Forecasting Las Vegas Visitor Volume

Jasmine Mohammed

Abstract

This project centers on analyzing and using a time series data set to forecast future Las Vegas visitor volume. To do this, I first plot Las Vegas visitor volume data and analyze any characteristics of non-stationarity. Once identified, I perform differencing to make the data stationary after setting aside some test points to validate the forecasting later. When I have a stationary dataset to work with, I analyze plots of the ACF and PACF to preliminarily identify models for forecasting. Several models are fitted and tweaked via estimating coefficients. The models with the lowest AICcs are chosen and checked to see if they are stationary and invertible by checking if the roots of these models are outside of the unit circle. Once deemed stationary and invertible, diagnostics are performed on the data to see if it is suitable for forecasting. This is mainly done through analyzing the residuals of these models. Finally, the model with the best diagnostics is chosen and used to forecast the data. The forecast is compared with the test data to validate its accuracy. In this project, I was able to identify a model that forecasted future Las Vegas visitor volume accurately.

Introduction

The problem I am trying to address is whether or not I can predict future Las Vegas visitor volume. This is important for me because Las Vegas is one of my favorite vacation spots and I would like to know when it is most packed because that will provide insight into many things including cost of rooms and ease of obtaining access into events. As a broke 21 year old, this is of high importance to me.

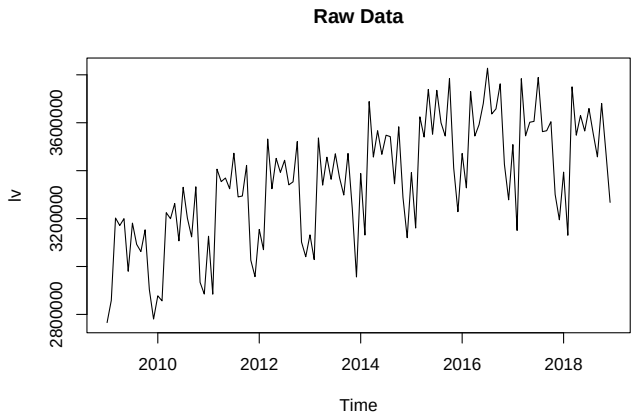
I plan to address this problem by using Las Vegas visitor volume data to forecast the future visitor volume of Las Vegas. The Las Vegas visitor volume dataset I will be working with contains monthly Las Vegas visitor volume data from January 2009 to December 2018. This time series data consists of monthly totals for the number of people that visited Las Vegas. I retrieved the dataset from the Las Vegas Convention and Visitors Authority. To analyze and forecast the data, I used RStudio. Techniques used to achieve this include data processing via differencing, model identification, estimation, and selection, and diagnostic checking to determine a model best fit for forecasting. Despite some non-ideal outcomes when completing this project–increased variance after removing trend, models being non-stationary and invertible, and models not completely passing the diagnostics–I was able to determine a model that accurately forecasted Las Vegas visitor volume.

Lav Vegas Visitor Volume Data
- Monthly Las Vegas visitor volume data (January 2009 - December 2018)
- Monthly totals in number of visitors $\{U_t, t = 1, 2, ..., 120\}$
- Data Source: Las Vegas Convention and Visitors Authority

Exploratory Data Analysis

To start I plot all the observations included in the Las Vegas visitor volume data to get a sense of what I am working with:

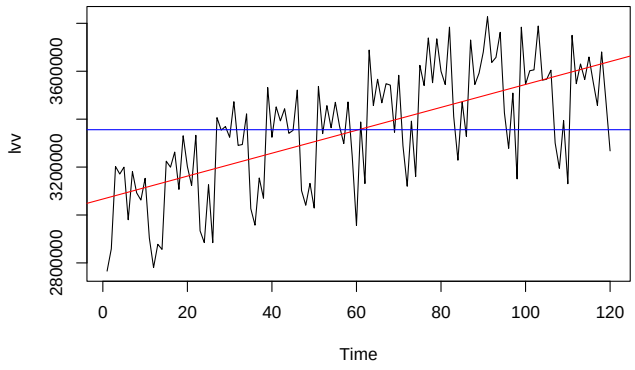
```
library(tsdl)
# read in data
LasVegasVisitors <- read.csv("/Users/jasminechung/Downloads/LasVegasVisitors.csv")
# plot data with years on x-axis
lv <- ts(LasVegasVisitors$MonthlyValue, start = c(2009, 1), end = c(2018, 12), frequency = 12)
plot.ts(lv, main = "Raw Data")
```



