Australian Beer Production Forecast

Changhee (Eric) Yoon

Abstract

In business, it is very important to know how much goods need to be produced in certain time of the year. In this project, I've implemented a mathematical model to predict monthly beer production using the past data. To be specific, in order to predict 24 months of future Australian beer production, data from 20 past years was used to formulate a mathematical model that considers both the production values from the same time of the year from the past and the production values from the past months. In conclusion, the model provides quite accurate forecast that closely resembles the actual beer production values.

Introduction

The goal of this project is to implement time series model with parameters tuned on 20 years worth data from 1956 to 1976 to make a 2 year forecast on the monthly beer production (from 1976~1978). The dataset from Kaggle consists of two columns: date and beer production in million liters.

It is crucial for the businesses, not only for just the beer companies but the suppliers for beer companies, to predict how much production of the product is needed for each month/time of the year, and having a good, accurate model will provide forecasts that will be beneficial insights for businesses to prepare resources to produce the appropriate amount of product needed each month. Inaccurate forecast would result in over or under producing the goods and result in financial losses for the businesses. The Australian beer industry produced about 1.6 billion liters (about 422 million gallons) of beer in 2018, and the beer industry contributed 4.6 billion Australian dollars to the Australian economy. In an industry of this size, it's crucial to have an accurate forecast of the production.

In order to make an accurate forecast, first, the non-seasonality of the data was confirmed by plotting it as a time series. Box-cox transformation was used to transform the data and stabilize the variance. Decomposition of the time series and the graph was utilized to identify where to difference the time series. ACF and PACF graphs of transformed and differenced data were analyzed to identify potential parameters for seasonal ARIMA models. Then, to find the optimal model that fits the data well, several SARIMA models with all the combination of parameters were fitted and their AIC values were compared (to pick models with lower AIC values). Two candidate models were then tested for stationarity and invertibility by checking if the roots of characteristic functions of MA, AR, SMA and SAR parts of SARIMA models lie outside of unit circle. Since Model i had $\phi_2 = -1$, the model was concluded to fail the stationary test. Model ii had all the roots outside of unit circle, so its stationarity and invertibility were checked. Then, for diagnostics, the normality of residuals was first assessed using histogram, qqPlot, and Shapiro Wilk test. Then, to check for independence of residuals (residuals should resemble Gaussian WN(0,1)), the ACF and PACF plot of residuals was assessed (should be within the C.I.), and linear and non linear dependence of residuals were tested using Box-Pierce test, Ljung-Box test, and McLeod-Li test. In the process, unfortunately, the model didn't pass the Box-Pierce and Ljung-Box test at confidence level 95%. Then, residuals were tested using yule-walker method, if the order selected is 0, which means that the residuals resemble AR(0) process, white noise. Although the model didn't pass the box tests, the histogram, qqPlot, ACF plot, and PACF plot all looked decent and it passed other diagnostic tests, the model SARIMA (p=1,d=1,q=3)x(P=0,D=1,Q=5)$_{s=12}$ was used for the forecast. Finally, the generated forecast of monthly beer production from 1976~1978 was

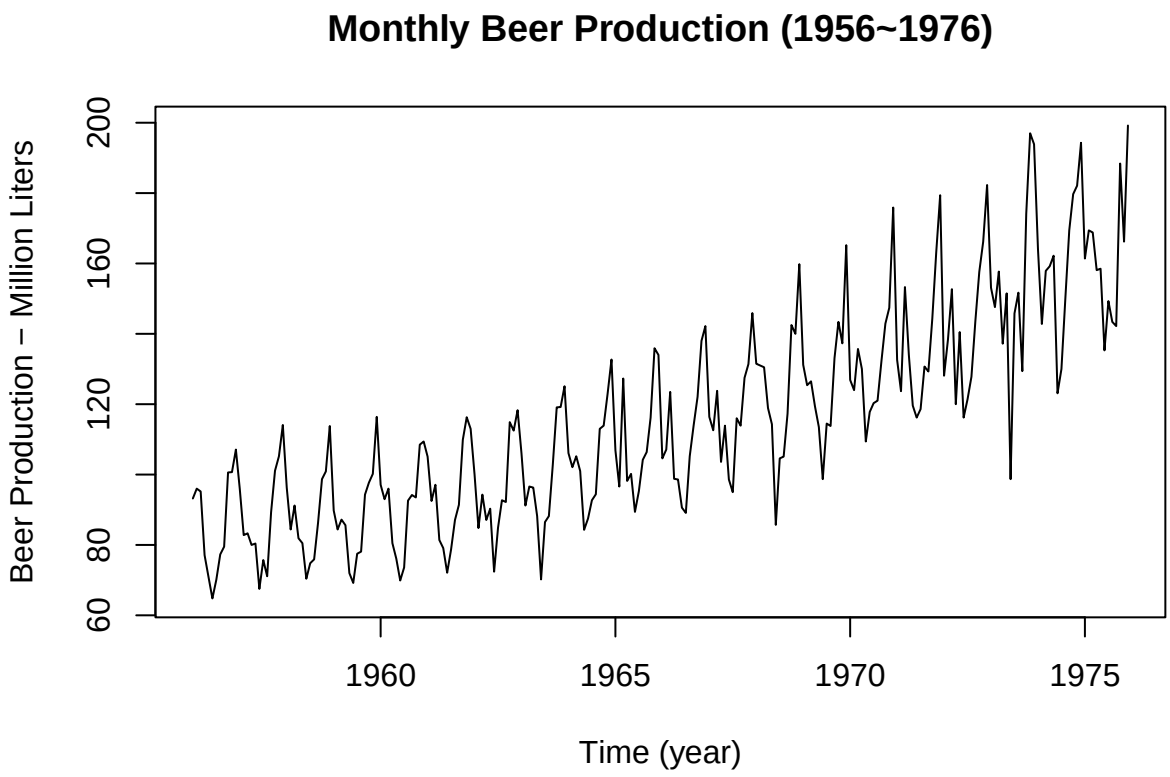
compared to the actual production values from the test set. The result was quite successful with actual data aligning with the forecast values closely.

For all the visualization and calculation, R was used.

Initial Data exploration and transformation Before looking into the data, the data was split into train and test sets. Train set consists of 20 years worth of data from 1956~1976, and test set consists of 2 years worth of data from 1976 to 1977.

First, to assess the trend and the seasonality, graph the data (U_t)

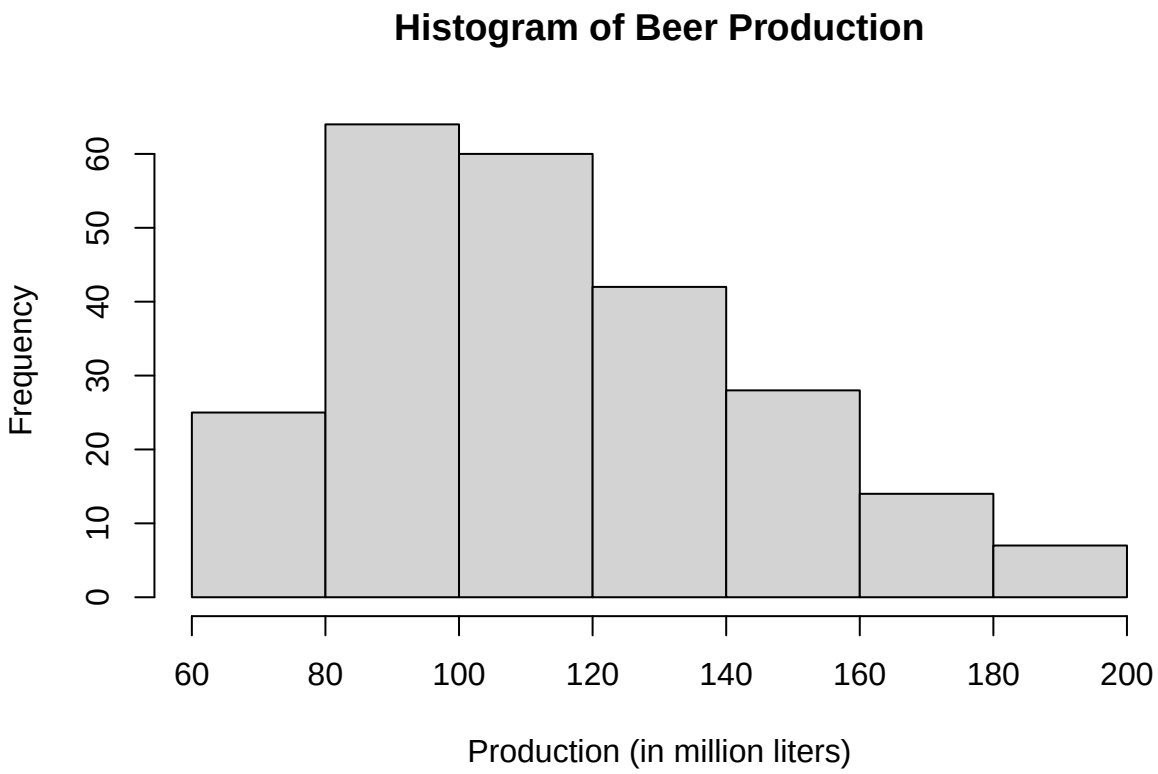
```
ts.plot(train_ts, ylab="Beer Production - Million Liters", xlab="Time (year)", main="Monthly Beer Produ
```



The graph of time series shows that there is a positive trend over time. The data appears to have a seasonal component with yearly peaks and troughs. There aren't any apparent sharp changes in behavior. Looking at the increase in the difference between the high's and the low's, there appears to be an increase in variance over time.

