Forecasting Company Sales Over Time

PSTAT 174, University of California, Santa Barbara

Jacob Gerhard

12/03/2021

Abstract

Time series analysis consists of interpretation of metered data. Financial data is frequently collected at equally spaced intervals of time, and business analytics is often an implementation of such data. Specifically, companies record daily, weekly, monthly, and yearly sales for forecasting and diagnostics, i.e. evaluating company strengths and weaknesses. This project will determine a SARIMA model (seasonal autoregressive integrated moving average) that will predict future values of a company's monthly sales. Using the Box-Jenkins method, this project will cover transformations, differencing, model selection, stationarity and invertibility, and diagnostic checking using residuals. The analysis performed in this paper identifies a model within the scope of linear time series analysis that provides the ideal fit for the sales data provided.

Introduction

Time series data is made up of dependent observations that are time-ordered and available at equally spaced intervals of time. The use of time series data within data analytics can be incredibly beneficial for a wide range of topics, such as understanding stochastic mechanisms or forecasting future data values. In this project, I will implement time series analysis with R on the data set "Sales of Company X, Jan. 1965 to May 1971" (Hipel and McLeod, 1994) in order to predict the expected sales of Company X over the course of May 1971 to December 1971. This can provide information such as when the company should focus on marketing to maximize sales, what months the company can improve sales, or the amount of product that needs to be produced to meet monthly demand. Implementing the Box-Jenkins method of analysis, I will determine a SARIMA model that will fit the historic data that I will utilize to predict future values. I found this particular data set interesting because Company X represents a general company, meaning the data analysis performed in this project can be applied to the sales of any company, where the industry does not influence my analysis. From my analysis, I concluded that the model that best fit the data is

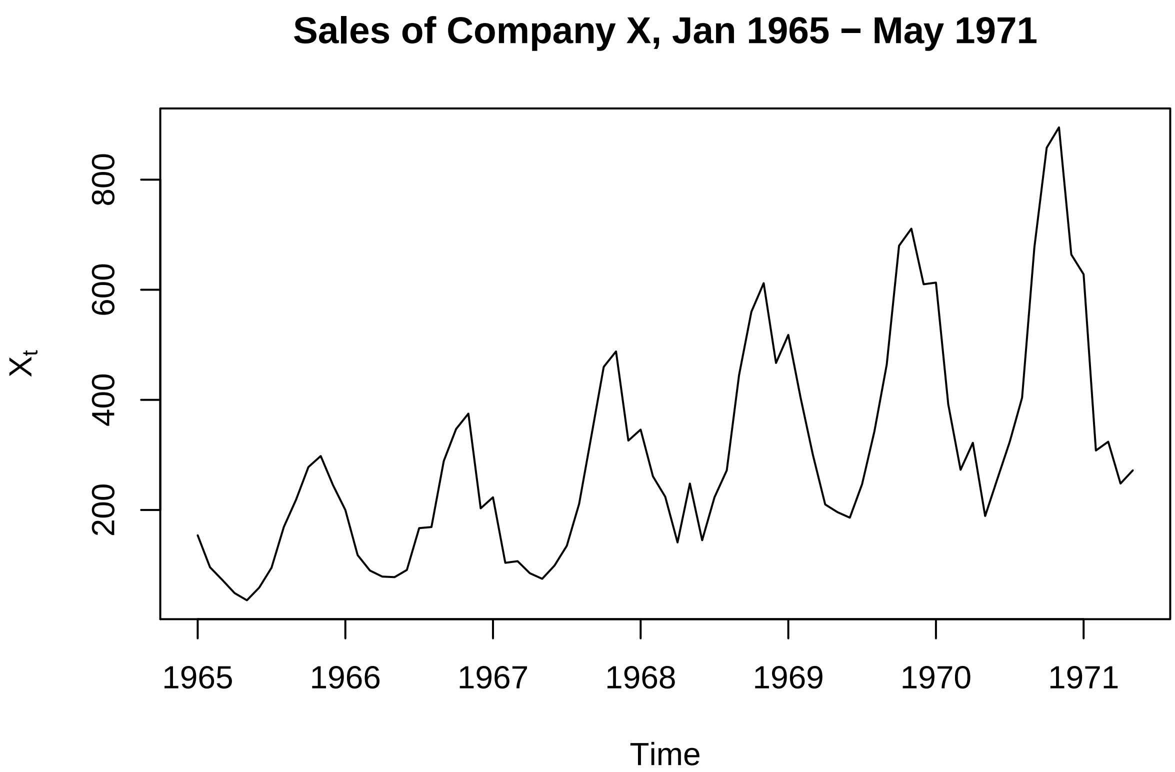
$$\nabla_1 \nabla_{12}(1-0.2639_{(0.1263)}B-0.3596_{(0.1289)}B^2)\sqrt{U_t} = (1-1_{(0.0956)}B)(1-0.5127_{(0.2105)}B^{12})Z_t, \quad Z_t \sim N(0, 1.783)$$

. This model is used to predict the future monthly sales of Company X. Understanding model limitations should be considered when quantifying prediction inaccuracies

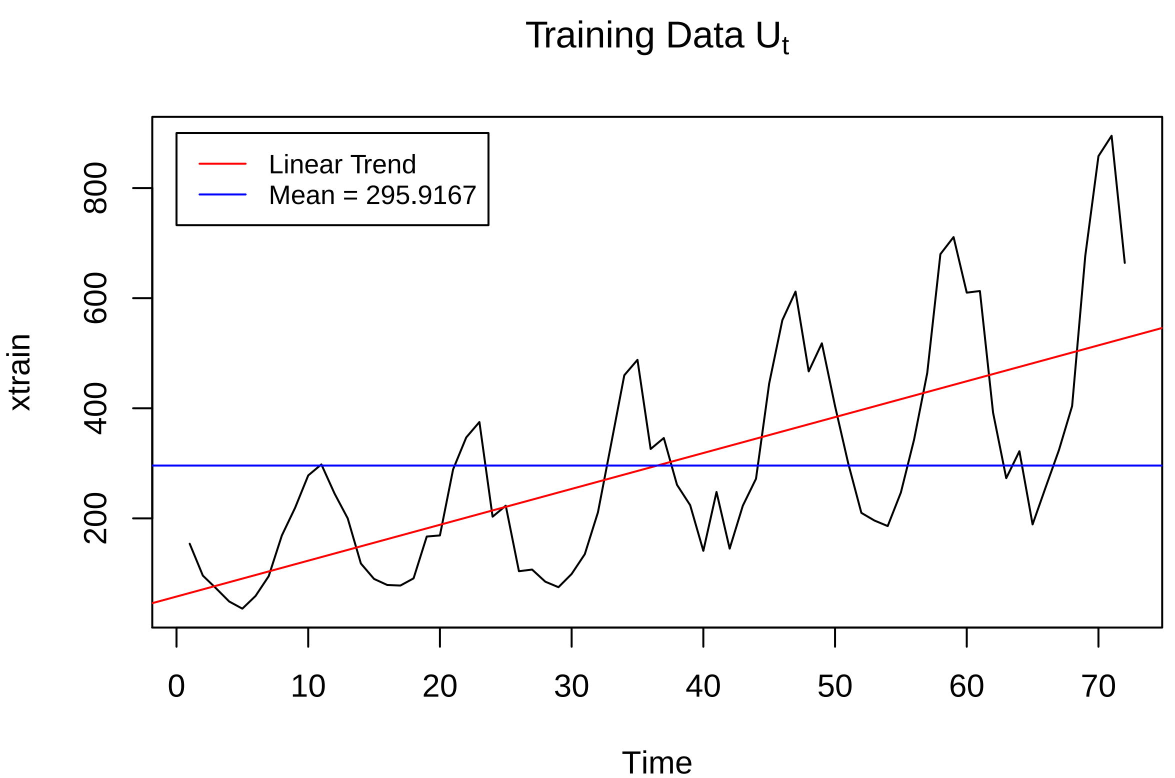
Data Analysis

Understanding Data

The data is a univariate time series consisting of 1 variable and 77 observations. The data is modeled by $X_t =$ Sales of Company X, $t = 1, 2, 3, \ldots, 77$. A plot of the data can be seen below:



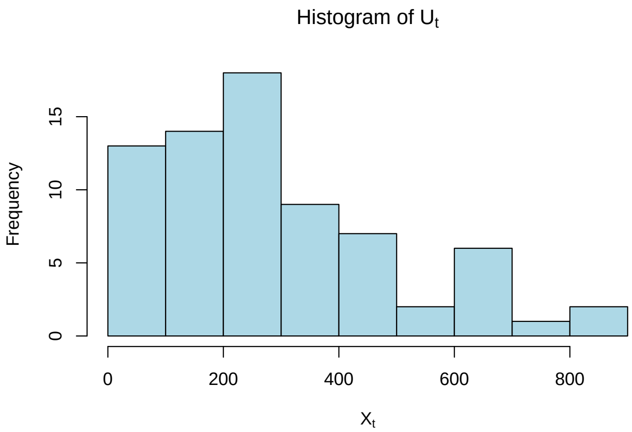
The data is split into a training set of 72 observations and a test set of 5 observations, which will be used to determine the accuracy of our predictions later on. Analysis will be performed on the training data $U_t = X_1, X_2, \ldots, X_{68}$. The plot of the training data is below:

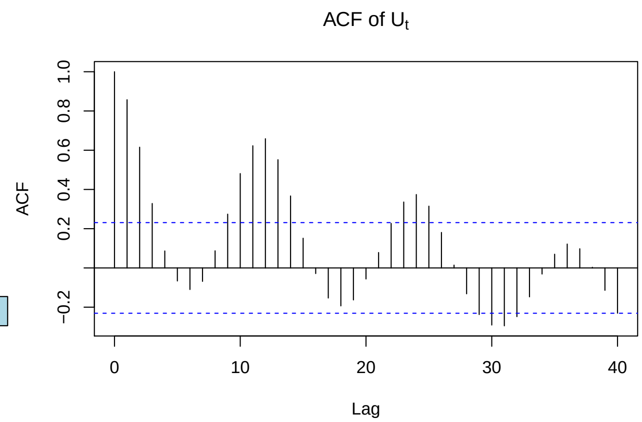


Evaluating the data, there is obvious positive trend, a yearly seasonal component, and a few sharp changes in behavior at $t \approx 41, 49, 64$. This model also has increasing variance. Transforming and differencing the data to a stationary series will generate constant variance and mean.