I also plot the time series with lines for trend (red) and constant mean (blue) to get an even better visualization of the data:

```
# plot data with trend and mean
lvv <- ts(LasVegasVisitors$MonthlyValue)
plot.ts(lvv)
fit <- lm(lvv ~ time(lvv)); abline(fit, col="red") # added trend to data plot
abline(h=mean(lvv), col="blue") # added mean (constant) to data plot)
mean(lvv) # 3355972
```

```
## [1] 3355972
```

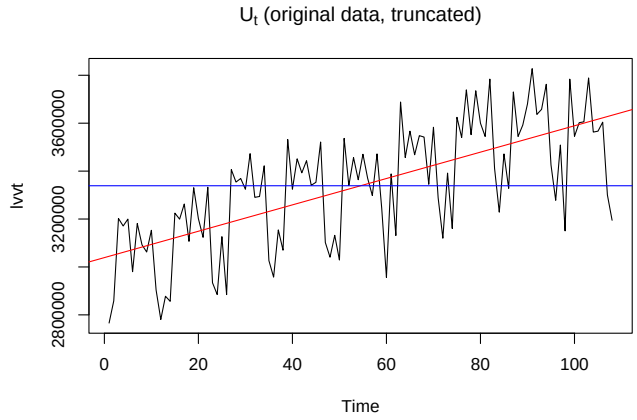


Looking at the graphs above, my immediate observations are that there seems to be an increasing trend as well as seasonality. This leads me to believe that the data is highly non-stationary. The data also seems to have a constant variance throughout. The mean of the data is 3355972.

Since I don't have any new data to check the validity of the model I will develop, I partition the dataset into two parts: one part for model training and one for model validation. The training dataset U_t will include all but the last 12 data points (108 of the 120 observations). These 12 data points will be instead set aside in a test dataset to be used for model validation later. Plotting the training dataset gives me:

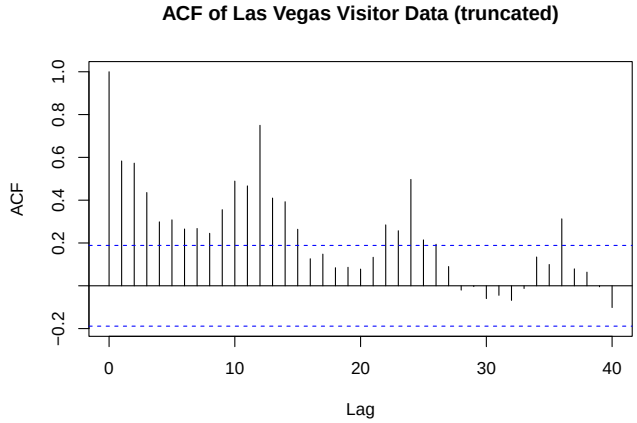
```
lvvt = lvv[c(1: 108)] # training dataset U_t, to build a model
lvv.test = lvv[c(109:120)] # test dataset
# plot the truncated dataset U_t
```

```
plot.ts(lvvt, main = expression("U"[t]*" (original data, truncated)"))
fit <- lm(lvvt ~ time(lvvt))
abline(fit, col="red")
abline(h=mean(lvvt), col="blue")
#mean(lvvt)
```



My immediate observations hold for the training dataset; the data is highly non-stationary as can be seen clearly through the increasing trend as well as the seasonality. The variance, however, seems to be constant. To confirm my suspicions of non-stationarity, I plot the ACF of U_t:

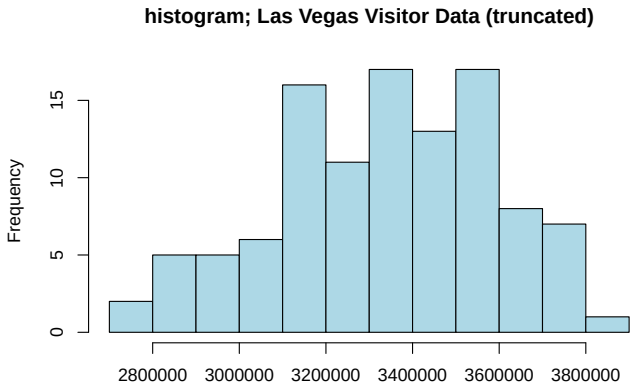
```
acf(lvvt,lag.max=40, main="ACF of Las Vegas Visitor Data (truncated)")
```



Looking at the ACF, I see that it remains large and periodic, which supports my immediate observations of non-stationarity via the presence of trend and seasonality.

