PSTAT 174

Final Project:

Quarterly Beer Production in Australia Time Series Analysis

By Micheal Alexander, Lindsay Hayden, Keanu Izadian, Randy Zhu, Sean Kinsman

Professor: Sudeep Bapat

Abstract

Beer Production in Australia is a $16.9 billion ($13B USD) a year industry and accounts for 1.02% of the country's overall GDP. Australian beer production is important to Australians; as roughly 80% of Australians consume alcohol and 95% of all beer sold in Australia is produced within the country creating upwards of 143,000 jobs.[1] In our report, we analyze quarterly beer production in Australia between the first quarter of 1956 and the second quarter of 1994. We first transform the data with a square-root transformation and then remove any traces of non-stationary behavior by decomposing our time series and negating the effects of seasonality and trend. After which, we fit a SARIMA model that we gauge by minimizing the AICc. Finally we perform diagnostic checks and residual analysis prior to our main goal of forecasting future behavior.

1.) Introduction

Beer is a fermented alcoholic beverage that usually, in its most basic form, is comprised of barley, hops, yeast, and water. All of these ingredients can be sourced locally which aid in production. Despite Australia only being the twenty-first largest beer producer in the world[2], the industry is important to the country nonetheless; as it is the largest producer of beer consumed within Australia by far, as well as contributing to the nation's overall GDP.

In our project, we seek to use a time-dependent dataset of beer production in Australia between the years 1956 and 1994 to predict how beer producers can produce the optimal volume of their product so as to not overproduce or underproduce. We believe that 38 years of data in a nation where beer is very popular can do just that. We obtained this dataset from datamarket.com.[3] The dataset contains 154 observations with 2 variables. One variable being time, represented by each quarter between 1956 and 1994, and the other being beer production measured in megaliters.

[1] https://www.brewers.org.au/beer-facts.html
[2] https://www.bluemarblecitizen.com/rankings/top-beer-producing-countries
[3] https://datamarket.com/data/set/22ry/quarterly-beer-production-in-australia-megalitres-march-1956-june-1994#!ds=22ry&display=line

From plotting the raw data as a time series using a statistical software (R) , we noticed that the series illustrates high levels of seasonality, as well as an upward trend. Our first step was to decompose our data and remove any traces of non-stationary behavior by removing seasonality, linear trend, and performing a square-root transformation. After analyzing the ACF, PACF and AICc of a variety of possible models, we decided to compare two possible models:

$$\text{Model 1: SARIMA}(2,1,2) \times (0, 1, 1)_4$$
$$\text{Model 2: SARIMA}(3,1,4) \times (0,1,0)_4$$

Prior to forecasting, we performed various diagnostic checks on the residual errors of our two models such as linear independence, serial correlation and normality. We used this model to achieve our main goal of forecasting future beer production. Our model accurately predicted the last 10 values of our dataset, as the behavior of the actual values was reflected in our forecasted values.

2.) Exploratory Analysis

2-1 Preliminary Data Exploration

As mentioned earlier, our data has 154 observations of two variables, the first being each quarter between 1956 and 1994, and the second being the production of beer measured in megaliters. After plotting the raw time series data, one can observe a strong presence of non-stationary behavior such as a positive linear trend and seasonality as illustrated by the repeated peak pattern and increasing mean (see Figure 1). Since our data spans close to four decades, the seasonal component is obscured. To acquire a better understanding, we used a seasonal plot (see Figure 2).

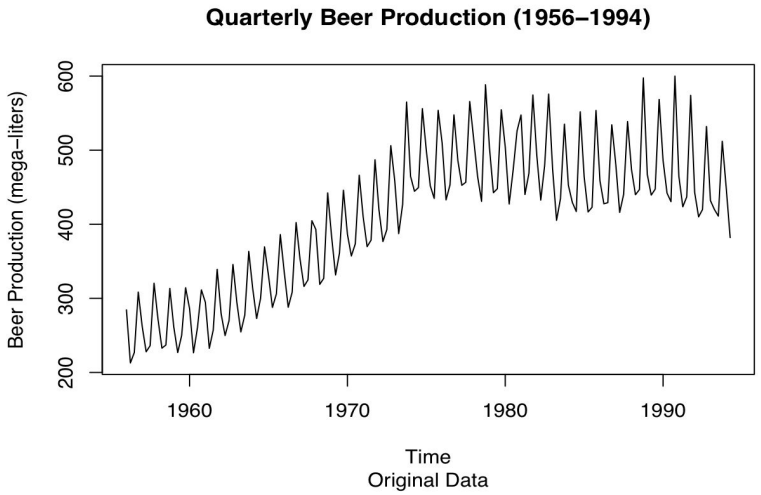


Figure 1: A time series plot the original time series data.

Illustrated below is our seasonal plot, which places our quarters on the x-axis and graphs the behavior of each year separately and independently. Our seasonal plot provides us with a digestible means of analyzing how beer production varies within a single year, which is particularly useful given the strong linear trend in the original time series plot.

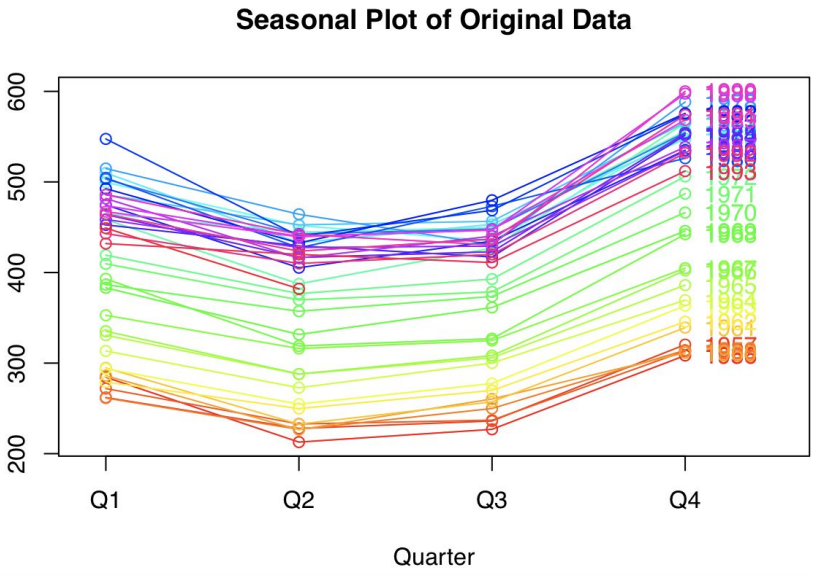


Figure 2: A seasonal plot of the original time series data.

In our seasonal plot, it appears that regardless of the year, beer production follows a parabolic pattern in that more beer is produced in the fourth quarter and less is produced in the second

quarter. The first quarter starts off strong and is followed by a decline in the second quarter and an exponential increase across the third and fourth quarters. This relatively parabolic phenomena could be explained by the fact that Australia's seasons are opposite those of the northern hemisphere, therefore the fourth quarter is summer. As a result, the beer brewers brew more beer in anticipation of people craving a cold beer in the hot summer months. Similarly, the beer brewers exhibit lowered production in the second quarter, as the weather gets colder, people consume less cold beverages. However, this seasonal plot alone is not sufficient enough to draw conclusions and make a forecast. We have to delve deeper, and decompose the time series and explicitly difference its seasonal and trend components to further improve our model.

2-2 Decomposing our Model

According to the decomposition plot, our initial observation regarding our data being non-stationary was correct in that it reveals a recurring seasonal component as well as an upward trend (see Figure 3). The seasonal component illustrates the various peaks and dips (which from our seasonal plot we saw took place during quarters 4 and 2 respectively). The trend component also does indicate an initial upward trend followed by a leveling out. We can now say for certain that our data is not stationary, and in the next section we will remove this non-stationarity through differencing and transformation.

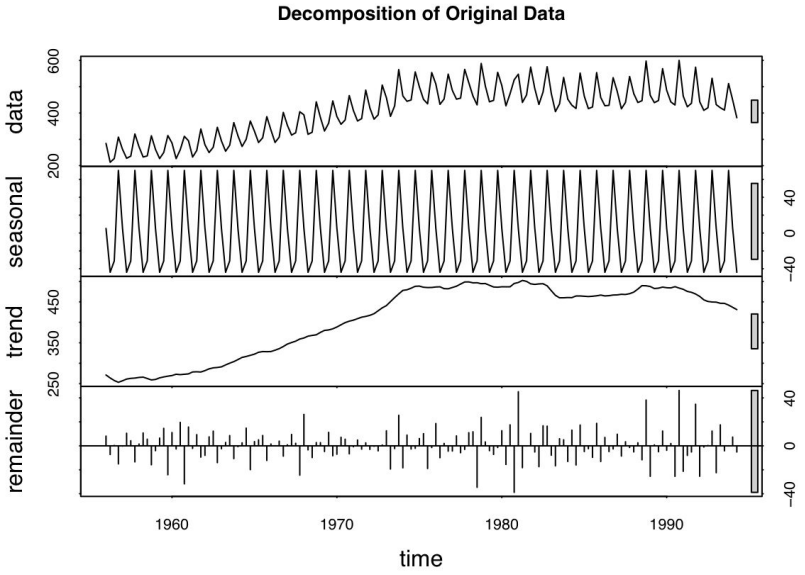


Figure 3: A plot of the decomposition of the original time series data.

3.) Data Transformation

In the following subsections, we will break down how we removed nonstationarity through applying a square root transformation to our data as well as differencing of our data to account for seasonality and linear trend.

3-1 Stabilizing the Variance Through Square Root Transformation

During our exploratory data analysis, we found that the variance of our response appears to be very high, specifically being 9,525.48. Performing a log transformation did not appear to have a significant influence on our variance. Therefore to properly stabilize/shrink this value, we performed a Box-Cox transformation using a built-in method in R. The figure below represents a graph of our data's log-likelihood as a function of λ. The λ is the parameter which we implement into our transformed data to minimize variance. The optimal λ will be the one which maximizes log-likelihood.

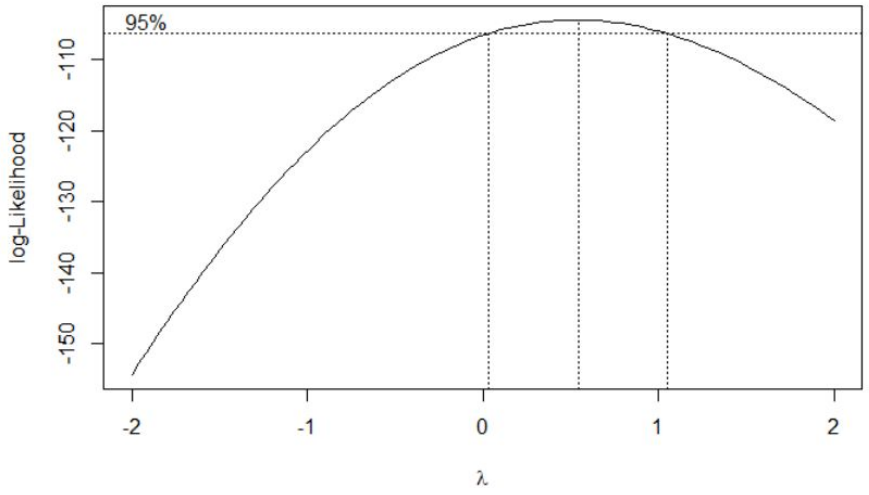


Figure 4: A plot of our log-likelihood to determine the best lambda for Box-Cox Transformation.