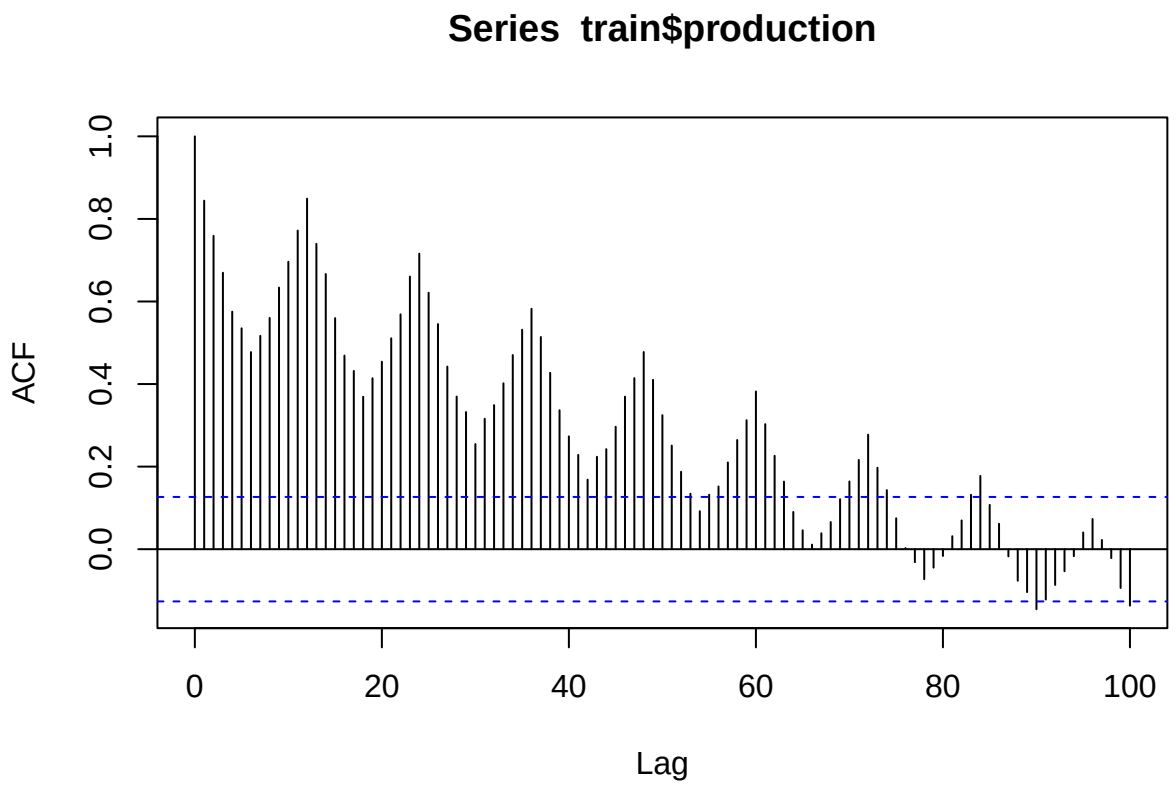
Now, check the histogram.

```
hist(train$production, xlab="Production (in million liters)", main = "Histogram of Beer Production")
```



The histogram is skewed and indicates that some transformation could help with normalizing the data.

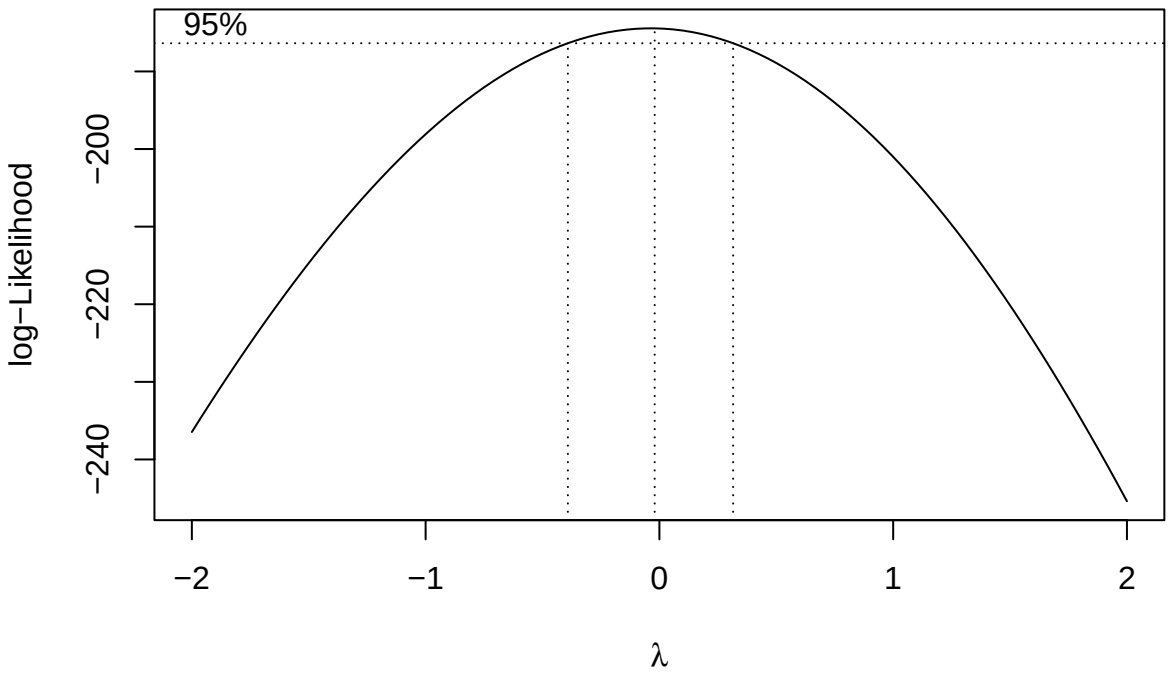
```
acf(train$production, lag.max=100)
```



The ACF remains large for a long time and it appears to have a period. This ACF graph makes it obvious that the data is non stationary and periodic.

Now, to stabilize the variance and make the data less skewed, transform the data. Run the Box-Cox transformation to find the optimal lambda.

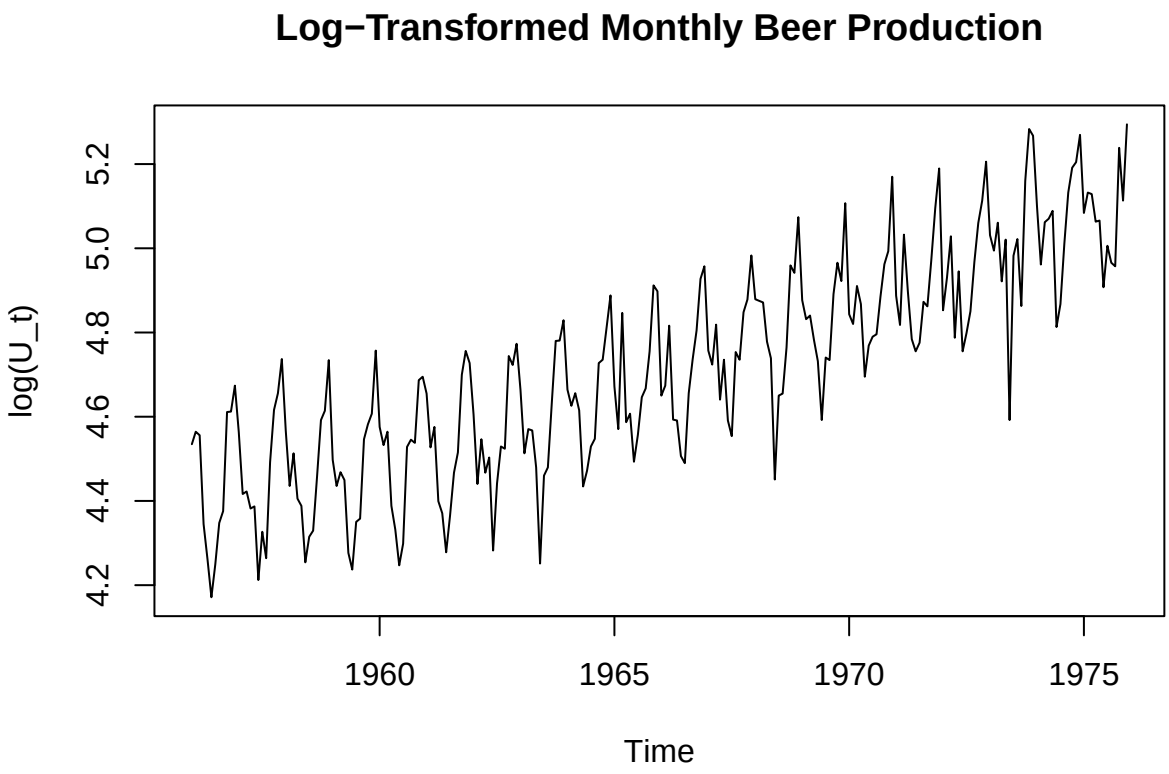
```
trainTrans <- boxcox(train$production~ as.numeric(1:length(train$production)))
```




```
lambda <- trainTrans$x[which(trainTrans$y == max(trainTrans$y))]
```