Next, I plot the histogram of U_t:

```
hist(lvvt, col="light blue", xlab="", main="histogram; Las Vegas Visitor Data (truncated)")
```



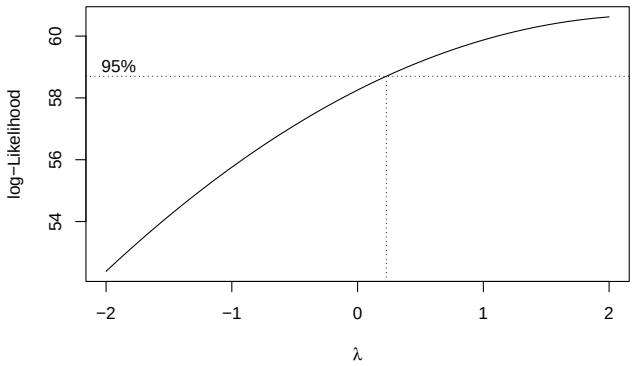
The histogram looks pretty normal, though slightly skewed, indicating that I may need to stabilize my variance as well.

Data Processing

Now I work on making my data stationary. First, to check if I actually do need to stabilize the variance, I check and see what Box-Cox gives me for the value of λ.

```
library(MASS)
bcTransform <- boxcox(lvvt~time(lvvt))
lambda=bcTransform$x[which(bcTransform$y == max(bcTransform$y))]
lambda
```

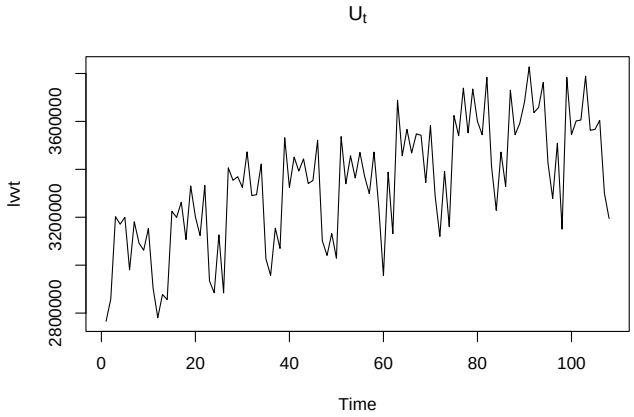
```
## [1] 2
```



Box cox gives me $\lambda = 2$ and the above graph. The confidence interval contains the value lambda $= 1$, so a transformation may not be necessary. However, since I got a value of 2 for lambda, I will apply a transformation to U_t by squaring it and then compare U_t^2 to U_t to see if the transformation was necessary.

Plotting U_t and U_t^2 gives me:

```
library(gridExtra)
lvvt.sq <- lvvt^2
plot.ts(lvvt, main = expression("U"[t]))
```

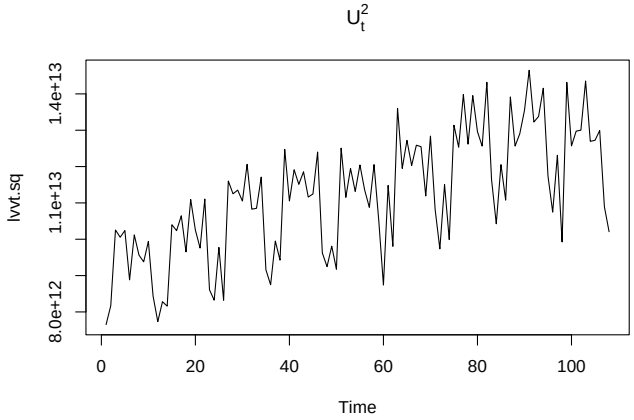



```
plot.ts(lvvt.sq, main = expression("U"[t]^2))
var(lvvt)
```

```
## [1] 63881794951
```

```
var(lvvt.sq)
```