The above figure, gives us 0.545 as our optimal λ value. As this value is very close to .5 we can conclude that a square root transformation would be appropriate (see Figure 5). We then calculated our variance and saw that the square root transformation shrunk our variance from 9525.484 to 6.18299. Overall, the square root transform would be more intuitive and smooth to perform on our data, because it could easily be done and undone as we're going between the transformed and untransformed data.

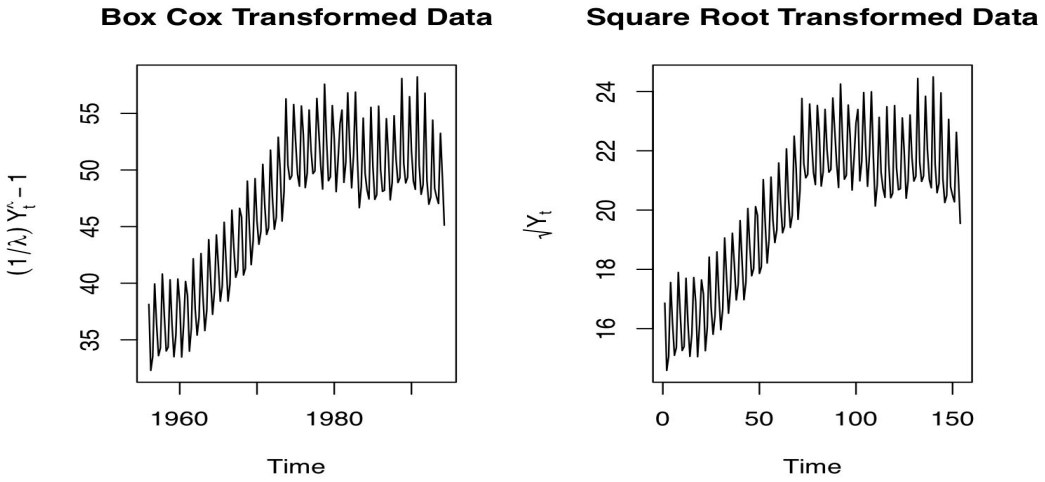


Figure 5: A plot for each transformation applied to our original time series.

3-2 Differencing Seasonality and Trend

Evidenced by the positive linear trend as well as the repeated seasonal trend shown in Figure 5, we proceeded to difference our data to account for these issues. Our first step was to difference our square-root transformed data at lag 1 due to evidence of a linear trend. This differencing yielded a time series with constant mean as well as a decrease in variance from 6.18299 to 2.789639, but a seasonal component is still present (see Figure 6).

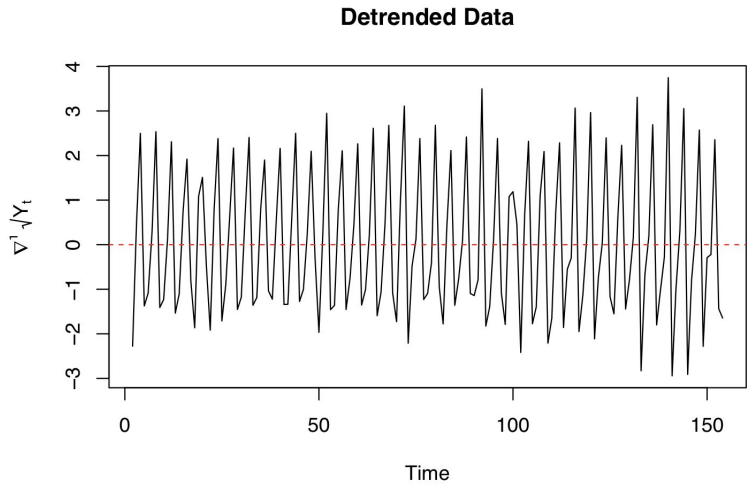


Figure 6: Data after differencing at lag 1

To address the issue of seasonality, we proceeded to difference the detrended data one more time at lag 4 since our seasonal component depends on quarters in a year. This differencing yielded a time series with no clear repeated pattern of peaks and valleys as well as a decrease in variance from 2.789639 to .4203988 thereby indicating that this differencing removed the seasonal component evidenced in our raw data (see Figure 7).

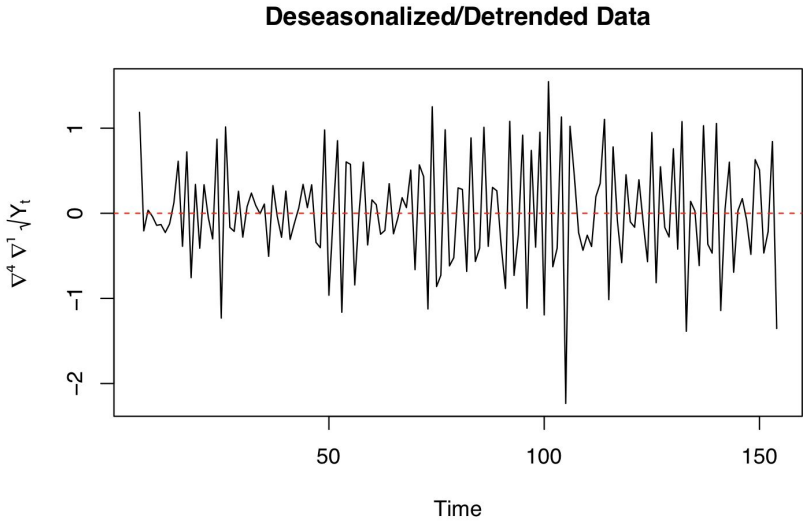


Figure 7: Data differenced at lag 1 and then differenced again at lag 4

To ensure that our data is in fact stationary, we proceeded to run two tests: another differencing at lag 1 to see if it yields a decrease in variance and an augmented Dickey-Fuller test. The first

test of differencing our detrended and deseasonalized data again yielded an increase of variance, implying the additional differencing is unnecessary.

The second test we're running is an augmented Dickey-Fuller test, also known as the unit root test. This is a statistical test that can be applied to time series to test how strongly a series is influenced by trend, which is a major aspect of nonstationarity. Our null hypothesis is that the data has some time dependence and is therefore not stationary, while the alternative hypothesis that the data is stationary. The figure below illustrates that after having done a Dicky-Fuller test using R, we observe a p-value less than our 0.01 significance level. Therefore we reject the null hypothesis, and we're able to finally conclude that our data is indeed stationary. Now that we have purged any aspects of nonstationarity from our model, we can go on to identifying our seasonal model and its parameters.

```
## Warning in adf.test(sqdif1dif4): p-value smaller than printed p-value
##
##  Augmented Dickey-Fuller Test
##
## data:  sqdif1dif4
## Dickey-Fuller = -8.7579, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

Figure 8: Augmented Dickey-Fuller test.

4.) Model Identification

4-1 Identifying Parameters

Figures 9 and 10 show the ACF and PACF plots of our data set after differencing once at lag 1 and once at lag 4 to remove the trend and seasonality respectively.

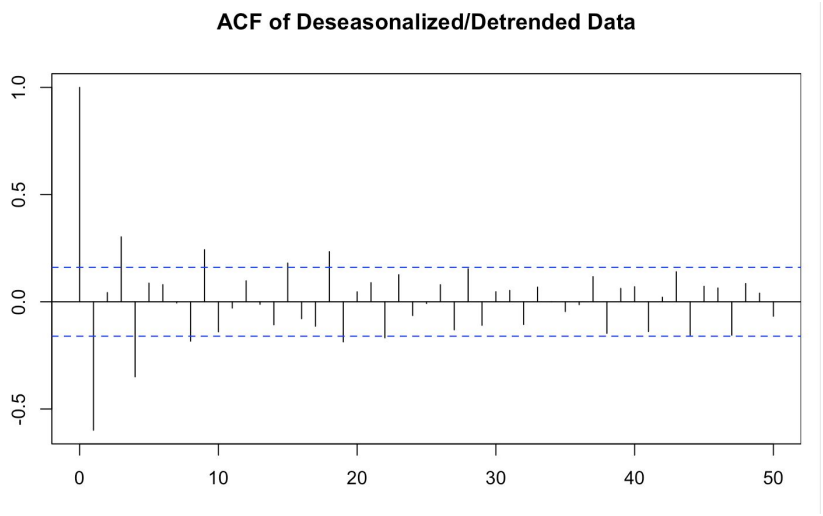


Figure 9: ACF of differenced data

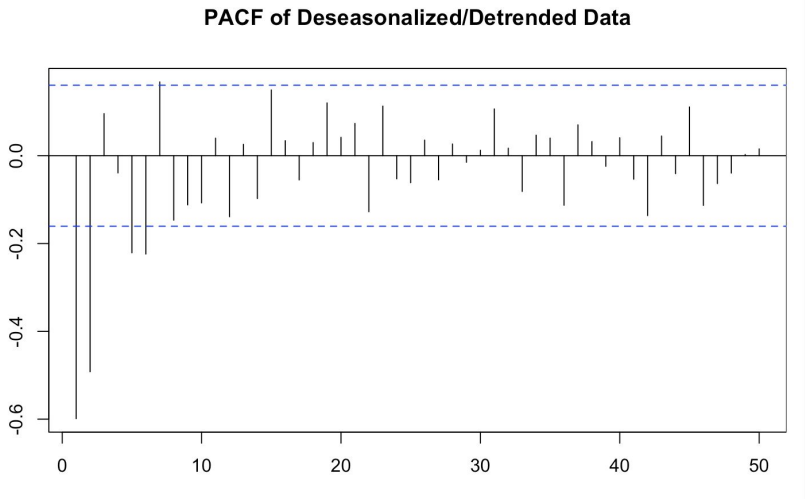


Figure 10: PACF of differenced data

For lags at multiples of our period (4), the PACF shows all are 0 and the ACF cuts off after each seasonal lag. Hence, we will fix our seasonal parameters: P = 0 and Q = 1. We also know that d = 1 and D = 1 (lag = 4) from when we differenced once at lag 1 and again at 4 as explained in prior sections.

Now we move to determine p and q. Looking at lag = 1, 2 or 3 we can see that both the PACF and ACF seem to tail off after lag = 2, suggesting that the model is some variation of SARIMA(p,1,q)

x (P,1,Q). We then used for loops to determine all possible combinations of ARIMA models up to p = q = 2, with fixed seasonal components, using their AICc's to determine the most valid models (see Figure 11).