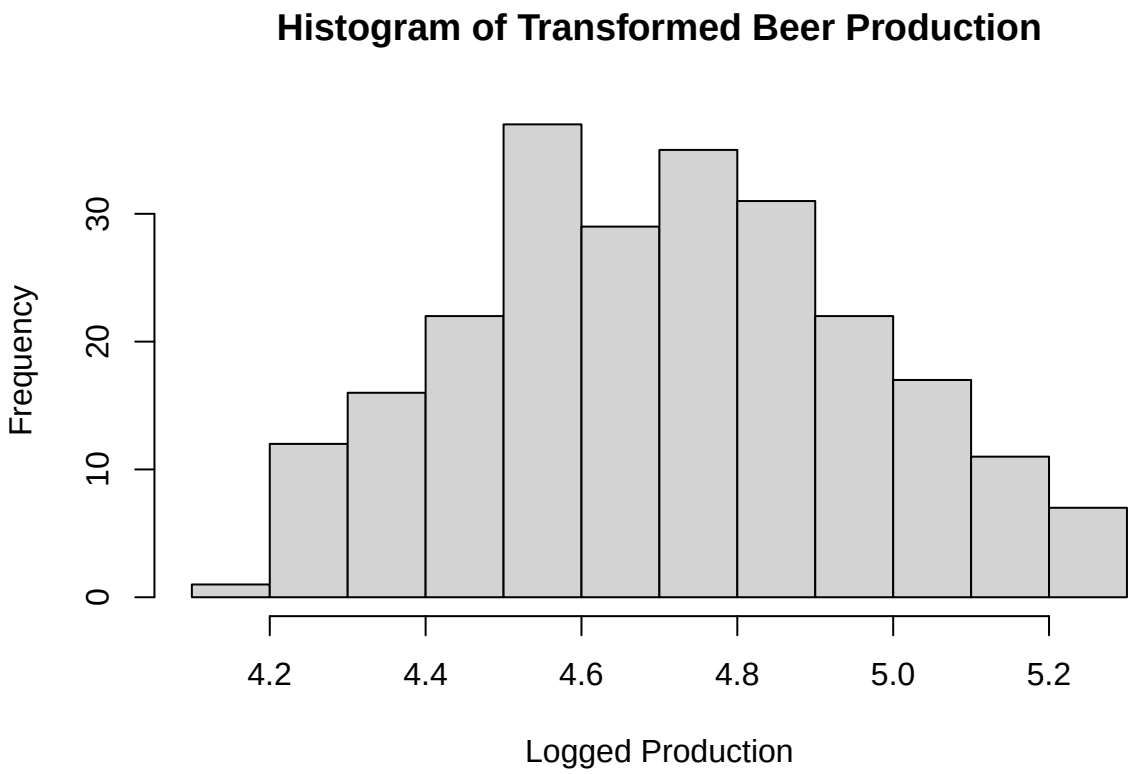
The lambda calculated from BoxCox is very close to zero, and it will be much simpler to do a log transformation than a the exact Box-cox value, so log transform the data.

```
plot.ts(train_log, main="Log-Transformed Monthly Beer Production", ylab="log(U_t)")
```



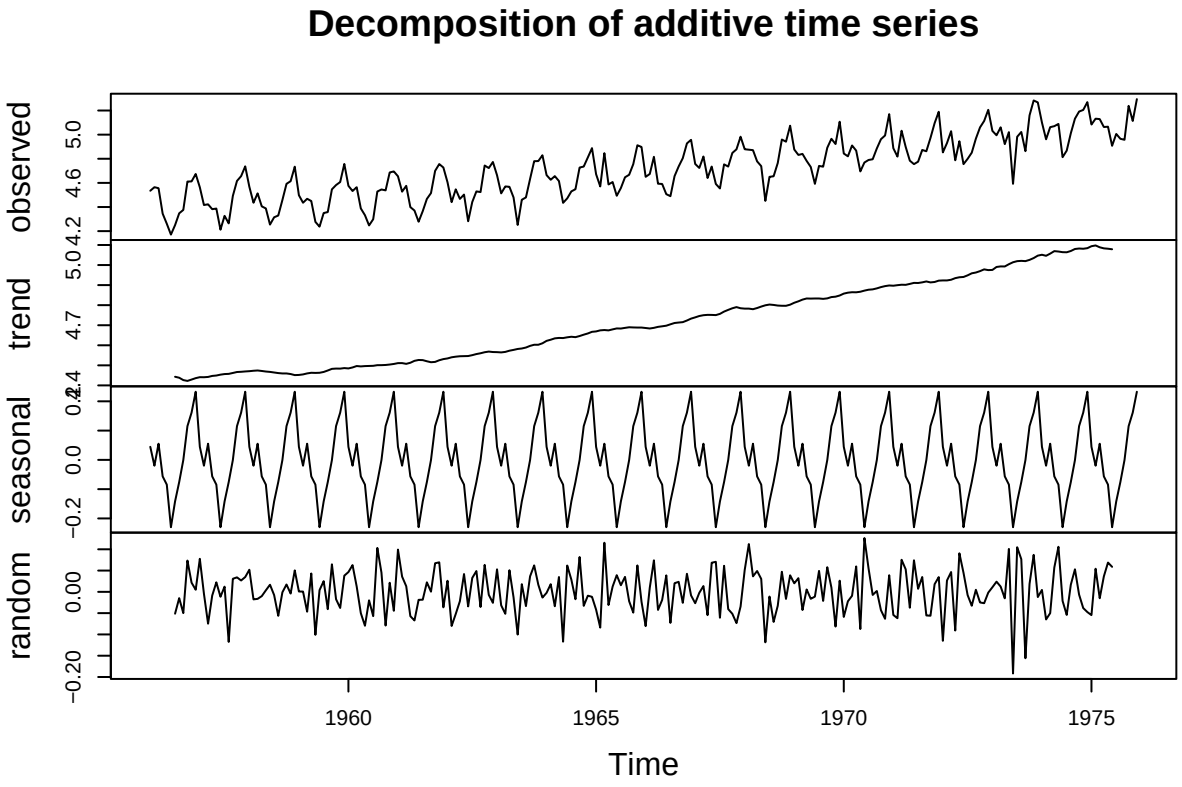
The variance looks more stable after the transform with more uniform distance between the ups and downs.

```
hist(train_log, xlab="Logged Production", main = "Histogram of Transformed Beer Production")
```



The histogram now doesn't look skewed. The data seem more clustered around the center and symmetric.

```
plot(decompose(train_log))
```



Decomposition
Here's a decomposition of the transformed data. There is a positive trend ("trend" column). There is a seasonal component to the data ("seasonal" column), and the variance appears to be consistent (looking at the "random" column).

Differencing and Identifying Potential Parameters for Models Before differencing, check the variance of the $\log(U_t)$ to compare it with the differenced data

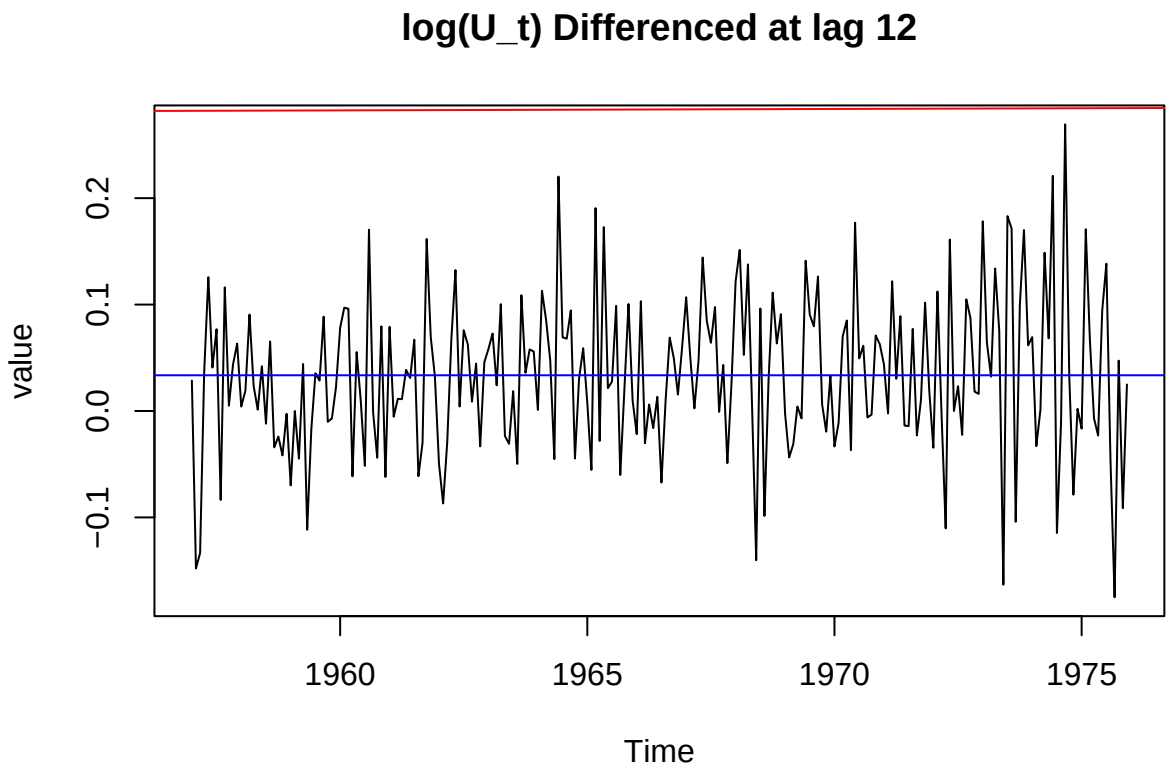
```
var(train_log)
```

```
## [1] 0.06423776
```

Since there is a consistent ups and downs every year, 12 months, so **difference at lag=12** to remove the seasonality. Also check if the variance is less after differencing

```
# difference at lag 12
train_log_12 <- diff(train_log, lag=12)
# get the variance after differencing at lag 12
var(train_log_12)
```

```
## [1] 0.00540387
```

