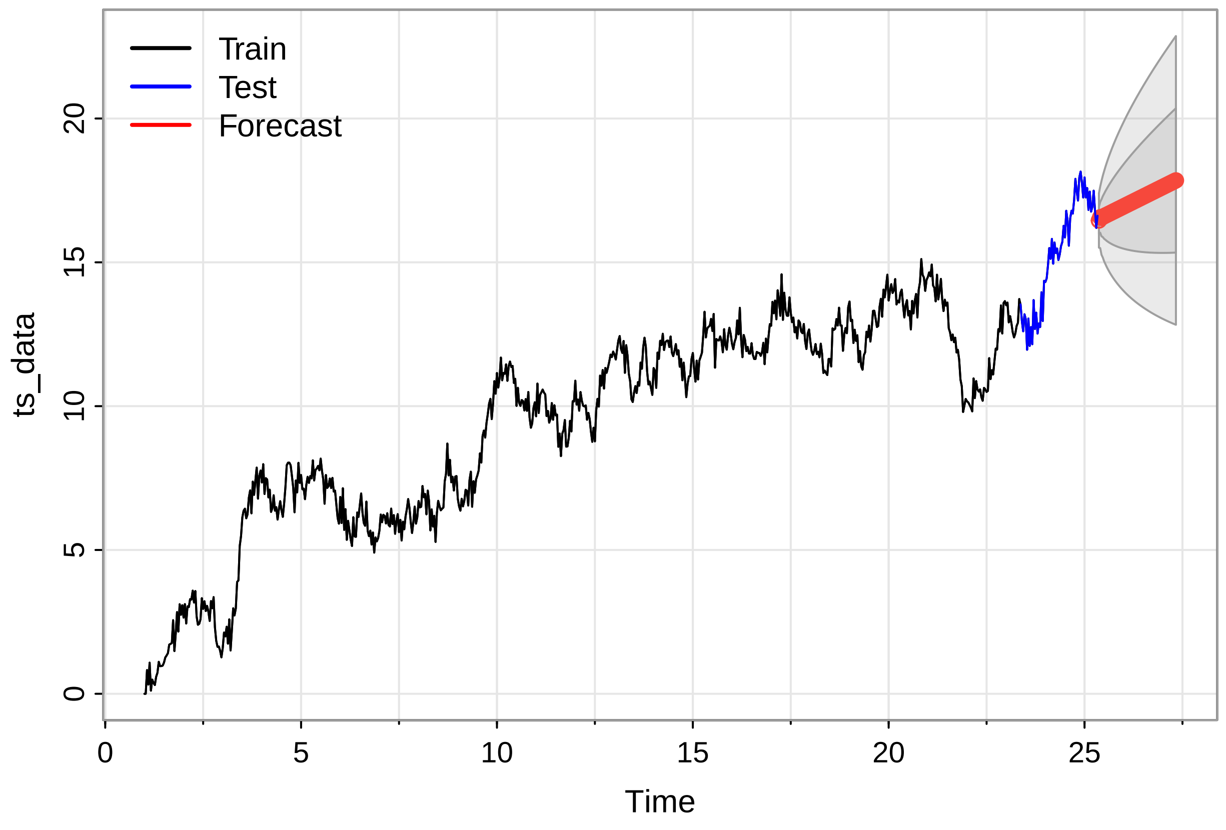
The model appears to do a satisfactory job with predictions compared against the test data. While not perfect, all the test values are within the confidence bounds of the predictions. Future predictions will also be made on the original data set, going another 60 days out.

```
# forecast 60 days into the future using the whole data set
sarima.for(ts_data,
           n.ahead = 60,
           1,1,0,
           plot.all = TRUE)
```

```
## $pred
## Time Series:
## Start = c(25, 12)
## End = c(27, 11)
## Frequency = 30
##  [1] 16.45412 16.56554 16.54647 16.58862 16.60205 16.62896 16.64954 16.67309
##  [9] 16.69525 16.71806 16.74056 16.76321 16.78579 16.80840 16.83100 16.85360
## [17] 16.87620 16.89881 16.92141 16.94401 16.96661 16.98922 17.01182 17.03442
## [25] 17.05702 17.07962 17.10223 17.12483 17.14743 17.17003 17.19263 17.21524
## [33] 17.23784 17.26044 17.28304 17.30564 17.32825 17.35085 17.37345 17.39605
## [41] 17.41865 17.44126 17.46386 17.48646 17.50906 17.53167 17.55427 17.57687
## [49] 17.59947 17.62207 17.64468 17.66728 17.68988 17.71248 17.73508 17.75769
## [57] 17.78029 17.80289 17.82549 17.84809
##
## $se
## Time Series:
```

```
## Start = c(25, 12)
## End = c(27, 11)
## Frequency = 30
##  [1] 0.4724387 0.5348617 0.6418280 0.7110267 0.7833992 0.8455275 0.9051811
##  [8] 0.9603464 1.0128628 1.0626297 1.1102387 1.1558565 1.1997554 1.2420973
## [15] 1.2830456 1.3227254 1.3612498 1.3987131 1.4351991 1.4707801 1.5055205
## [22] 1.5394771 1.5727007 1.6052369 1.6371265 1.6684067 1.6991112 1.7292705
## [29] 1.7589128 1.7880638 1.8167471 1.8449845 1.8727962 1.9002009 1.9272159
## [36] 1.9538575 1.9801406 2.0060794 2.0316871 2.0569760 2.0819577 2.1066432
## [43] 2.1310428 2.1551662 2.1790225 2.2026205 2.2259683 2.2490737 2.2719442
## [50] 2.2945867 2.3170080 2.3392144 2.3612119 2.3830064 2.4046033 2.4260080
## [57] 2.4472255 2.4682606 2.4891180 2.5098020
```

```r
lines(train, col = "black")
lines(test, col = "blue")
legend("topleft",
       legend = c("Train","Test","Forecast"),
       col = c("black","blue","red"),
       lty = 1,
       lwd = 2,
       bty = "n")
```