```
0 0[1] 596.3526
0 1[1] 512.9377
0 2[1] 515.1489
1 0[1] 592.9315
1 1[1] 514.8151
1 2[1] 511.7264
2 0[1] 501.7827
2 1[1] 397.4942
2 2[1] 316.3044
```

Figure 11: AICc values of various ARIMA(p,1,q) combinations

Our for loops suggest that the model with the lowest AIC is when both p and q are 2, which would be a SARIMA(2, 1, 2) x $(0, 1, 1)_4$.

We also chose to fit an auto arima model to our data which proposed SARIMA(3, 1, 4) x $(0, 1, 1)_4$ as the best model. We will thus compare the following two models:

Model 1: SARIMA(2, 1, 2) x $(0, 1, 1)_4$
Model 2: SARIMA(3, 1, 4) x $(0, 1, 1)_4$

4-2 Finding the Coefficients

To estimate the coefficients of our models, we used the MLE method. It produced the following results for our models:

Components	Model 1 SARIMA(2, 1, 2) x $(0, 1, 1)_4$ Coefficients	Model 2 SARIMA(3, 1, 4) x $(0, 1, 0)_4$ Coefficients
AR(1)	0.1636	-1.0314
AR(2)	0.0701	-0.6560

AR(3)	-	-0.2806
MA(1)	-1.1306	0.0754
MA(2)	0.5095	0.0829
MA(3)	-	0.0589
MA(4)	-	-0.8155
SMA(1)	-0.8404	-

The algebraic representation of our first model is:

$$(1 - 0.1636B - 0.0701B^2)\, X_t = (1 - 1.1306B + 0.5095B^2)\,(1 - 0.8404B^4)\, Z_t$$

The algebraic representation of our second model is:

$$(1 + 1.0314B + 0.6560B^2 + 0.28068B^3)\, X_t = (1 + 0.0754B + 0.0829B^2 + 0.0589B^3 - 0.8155B^4) Z_t$$

5.) Diagnostics

The diagnostic tests we ran on our models were checking for normality, serial correlation, and for heteroskedasticity in our residuals.

5-1 Normality

To check normality, we conducted a Shapiro-Wilkes test as well as observed the behavior of the histogram and normal QQ plot of our residuals.

```
        Shapiro-Wilk normality test

data:  r1
W = 0.99242, p-value = 0.5922


        Shapiro-Wilk normality test

data:  r2
W = 0.99282, p-value = 0.6624
```

Figure 12: Results of Shapiro-Wilk Test on both models.

Both models passed the Shapiro-Wilk normality test with p-values greater than 0.05. Therefore, we accept the null hypothesis that the residuals are normal. Next, we assessed the residuals behavior via histograms and quantile-quantile plots (see Figures 13 and 14).

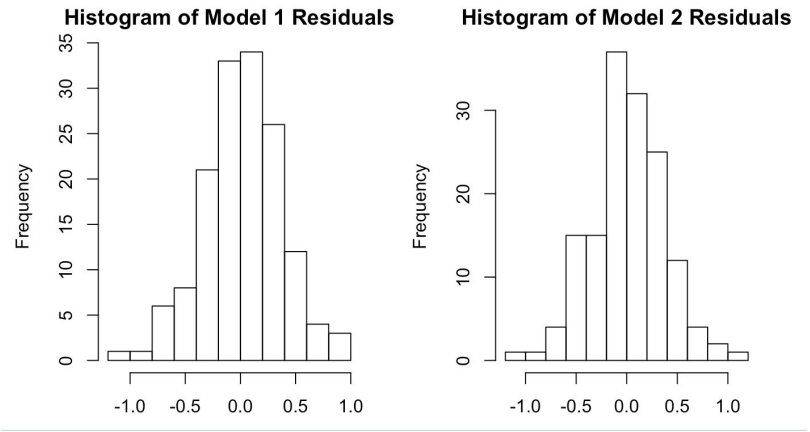


Figure 13: Histogram of residuals to check for normality

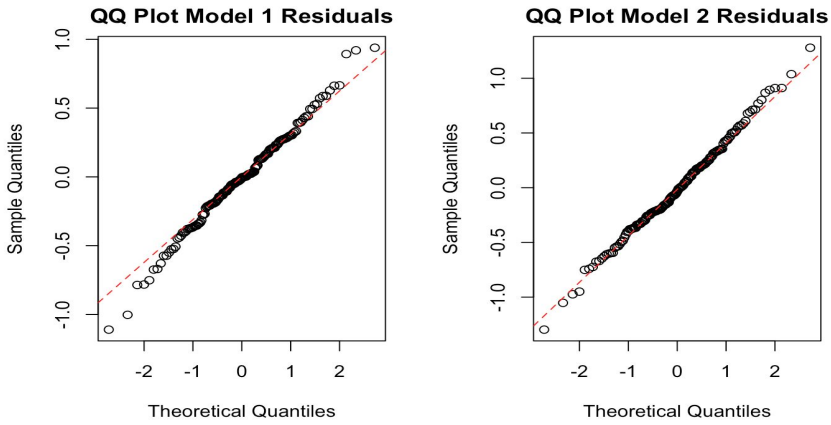


Figure 14: QQ-Plot of residuals to check for normality

The histograms of each model and their normal quantile-quantile plots also corroborate the conclusion that the residuals are normal. The histogram has the rough shape of a normal distribution, and most of the points on the QQ plots are on the QQ line.

5-2 Serial Correlation

Serial correlation was diagnosed using the Ljung-Box and Box-Pierce tests, the results of which are seen in Figure 15.