Transform Data to Stationary Series

Currently, the data has histogram and acf as follows:

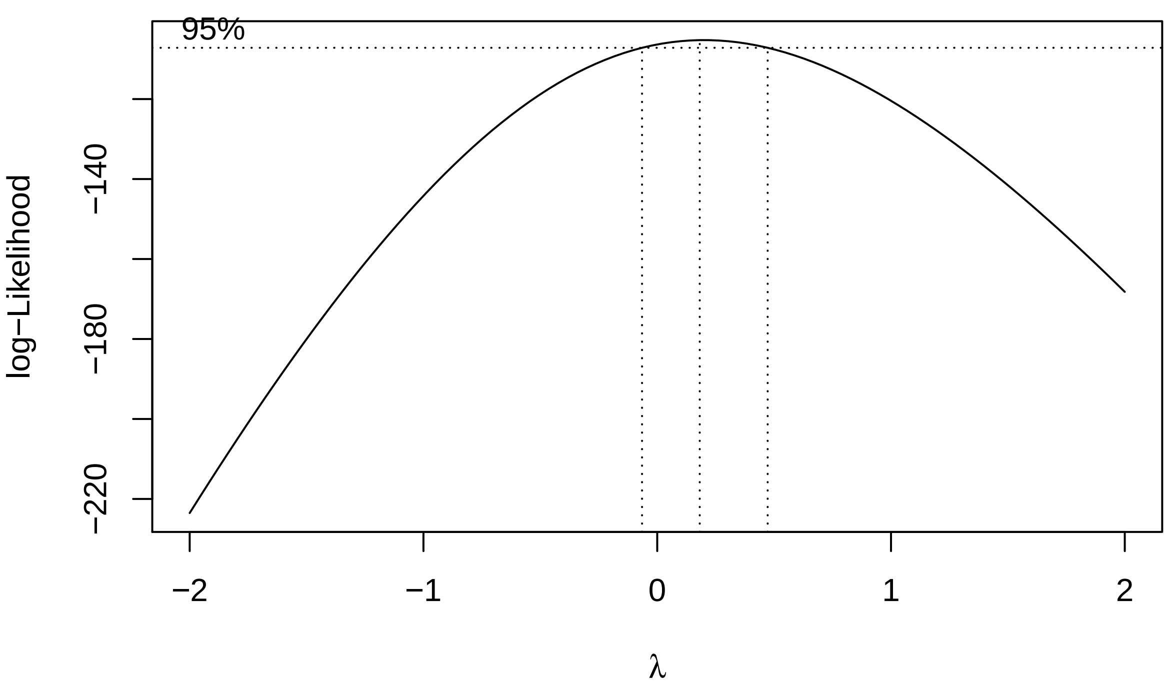




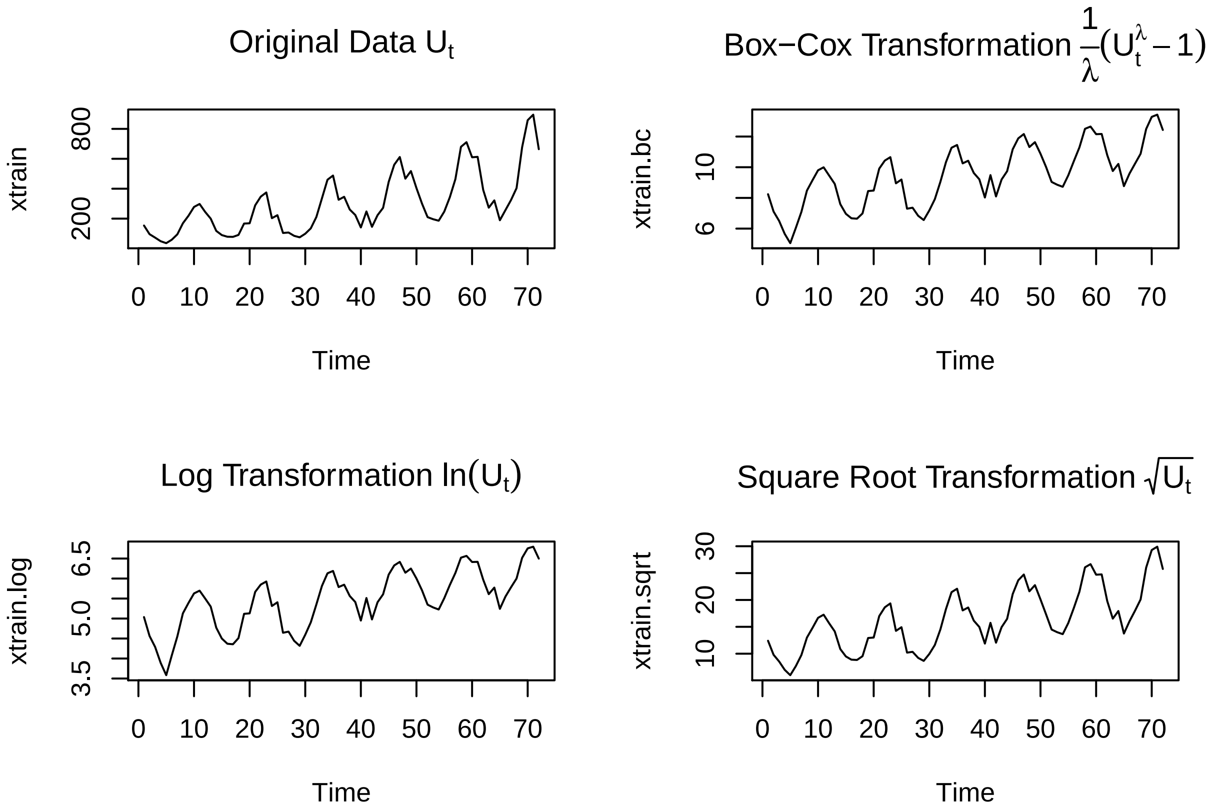
The histogram is skewed right, further demonstrating non-constant variance. The ACF has significant values that represent seasonality and trend. The three transformations that are performed on the data are Box-Cox, logarithmic, and square root.

Choosing Transformation

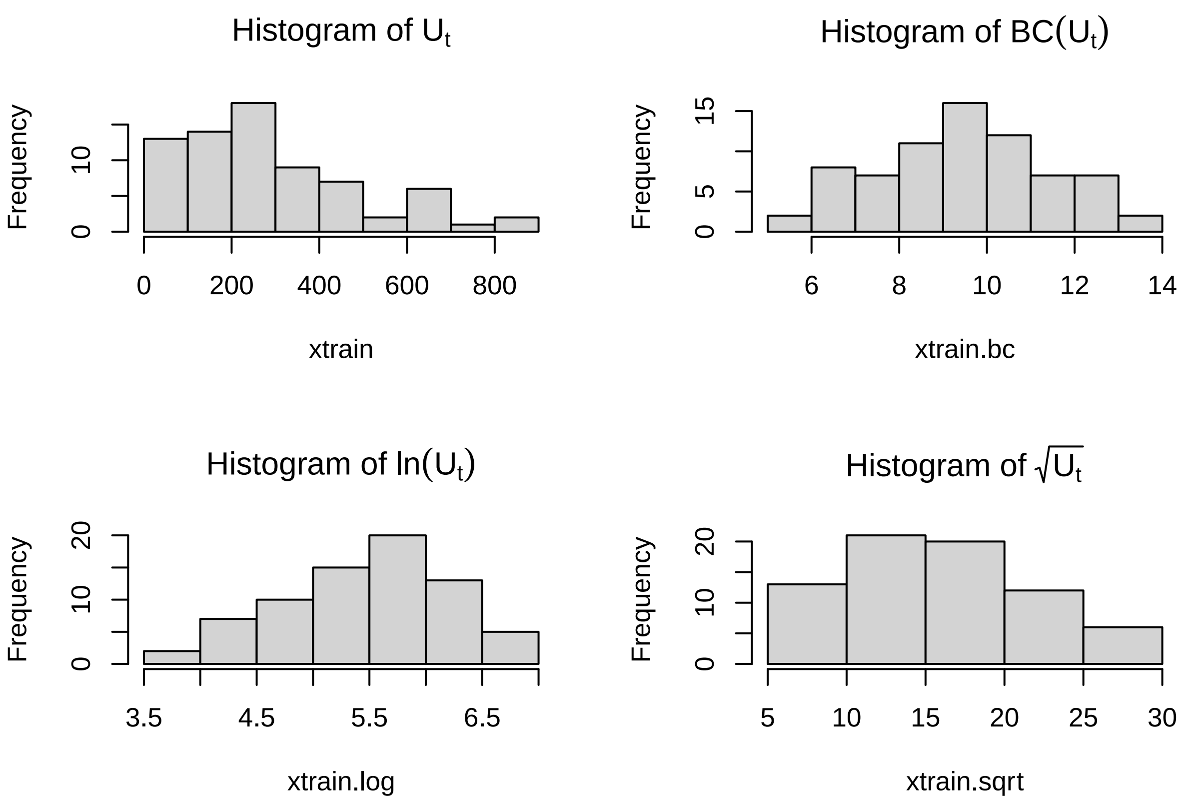
The Box-Cox Transformation plot helps determine possible transformations of series U_t:



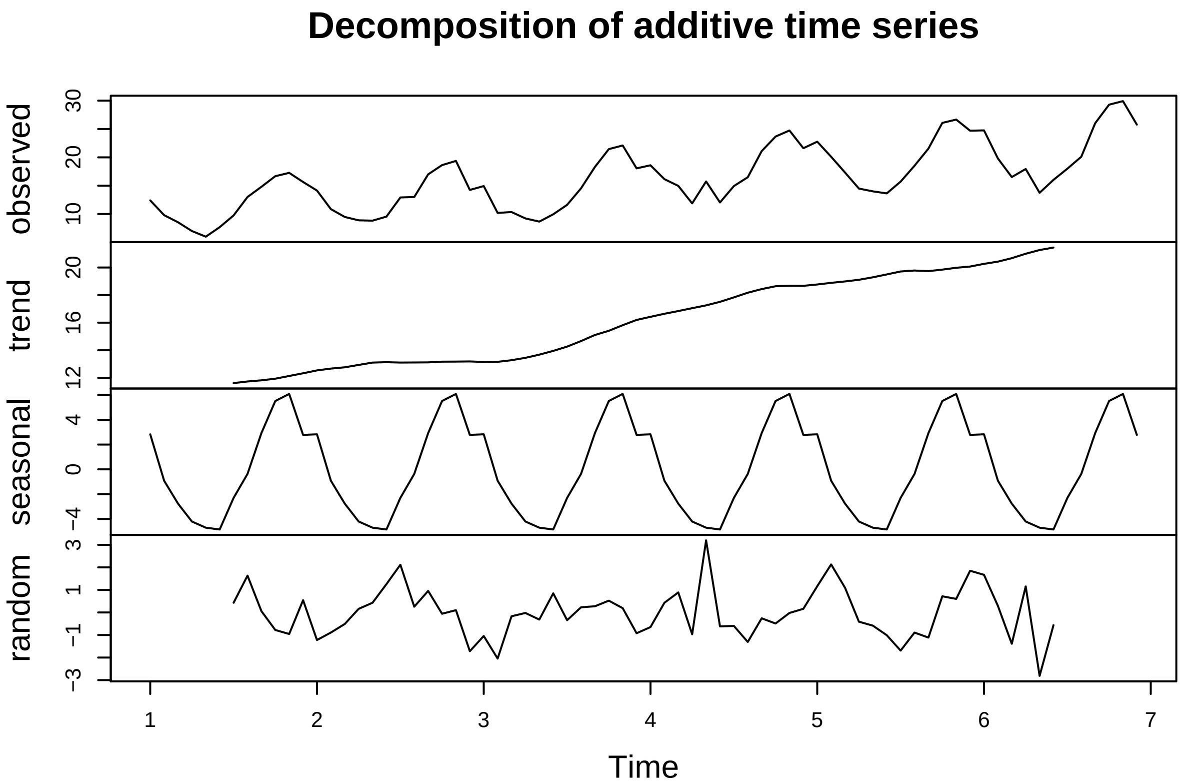
The predictive `boxcox()` command produces $\lambda = 0.1818$. The confidence interval is $\lambda \approx (-0.05, 0.5)$. Since both $\lambda = 0, \frac{1}{2}$ lie in the confidence interval, logarithmic and square root transformations should also be considered. The plots of the transformations and the original data are:



The transformations have reduced the variance. Plots of the histograms will further help determine the transformation that will most stabilize the variance.



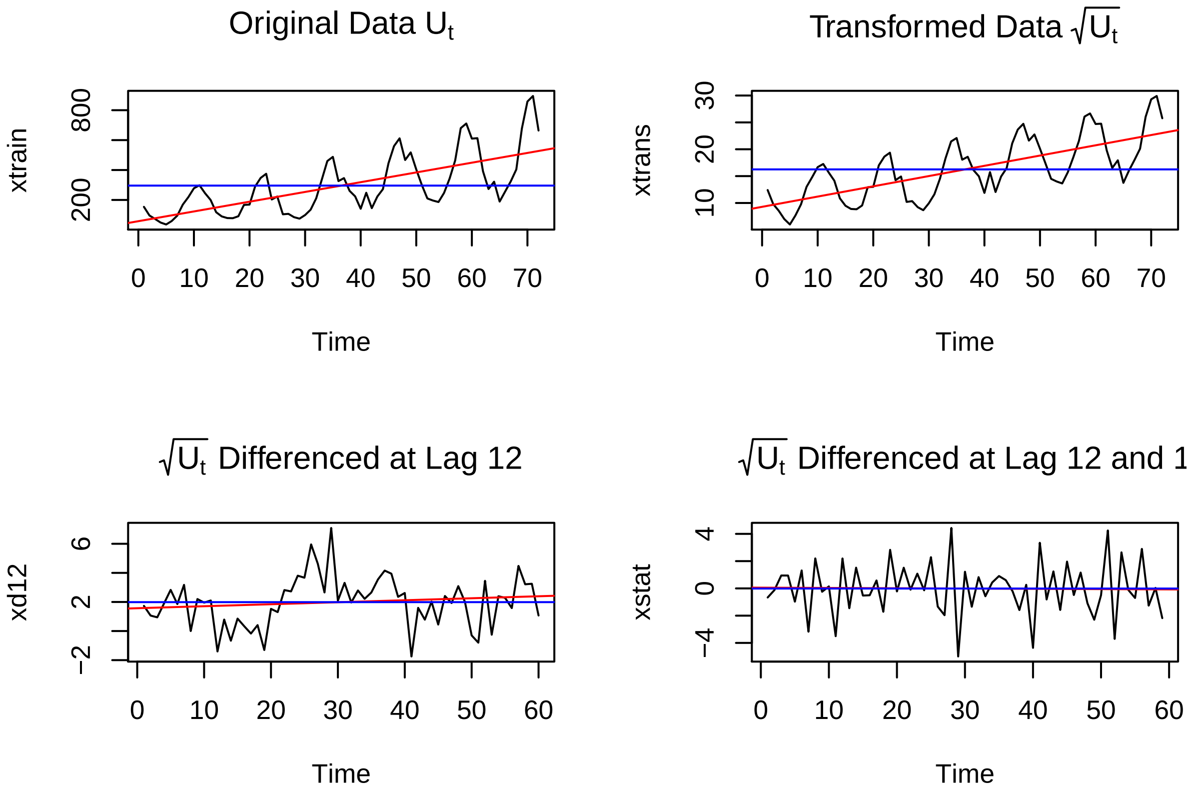
The histogram of $\sqrt{U_t}$ has the flattest curve, and therefore, most stabilized variance. The general result of the log transformation can be found in the Conclusion. The decomposition of $\sqrt{U_t}$ is below:



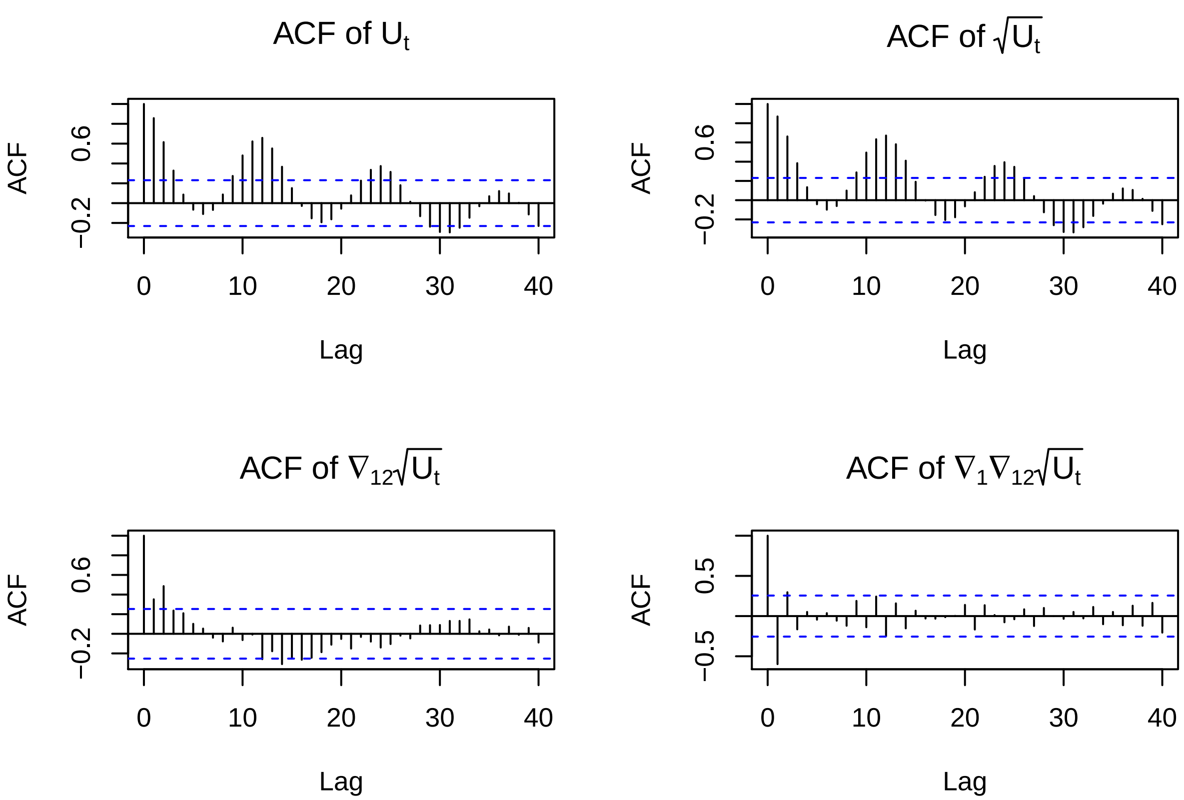
The variance-stabilized data still has seasonality and trend. Differencing the data can remove these features.

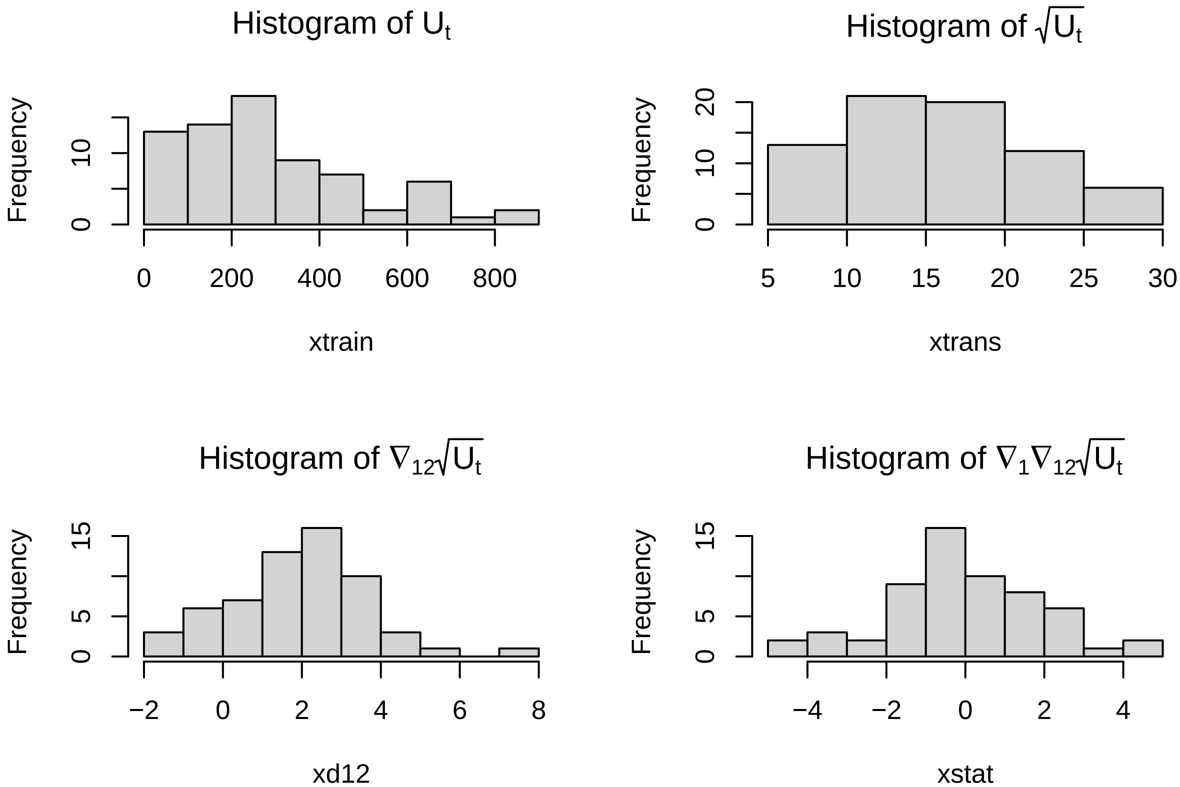
Differencing Data

The data has seasonality every 12 months, so a difference at lag 12, $\nabla_{12}\sqrt{U_t}$ will remove the seasonal component. Differencing at lag 1 removes trend, represented as $\nabla_1\nabla_{12}\sqrt{U_t}$.