D4, Analysis Output and Calculations

All code, calculations, and output are provided throughout the document. Below is a comparison of the unseen test data (`actual`) to the predicted values (`forecast`) based on the training data. The average variance is also calculated, which comes in at roughly $1.23M.

```
arima_pred <- sarima.for(train,
                         n.ahead = 60,
                         1,1,0,
                         plot = FALSE)

pred_values <- arima_pred$pred
pred_df <- data.frame(
  actual = test,
  forecast = pred_values
)
pred_df <- pred_df %>%
  mutate(
    variance = actual - forecast
  )
pred_df
```

```
##      actual forecast     variance
## 1  13.53928 13.37954  0.159736444
## 2  12.96144 13.56526 -0.603816386
## 3  12.60190 13.51020 -0.908299302
## 4  13.19403 13.56445 -0.370413861
## 5  13.02495 13.56907 -0.544128193
## 6  11.95978 13.59623 -1.636441924
## 7  13.04856 13.61315 -0.564589230
## 8  12.09274 13.63472 -1.541982502
## 9  12.74881 13.65418 -0.905376377
## 10 12.15195 13.67460 -1.522651576
## 11 13.68802 13.69458 -0.006565595
## 12 12.68175 13.71476 -1.033016475
## 13 13.25007 13.73485 -0.484784126
## 14 12.52233 13.75499 -1.232652336
## 15 12.87797 13.77510 -0.897128052
## 16 12.74761 13.79522 -1.047612391
## 17 13.95947 13.81534  0.144129761
## 18 12.95735 13.83546 -0.878109373
## 19 14.35757 13.85558  0.501992467
## 20 14.31368 13.87570  0.437978562
## 21 14.44175 13.89582  0.545934457
## 22 14.89664 13.91593  0.980709680
## 23 15.49597 13.93605  1.559915735
## 24 15.12121 13.95617  1.165032526
## 25 15.81472 13.97629  1.838432933
## 26 14.95180 13.99641  0.955391548
## 27 15.69224 14.01653  1.675712153
## 28 15.32256 14.03665  1.285913798
## 29 15.48149 14.05677  1.424721024
## 30 15.07793 14.07689  1.001042919
## 31 15.26983 14.09701  1.172822814
## 32 15.56393 14.11713  1.446809829
## 33 15.70968 14.13724  1.572439495
## 34 16.27058 14.15736  2.113213050
## 35 15.86465 14.17748  1.687165905
## 36 16.79198 14.19760  2.594379751
## 37 16.44670 14.21772  2.228977526
## 38 15.57683 14.23784  1.338993961
## 39 16.51737 14.25796  2.259416676
## 40 16.79290 14.27808  2.514823112
## 41 16.69296 14.29820  2.394768577
## 42 17.15234 14.31832  2.834026102
## 43 17.90319 14.33843  3.564754687
## 44 17.47107 14.35855  3.112513403
## 45 17.14585 14.37867  2.767177778
## 46 17.94883 14.39879  3.550042653
## 47 18.15477 14.41891  3.735858688
## 48 17.76882 14.43903  3.329787604
## 49 17.25085 14.45915  2.791697869
## 50 17.95228 14.47927  3.473010364
## 51 17.25593 14.49939  2.756547720
## 52 17.58363 14.51951  3.064120985
## 53 16.82176 14.53962  2.282133520
## 54 17.45906 14.55974  2.899312565
## 55 16.76128 14.57986  2.181415651
## 56 16.93156 14.59998  2.331577036
## 57 17.49067 14.62010  2.870565531
## 58 16.80364 14.64022  2.163418306
## 59 16.19481 14.66034  1.534474802
## 60 16.62080 14.68046  1.940340777
```

```
mean(pred_df$variance)
```

```
## [1] 1.233428
```