```
        Box-Ljung test

data:  r1
X-squared = 5.5951, df = 8.4097, p-value = 0.7305


        Box-Ljung test

data:  r2
X-squared = 5.2267, df = 9.4097, p-value = 0.8415


        Box-Pierce test

data:  r1
X-squared = 5.2295, df = 8.4097, p-value = 0.7684


        Box-Pierce test

data:  r2
X-squared = 4.859, df = 9.4097, p-value = 0.8708
```

Figure 15: Results of Box-Ljung and Box-Pierce tests

Both models pass because the p-values are greater than 0.05. We conclude that the residuals of both models are not serially correlated.

5-3 Heteroskedasticity

Next we checked for heteroskedasticity of our residuals by looking at the ACF and PACF plots of the squared residuals.

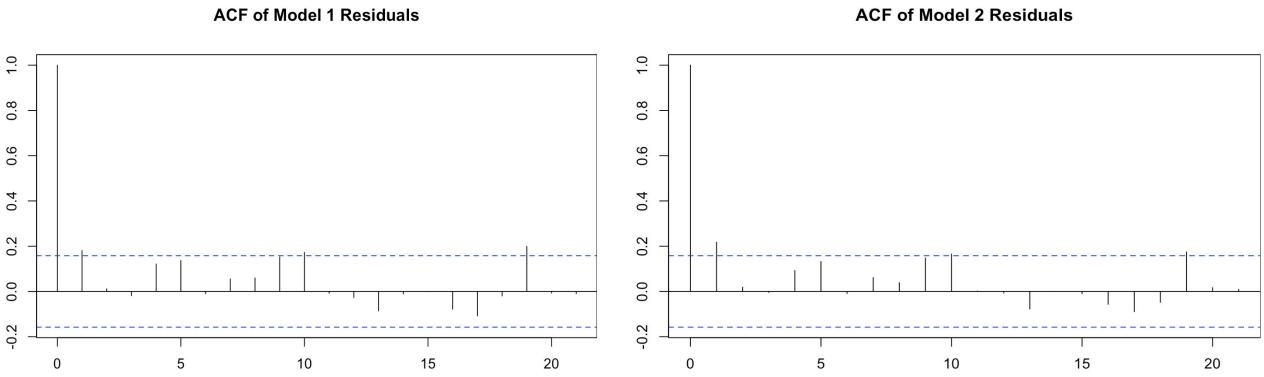


Figure 16: ACF of residuals for model 1 and model 2

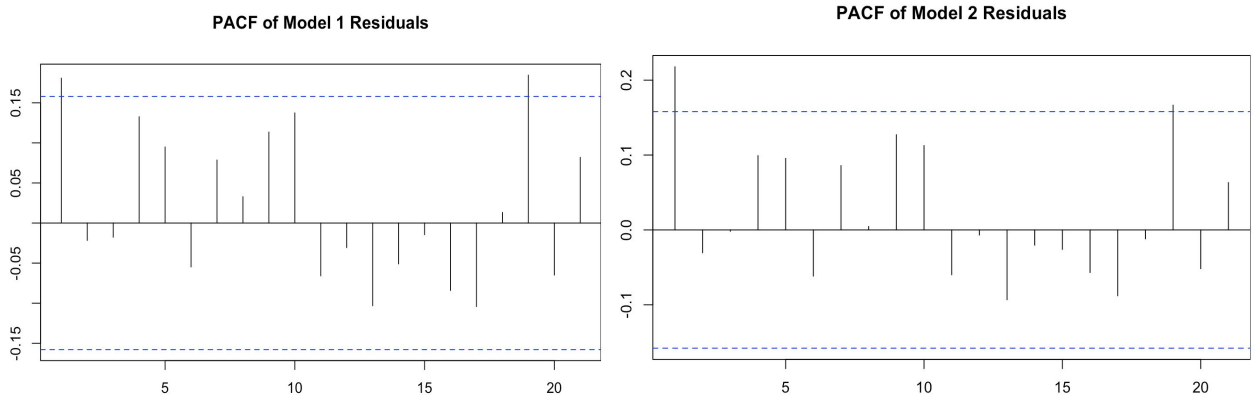


Figure 17: PACF of residuals for model 1 and model 2.

In the ACF plots, the lag spikes for Model 1 are barely outside of the confidence interval whereas the lag spikes for Model 2 are more clearly outside the bounds of the interval. The same behavior is found in the PACF plots. Because the lags are barely outside the bounds of the 95% confidence interval, we decided this was not a significant enough deviation to invalidate either model. We also tried fitting many other models and conducted this test, but the spikes were consistently outside of the bounds with each model.

After running diagnostic checks on both models, we determined that Model 1, SARIMA(2,1,2) x $(0,1,1)_4$ was our best model. Even though Model 2 passed the same amount of diagnostic checks, the Principle of Parsimony suggests we use the model with fewer parameters. This combined with the fact that Model 1 had a slightly lower AIC (138.66 vs 141.17) allows us to confidently

choose Model 1 as our final model. This also means that the model obtained by using the AICc metric is the same one that was suggested by our ACF and PACF plots.

Even though our final model is not completely without changing variance, we determined that the model was still satisfactory based on the performance of the residuals in other tests. For some that exceeds the bound, they can be seen as outliers in our dataset.

6.) Forecasting

This part is the main purpose of our project. We removed the last 10 observations in the original data set to be used as testing data and trained our model on the rest of it. After applying our model, the following two graphs show the forecasted values on the original data and transformed data, respectively.

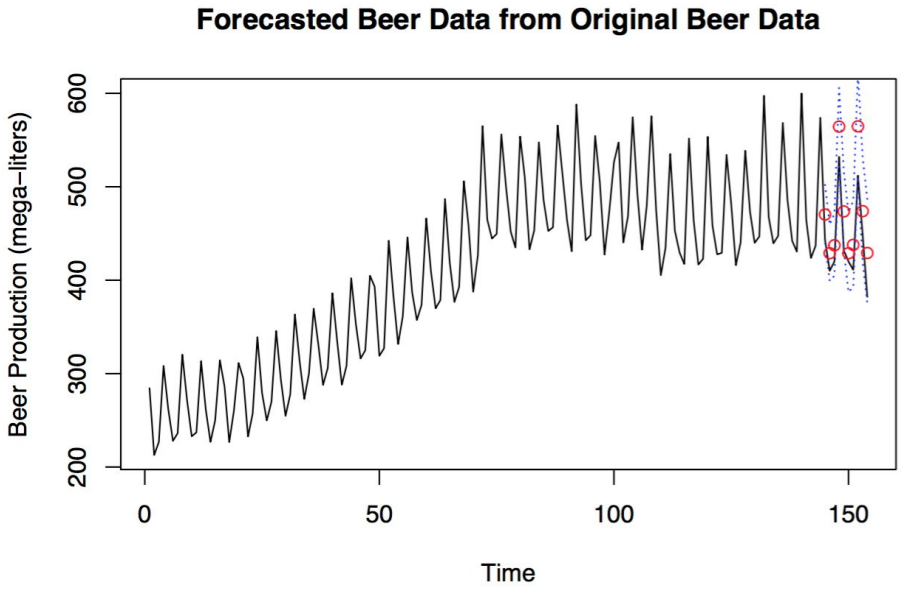


Figure 18: Forecasted values based on original time series data.

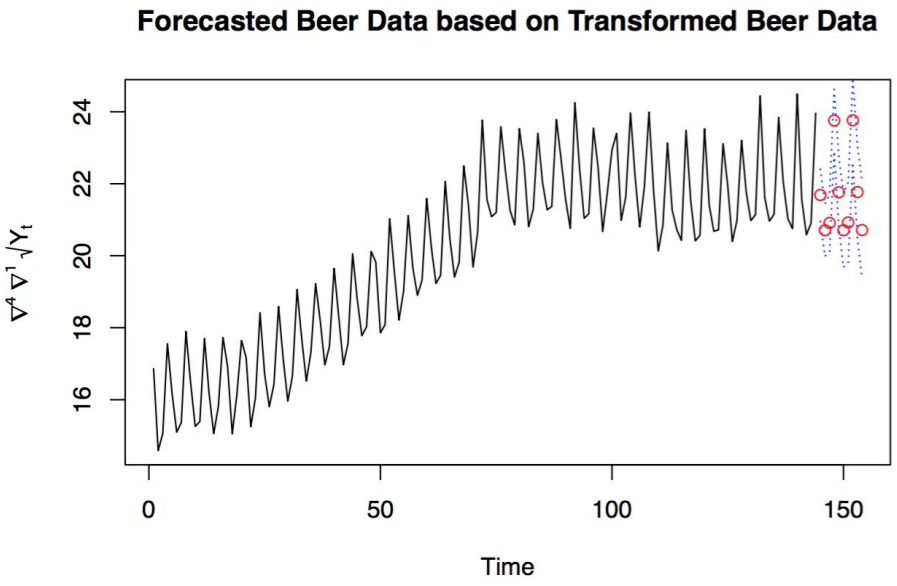


Figure 19: Forecasted values based on transformed time series data.

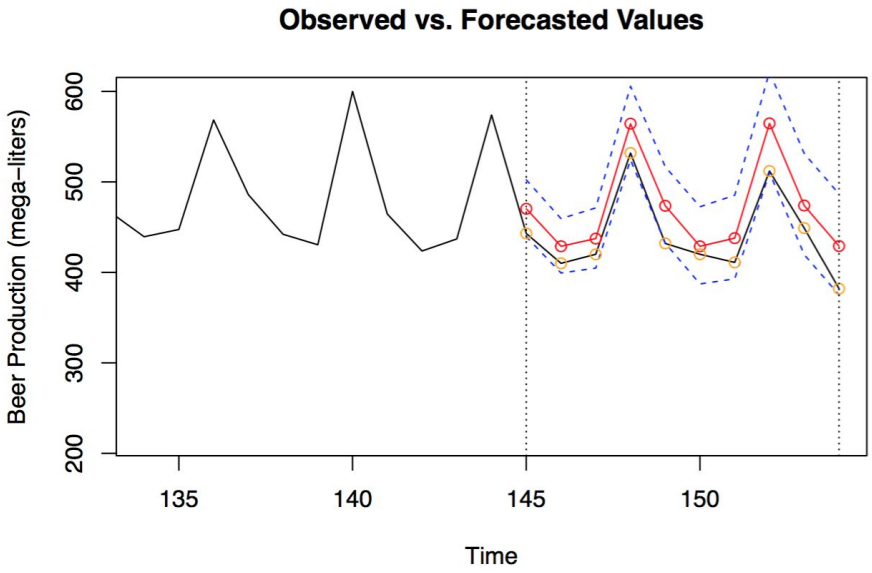


Figure 20: Comparison of forecasted values against observed values.

From these plots we can see that the forecasted values based on our SARIMA model are within the 95% confidence interval. The behavior of the forecasted values is very much the same as the actual values. Both suggest a sudden increase and decrease in production of beer, confirming that our model was able to successfully forecast beer production with our dataset.

7.) Future Study

Though our model acts decently when predicting future beer production, there are still some factors that we can't include within the model. First, those major incidents that would cause the time series to act abnormally. These incidents cannot be eliminated by differencing. Secondly, our sample size is not large enough. We only collected the data from 1956 to 1994. If we have a larger sample size the model will be much more precise. Thirdly, this time period is dated back more than 20 years ago so it will be obsolete if we try to predict the latest beer production. We can use some other statistical methods to eliminate these factors such as machine learning or regression analysis.

8.) Conclusion

To build a forecasting model, we take out the last ten values of the original dataset and train our model on the rest of it. To make the series stationary, we difference the data at d=1 and lag=4. We also take the square root transform. We then used auto arima and for loop to filtrate the two potential models with lowest AICc. We run several diagnostics checks on them and pick the one with lower AICc to be our final model. After applying the model to forecast, the result shows that our model gives very reliable predictions. This result validates our work and thus indicates that our project is successful.

9.) Appendix

PSTAT 174 Final Project

Micheal Alexander, Lindsay Hayden, Keanu Izadian, Randy Zhu, Sean Kinsman
March 14, 2019

Data Setup

Load libraries

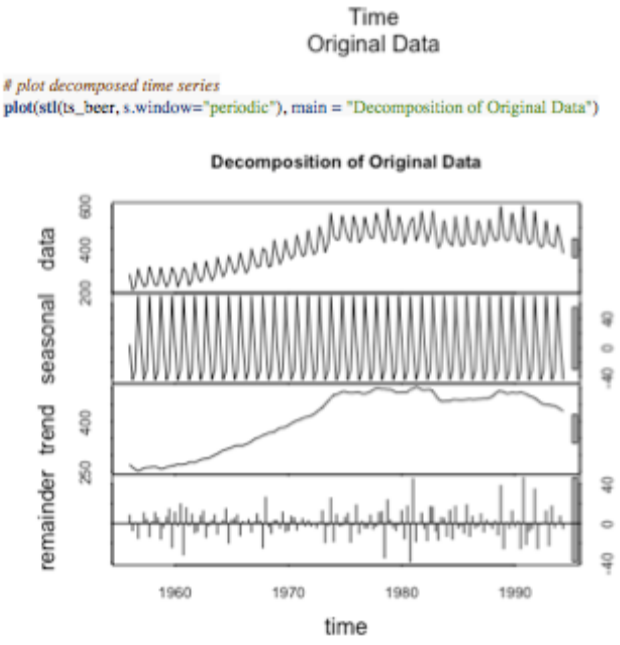
```
library(astsa)
library(forecast)
library(ggplot2)
library(tseries)
library(MASS)
library(qpcR)
```

Load data

```
beer = na.omit(read.csv("/Users/dailyuse/Desktop/beer.csv"))
colnames(beer) = c("quarters", "beer")

head(beer)
##   quarters  beer
## 1   1956Q1 284.4
## 2   1956Q2 212.8
## 3   1956Q3 226.9
## 4   1956Q4 308.4
## 5   1957Q1 262.0
## 6   1957Q2 227.9
tail(beer)
##     quarters beer
## 149   1993Q1  432
## 150   1993Q2  420
## 151   1993Q3  411
## 152   1993Q4  512
## 153   1994Q1  449
## 154   1994Q2  382
```

Explore original data

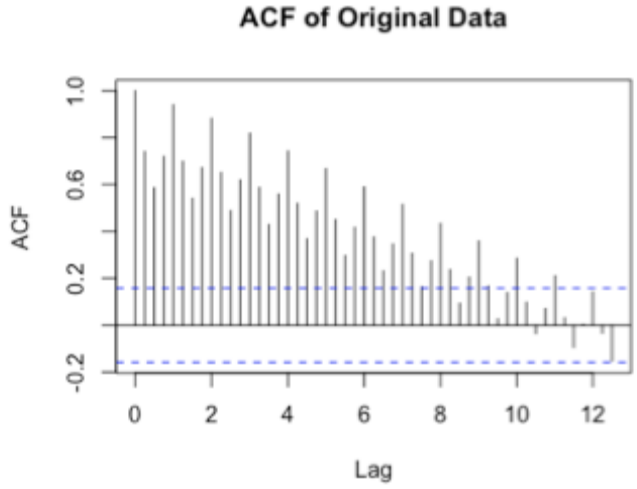
```
# plot original time series
ts_beer = ts(data = beer$beer, start = c(1956, 1), end = c(1994, 2), frequency = 4)
plot(ts_beer, ylab = "Beer Production (mega-liters)",
   main = "Quarterly Beer Production (1956-1994)",
   sub = "Original Data")
```



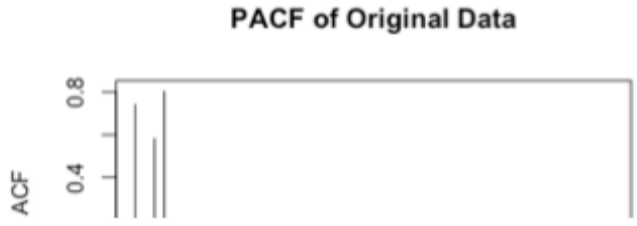