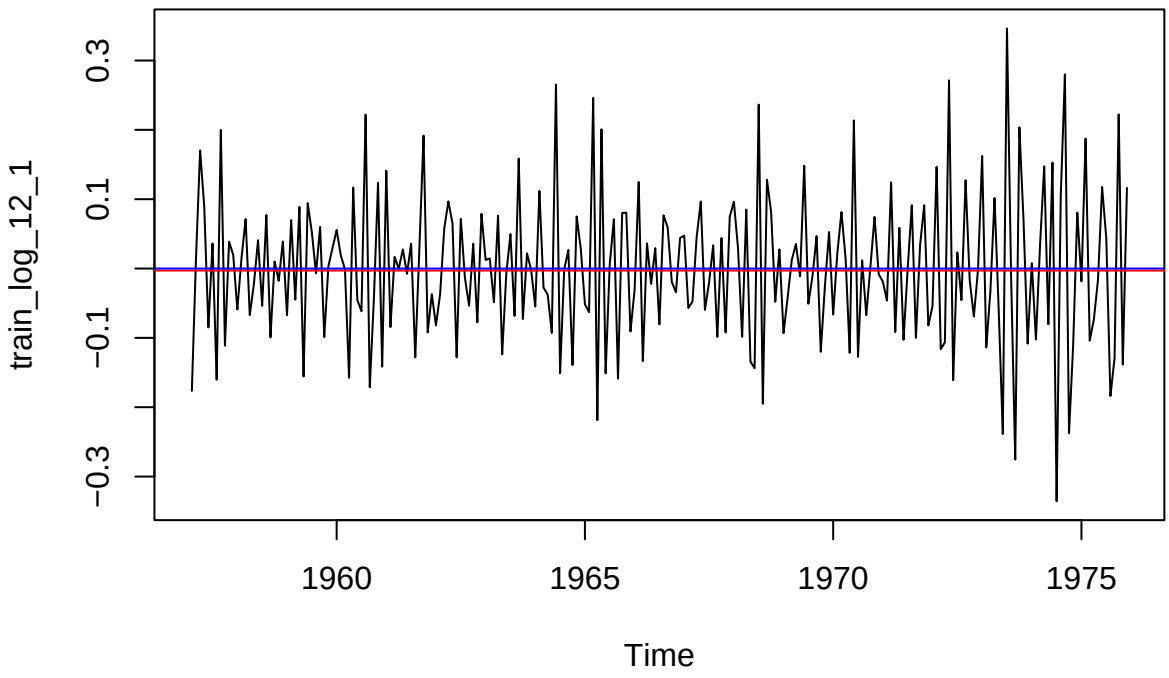


The variance has decreased quite significantly after differencing at 12, and seasonality appears to be eliminated. There seem to be a very slight positive trend, but it could be negligible.

Now, for differencing at **lag 12 and 1**

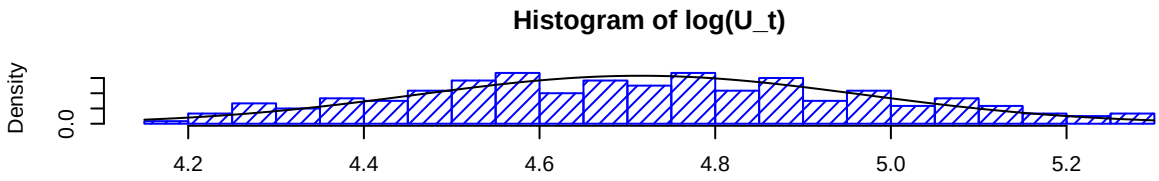
```
# Differencing at lag 12 and 1
train_log_12_1 <- diff(train_log_12, lag=1)
var(train_log_12_1)
```

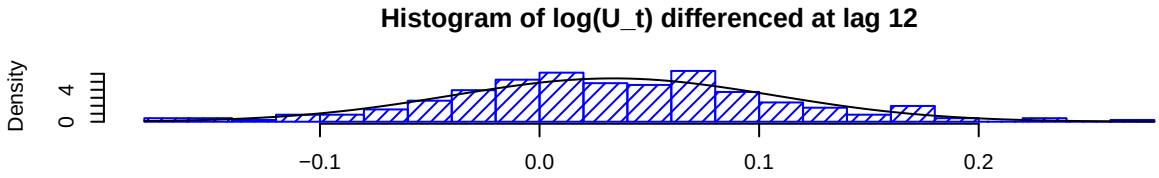
```
## [1] 0.01160582
```

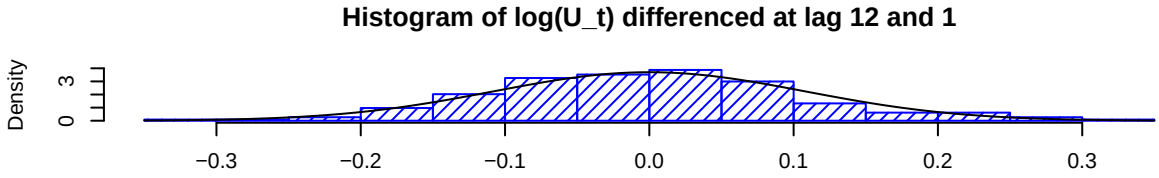


No seasonarity, no trend after differentiating at lag 12 and then at lag 1. However, the variance is larger, so I will be comparing both models with differencing at lag 12 and lag 12 & 1.

```
par(mfrow = c(3, 1))

hist(train_log, density=20,breaks=20, col="blue", xlab="", prob=TRUE, main = "Histogram of log(U_t)")
m<-mean(train_log)
std<- sqrt(var(train_log))
curve( dnorm(x,m,std), add=TRUE )

hist(train_log_12, density=20,breaks=20, col="blue", xlab="", prob=TRUE, main = "Histogram of log(U_t) d
m<-mean(train_log_12)
std<- sqrt(var(train_log_12))
curve( dnorm(x,m,std), add=TRUE )

hist(train_log_12_1, density=20,breaks=20, col="blue", xlab="", prob=TRUE, main = "Histogram of log(U_t
m<-mean(train_log_12_1)
std<- sqrt(var(train_log_12_1))
curve( dnorm(x,m,std), add=TRUE )
```