D5, Code Support

All code has been provided throughout the document.

E1, Analysis Results

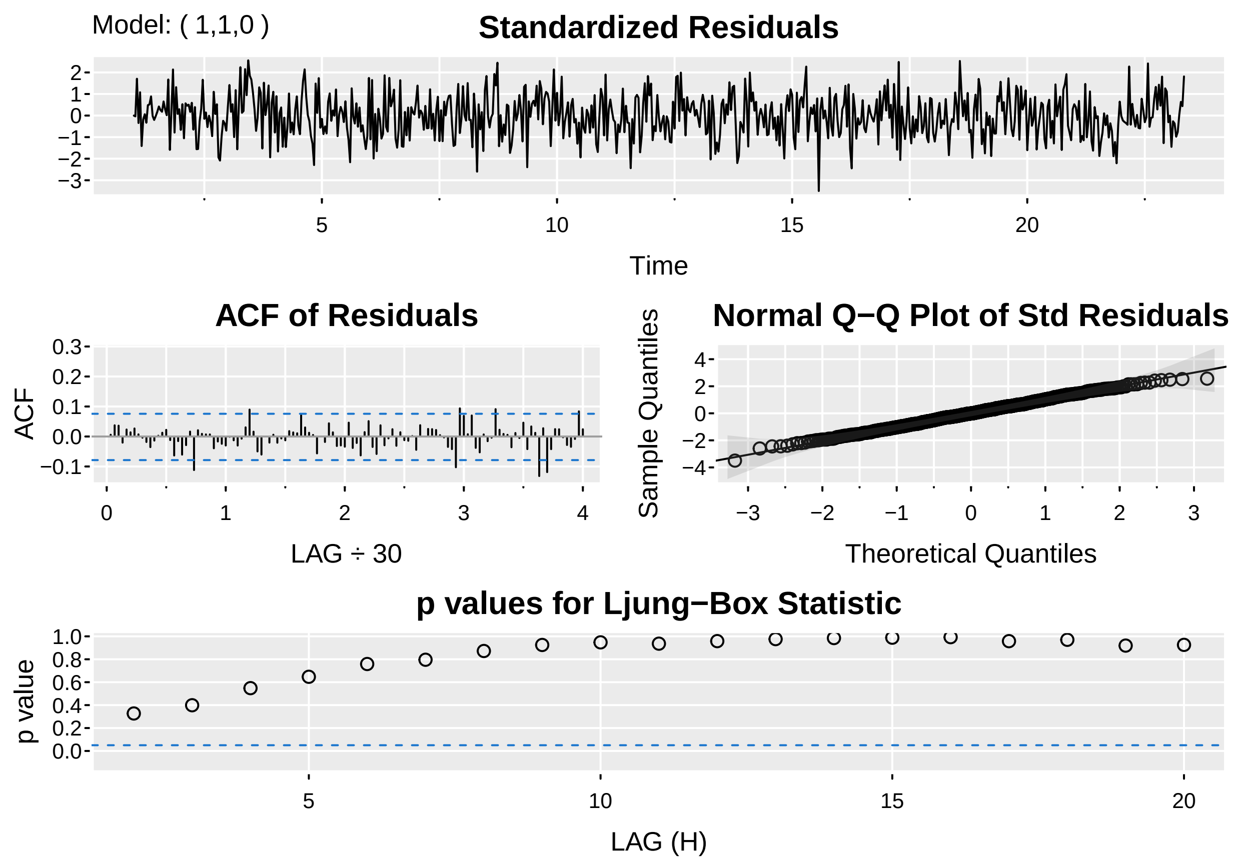
Selection of ARIMA Model
The ACF and PACF plots suggested two potential models: ARIMA(1,1,0) and ARIMA(0,1,1). The spectral density plot of the differenced data indicated no strong seasonality, so the seasonal component was not considered for modeling purposes. Three additional models were considered for model creation: ARIMA(2,1,0), ARIMA(0,1,2), and ARIMA(1,1,1). The models were fit using the training data. Akaike Information Criterion (*AIC*) was used to determine the best model fit. The AIC scores from each model were stored in a data frame and compared. The ARIMA(1,1,0) model produced the lowest AIC score, which was used as the final model.

```r
print(AIC_scores)
```

```
##           Model      AIC
## 1 ARIMA (1,1,0) 1.319860
## 2 ARIMA (0,1,1) 1.366350
## 3 ARIMA (2,1,0) 1.322638
## 4 ARIMA (0,1,2) 1.324594
## 5 ARIMA (1,1,1) 1.322680
```

```r
sarima(train,
       1,1,0,
       gg=TRUE)
```

```
## initial  value -0.647595
## iter   2 value -0.762911
## iter   3 value -0.762912
## iter   4 value -0.762913
## iter   5 value -0.762913
## iter   5 value -0.762913
## iter   5 value -0.762913
## final  value -0.762913
## converged
## initial  value -0.763486
## iter   2 value -0.763486
## iter   3 value -0.763486
## iter   3 value -0.763486
## iter   3 value -0.763486
## final  value -0.763486
## converged
## <><><><><><><><><><><><><><>
##
## Coefficients:
##          Estimate     SE  t.value p.value
## ar1       -0.4540 0.0345 -13.1725  0.0000
## constant   0.0201 0.0124   1.6242  0.1048
##
## sigma^2 estimated as 0.2171175 on 668 degrees of freedom
##
## AIC = 1.31986  AICc = 1.319887  BIC = 1.340042
##
```