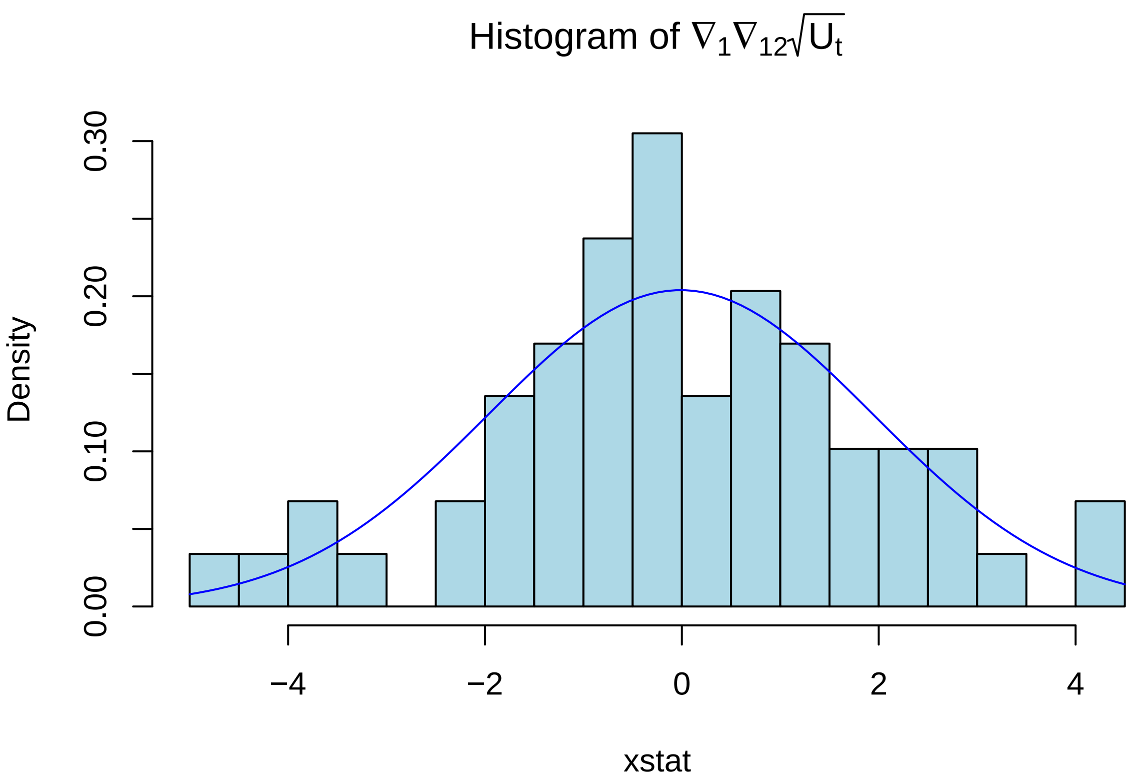


The stationary data set $S_t = \nabla_1 \nabla_{12} \sqrt{U_t}$. Series S_t has $\mu_{S_t} = -0.0123$ and $\mathrm{Var}(S_t) = 3.8259$. The graph of S_t displays constant variance and mean near 0, meaning $S_t \sim N(-0.0123, 3.8259)$. The plots of the various ACF and histograms are below:





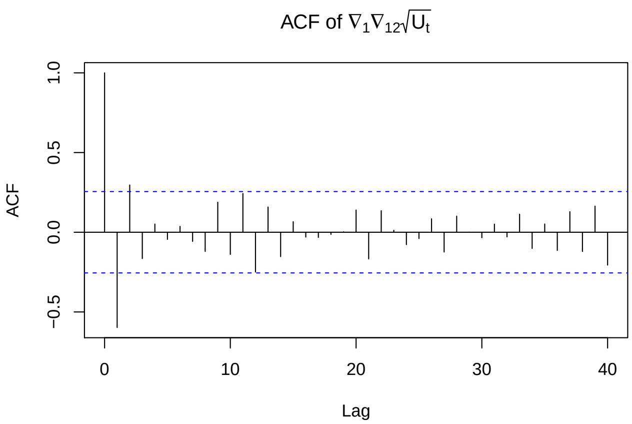
The histogram below is a more detailed version of the differenced and transformed data, displaying density rather than frequency, overlayed with a normal curve $N(\mu_{S_t}, \sigma^2_{S_t})$:

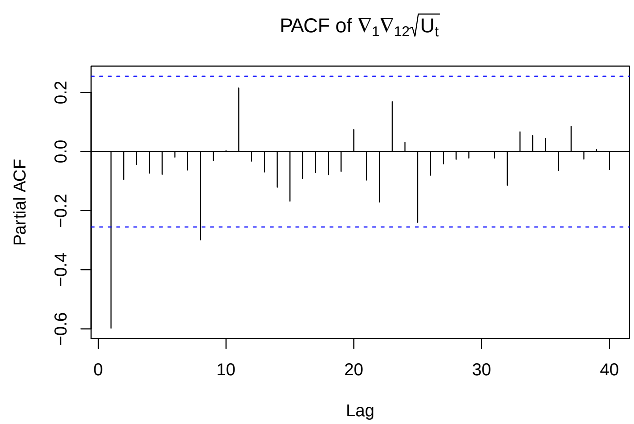


Now that the data represents a somewhat normal distribution, the ACF and PACF provide the information necessary to identify the ideal SARIMA model.

Model Selection

The SARIMA model can be identified by looking at lags in which the ACF and PACF have significant values.





The ACF has significant (non-zero) values at lags $k = 1, 2$. The PACF has significant values at lags $k = 1, 8$. These are used to determine values p, P, q, Q in SARIMA$(p, d, q) \times (P, D, Q)_Q$. It is known that $s = 12, d = 1, D = 1$ from the previous transformations. Possible models have values $p = 1, 2; q = 0, 1, 8; P = 0, 1; Q = 0, 1$. The model with the best fit will have the lowest AICc. After checking all possible models, the three with the lowest AICc are:
SARIMA$(1, 1, 0) \times (0, 1, 1)_{12}$ with AICc $= 220.2807$
SARIMA$(1, 1, 0) \times (1, 1, 1)_{12}$ with AICc $= 220.7522$
SARIMA$(2, 1, 1) \times (0, 1, 1)_{12}$ with AICc $= 220.1518$

Implementing the `arima()` function, R estimates coefficients for $\phi_p, \Phi_P, \theta_q, \Theta_Q$.

```
##
## Call:
## arima(x = xtrans, order = c(2, 1, 1), seasonal = list(order = c(0, 1, 1), period = 12),
##     method = "ML")
##
## Coefficients:
##          ar1     ar2      ma1     sma1
##       0.2639  0.3596  -1.0000  -0.5217
## s.e.  0.1263  0.1289   0.0956   0.2105
##
## sigma^2 estimated as 1.783:  log likelihood = -104.51,  aic = 219.02
```

```
##
## Call:
## arima(x = xtrans, order = c(1, 1, 0), seasonal = list(order = c(0, 1, 1), period = 12),
##     method = "ML")
##
## Coefficients:
##           ar1     sma1
##       -0.5434  -0.5403
## s.e.   0.1133   0.2133
##
## sigma^2 estimated as 2.035:  log likelihood = -106.92,  aic = 219.84
```

Model A

$$\nabla_1\nabla_{12}(1-0.2639_{(0.1263)}B-0.3596_{(0.1289)}B^2)\sqrt{U_t} = (1-1_{(0.0956)}B)(1-0.5127_{(0.2105)}B^{12})Z_t, \quad Z_t \sim N(0,1.783)$$