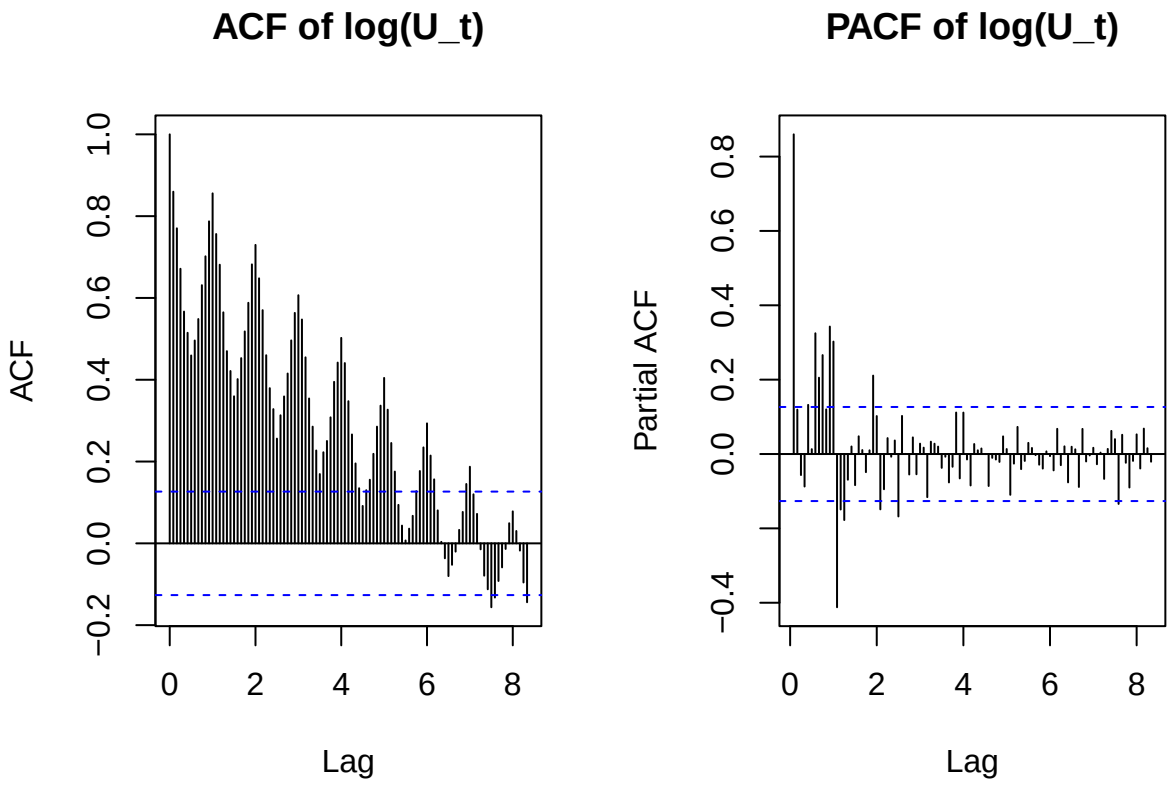




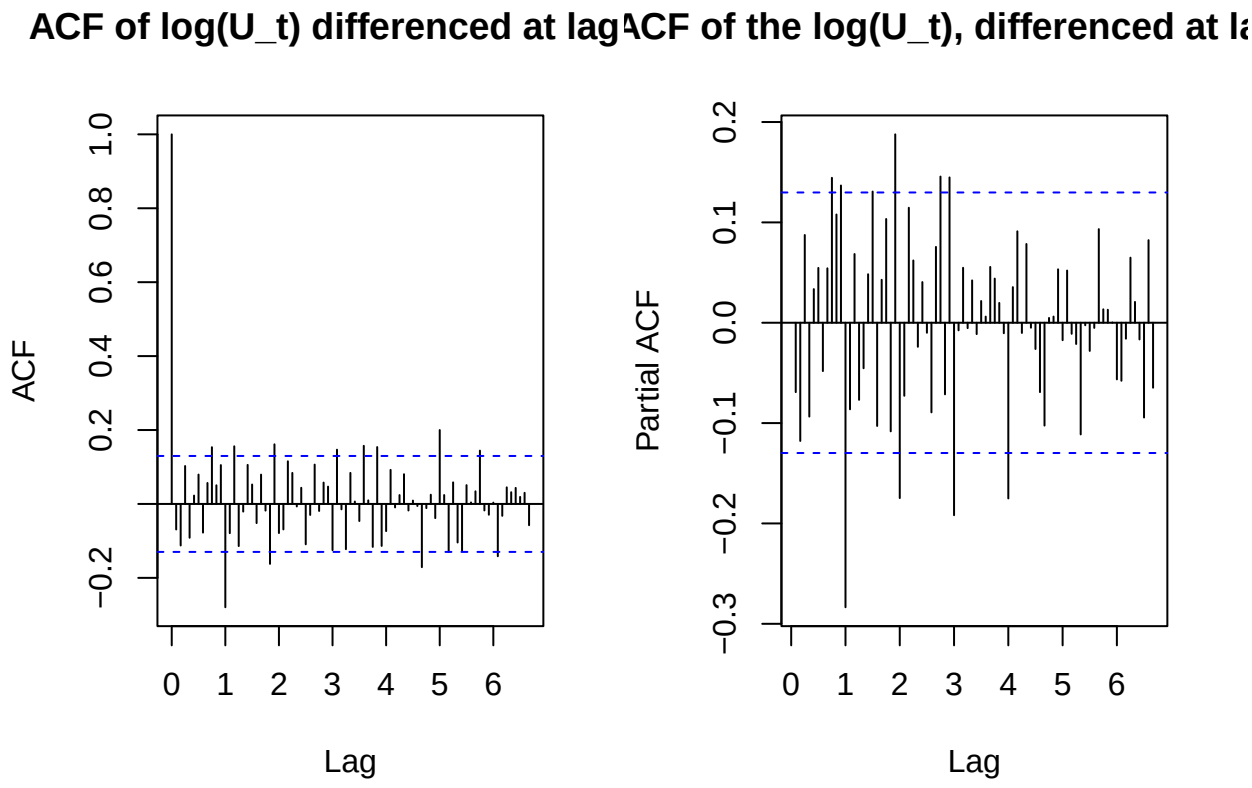


Now, analyze the ACF & PACF to determine the potential parameters for the model

```
par(mfrow = c(1, 2))
acf(train_log, lag.max=100, main="ACF of log(U_t)")
pacf(train_log, lag.max=100, main="PACF of log(U_t)")
```




```
par(mfrow = c(1, 2))
acf(train_log_12, lag.max=80, main="ACF of log(U_t) differenced at lag 12")
pacf(train_log_12, lag.max=80, main="PACF of the log(U_t), differenced at lags 12")
```

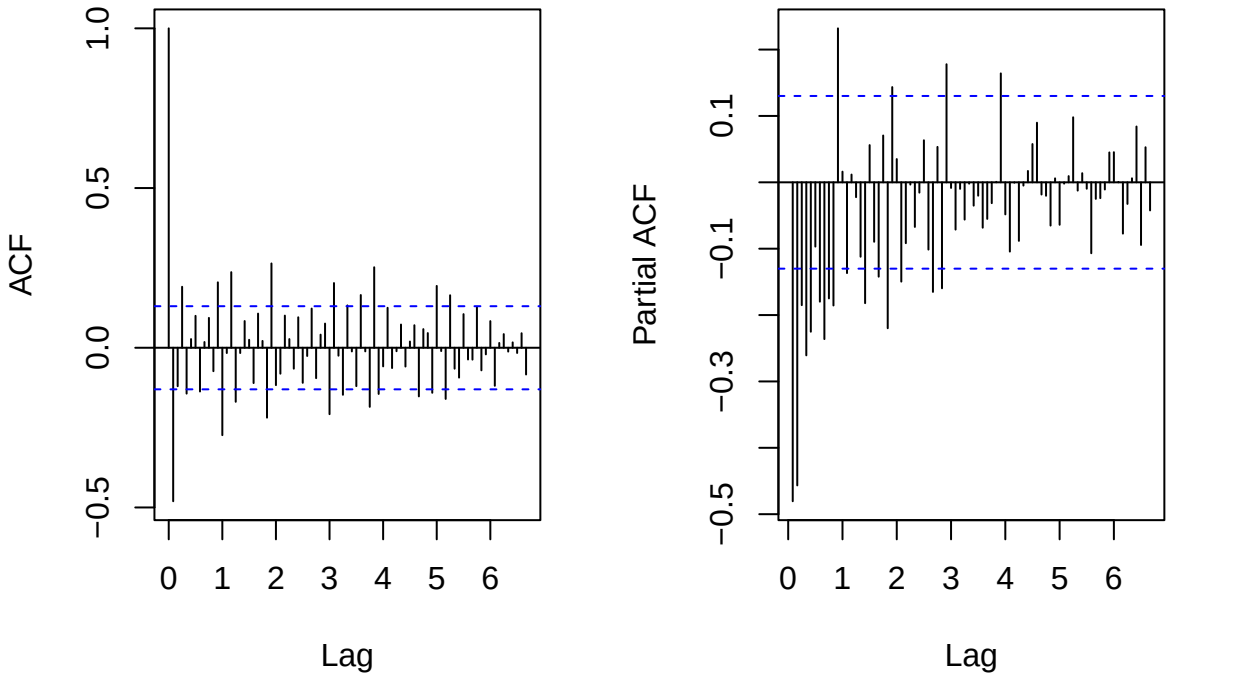


ACF plot for differenced at lag 12 quite nice compared to the ACF for non-differenced data.

Determining parameters for SARIMA for $log(U_t)$: - We applied one seasonal differencing so D=1 at lag s=12 - d=0 since no additional differencing was applied to remove the trend. - ACF seems significant (outside of 95% CI) at h=1s and 5s, so Q=1 or 5 - ACF shows no strong peak for non-seasonal ACF and a small peak at h=9, so q=0 or 3 - PACF shows strong peaks at h=1s, 2s, 3s and 4s, so P=4 - PACF shows a weak peak at h = 11 and 9, so p=0 or 1 or 3

=> SARIMA (p=0 or 1 or 3,d=0,q=0 or 3)(P=4, D=1, Q=1 or 5)$_{s=12}$

```
par(mfrow = c(1, 2))
acf(train_log_12_1, lag.max=80, main="ACF of log(U_t) differenced at lag 12 and 1")
pacf(train_log_12_1, lag.max=80, main="PACF of the log(U_t), differenced at lags 12 and 1")
```



The ACF graph for twice differenced data looks quite similar to the one for data differenced at lag 12, so I will be proceeding with comparing both models with different differencing coefficients.

Determining parameters for SARIMA for $log(U_t)$: - We applied one seasonal differencing so D=1 at lag s=12 - We applied one differencing to remove the trend, so d=1 - ACF seems significant (outside of 95% CI) at h=1s, 3s, and 5s, so Q=1 or 3 or 5 - ACF shows strong peak at h=1 and smaller peak at h=3 and maybe at h=4, so q=1 or 3 or 4 - PACF appears to have no particular peak at any seasonal lags, so P=0 - PACF shows strong peaks peaks at h = 1, 2 and smaller peaks until h=5 and there is a quite strong peak at h=11, so p= 1 or 2 or 5

=> SARIMA (p=1 or 2 or 5, d=1, q=1 or 3 or 4)(P=0, D=1, Q=1 or 3 or 5)$_{s=12}$

Choosing candidate models with AIC values First, using R commands, all the potential parameters for each models are fitted and their AIC values are calculated. After calculating the AIC value, each model, in the order of lowest to highest AIC, were fed into a diagnostic pipeline with Shapiro Wilk normality test of residuals, and other diagnostic tests. Unfortunately, none of the models with d=0 did not pass Shapiro Wilks normality test on the residuals. In addition, all the models that produced NaN error or "possible convergence error" in running the R code were eliminated, and models with very close to zero coefficients had their coefficients fixed to zero and the AIC for those models were re-calculated. The re-calculated models generally outputted higher AIC values.

After calculating AIC values for all potential models with differencing at lag 12 and 1, a couple models with low AIC values were picked:

Model i: SARIMA (p=2,d=1,q=3)x(P=0,D=1,Q=3)$_{s=12}$

Model i had the lowest AIC score (-630.51) among all the models that didn't result in error when fitting the model.

Model ii: SARIMA (p=1,d=1,q=3)x(P=0,D=1,Q=5)$_{s=12}$

Model ii had the second lowest AIC score (-603) among the models with no errors.

Check for stationarity and invertibility before diagnostic *Model i* SARIMA (p=2,d=1,q=3)x(P=0,D=1,Q=3)$_{s=12}$

```
fit.i <- sarima(xdata = train_log,
  p = 2, d = 1, q = 3,
  P = 0, D = 1, Q = 3, S = 12,
  details = F)


resi <- fit.i$fit$residuals

fit.i$fit$coef
```

```
##         ar1         ar2         ma1         ma2         ma3        sma1
## -1.15222700 -0.99978737  0.24957499 -0.05748633 -0.91087732 -0.83694557
##        sma2        sma3
## -0.03719675  0.09443777
```

In mathmatical equation:

$$\left(1-\phi_1 B-\phi_2 B^2\right)(1-B)\left(1-B^{12}\right) log(X_t) = \left(1+\theta_1 B+\theta_2 B^2+\theta_3 B^3\right)\left(1+\Theta_1 B^{12}+\Theta_2 B^{24}+\Theta_3 B^{36}\right) Z_t$$

Before the diagnostic checking, check the stationarity and invertibility of the model:

Unfortunately, the coefficient ϕ_2 appears to be -.99979, which is -1. Since $\phi_2 = 1$ for AR(2) part of the model, the model isn't stationary, and I'll move onto next model.