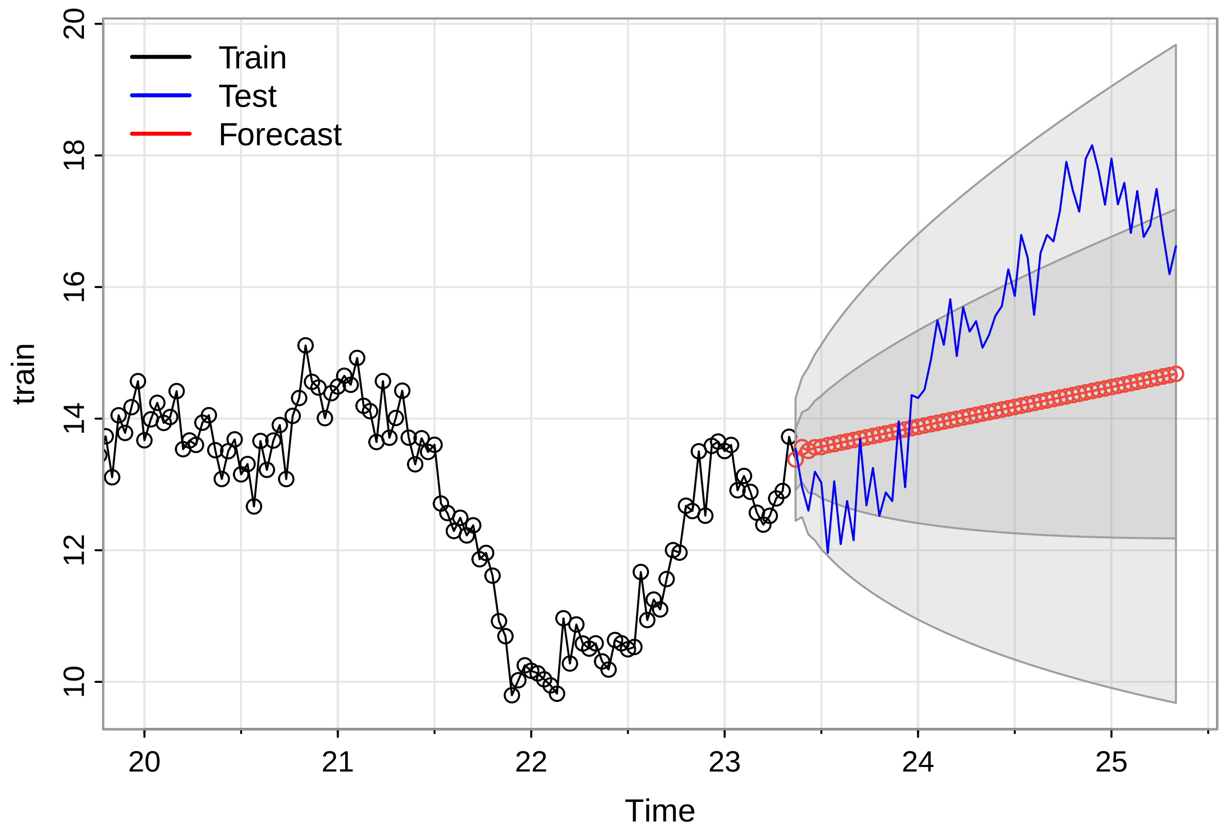


Prediction Interval
Using the ARIMA(1,1,0) model, the training data was forecasted 60 days out to compare to the unseen test data. The model predictions appeared satisfactory. The predictions do not precisely follow the test values, but all test points are within the prediction bounds.

```
sarima.for(train,
           n.ahead = 60,
           1,1,0)
```

```
## $pred
## Time Series:
## Start = c(23, 12)
## End = c(25, 11)
## Frequency = 30
##  [1] 13.37954 13.56526 13.51020 13.56445 13.56907 13.59623 13.61315 13.63472
##  [9] 13.65418 13.67460 13.69458 13.71476 13.73485 13.75499 13.77510 13.79522
## [17] 13.81534 13.83546 13.85558 13.87570 13.89582 13.91593 13.93605 13.95617
## [25] 13.97629 13.99641 14.01653 14.03665 14.05677 14.07689 14.09701 14.11713
## [33] 14.13724 14.15736 14.17748 14.19760 14.21772 14.23784 14.25796 14.27808
## [41] 14.29820 14.31832 14.33843 14.35855 14.37867 14.39879 14.41891 14.43903
## [49] 14.45915 14.47927 14.49939 14.51951 14.53962 14.55974 14.57986 14.59998
## [57] 14.62010 14.64022 14.66034 14.68046
##
## $se
## Time Series:
## Start = c(23, 12)
## End = c(25, 11)
## Frequency = 30
##  [1] 0.4659587 0.5308976 0.6361384 0.7062844 0.7781646 0.8405084 0.8999868
##  [8] 0.9551492 1.0075626 1.0572654 1.1047846 1.1503205 1.1941308 1.2363856
## [15] 1.2772450 1.3168365 1.3552722 1.3926473 1.4290454 1.4645391 1.4991927
## [22] 1.5330633 1.5662015 1.5986529 1.6304586 1.6616556 1.6922776 1.7223552
## [29] 1.7519165 1.7809872 1.8095910 1.8377496 1.8654833 1.8928106 1.9197489
## [36] 1.9463145 1.9725223 1.9983864 2.0239200 2.0491355 2.0740445 2.0986578
## [43] 2.1229858 2.1470381 2.1708240 2.1943520 2.2176305 2.2406671 2.2634692
## [50] 2.2860440 2.3083980 2.3305375 2.3524688 2.3741974 2.3957290 2.4170688
## [57] 2.4382218 2.4591929 2.4799867 2.5006075
```

```r
lines(test, col = "blue")
legend("topleft",
       legend = c("Train","Test","Forecast"),
       col = c("black","blue","red"),
       lty = 1,
       lwd = 2,
       bty = "n")
```