```
# plot decomposed time series
plot(stl(ts_beer, s.window="periodic"), main = "Decomposition of Original Data")
```

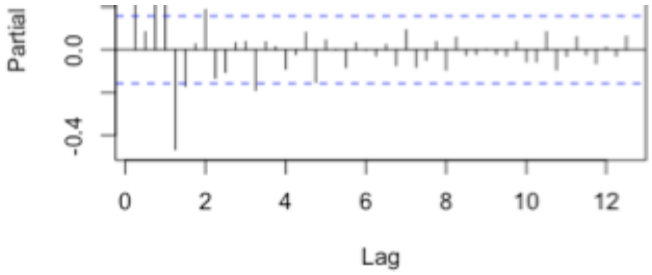


```
# plot acf and pacf
acf(ts_beer, main="ACF of Original Data", lag.max = 50)
```

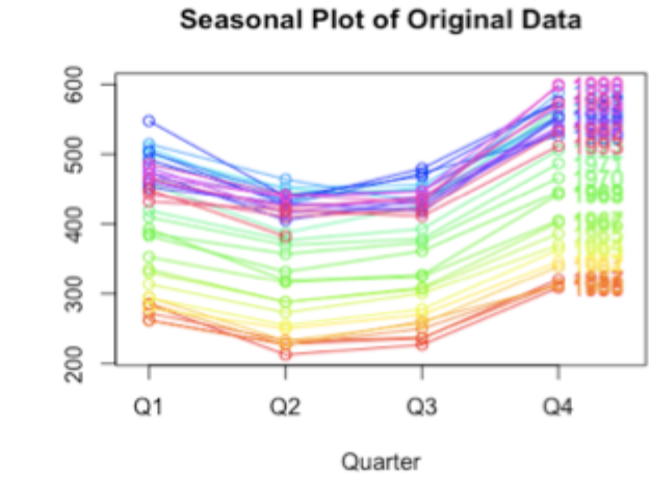



```
pacf(ts_beer, main="PACF of Original Data", lag.max = 50)
```






```
# plot seasonal
seasonplot(ts_beer, 4, col = rainbow(39), year.labels = TRUE,
       main = "Seasonal Plot of Original Data")
```

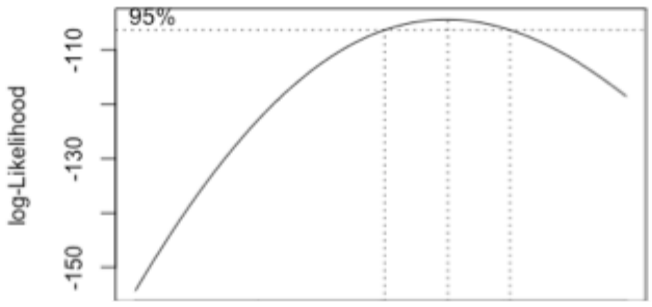


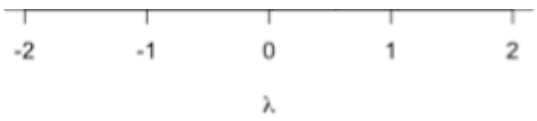

```
# calculate variance
var(ts_beer)
## [1] 9525.484
```

Variance transformation

Box-Cox transformation

```
# plot boxcox
bcTransform = boxcox(ts_beer ~ as.numeric(1:length(ts_beer)))
```





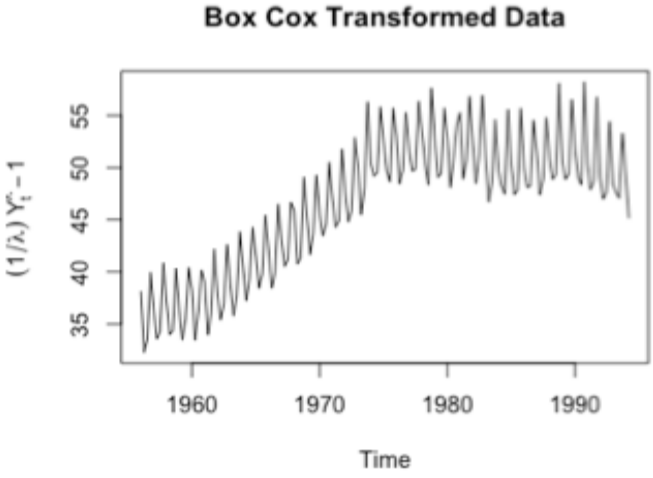

```
# find best lambda
(lambda = bcTransform$x[which(bcTransform$y == max(bcTransform$y))])
## [1] 0.5454545
# transform data
bc_beer = (1/lambda)*(ts_beer^lambda-1)

# calculate variance
var(bc_beer)
## [1] 42.43582
```

Square root transformation

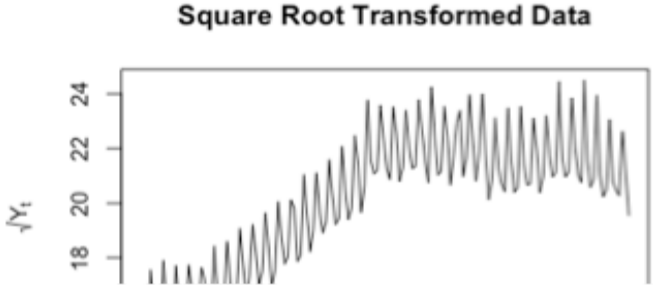
```
# apply square root transformation
sq_beer = as.ts(beer$beer**(1/2))

var(sq_beer)
## [1] 6.182991
```

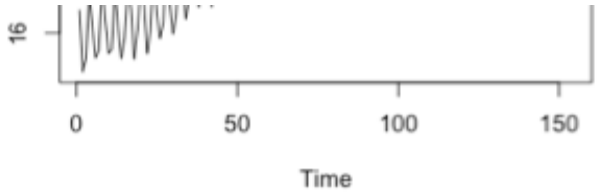
Comparision

```
# plot boxcox time series
plot(bc_beer, ylab = expression((1/lambda)~Y[t]^lambda-1),
   main = "Box Cox Transformed Data")
```




```
# plot square root time series
plot(sq_beer, ylab = expression(sqrt(Y[t])),
   main = "Square Root Transformed Data")
```

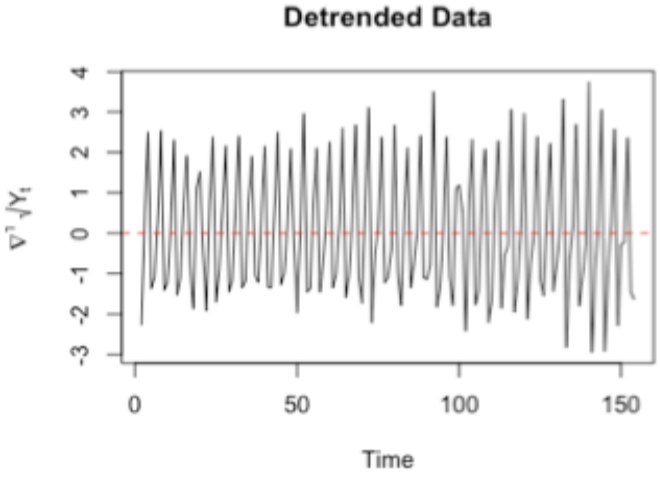




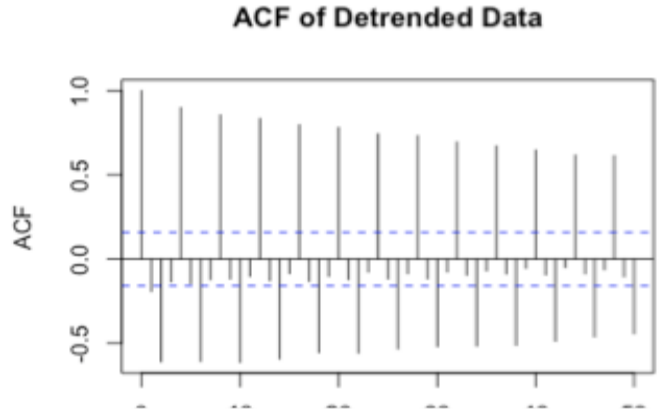
Since both are very similar in transforming our data, we will continue with the square root transformation due to it being a simpler transformation.

Remove trend

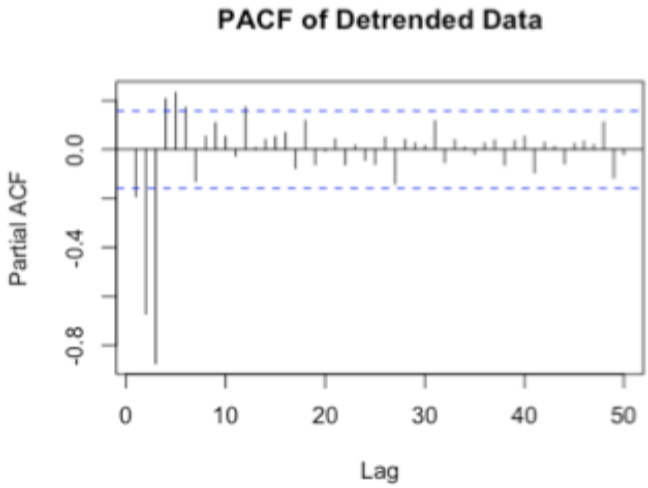
```
# difference at lag 1
sqdif1 <- diff(sq_beer, differences = 1)

# plot differenced data
plot(sqdif1,
   ylab = expression(nabla^{1}~sqrt(Y[t])),
   main = "Detrended Data")
abline(h = 0, lty = 2, col = 2)
```




```
# plot acf and pacf of differenced time series
acf(sqdif1, main="ACF of Detrended Data", lag.max = 50)
```



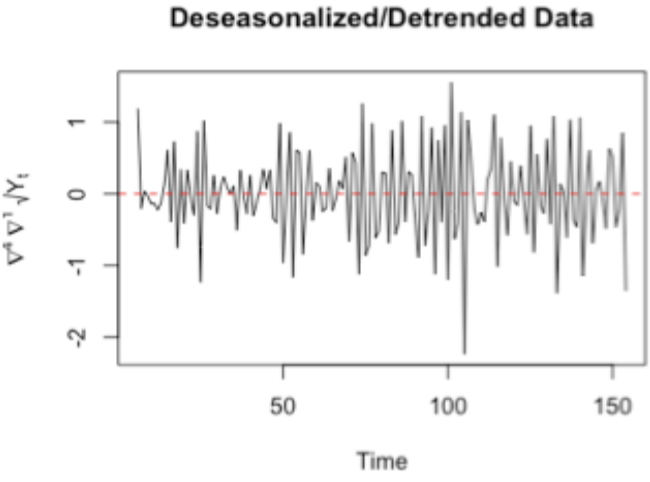