Model ii SARIMA (p=1,d=1,q=3)x(P=0,D=1,Q=5)$_{s=12}$

```
fit.ii <- sarima(xdata = train_log,
  p = 1, d = 1, q = 3,
  P = 0, D = 1, Q = 5, S = 12,
  details = F)

resii <- fit.ii$fit$residuals

fit.ii$fit$coef
```

```
##         ar1         ma1         ma2         ma3        sma1        sma2
## -0.54486539 -0.43296268 -0.58201823  0.14990599 -0.71879831 -0.16573458
##        sma3        sma4        sma5
## -0.10873247  0.09960884  0.25077335
```

In mathmatical equation:

$$(1+\phi_1 B)(1-B)\left(1-B^{12}\right) log(X_t) = \left(1+\theta_1 B+\theta_2 B^2+\theta_3 B^3\right)\left(1+\Theta_1 B^{12}+\Theta_2 B^{24}+\Theta_3 B^{36}+\Theta_4 B^{48}+\Theta_5 B^{60}\right) Z_t$$

Before the diagnostics, check for stationarity and invertibility of the model For AR component, MA component, and seasonal MA component respectively:

```
par(mfrow = c(2, 2))
uc.check(pol_ = c(1, 0.54486539), plot_output = TRUE)
```

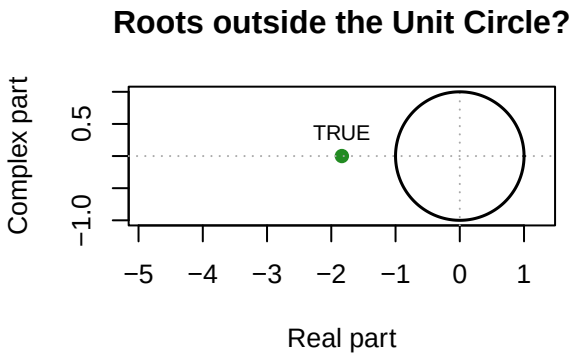
```
##        real complex outside
## 1 -1.835316       0    TRUE
## *Results are rounded to 6 digits.
```

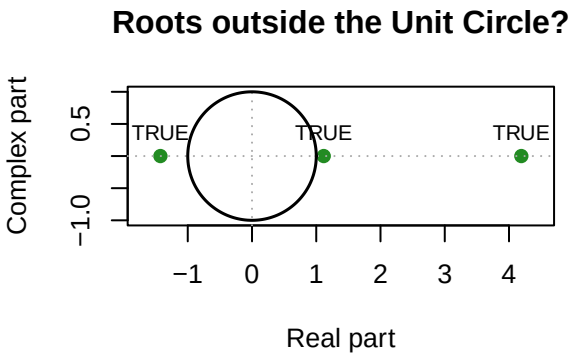
```
uc.check(pol_ = c(1, -0.43296268, -0.58201823,  0.14990599), plot_output = TRUE)
```

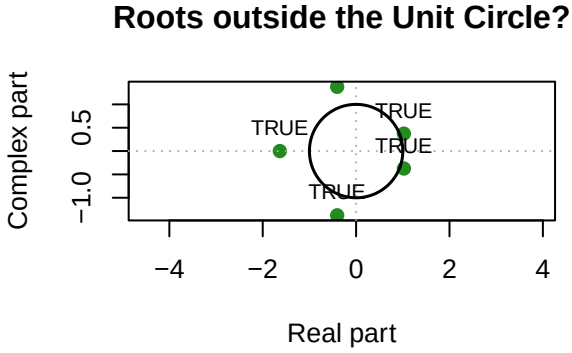
```
##        real complex outside
## 1  1.116251       0    TRUE
## 2 -1.425625       0    TRUE
## 3  4.191929       0    TRUE
## *Results are rounded to 6 digits.
```

```
uc.check(pol_ = c(1, -0.71879831, -0.16573458, -0.10873247,  0.09960884,  0.25077335 ), plot_output = TR
```

```
##        real   complex outside
## 1  1.023340  0.373166    TRUE
## 2 -0.405318  1.376057    TRUE
## 3 -0.405318 -1.376057    TRUE
## 4  1.023340 -0.373166    TRUE
## 5 -1.633250  0.000000    TRUE
## *Results are rounded to 6 digits.
```





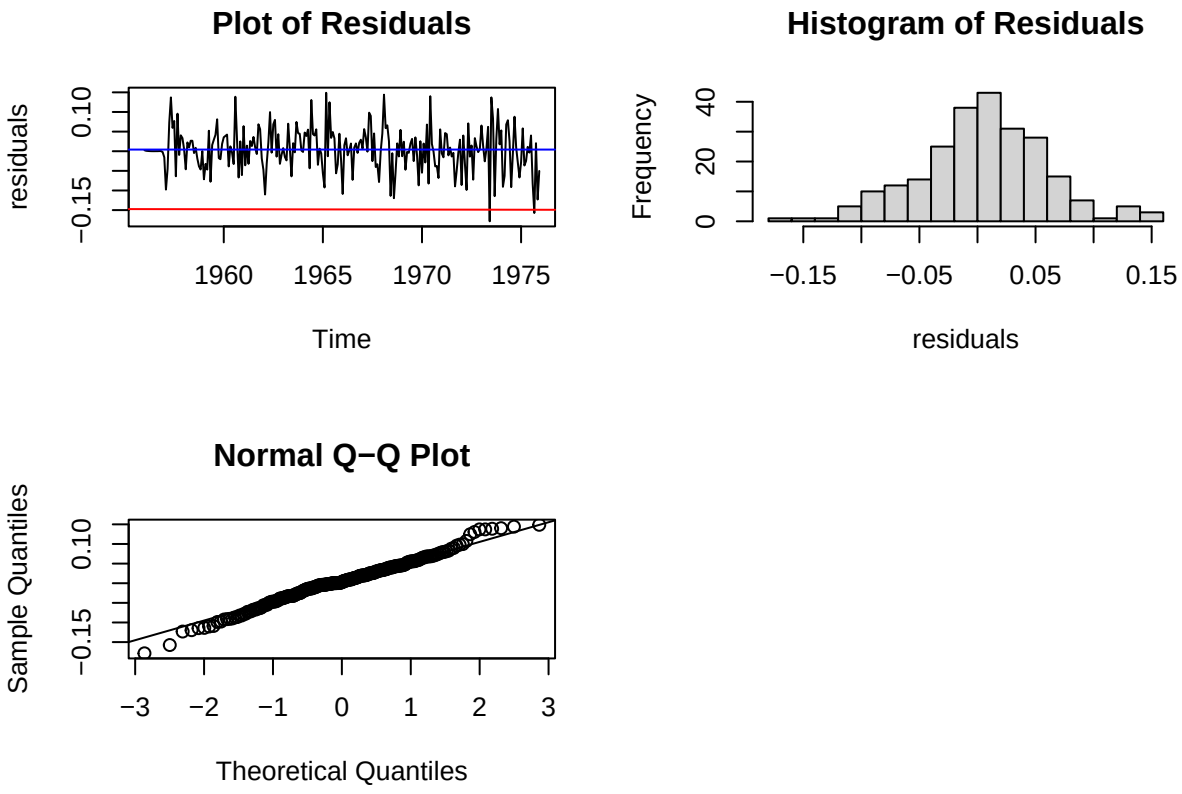


Since all the roots of the polynomial of MA and AR part and characteristic polynomial of seasonal MA part are out of the unit circle, the model is stationary and invertible.

Now, onto diagnostics:

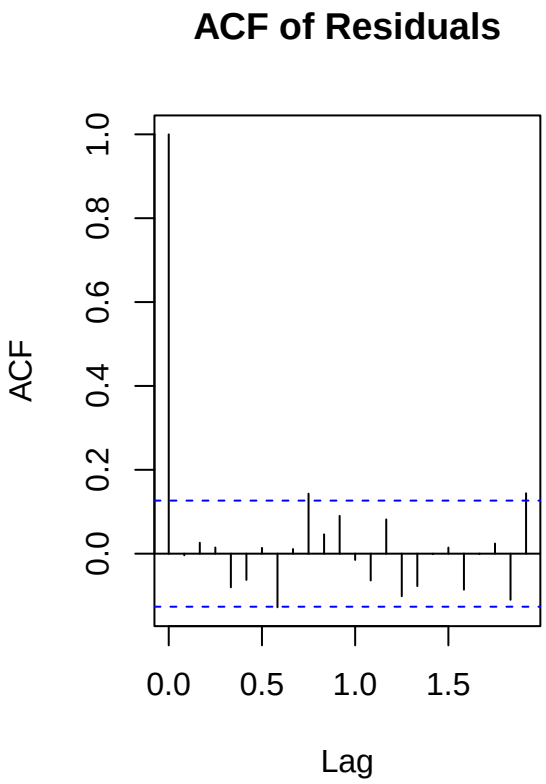
First, a good fitting model's residual should resemble a Gaussian white noise. So check for normality of the residuals:

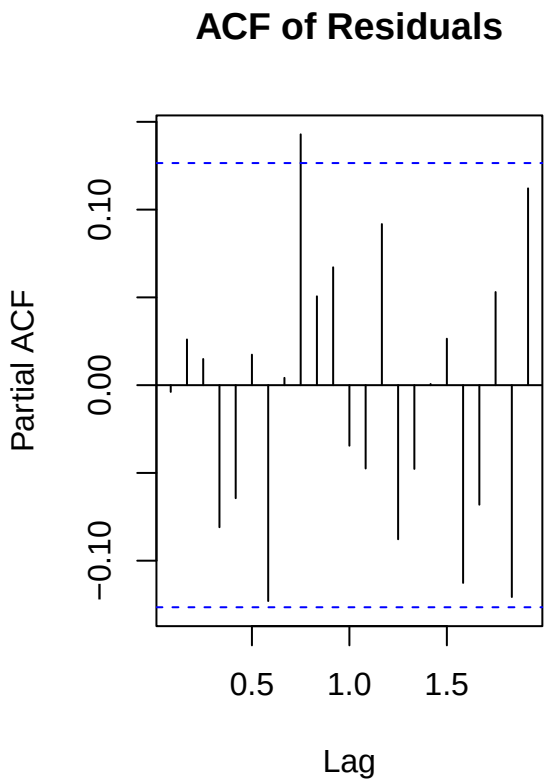
```
par(mfrow = c(2, 2))
fit <- lm(resii ~ as.numeric(1:length(resii)))

plot.ts(resii, ylab="residuals", main="Plot of Residuals")

abline(fit, col="red")
abline(h=mean(resii), col="blue")

hist(resii, breaks = 20, xlab="residuals", main = "Histogram of Residuals")

qqnorm(resii)
qqline(resii)
```



Both histogram and qqPlot indicates that the residuals are mostly normal.

```
par(mfrow = c(1, 2))
acf(resii, main="ACF of Residuals")
pacf(resii, main="ACF of Residuals")
```





ACF and PACF plots also look quite good with no obvious peaks that indicate correlation between the residuals.