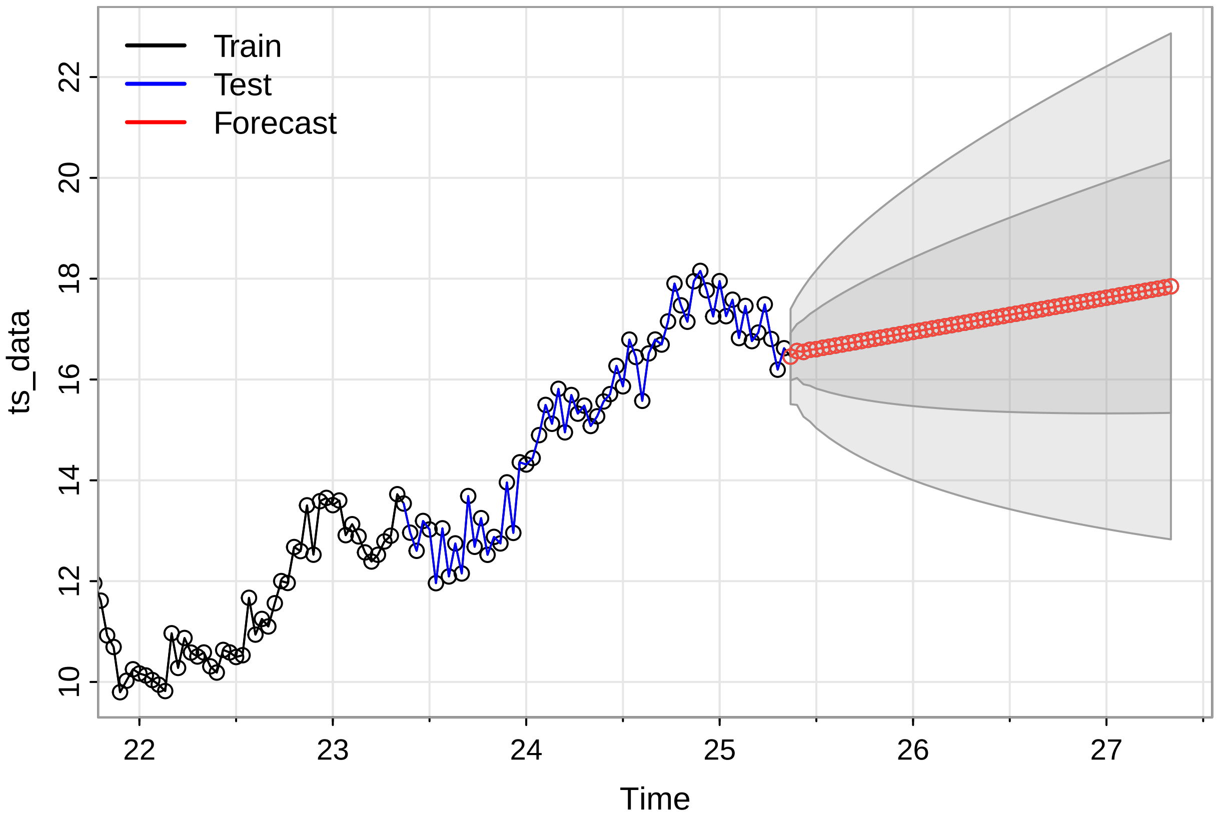


A second prediction was made using the ARIMA(1,1,0) model after seeing satisfactory results compared to test data. This time, using the complete data set, predictions were made 60 days into the future.

```r
sarima.for(ts_data,
           n.ahead = 60,
           1,1,0)
```

```
## $pred
## Time Series:
## Start = c(25, 12)
## End = c(27, 11)
## Frequency = 30
##  [1] 16.45412 16.56554 16.54647 16.58862 16.60205 16.62896 16.64954 16.67309
##  [9] 16.69525 16.71806 16.74056 16.76321 16.78579 16.80840 16.83100 16.85360
## [17] 16.87620 16.89881 16.92141 16.94401 16.96661 16.98922 17.01182 17.03442
## [25] 17.05702 17.07962 17.10223 17.12483 17.14743 17.17003 17.19263 17.21524
## [33] 17.23784 17.26044 17.28304 17.30564 17.32825 17.35085 17.37345 17.39605
## [41] 17.41865 17.44126 17.46386 17.48646 17.50906 17.53167 17.55427 17.57687
## [49] 17.59947 17.62207 17.64468 17.66728 17.68988 17.71248 17.73508 17.75769
## [57] 17.78029 17.80289 17.82549 17.84809
##
## $se
## Time Series:
## Start = c(25, 12)
## End = c(27, 11)
## Frequency = 30
##  [1] 0.4724387 0.5348617 0.6418280 0.7110267 0.7833992 0.8455275 0.9051811
##  [8] 0.9603464 1.0128628 1.0626297 