```
pacf(sqdif1, main="PACF of Detrended Data", lag.max = 50)
```




```
# calculate variance
var(sqdif1)
## [1] 2.789639
```

Remove seasonality

```
# difference at lag 4
sqdif1dif4 <- diff(sqdif1, lag = 4, differences = 1)

# plot differenced data
plot(sqdif1dif4,
   ylab = expression(nabla^{4}~nabla^{1}~sqrt(Y[t])),
   main = "Deseasonalized/Detrended Data")
abline(h = 0, lty = 2, col = 2)
```



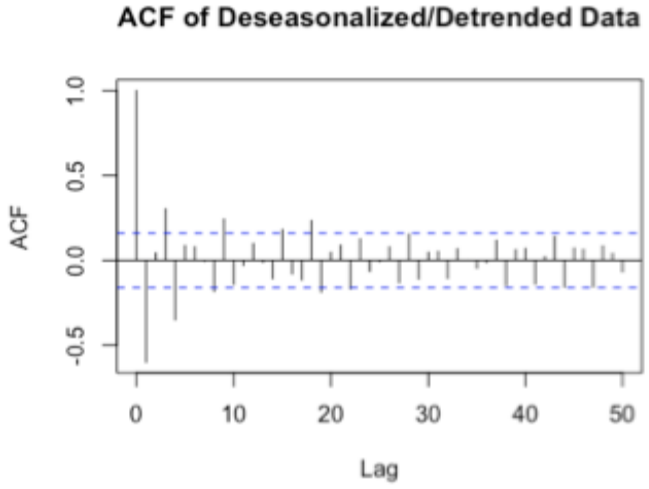
```
# calculate variance
var(sqdif1dif4)
## [1] 0.4203988
#see if further differencing needed
var(diff(sqdif1dif4, 1)) < var(sqdif1dif4)
## [1] FALSE
```

Since further differencing is not warranted, we assume our model is stationary but will verify using a Dicky Fuller test.

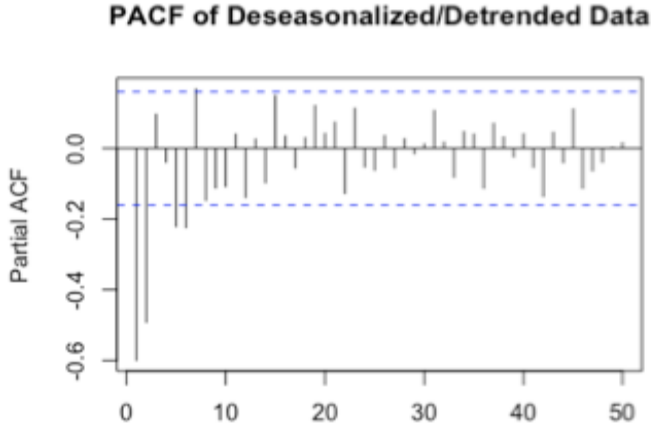
```
# check data is stationary
adf.test(sqdif1dif4) #yes
## Warning in adf.test(sqdif1dif4): p-value smaller than printed p-value
##
##  Augmented Dickey-Fuller Test
##
## data:  sqdif1dif4
## Dickey-Fuller = -8.7579, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

Model Selection

```
# plot acf and pacf of differenced time series
acf(sqdif1dif4, main="ACF of Deseasonalized/Detrended Data", lag.max = 50)
```




```
pacf(sqdif1dif4, main="PACF of Deseasonalized/Detrended Data", lag.max = 50)
```



Lag

For lags at multiples of our period (4), the PACF shows all are 0 and the ACF seems to tail / cut off after 1. Hence, P = 0 and Q = 1. We also know that d = 1 and D = 1 (lag = 4). Now we move to determine p and q. Looking at lag = 1, 2 or 3 we can see that both the PACF and ACF seem to tail off after lag = 2, so we will check models up to p = 2 and q = 2.

```
# compare AIC
for (i in 0:2)
{
 for (j in 0:2)
 {
  cat(i,j, sep = " ")
  print(AICc(arima(sq_beer, order = c(i,1,j),
                   seasonal = list(order = c(0,1,1), lag = 4), method = "ML")))
 }
}
## 0 0[1] 596.3526
## 0 1[1] 512.9377
## 0 2[1] 515.1489
## 1 0[1] 592.9315
## 1 1[1] 514.8151
## 1 2[1] 511.7264
## 2 0[1] 501.7827
## 2 1[1] 397.4942
## 2 2[1] 316.3044
## first: 2 2 316.3044
## second: 2 1 397.4942
## third: 2 0 501.7827
(mod = arima(sq_beer, order = c(2,1,2), seasonal = list(order = c(0,1,1), period = 4), method =
"ML"))
##
## Call:
## arima(x = sq_beer, order = c(2, 1, 2), seasonal = list(order = c(0, 1, 1), period = 4),
##     method = "ML")
##
## Coefficients:
##         ar1     ar2     ma1     ma2     sma1
##      0.1636  0.0701  -1.1306  0.5095  -0.8404
## s.e.  0.2523  0.1601   0.2384  0.1402   0.0553
##
## sigma^2 estimated as 0.1323:  log likelihood = -63.33,  aic = 138.66
AICc(mod)
## [1] 139.0685
(auto_mod = auto.arima(sqdif1dif4, d=0))
## Series: sqdif1dif4
## ARIMA(3,0,4) with zero mean
##
## Coefficients:
##         ar1     ar2     ar3     ma1     ma2     ma3     ma4
##      -1.0314  -0.6560  -0.2806  0.0754  0.0829  0.0589  -0.8155
## s.e.  0.1076   0.1358   0.0997  0.0722  0.0624  0.0606   0.0620
##
## sigma^2 estimated as 0.1372:  log likelihood=-62.59
## AIC=141.17   AICc=142.2   BIC=165.2
AICc(auto_mod)
## [1] 141.9659
```

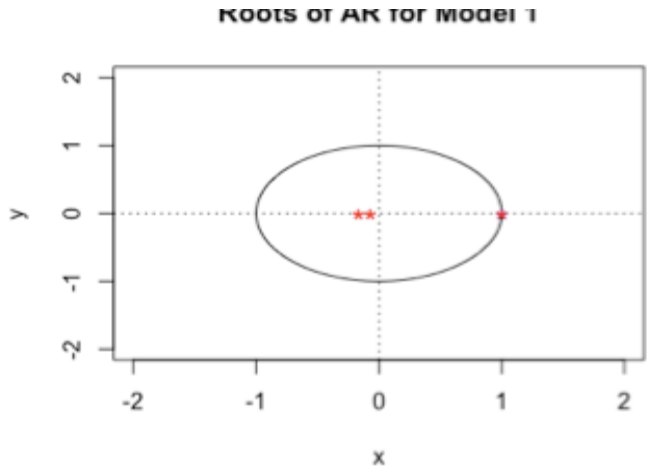
The first model is .

The second model is

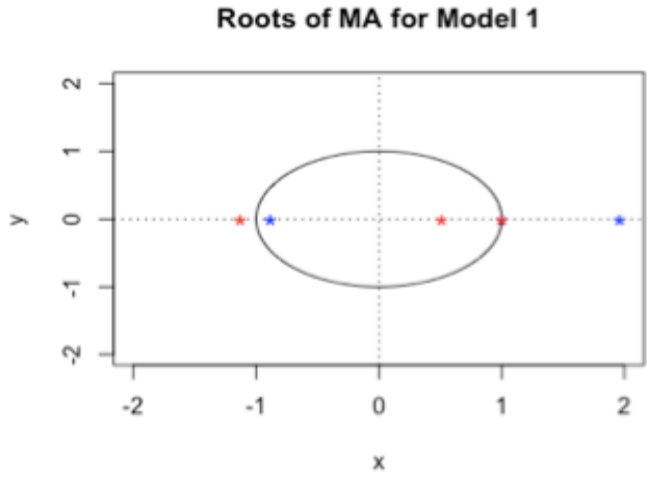
Diagnostic checks

Plot Roots

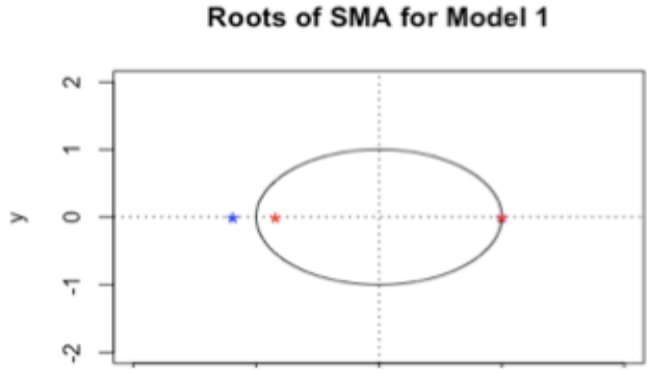
```
source("/Users/dailyuse/Desktop/plot.roots.R")

# model 1
plot.roots(NULL, c(1, -.1636, -.0701), main = "Roots of AR for Model 1")
```

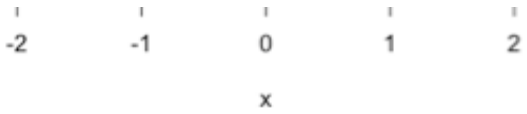



```
# all roots inside circle, hence not invertible
plot.roots(NULL, c(1, -1.1306, .5095), main = "Roots of MA for Model 1")
```

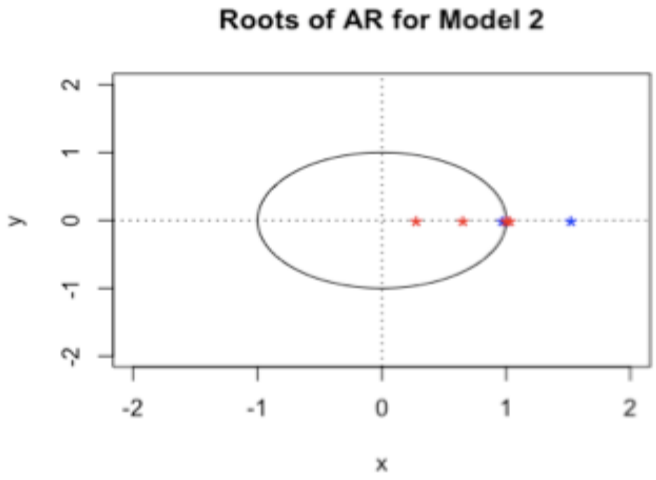



```
# only one root outside circle, hence not causal
plot.roots(NULL, c(1, -.8404), main = "Roots of SMA for Model 1")
```