Model B

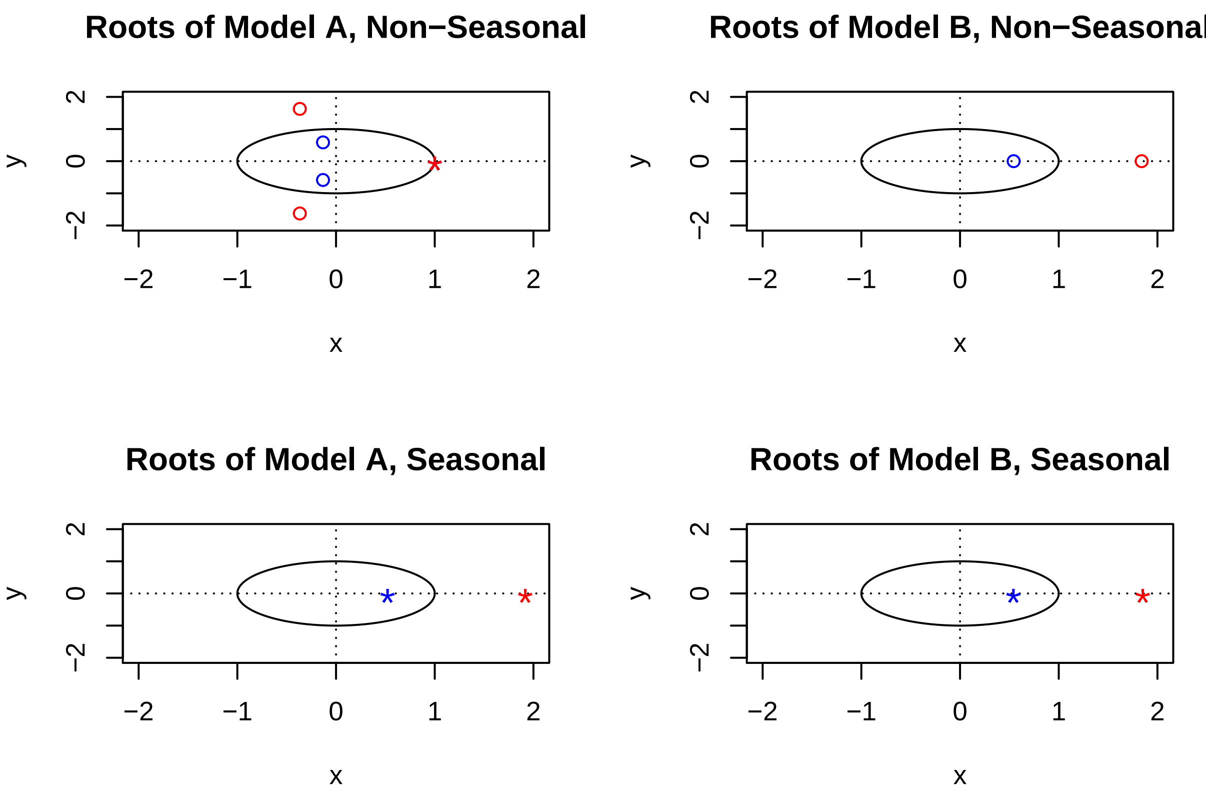
$$\nabla_1\nabla_{12}(1+0.5434_{(0.1133)}B)\sqrt{U_t} = (1-0.5403_{(0.2133)}B^{12})Z_t, \quad Z_t \sim N(0,2.035)$$

Diagnostic checking is then performed on the two best models to estimate model accuracy and help choose which model will be used for prediction.

Diagnostic checking

Stationarity and Invertibility

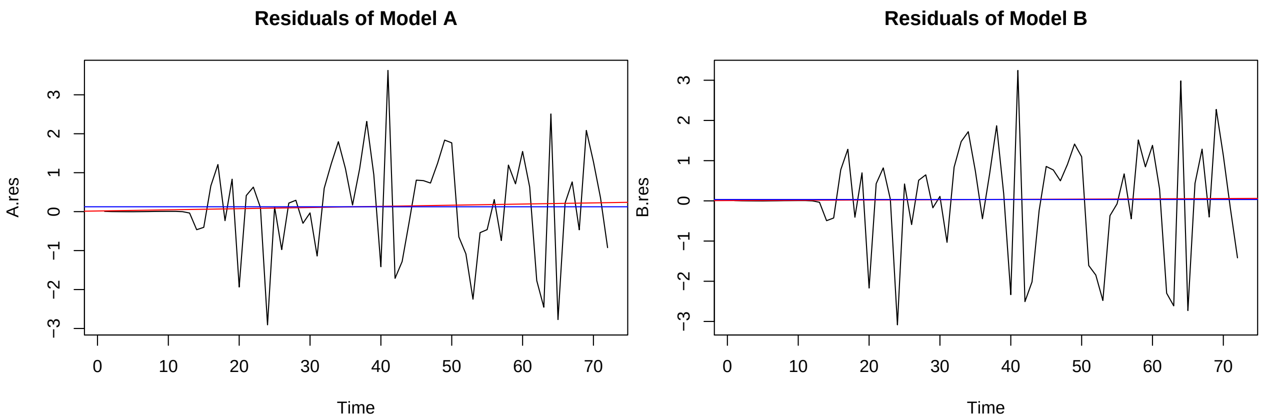
The `plot.roots()` function displays the roots of the models. The models are invertible if the MA roots ($\star$) lie outside of the unit circle. The models are stationary if the AR roots ($\circ$) lie outside of the unit circle. Obviously, the models are going to be stationary due to the transformations performed in Transforming Data to Stationary Series.



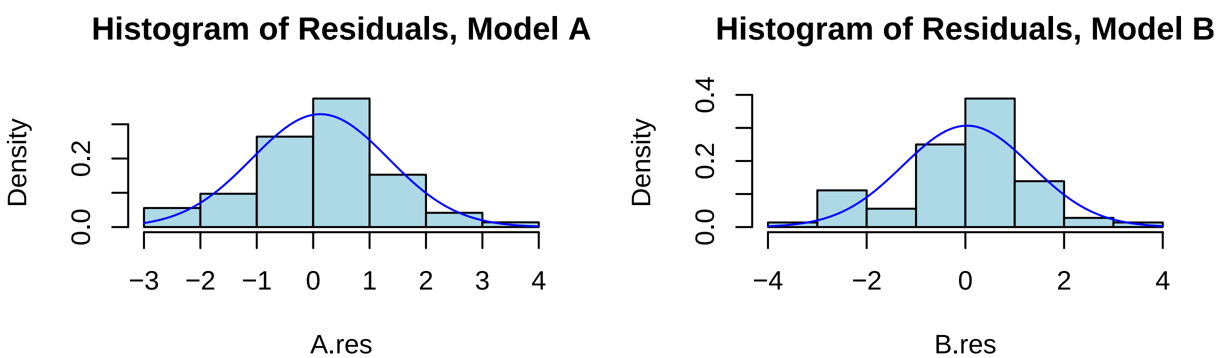
Model A is stationary but not invertible, Model B is stationary and invertible.

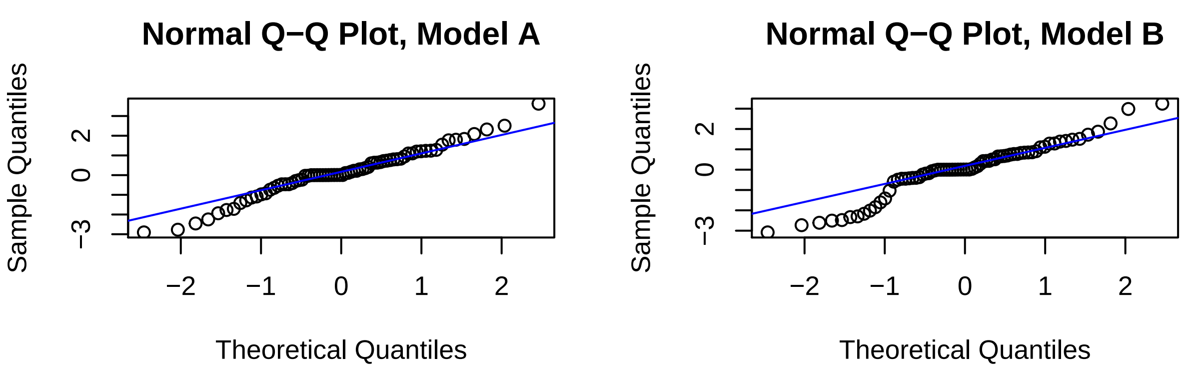
Analysis of Residuals

The next step in diagnostic checking is analysis of residuals. The residuals of an accurate model should resemble Gaussian white noise

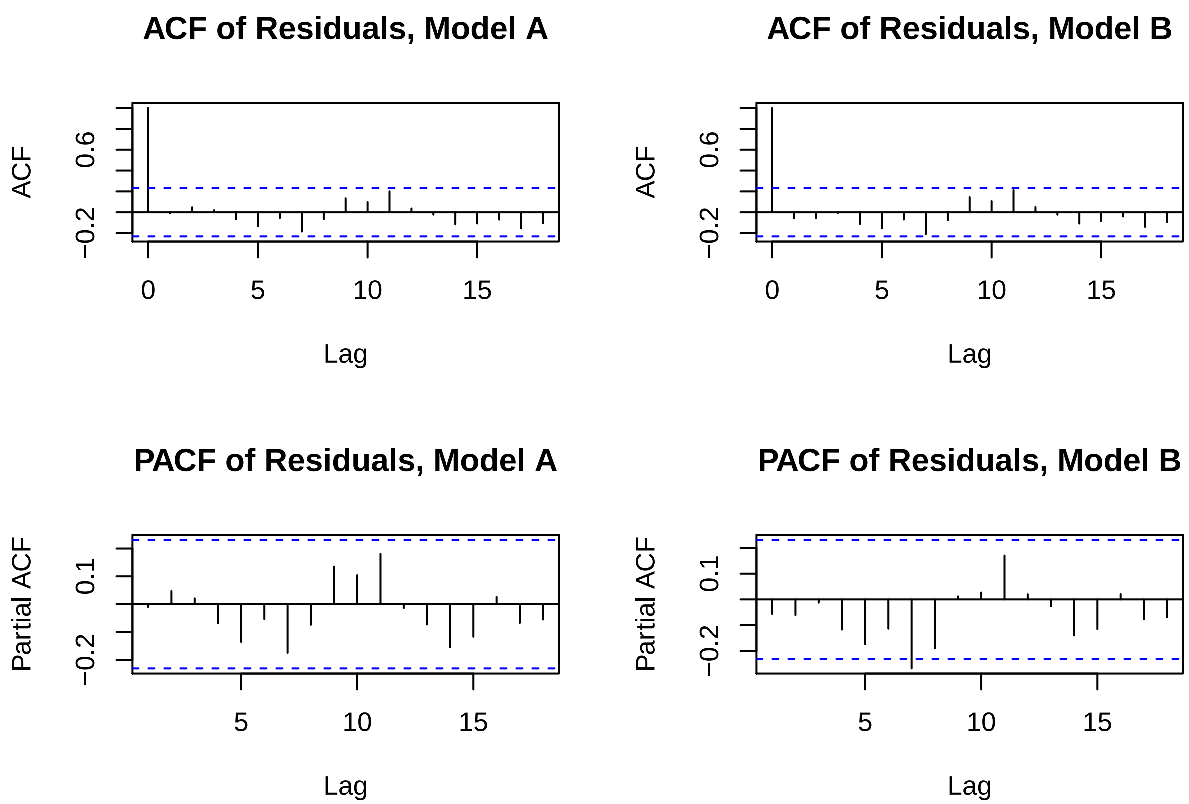


From the graphs above, Model B seems to be a better fit of the data. The graphs display data that resembles white noise with means $\mu_A = 0.1249$ and $\mu_B = 0.0335$. Plotting the Q-Q Plots and Histograms of the residuals can further display the distributions of models A and B.





After viewing the histograms and Q-Q plots, residuals Model A seems to have a more normal distribution. Residuals of Model B are left-skewed. Examining the ACF and PACF graphs of the residuals will reveal if the residuals are correlated.