1.1102387 1.1558565 1.1997554 1.2420973
## [15] 1.2830456 1.3227254 1.3612498 1.3987131 1.4351991 1.4707801 1.5055205
## [22] 1.5394771 1.5727007 1.6052369 1.6371265 1.6684067 1.6991112 1.7292705
## [29] 1.7589128 1.7880638 1.8167471 1.8449845 1.8727962 1.9002009 1.9272159
## [36] 1.9538575 1.9801406 2.0060794 2.0316871 2.0569760 2.0819577 2.1066432
## [43] 2.1310428 2.1551662 2.1790225 2.2026205 2.2259683 2.2490737 2.2719442
## [50] 2.2945867 2.3170080 2.3392144 2.3612119 2.3830064 2.4046033 2.4260080
## [57] 2.4472255 2.4682606 2.4891180 2.5098020
```

```r
lines(train, col = "black")
lines(test, col = "blue")
legend("topleft",
       legend = c("Train","Test","Forecast"),
       col = c("black","blue","red"),
       lty = 1,
       lwd = 2,
       bty = "n")
```



Forecast Length
The forecast length of 60 days was selected and deemed reasonable. The complete data set is 731 revenue days, including 671 in the training set. The model should have been able to pick up on patterns within that time frame. From a business point-of-view, predicting revenues 60 days into the future with moderate accuracy is beneficial. Ideally, a longer time frame would be desired, but it would likely sacrifice accuracy.