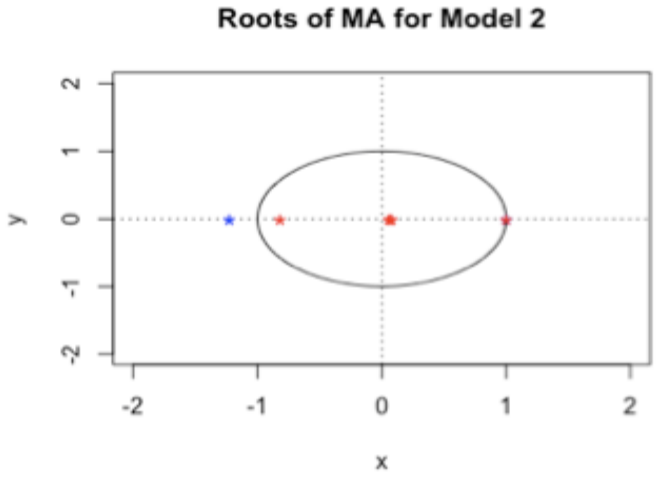





```
# all roots inside circle, hence not causal

# model 2
plot.roots(NULL, c(1, 1.0314, .6560, .2806), main = "Roots of AR for Model 2")
```




```
# all roots inside circle, hence not causal
plot.roots(NULL, c(1, .0754, .0829, .0589, -.8155), main = "Roots of MA for Model 2")
```

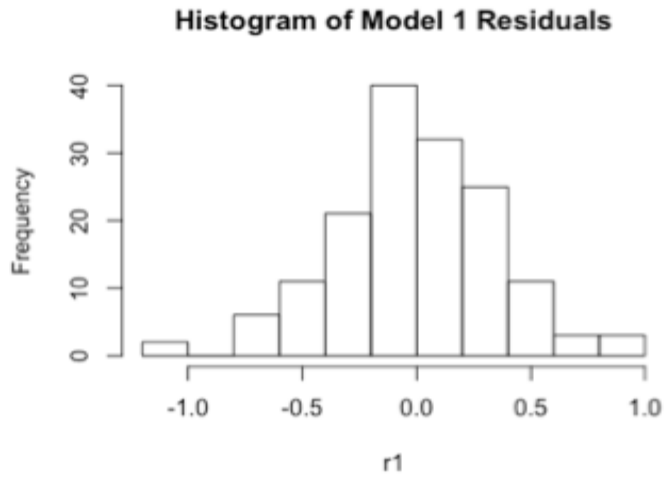



```
# all roots inside circle, hence not invertible
```

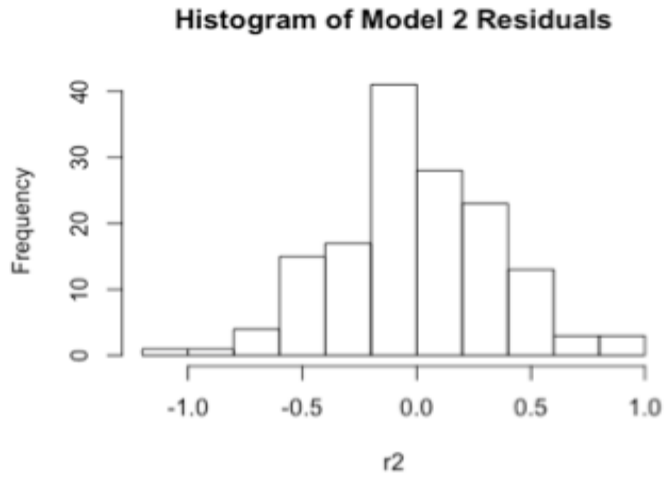
Normality

```
r1 = residuals(mod)
r2 = residuals(auto_mod)

# histograms
```

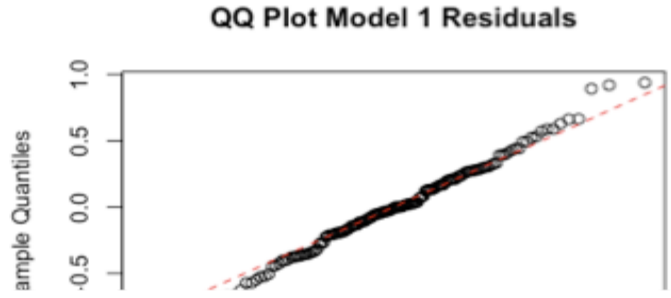
```
hist1 = hist(r1, main = "Histogram of Model 1 Residuals")
```

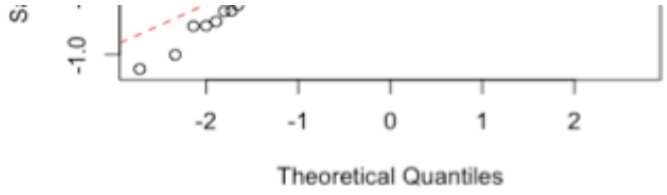



```
hist(r2, main = "Histogram of Model 2 Residuals")
```

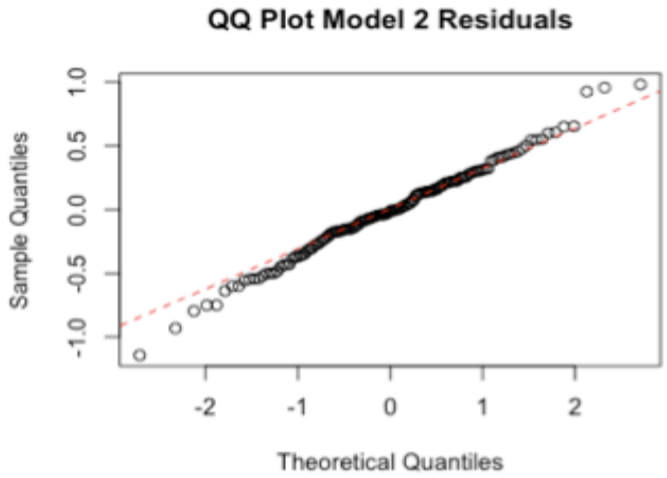



```
# qq plot
qqnorm(r1, main = "QQ Plot Model 1 Residuals")
qqline(r1, lty = 2, col = 2)
```






```
qqnorm(r2, main = "QQ Plot Model 2 Residuals")
qqline(r2, lty = 2, col = 2)
```




```
# shapiro-wilke
shapiro.test(r1)
##
##  Shapiro-Wilk normality test
##
## data:  r1
## W = 0.99242, p-value = 0.5922
shapiro.test(r2)
##
##  Shapiro-Wilk normality test
##
## data:  r2
## W = 0.99282, p-value = 0.6624
## both normal
```

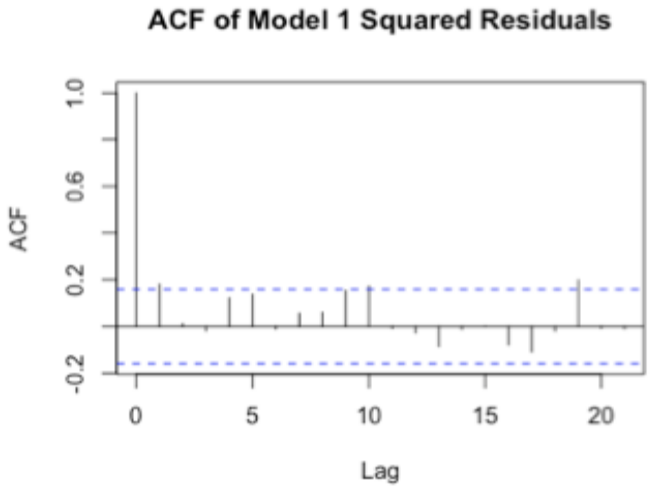
Serial Correlation

```
## ljung-box
Box.test(r1, lag = sqrt(154), type= c("Ljung-Box"), fitdf = 4)
##
##  Box-Ljung test
##
## data:  r1
## X-squared = 5.5951, df = 8.4097, p-value = 0.7305
#Box.test(r2, lag = 1, type= c("Ljung-Box"), fitdf = 0)
Box.test(r2, lag = sqrt(154), type= c("Ljung-Box"), fitdf = 3)
##
##  Box-Ljung test
##
## data:  r2
## X-squared = 5.2267, df = 9.4097, p-value = 0.8415
## box-pierce
Box.test(r1, lag = sqrt(154), type= c("Box-Pierce"), fitdf = 4)
##
##  Box-Pierce test
##
```

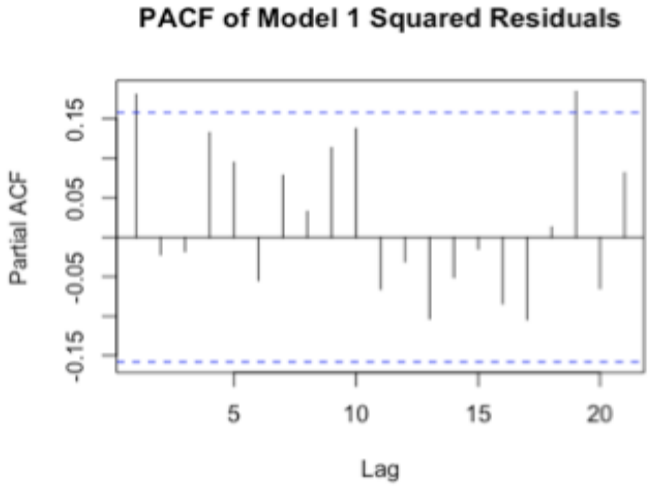
```
## data:  r1
## X-squared = 5.2295, df = 8.4097, p-value = 0.7684
#Box.test(r2, lag = 1, type= c("Box-Pierce"), fitdf = 0)
Box.test(r2, lag = sqrt(154), type= c("Box-Pierce"), fitdf = 3)
##
##  Box-Pierce test
##
## data:  r2
## X-squared = 4.859, df = 9.4097, p-value = 0.8708
## both models pass
```

ACF / PACF

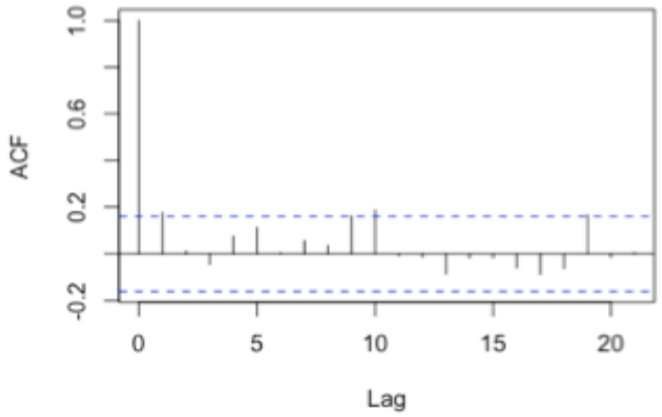
```
# model 1
acf(r1**2, main="ACF of Model 1 Squared Residuals")
```




```
pacf(r1**2, main="PACF of Model 1 Squared Residuals")
```




```
# model 2
acf(r2**2, main="ACF of Model 2 Squared Residuals")
```




```
pacf(r2**2, main="PACF of Model 2 Squared Residuals")
```


```
# both have majority of values lie within confidence intervals
```

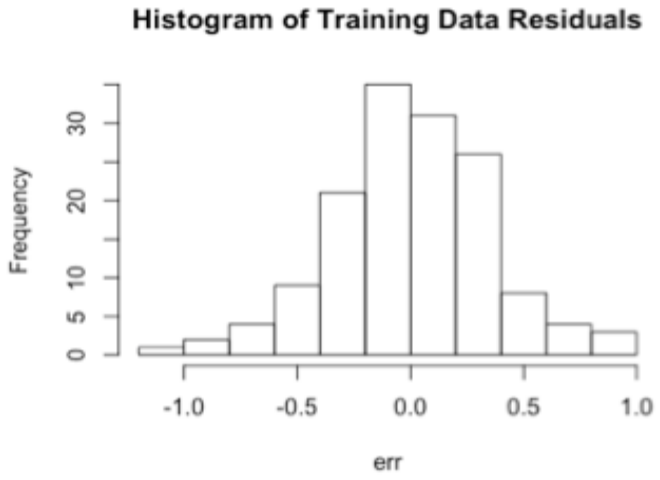
Since both models seem to pass our diagnostic checks, we will apply the principle of parsimony and choose the model with less parameters i.e.