These plots show that Model A has residuals that completely resemble white noise. Further tests that are performed to help with model diagnostics are the (1) Shapiro-Wilkes Normality Test, (2) Box-Pierce Test, (3) Ljung-Box Test, and (4) McLeod-Li Test.

Model A Tests

```
##
##  Shapiro-Wilk normality test
##
## data:  A.res
## W = 0.97665, p-value = 0.2004
```

```
##
##  Box-Pierce test
##
## data:  A.res
## X-squared = 4.7433, df = 4, p-value = 0.3147
```

```
##
##  Box-Ljung test
##
## data:  A.res
## X-squared = 5.328, df = 4, p-value = 0.2553
```

```
##
##  Box-Ljung test
##
## data:  (A.res)^2
## X-squared = 6.5406, df = 8, p-value = 0.5869
```

Model B Tests

```
##
##  Shapiro-Wilk normality test
##
## data:  B.res
## W = 0.94848, p-value = 0.005179
```

```
##
##  Box-Pierce test
##
## data:  B.res
## X-squared = 6.9983, df = 6, p-value = 0.321
```

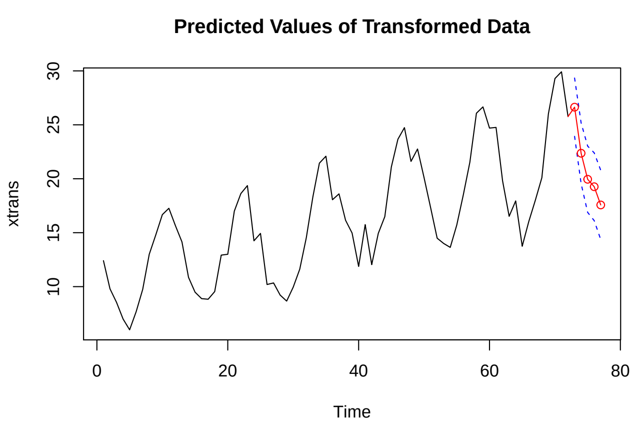
```
##
##  Box-Ljung test
##
## data:  B.res
## X-squared = 7.8235, df = 6, p-value = 0.2513
```

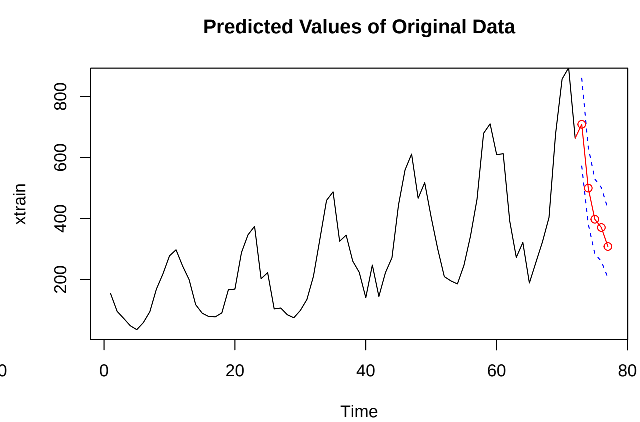
```
##
##  Box-Ljung test
##
## data:  (B.res)^2
## X-squared = 16.381, df = 8, p-value = 0.03724
```

Model A passes all tests because all p-values > 0.05. The residuals are normally distributed, uncorrelated, linearly independent, and non-linearly independent. Model B fails the Shapiro-Wilkes Normality Test and the McLeod-Li Test, meaning the residuals are not normally distributed and have some form of non-linear dependence. Model A passes diagnostic checking and will be used for forecasting.

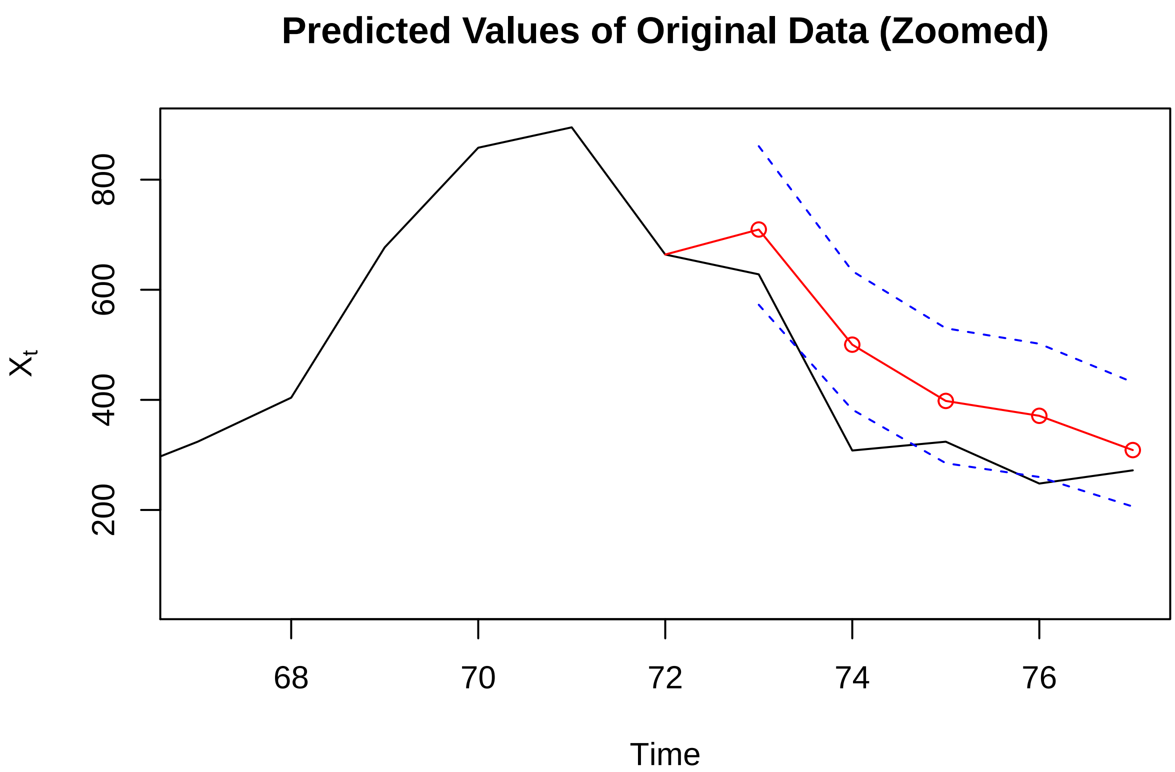
Forecasting

Using Model A, R can predict future values of our original time series.

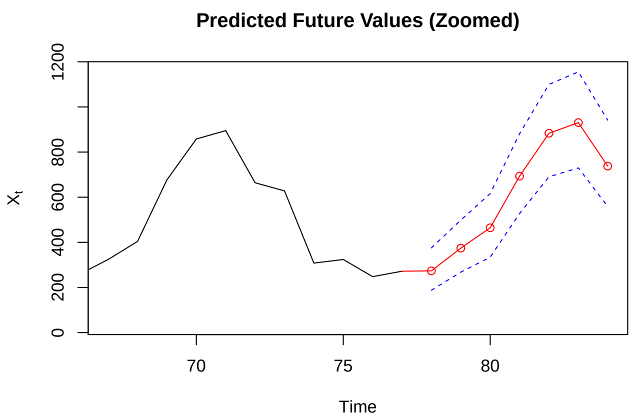


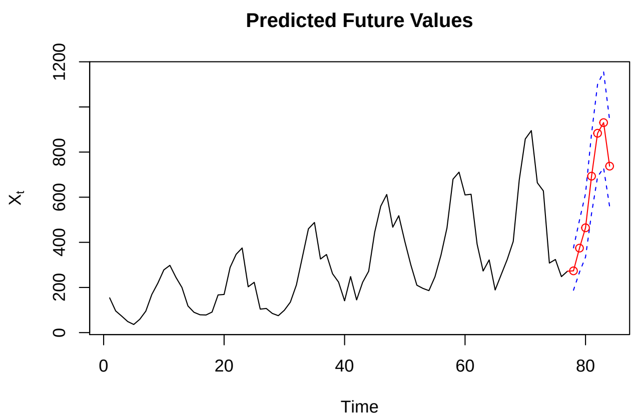


The transformed data has MSE = 8.6022. The original data has MSE = 13116.72.



The previous predictive model can be modified to predict values past the range of the test set.





Conclusion

The Box-Jenkins model of the data predicted future values of Company X's sales over the course of the rest of 1971. The predicted values had large MSE's, and the test data had points outside of the 95% prediction confidence interval. We can, however, assume that most of the predicted future data will lie within the 95% confidence interval provided except for a few outliers. We have still determined that the model

$$\nabla_1\nabla_{12}(1-0.2639_{(0.1263)}B-0.3596_{(0.1289)}B^2)\sqrt{U_t} = (1-1_{(0.0956)}B)(1-0.5127_{(0.2105)}B^{12})Z_t, \quad Z_t \sim N(0, 1.783)$$

is the best model for linear time series analysis on the data because of the normality and independence of residuals.If different transformations are performed on the data, such as logarithmic or Box-Cox, the residuals will display non-linear dependence and would require a different model representation for accurate prediction.The same goes for the model that was selected. Obviously, the model is not a perfect representation of the data, but it accurately reflects linear time series analysis using the Box-Jenkins Method. With more knowledge of time series analysis, a better model could be created to better predict sales of Company X. This project was completed with the help of Professor Feldman.

References

[1] R Core Team (2017). R: A language and environment for statistical computing. R Foundation for Statistical Computing,Vienna, Austria. URL https://www.R-project.org/.

[2] Hipel and McLeod (1994) "Sales of Company X, Jan. 1965 to May 1971".

[3] Feldman, R. 2021, *PSTAT 174 Lecture Notes*, Time Series PSTAT174, University of California, Santa Barbara.

[4] Shumay, R.H. & Stoffer, D.S., (2017), *Time Series Analysis and Its Applications: With R Examples*, Fourth Edition, Springer.