Now, run a Shapiro Wilk normality test to confirm the normality of residuals

```
shapiro.test(resii)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resii
## W = 0.98884, p-value = 0.0601
```

Shapiro Wilk normality test has p-value greater than .05, so fail to reject the null that residuals are normally distributed at 95% confidence level. It passes the normality test.

More diagnostic tests:

h is can be estimated using sqrt(n), which is sqrt(240) = 15.49 that rounds down to 15 Degrees of freedom for the box pierce test is h-(p+q+P+Q) = 15 - (4+5) = 6

Box-Pierce test and Box-Ljung tests assess the white noise assumption of residuals and McLeod-li tests for non-linear dependence of residuals.

```
Box.test(resii, lag=15, type = c("Box-Pierce"), fitdf = 9)
```

```
##
```

```
##  Box-Pierce test
##
## data:  resii
## X-squared = 19.17, df = 6, p-value = 0.003887
```

```
Box.test(resii, lag=15, type = c("Ljung-Box"), fitdf = 9)
```

```
##
##  Box-Ljung test
##
## data:  resii
## X-squared = 20.134, df = 6, p-value = 0.002621
```

```
Box.test(resii, lag=15, type = c("Ljung-Box"), fitdf = 0)
```

```
##
##  Box-Ljung test
##
## data:  resii
## X-squared = 20.134, df = 15, p-value = 0.1669
```

Unfortunately, the model doesn't pass the Box-Pierce and Ljung-box test due to low p-values. With the p-values lower than .05, the white noise hypothesis is rejected. The model passes the McLeod-li test though.

Lastly, the residuals should resemble a AR(0) process, white noise. So use Yule-Walker method to see if order 0 is selected for the model.

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

```
##
## Call:
## ar(x = resii, aic = TRUE, order.max = NULL, method = c("yule-walker"))
##
##
## Order selected 0  sigma^2 estimated as  0.003046
```

Order selected is 0 and residuals resemble an AR(0) process, which is white noise.

Although the model didn't pass the Box-Pierce and Ljung-Box tests, all the other diagnostics indicate that the residuals are decently normal and I will proceed to forecast with the model ii: SARIMA (p=1,d=1,q=3)x(P=0,D=1,Q=5)$_{s=12}$

$$(1+\phi_1 B)\,(1-B)\,\left(1-B^{12}\right)\,log(X_t) = \left(1+\theta_1 B+\theta_2 B^2+\theta_3 B^3\right)\left(1+\Theta_1 B^{12}+\Theta_2 B^{24}+\Theta_3 B^{36}+\Theta_4 B^{48}+\Theta_5 B^{60}\right) Z_t$$

Forcast

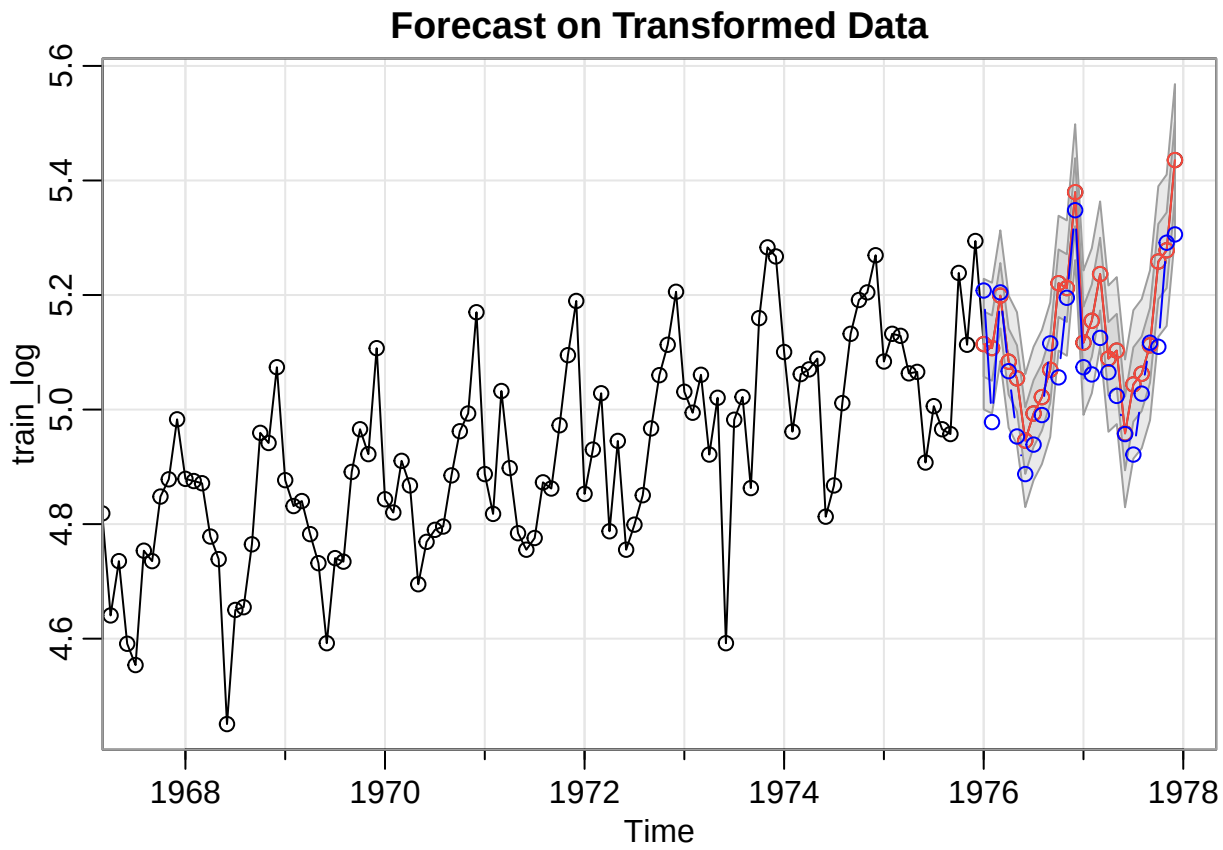
Using the data and the model selected, forecast the next two years (24 months) of monthly beer production.

```
train.log <- log(train$production)
fit <- arima(train.log, order=c(1,1,3), seasonal = list(order = c(0,1,5), period = 12), method="ML")
pred.tr <- predict(fit, n.ahead=24)
```

```
sarima.for(train_log, n.ahead=24 ,p=1, d=1, q=3, P=0, D=1, Q=5, S=12, main="Forecast on Transformed Data
```

```
## $pred
##           Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
## 1976 5.114296 5.107398 5.198774 5.083591 5.054456 4.945772 4.993151 5.021742
## 1977 5.116854 5.154988 5.236541 5.088961 5.102939 4.958440 5.043897 5.062535
##           Sep      Oct      Nov      Dec
## 1976 5.069590 5.220571 5.212137 5.379589
## 1977 5.112220 5.258466 5.278108 5.435583
##
## $se
##             Jan        Feb        Mar        Apr        May        Jun
## 1976 0.05701337 0.05702734 0.05704802 0.05768256 0.05776275 0.05807708
## 1977 0.06304251 0.06326899 0.06343060 0.06386543 0.06413296 0.06448386
##             Jul        Aug        Sep        Oct        Nov        Dec
## 1976 0.05824381 0.05848382 0.05868101 0.05889990 0.05910492 0.05931626
## 1977 0.06478551 0.06511127 0.06542092 0.06573763 0.06604595 0.06635621
```