Forecasting

```
# set aside last 10 values for forecasting
train_beer = ts(sq_beer[1:(length(sq_beer)-10)])
# create training model
(train_model = arima(train_beer, order = c(2,1,2), seasonal = list(order = c(0,1,1), period = 4),
method = "ML"))
##
## Call:
## arima(x = train_beer, order = c(2, 1, 2), seasonal = list(order = c(0, 1, 1),
##     period = 4), method = "ML")
##
## Coefficients:
##         ar1     ar2      ma1     ma2     sma1
##      0.1644  0.0892  -1.1294  0.4852  -0.8358
## s.e.  0.2666  0.1591   0.2531  0.1621   0.0666
##
## sigma^2 estimated as 0.1314:  log likelihood = -58.76,  aic = 129.52
```

Diagnostic Checks

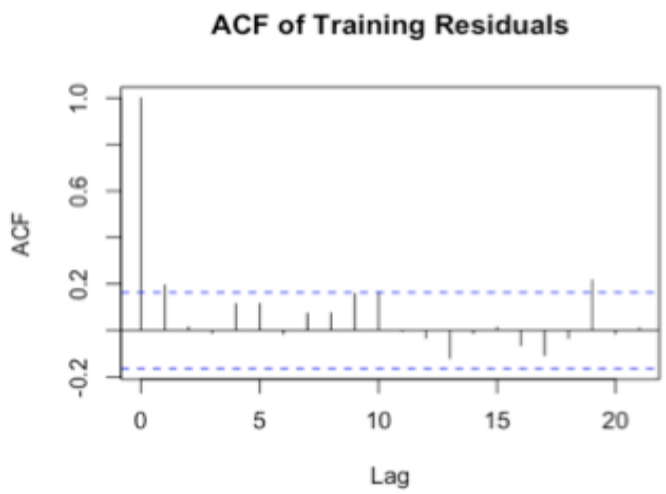
```
err = residuals(train_model)
hist(err, main = "Histogram of Training Data Residuals")
```




```
qqnorm(err, main = "QQ Plot of Training Data Residuals")
qqline(err, lty = 2, col = 2)
```


```
shapiro.test(err)
##
##  Shapiro-Wilk normality test
##
## data:  err
## W = 0.98815, p-value = 0.2589
Box.test(err, lag = sqrt(length(sq_beer)), type= c("Ljung-Box"), fitdf = 4)
##
##  Box-Ljung test
##
## data:  err
## X-squared = 6.3169, df = 8.4097, p-value = 0.6529
Box.test(err, lag = sqrt(length(sq_beer)), type= c("Box-Pierce"), fitdf = 4)
##
##  Box-Pierce test
```

```
##
## data:  err
## X-squared = 5.8572, df = 8.4097, p-value = 0.7026
acf(err**2, main = "ACF of Training Residuals")
```




```
pacf(err**2, main = "PACF of Training Residuals")
```

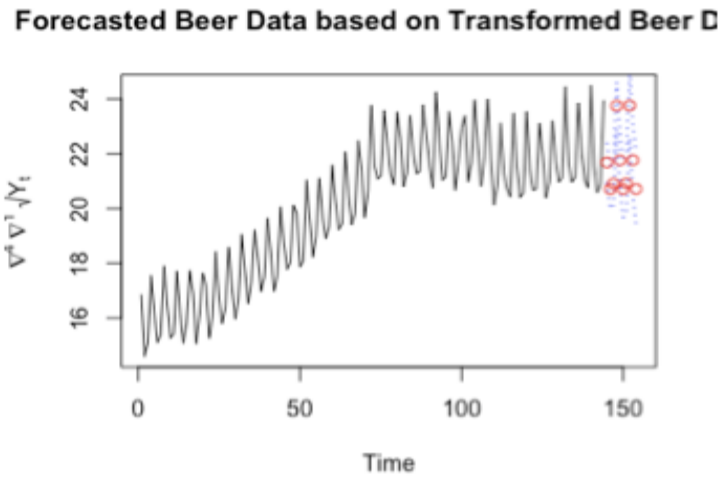
Predictions

```
train_pred = predict(train_model, n.ahead = 10)

# create confidence interval
up_CI = train_pred$pred + 2*train_pred$se
low_CI = train_pred$pred - 2*train_pred$se
```

Forecasted Data on Transformed Data

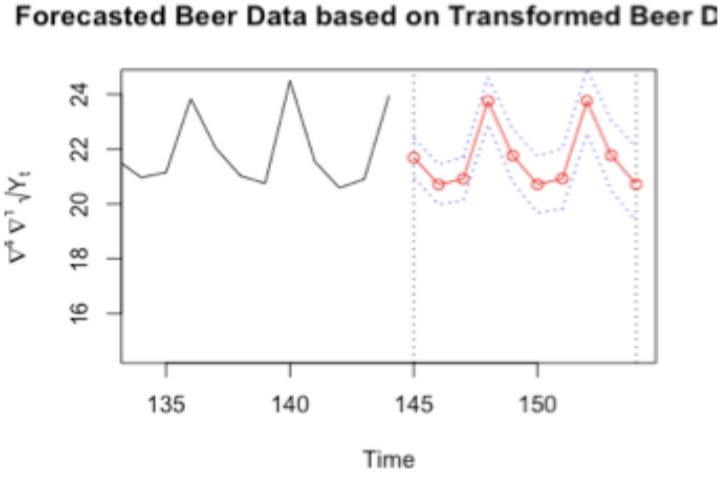
```
ts.plot(train_beer, xlim = c(1, length(train_beer) + 10),
        main = "Forecasted Beer Data based on Transformed Beer Data",
        ylab = expression(nabla^{4}~nabla^{1}~sqrt(Y[t])))
## confidence interval lines
lines(up_CI, col = "blue", lty = "dotted")
lines(low_CI, col = "blue", lty = "dotted")
## predicted values
```

points((length(train_beer)+1):(length(train_beer)+10), train_pred$pred, col = "red")



Forecasted Data on Transformed Data (Zoomed In)

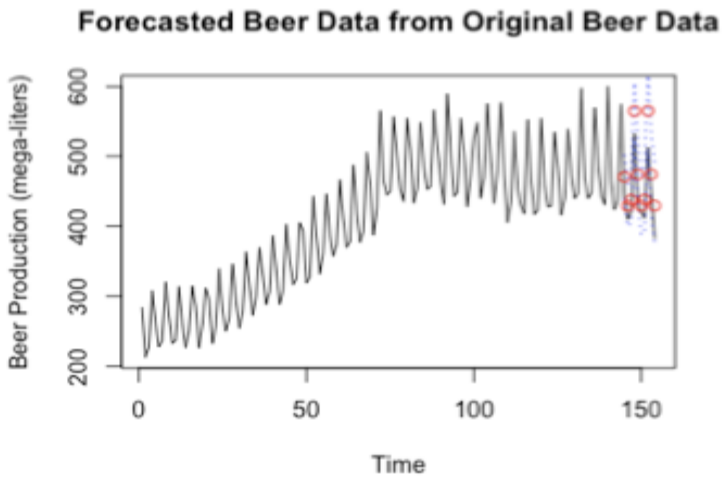
```
ts.plot(train_beer, xlim = c(length(train_beer)-10, length(train_beer)+10),
        main = "Forecasted Beer Data based on Transformed Beer Data",
        ylab = expression(nabla^{4}~nabla^{1}~sqrt(Y[t])))
## confidence interval lines
lines(up_CI, col = "blue", lty = "dotted")
lines(low_CI, col = "blue", lty = "dotted")
## predicted values
points((length(train_beer)+1):(length(train_beer)+10), train_pred$pred, col = "red")
lines((length(train_beer)+1):(length(train_beer)+10), train_pred$pred, lty = 1, col = "red")
## forecasted section
abline(v = 145, lty = 3)
abline(v = 154, lty = 3)
```



Forecasted Data on Original Data

```
# get CI for original data
pred_orig = train_pred$pred^(1/.5)
up_orig = up_CI^(1/.5)
low_orig = low_CI^(1/.5)
orig_beer = ts(beer[,2])
```

```
ts.plot(orig_beer, xlim = c(1, length(orig_beer)),
    main = "Forecasted Beer Data from Original Beer Data",
    ylab = "Beer Production (mega-liters)")
## confidence interval lines
lines(up_orig, col = "blue", lty = "dotted")
lines(low_orig, col = "blue", lty = "dotted")
## predicted values
points((length(train_beer)+1):(length(train_beer)+10), pred_orig, col = "red")
```



Forecasted Data on Original Data (Zoomed In)

```
ts.plot(orig_beer, xlim = c(length(orig_beer)-20, length(orig_beer)),
    main = "Observed vs. Forecasted Values",
    ylab = "Beer Production (mega-liters)")
## confidence interval lines
lines((length(train_beer)+1):(length(train_beer)+10), up_orig, lty = 2, col = "blue")
lines((length(train_beer)+1):(length(train_beer)+10), low_orig, lty = 2, col = "blue")
## prediction line / point
lines((length(train_beer)+1):(length(train_beer)+10), pred_orig, lty = 1, col = "red")
points((length(train_beer)+1):(length(train_beer)+10), pred_orig, col = "red")
## actual values
points((length(train_beer)+1):(length(train_beer)+10), orig_beer[145:154], col = "orange")
## forecasted section
abline(v = 145, lty = 3)
abline(v = 154, lty = 3)
```