Model Evaluation
AIC scores helped determine which model fit was best, but they do not indicate whether the model is good overall. The Root Mean Square Error (*RMSE*) was calculated to evaluate the model's quality. The resulting RMSE was 1.96, which seems fair given that the average value of the test set was 15.3. The ARIMA(1,1,0) model can be considered adequate and suitable for an initial pass.

```
rmse(pred_df$actual,pred_df$forecast)
```

```
## [1] 1.967612
```

```
mean(pred_df$actual)
```

```
## [1] 15.31907
```

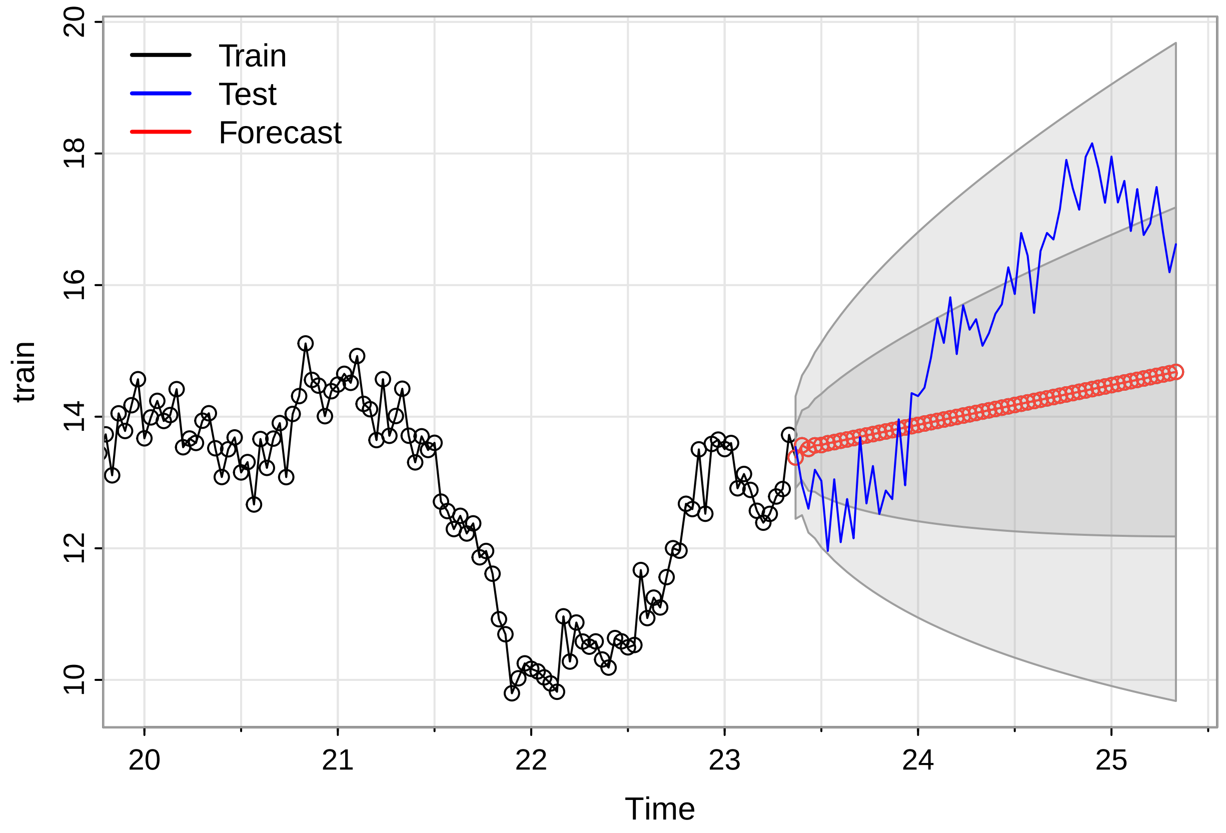
E2, Annotated Visualization

Below is the plot of the training data, test data, and model forecast.

```r
sarima.for(train,
           n.ahead = 60,
           1,1,0)
```

```
## $pred
## Time Series:
## Start = c(23, 12)
## End = c(25, 11)
## Frequency = 30
##  [1] 13.37954 13.56526 13.51020 13.56445 13.56907 13.59623 13.61315 13.63472
##  [9] 13.65418 13.67460 13.69458 13.71476 13.73485 13.75499 13.77510 13.79522
## [17] 13.81534 13.83546 13.85558 13.87570 13.89582 13.91593 13.93605 13.95617
## [25] 13.97629 13.99641 14.01653 14.03665 14.05677 14.07689 14.09701 14.11713
## [33] 14.13724 14.15736 14.17748 14.19760 14.21772 14.23784 14.25796 14.27808
## [41] 14.29820 14.31832 14.33843 14.35855 14.37867 14.39879 14.41891 14.43903
## [49] 14.45915 14.47927 14.49939 14.51951 14.53962 14.55974 14.57986 14.59998
## [57] 14.62010 14.64022 14.66034 14.68046
##
## $se
## Time Series:
## Start = c(23, 12)
## End = c(25, 11)
## Frequency = 30
##  [1] 0.4659587 0.5308976 0.6361384 0.7062844 0.7781646 0.8405084 0.8999868
##  [8] 0.9551492 1.0075626 1.0572654 1.1047846 1.1503205 1.1941308 1.2363856
## [15] 1.2772450 1.3168365 1.3552722 1.3926473 1.4290454 1.4645391 1.4991927
## [22] 1.5330633 1.5662015 1.5986529 1.6304586 1.6616556 1.6922776 1.7223552
## [29] 1.7519165 1.7809872 1.8095910 1.8377496 1.8654833 1.8928106 1.9197489
## [36] 1.9463145 1.9725223 1.9983864 2.0239200 2.0491355 2.0740445 2.0986578
## [43] 2.1229858 2.1470381 2.1708240 2.1943520 2.2176305 2.2406671 2.2634692
## [50] 2.2860440 2.3083980 2.3305375 2.3524688 2.3741974 2.3957290 2.4170688
## [57] 2.4382218 2.4591929 2.4799867 2.5006075
```

```r
lines(test, col = "blue")
legend("topleft",