Appendix

1 Development Code

1.1 Necessary Libraries

```
library(tsdl)
library(MASS)
library(ggplot2)
library(ggfortify)
library(forecast)
library(MuMIn)
library(astsa)
```

1.2 Understanding Data

```
# get data
xsales = tsdl[[358]]

# data information
length(xsales)
```

```
## [1] 77
```

```
attr(xsales, "description")
```

```
## [1] "Sales of company X, Jan. 1965 to May 1971"
```

```
attr(xsales, "source")
```

```
## [1] "Hipel and McLeod (1994)"
```

```
# set to ts
xts = ts(xsales, start = c(1965,1), end = c(1971,5), frequency = 12)

# view original plot
ts.plot(xsales, main = "Sales of Company X, Jan 1965 - May 1971", ylab = expression(X[t]))
```

```
# create training and test sets
xts = ts(xsales, start = c(1965,1), end = c(1971,5), frequency = 12)
xtrain = xts[c(1:72)]
xtest = xts[c(73:77)]

# mean and variance of xtrain
mean(xtrain)
```

```
## [1] 295.9167
```

```
var(xtrain)
```

```
## [1] 40251.01
```

```
# view plot of training set
ts.plot(xtrain, main = expression(paste("Training Data ", U[t])))
xfit = lm(xtrain ~ as.numeric(1:length(xtrain)));abline(xfit, col = 'red')
abline(h = mean(xtrain), col = 'blue')
legend(0, 900, legend=c("Linear Trend", "Mean = 295.9167"), col=c("red", "blue"),
       lty=1, cex=0.8)
```

1.3 Transforming Data to Stationary Series

```
# view hist and acf of training data
hist(xtrain, col="light blue", xlab = expression(X[t]),
     main = expression("Histogram of U"[t]))
```

```
acf(xtrain, lag.max = 40, main = expression("ACF of U"[t]))
```

```
# box cox transform
bcTransform = boxcox(xtrain ~ as.numeric(1:length(xtrain)), plotit = TRUE)
```

```
lambda = bcTransform$x[which(bcTransform$y == max(bcTransform$y))]
lambda
```

```
## [1] 0.1818182
```

```
xtrain.bc = (1/lambda)*((xtrain^lambda)-1)

# log transform
xtrain.log = log(xtrain)

# sqrt transform
xtrain.sqrt = sqrt(xtrain)

# plot all transformations
par(mfrow = c(2,2))
par(mfrow = c(2,2))
ts.plot(xtrain, main = expression(paste("Original Data ", U[t])))
ts.plot(xtrain.bc, main = expression(paste("Box-Cox Transformation ",
                                     frac(1,lambda)(U[t]^lambda-1))))
ts.plot(xtrain.log, main = expression(paste("Log Transformation ", ln(U[t]))))
ts.plot(xtrain.sqrt, main = expression(paste("Square Root Transformation ", sqrt(U[t]))))
```

```
# plot all histograms
par(mfrow = c(2,2))
```

```
hist(xtrain, main = expression(paste("Histogram of ", U[t])))
hist(xtrain.bc, main = expression(paste("Histogram of BC", (U[t]))))
hist(xtrain.log, main = expression(paste("Histogram of ", ln(U[t]))))
hist(xtrain.sqrt, main = expression(paste("Histogram of ", sqrt(U[t]))))
```

```
# final transformed data
xtrans = xtrain.sqrt

# decomposition of data
decomp = decompose(ts(as.ts(xtrans), frequency = 12))
plot(decomp)
```

```
# differencing lag 12 to remove seasonality
xd12 = diff(xtrans, 12)
var(xd12)
```

```
## [1] 2.912886
```

```
mean(xd12)
```

```
## [1] 1.988875
```

```
# differencing at lag 1 to remove trend
xstat = diff(xd12, 1)
var(xstat)
```

```
## [1] 3.825941
```

```
mean(xstat)
```

```
## [1] -0.01122784
```

```
# plots of data
par(mfrow = c(2,2))

ts.plot(xtrain, main = expression(paste("Original Data ", U[t])))
xfit = lm(xtrain ~ as.numeric(1:length(xtrain)));abline(xfit, col = 'red')
abline(h = mean(xtrain), col = 'blue')

ts.plot(xtrans, main = expression(paste("Transformed Data ",sqrt(U[t]))))
xtrans.fit = lm(xtrans ~ as.numeric(1:length(xtrans)));abline(xtrans.fit, col = 'red')
abline(h = mean(xtrans), col = 'blue')

ts.plot(xd12, main = expression(paste(sqrt(U[t]), " Differenced at Lag 12")))
xd12.fit = lm(xd12~as.numeric(1:length(xd12)));abline(xd12.fit, col = "red")
abline(h=mean(xd12), col = "blue")

ts.plot(xstat, main = expression(paste(sqrt(U[t]), " Differenced at Lag 12 and 1")))
xstat.fit = lm(xstat~as.numeric(1:length(xstat)));abline(xstat.fit, col = "red")
abline(h=mean(xstat), col = "blue")
```

```
# acfs of different data
par(mfrow=c(2,2))
acf(xtrain, lag.max = 40, main = expression(paste("ACF of ", U[t])))
acf(xtrans, lag.max = 40, main = expression(paste("ACF of ", sqrt(U[t]))))
acf(xd12, lag.max = 40,
    main = expression(paste("ACF of ", nabla[12], sqrt(U[t]))))
acf(xstat, lag.max = 40,
```

```r
    main = expression(paste("ACF of ", nabla[1], nabla[12], sqrt(U[t]))))
```

```r
# histograms of different data
hist(xtrain, main = expression(paste("Histogram of ", U[t])))
hist(xtrans, main = expression(paste("Histogram of ", sqrt(U[t]))))
hist(xd12, main = expression(paste("Histogram of ", nabla[12], sqrt(U[t]))))
hist(xstat, main = expression(paste("Histogram of ", nabla[1], nabla[12], sqrt(U[t]))))
```

```r
# plot density histogram
hist(xstat, breaks = 15, col ='light blue', prob = TRUE,
     main = expression(paste("Histogram of ", nabla[1], nabla[12], sqrt(U[t]))))
curve(dnorm(x, mean(xstat), sqrt(var(xstat))), add = TRUE, col = 'blue')
```

1.4 Model Selection

```r
# acf and pacf of stationary data
acf(xstat, lag.max = 40,
    main = expression(paste("ACF of ", nabla[1], nabla[12], sqrt(U[t]))))
```

```r
pacf(xstat, lag.max = 40,
     main = expression(paste("PACF of ", nabla[1], nabla[12], sqrt(U[t]))))
```

```r
# finding lowest AICc
#for (i in 1:2){
#  for (j in c(0,1,8)){
#    for (k in 0:1){
#      for (l in ):1{
#        print(i); print(j); print(k); print(l);
#        print(AICc(arima(xtrans, order = c(i,1,j),
#                          seasonal = list(order = c(k,1,l),period = 12),
#                          method = "ML")))}}}}

# possible models
A = arima(xtrans, order=c(2,1,1),
          seasonal = list(order = c(0,1,1), period = 12),method="ML")
B = arima(xtrans, order=c(1,1,0),
          seasonal = list(order = c(0,1,1), period = 12), method="ML")
C = arima(xtrans, order=c(1,1,0),
          seasonal = list(order = c(1,1,1), period = 12), method="ML")
D = arima(xtrans, order=c(0,1,8),
          seasonal = list(order = c(0,1,0), period = 12), method = "ML")

# view model information
A
```

```
##
## Call:
## arima(x = xtrans, order = c(2, 1, 1), seasonal = list(order = c(0, 1, 1), period = 12),
##     method = "ML")
##
## Coefficients:
##          ar1     ar2      ma1     sma1
##       0.2639  0.3596  -1.0000  -0.5217
## s.e.  0.1263  0.1289   0.0956   0.2105
```

```
##
## sigma^2 estimated as 1.783:  log likelihood = -104.51,  aic = 219.02
```

```
B
```

```
##
## Call:
## arima(x = xtrans, order = c(1, 1, 0), seasonal = list(order = c(0, 1, 1), period = 12),
##     method = "ML")
##
## Coefficients:
##           ar1     sma1
##       -0.5434  -0.5403
## s.e.   0.1133   0.2133
##
## sigma^2 estimated as 2.035:  log likelihood = -106.92,  aic = 219.84
```

```
C
```

```
##
## Call:
## arima(x = xtrans, order = c(1, 1, 0), seasonal = list(order = c(1, 1, 1), period = 12),
##     method = "ML")
##
## Coefficients:
##           ar1   sar1     sma1
##       -0.5495  0.362  -0.9954
## s.e.   0.1118  0.201   1.2320
##
## sigma^2 estimated as 1.667:  log likelihood = -106.01,  aic = 220.01
```

```
D
```

```
##
## Call:
## arima(x = xtrans, order = c(0, 1, 8), seasonal = list(order = c(0, 1, 0), period = 12),
##     method = "ML")
##
## Coefficients:
##           ma1     ma2      ma3     ma4      ma5     ma6      ma7     ma8
##       -0.8288  0.3792  -0.2848  0.1225  -0.2691  0.0527  -0.4036  0.2319
## s.e.   0.1472  0.1633   0.1719  0.1638   0.1985  0.2019   0.1897  0.1810
##
## sigma^2 estimated as 1.872:  log likelihood = -104.3,  aic = 226.61
```