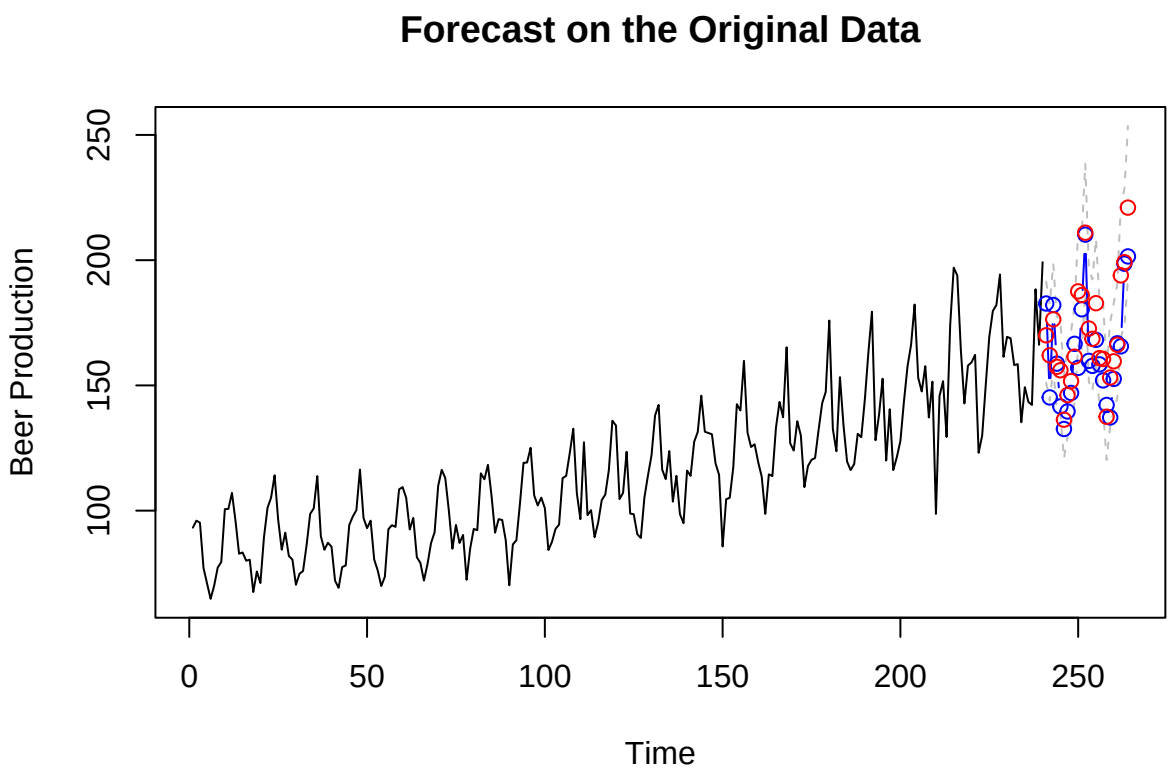
```
lines(log(test_ts), col="blue", type="b")
```



Red dots & line are the predicted values from the forecast, and the confidence interval is shaded around the red points. *Blue dots & line shows the actual data.*

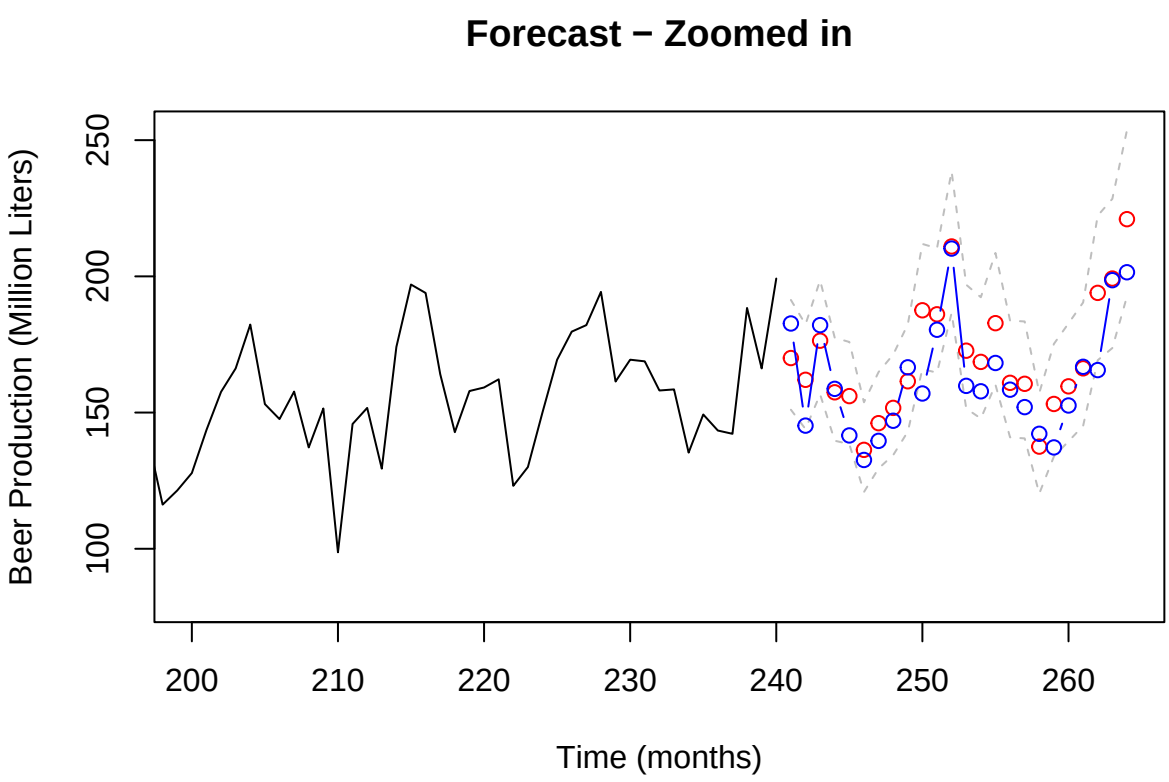
Here's forecast on the original data without transformation

```
U.tr= pred.tr$pred + 2*pred.tr$se
L.tr= pred.tr$pred - 2*pred.tr$se
pred.orig <- exp(pred.tr$pred)
U= exp(U.tr)
L= exp(L.tr)
ts.plot(train$production, xlim=c(1,length(train$production)+24), ylim = c(min(train$production),max(U))
        ylab="Beer Production", main="Forecast on the Original Data")
lines(U, col="grey", lty="dashed")
lines(L, col="grey", lty="dashed")
lines((length(train$production)+1):(length(train$production)+24), test$production, type="b", col="blue")
points((length(train$production)+1):(length(train$production)+24), pred.orig, col="red")
```



And lastly, a zoomed-in visualization of the forecast

```
ts.plot(train$production, xlim = c(200,length(train$production)+24), ylim = c(80,max(U)), ylab="Beer Pr
        xlab="Time (months)", main="Forecast - Zoomed in")
lines(U, col="grey", lty="dashed")
lines(L, col="grey", lty="dashed")
points((length(train$production)+1):(length(train$production)+24), pred.orig, col="red")
lines((length(train$production)+1):(length(train$production)+24), test$production, type="b", col="blue")
```



The forecast seems quite accurate. The actual data generally follows the forecast production values. There are only a couple of points barely outside of the 95% confidence interval, and they are not much off from the forecast.

Conclusion To conclude, this project was quite successful in meeting the goals set in the beginning. The forecast from the final model seem quite accurate and insightful for business uses. Although there is still a room to improve of course. The model not passing two of the diagnostic tests (Box-Pierce and Ljung-Box) indicates that the model isn't the best fit for the data, and there could be a better model with different parameters or something that's not SARIMA that could fit the model better. For now, the final model used: SARIMA (p=1,d=1,q=3)x(P=0,D=1,Q=5)$_{s=12}$

$$(1+\phi_1 B)(1-B)\left(1-B^{12}\right) log(X_t) = \left(1+\theta_1 B+\theta_2 B^2+\theta_3 B^3\right)\left(1+\Theta_1 B^{12}+\Theta_2 B^{24}+\Theta_3 B^{36}+\Theta_4 B^{48}+\Theta_5 B^{60}\right) Z_t$$

does a decent job at predicting the future beer production values each month with the real production values lying inside of the 95% confidence interval almost all the time. The model seems fine for providing a good idea for businesses how much supplies will be needed for beer production each month.

Acknowledge:

Thank you so much to Professor Feldman for providing with guidance and help.

Thank you to Lihao as well for helping me during sections and for the labs.

Apprehendix Kaggle dataset source: https://www.kaggle.com/datasets/sergiomora823/monthly-beer-production?select=datasets_56102_107707_monthly-beer-production-in-austr.csv

Code to try out all the combination of parameters for SARIMA models For differencing at lag 12: SARIMA (p=0 or 1 or 3,d=0,q=0 or 3)(P=4, D=1, Q=1 or 5)$_{s=12}$

```
# Candidate models for differencing at lag 12:
model_12 <- expand.grid(p=c(0,1,3), q=c(0,3), P=4, Q=c(1, 5))
model_12 <- cbind(model_12, AICc=NA)
# Compute AICc:
for (i in 1:nrow(model_12)) {
  sarima.obj <- NULL
  try(arima.obj <- arima(train_log, order=c(model_12$p[i], 0, model_12$q[i]),
                         seasonal=list(order=c(model_12$P[i], 1, model_12$Q[i]), period=12),
                         method="ML"))
  if (!is.null(arima.obj)) { model_12$AICc[i] <- AICc(arima.obj) }
  # print(model_12[i, ])
}
model_12
```

```
save(model_12,file="model_12.Rda")
```

For differencing at lag 12 and 1: SARIMA (p=1 or 2 or 5, d=1, q=1 or 3 or 4)(P=0, D=1, Q=1 or 3 or 5)$_{s=12}$

```
# Candidate models for differencing at lag 12 and 1:
model_12_1 <- expand.grid(p=c(1,2,5), q=c(1,3,4), P=0, Q=c(1,3,5))
model_12_1 <- cbind(model_12_1, AICc=NA)
# Compute AICc:
for (i in 1:nrow(model_12_1)) {
  sarima.obj <- NULL
  try(arima.obj <- arima(train_log, order=c(model_12_1$p[i], 1, model_12_1$q[i]),
                         seasonal=list(order=c(model_12_1$P[i], 1, model_12_1$Q[i]), period=12),
                         method="ML"))
  if (!is.null(arima.obj)) { model_12_1$AICc[i] <- AICc(arima.obj) }
  # print(model_12_1[i, ])
}
model_12_1
```

```
save(model_12_1,file="model_12_1.Rda")
```