       legend = c("Train","Test","Forecast"),
       col = c("black","blue","red"),
       lty = 1,
       lwd = 2,
       bty = "n")
```



E3, Recommendation

The ARIMA(1,1,0) model fit to the training data provided adequate results compared to the unseen test data. The company can use this model for high-level planning purposes for future months with the understanding that there can be some variance. In general, however, this time-series model is beneficial. Another recommended action is to continue fine-tuning the model parameters or feeding in more revenue data. It is possible that adding some seasonal components may increase model accuracy, though that does not seem likely given the results of the spectral density plot.

Appendix

G, Web Sources

Stoffer, D. (n.d.). *ARIMA Models in R* [MOOC]. DataCamp. (https://app.datacamp.com/learn/courses/arima-models-in-r) Vijay. (April 04, 2025). *Understanding stationary time series analysis*. Analytics Vidhya. Retrieved June 05, 2025, from Analytics Vidhya Wainaina, P. (October 24, 2023). *The complete guide to time series forecasting models*. Medium. Retrieved June 05, 2025, from Medium WGU College of Information Technology (n.d.). *D213 ARIMA II Webinar* [Panopto Video]. Western Governors University. (https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=37007075-c29e-43f2-a106-af7800d45914)

H, Code Sources

Peterson, A. (n.d.). *Reporting with R Markdown* [MOOC]. DataCamp. (https://app.datacamp.com/learn/courses/reporting-with-rmarkdown) Stoffer, D. (n.d.). *ARIMA Models in R* [MOOC]. DataCamp. (https://app.datacamp.com/learn/courses/arima-models-in-r) Tierney, N. (n.d.). *Dealing with Missing Data in R* [MOOC]. DataCamp. (https://app.datacamp.com/learn/courses/dealing-with-missing-data-in-r) WGU College of Information Technology (n.d.). *D213 ARIMA II Webinar* [Panopto Video]. Western Governors University. (https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=37007075-c29e-43f2-a106-af7800d45914) WGU College of Information Technology (n.d.). *Getting Started with Duplicates* [PowerPoint slides]. Western Governors University. (https://westerngovernorsuniversity.sharepoint.com/sites/DataScienceTeam/Shared Documents/Forms/AllItems.aspx?id=%2Fsites%2FDataScienceTeam%2FShared%20Document