1.5 Diagnostic Checking

```
# plot roots function
plot.roots <- function(ar.roots=NULL, ma.roots=NULL, size=2, angles=FALSE, special=NULL, sqecial=NULL,my
{xylims <- c(-size,size)
      omegas <- seq(0,2*pi,pi/500)
      temp <- exp(complex(real=rep(0,length(omegas)),imag=omegas))
      plot(Re(temp),Im(temp),typ="l",xlab="x",ylab="y",xlim=xylims,ylim=xylims,main=main)
      abline(v=0,lty="dotted")
      abline(h=0,lty="dotted")
      if(!is.null(ar.roots))
```

```
        {
          points(Re(1/ar.roots),Im(1/ar.roots),col=first.col,pch=my.pch)
          points(Re(ar.roots),Im(ar.roots),col=second.col,pch=my.pch)
        }
      if(!is.null(ma.roots))
        {
          points(Re(1/ma.roots),Im(1/ma.roots),pch="*",cex=1.5,col=first.col)
          points(Re(ma.roots),Im(ma.roots),pch="*",cex=1.5,col=second.col)
        }
      if(angles)
        {
          if(!is.null(ar.roots))
            {
              abline(a=0,b=Im(ar.roots[1])/Re(ar.roots[1]),lty="dotted")
              abline(a=0,b=Im(ar.roots[2])/Re(ar.roots[2]),lty="dotted")
            }
          if(!is.null(ma.roots))
            {
              sapply(1:length(ma.roots), function(j) abline(a=0,b=Im(ma.roots[j])/Re(ma.roots[j]),lty="
            }
        }
      if(!is.null(special))
        {
          lines(Re(special),Im(special),lwd=2)
        }
      if(!is.null(sqecial))
        {
          lines(Re(sqecial),Im(sqecial),lwd=2)
        }
}

# plot roots, check stationarity of AR
par(mfrow = c(2,2))

A.arcoef = polyroot(c(1, 0.2639, 0.3596))
A.macoef = polyroot(c(1, -1))
A.smacoef = polyroot(c(1, -0.5217))

B.arcoef = polyroot(c(1, -0.5434))
B.smacoef = polyroot(c(1, -0.5403))

plot.roots(A.arcoef, A.macoef, main = "Roots of Model A, Non-Seasonal")
plot.roots(B.arcoef, NULL, main = "Roots of Model B, Non-Seasonal")
plot.roots(NULL, A.smacoef, main = "Roots of Model A, Seasonal")
plot.roots(NULL, B.smacoef, main = "Roots of Model B, Seasonal")
```

```
# set model
A.fit = A
B.fit = B

# get residuals
A.res = residuals(A.fit)
B.res = residuals(B.fit)
mean(A.res)
```

```
## [1] 0.1248805
```

```
var(A.res)
```

```
## [1] 1.46615
```

```
mean(B.res)
```

```
## [1] 0.03347683
```

```
var(B.res)
```

```
## [1] 1.690185
```

```
# plots of residuals
par(mfrow = c(2,2))

ts.plot(A.res, main = "Residuals of Model A")
A.res.fit = lm(A.res~as.numeric(1:length(A.res)));abline(A.res.fit, col ='red')
abline(h=mean(A.res), col='blue')

ts.plot(B.res, main = "Residuals of Model B")
B.res.fit = lm(B.res~as.numeric(1:length(B.res)));abline(B.res.fit, col ='red')
abline(h=mean(B.res), col='blue')
par(mfrow = c(2,2))
```

```
# hist of residuals
hist(A.res, col = "light blue", prob = TRUE,
     main = "Histogram of Residuals, Model A")
curve(dnorm(x, mean(A.res), sqrt(var(A.res))), add=TRUE, col = "blue")

hist(B.res, col = "light blue", prob = TRUE,
     main = "Histogram of Residuals, Model B")
curve(dnorm(x, mean(B.res), sqrt(var(B.res))), add=TRUE, col = "blue")


# qq plots
qqnorm(A.res, main = "Normal Q-Q Plot, Model A")
qqline(A.res,col="blue")

qqnorm(B.res, main = "Normal Q-Q Plot, Model B")
qqline(B.res,col="blue")
```

```
# acf and pacf of residuals
par(mfrow = c(2,2))

acf(A.res, main = "ACF of Residuals, Model A")
acf(B.res, main = "ACF of Residuals, Model B")

pacf(A.res, main = "PACF of Residuals, Model A")
pacf(B.res, main = "PACF of Residuals, Model B")
```

```
# shapiro test
shapiro.test(A.res)
```

```
##
##  Shapiro-Wilk normality test
##
```

```
## data:  A.res
## W = 0.97665, p-value = 0.2004
```

```
shapiro.test(B.res)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  B.res
## W = 0.94848, p-value = 0.005179
```

```
# box-pierce test
Box.test(A.res, lag = 8, type=c("Box-Pierce"), fitdf = 4)
```

```
##
##  Box-Pierce test
##
## data:  A.res
## X-squared = 4.7433, df = 4, p-value = 0.3147
```

```
Box.test(B.res, lag = 8, type=c("Box-Pierce"), fitdf = 2)
```

```
##
##  Box-Pierce test
##
## data:  B.res
## X-squared = 6.9983, df = 6, p-value = 0.321
```

```
# ljung-box test
Box.test(A.res, lag = 8, type=c("Ljung-Box"), fitdf = 4)
```

```
##
##  Box-Ljung test
##
## data:  A.res
## X-squared = 5.328, df = 4, p-value = 0.2553
```

```
Box.test(B.res, lag = 8, type=c("Ljung-Box"), fitdf = 2)
```

```
##
##  Box-Ljung test
##
## data:  B.res
## X-squared = 7.8235, df = 6, p-value = 0.2513
```

```
# mcleod-li test
Box.test((A.res)^2, lag = 8, type=c("Ljung-Box"), fitdf = 0)
```

```
##
##  Box-Ljung test
##
## data:  (A.res)^2
## X-squared = 6.5406, df = 8, p-value = 0.5869
```

```
Box.test((B.res)^2, lag = 8, type=c("Ljung-Box"), fitdf = 0)
```

```
##
##  Box-Ljung test
##
```

```
## data:  (B.res)^2
## X-squared = 16.381, df = 8, p-value = 0.03724
```

```r
# yule-walker
ar(A.res, aic = TRUE, order.max = NULL, method = c("yule-walker"))
```

```
##
## Call:
## ar(x = A.res, aic = TRUE, order.max = NULL, method = c("yule-walker"))
##
##
## Order selected 0  sigma^2 estimated as  1.466
```

```r
ar(B.res, aic = TRUE, order.max = NULL, method = c("yule-walker"))
```

```
##
## Call:
## ar(x = B.res, aic = TRUE, order.max = NULL, method = c("yule-walker"))
##
##
## Order selected 0  sigma^2 estimated as  1.69
```

1.5 Forecasting

```r
# forecast future values
fore.fit = A
forecast(fore.fit)
```

```
##    Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 73       26.63556 24.90297 28.36815 23.98580 29.28532
## 74       22.36703 20.56513 24.16893 19.61127 25.12279
## 75       19.95199 17.98383 21.92014 16.94195 22.96202
## 76       19.26201 17.24912 21.27490 16.18356 22.34046
## 77       17.57207 15.51353 19.63060 14.42381 20.72033
## 78       18.21599 16.13608 20.29590 15.03504 21.39694
## 79       20.36346 18.26680 22.46012 17.15689 23.57003
## 80       22.56029 20.45363 24.66695 19.33842 25.78215
## 81       27.22849 25.11442 29.34255 23.99530 30.46167
## 82       30.60340 28.48435 32.72244 27.36259 33.84420
## 83       31.27797 29.15501 33.40093 28.03118 34.52476
## 84       27.85039 25.72472 29.97606 24.59946 31.10132
## 85       28.44947 26.11587 30.78307 24.88054 32.01841
## 86       24.36934 22.01068 26.72799 20.76208 27.97659
## 87       21.90754 19.49959 24.31548 18.22490 25.59017
## 88       21.27297 18.84741 23.69854 17.56339 24.98255
## 89       19.58084 17.13773 22.02395 15.84442 23.31726
## 90       20.24411 17.79139 22.69683 16.49300 23.99523
## 91       22.39590 19.93536 24.85644 18.63282 26.15898
## 92       24.60083 22.13513 27.06653 20.82986 28.37179
## 93       29.27272 26.80294 31.74250 25.49552 33.04992
## 94       32.65152 30.17879 35.12424 28.86981 36.43322
## 95       33.32844 30.85310 35.80378 29.54274 37.11414
## 96       29.90288 27.42571 32.38005 26.11438 33.69138
```

```r
# transformed data with predictions
pred.A = predict(fore.fit, n.ahead = 5)
```

```r
up.A = pred.A$pred + 2*pred.A$se
low.A = pred.A$pred - 2*pred.A$se

ts.plot(xtrans, xlim = c(1, length(xtrans)+5), ylim = c(min(xtrans),max(up.A)),
        main = "Predicted Values of Transformed Data")
lines(up.A, col = "blue", lty = "dashed")
lines(low.A, col = "blue", lty = "dashed")
points((length(xtrans)+1):(length(xtrans)+5),pred.A$pred, col='red', pch = 1)
lines(c(xtrans[72],pred.A$pred), x = c(72:77), col = 'red')
lines(c(xtrans[72],sqrt(xtest)), x = c(72:77))
```

```r
# MSE training data
err.trans = (sqrt(xtest)-pred.A$pred)
mean(err.trans^2)
```

```
## [1] 8.602225
```

```r
# original data with predictions
pred.orig = (pred.A$pred)^2
up.orig = (up.A)^2
low.orig = (low.A)^2

ts.plot(xtrain, xlim = c(1, length(xtrain)+5), ylim = c(min(xtrain),max(up.orig)),
        main = "Predicted Values of Original Data")
lines(up.orig, col = 'blue', lty = 'dashed')
lines(low.orig, col = 'blue', lty = 'dashed')
points((length(xtrain)+1):(length(xtrain)+5), pred.orig, col = 'red')
lines(c(xtrain[72],pred.orig), x = c(72:77), col = 'red')
lines(c(xtrain[72],xtest), x = c(72:77))
```

```r
xnum = xts[0:77]
ts.plot(xnum, xlim = c(67,77), ylab = expression(X[t]),
        main = "Predicted Values of Original Data (Zoomed)")
lines(up.orig, col = 'blue', lty = 'dashed')
lines(low.orig, col = 'blue', lty = 'dashed')
points((length(xtrain)+1):(length(xtrain)+5), pred.orig, col="red")
lines(c(xnum[72],pred.orig), x = c(72:77), col = 'red')
```

```r
ftr.fit = arima(sqrt(xts), order=c(2,1,1),
                seasonal = list(order = c(0,1,1),period = 12), method="ML")

# MSE original data
err.orig = xtest - pred.orig
mean(err.orig^2)
```

```
## [1] 13116.72
```

```r
# predict future values past test set
ftr = (pred.ftr$pred)^2
up = (up.ftr[1:7])^2
low = (low.ftr[1:7])^2

ts.plot(xnum, xlim = c(67,84), ylim = c(min(xnum),max(up)),
        ylab = expression(X[t]), main = "Predicted Future Values (Zoomed)")
lines(up, x = c(78:84), col = 'blue', lty = 'dashed')
lines(low, x = c(78:84), col = 'blue', lty = 'dashed')
```

```r
points((length(xnum)+1):(length(xnum)+7), ftr, col="red")
lines(c(xnum[77],ftr), x = c(77:84), col = 'red')
```

```r
ts.plot(xnum, xlim = c(1,84), ylim = c(min(xnum),max(up)),
        ylab = expression(X[t]), main = "Predicted Future Values")
lines(up, x = c(78:84), col = 'blue', lty = 'dashed')
lines(low, x = c(78:84), col = 'blue', lty = 'dashed')
points((length(xnum)+1):(length(xnum)+7), ftr, col="red")
lines(c(xnum[77],ftr), x = c(77:84), col = 'red')
```