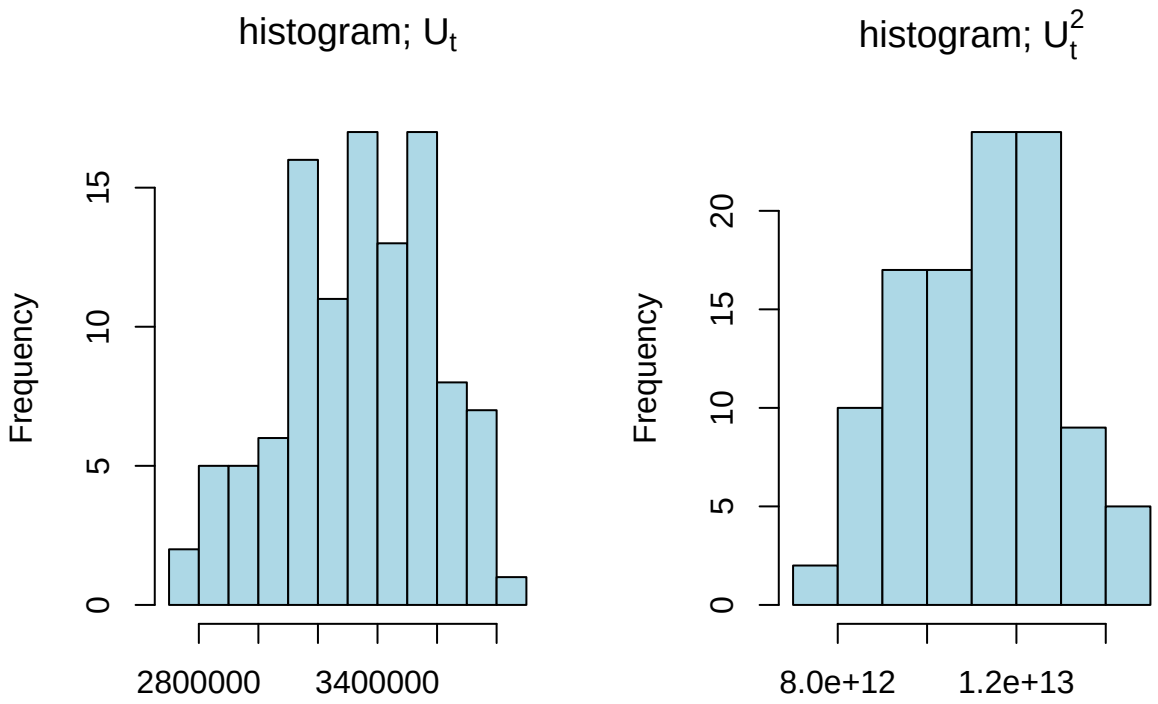
```
## [1] 2.805844e+24
```



Looking at the plots of U_t and U_t^2, I see little to no difference. Also, the variance of U_t is 6.388e+10 before transformation and 2.806e+24 after transformation (U_t^2). This large increase in variance and the lack of change in plots tells me that transforming U_t is unnecessary.

Just to be certain a transformation is not needed, I also plot the histograms of U_t and U_t^2:

```
par(mfrow=c(1,2))
hist(lvvt, col="light blue", xlab="", main=expression("histogram; U"[t]))
hist(lvvt.sq, col="light blue", xlab="", main=expression("histogram; U"[t]^2))
```



Looking at the histograms of U_t and U_t^2, it is apparent that the transformation did not improve the slight skew. Therefore since 1 is in the confidence interval for lambda and the attempt at transformation did not show significant improvements, I will move forward without transforming the data for the sake of simplicity.

Looking at the histograms of U_t and U_t^2 above, it is apparent that the transformation did not improve the slight skew. Therefore, since the attempt at transformation did not show significant improvements and since 1 is also in the confidence interval for lambda, I will move forward without transforming the data for the sake of simplicity.

I plot the decomposition of U_t as one last check to support my observations for trend, seasonality, and variance:

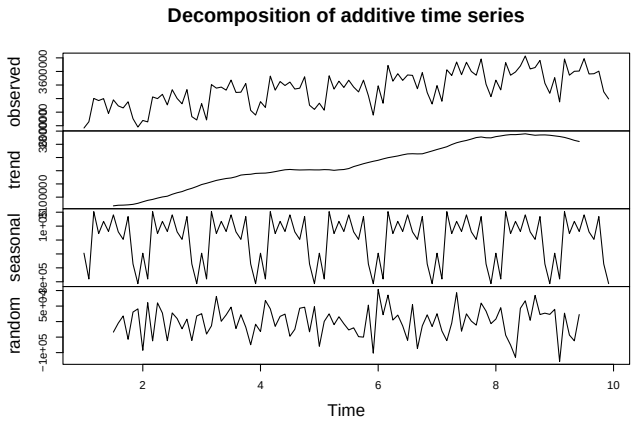
```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.1.2
```

```
library(ggfortify)
```

```
## Warning: package 'ggfortify' was built under R version 4.1.2
```

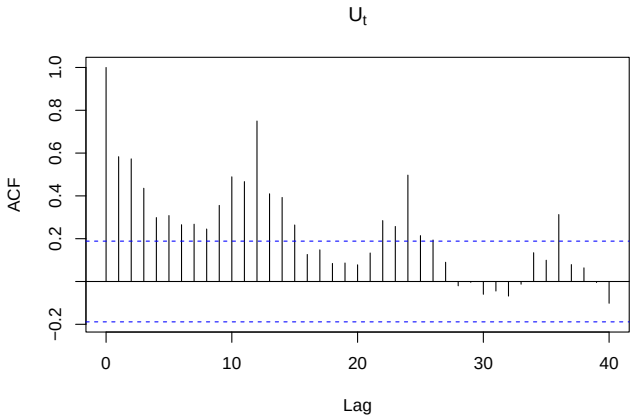
```
y <- ts(as.ts(lvvt), frequency = 12)
decomp <- decompose(y)
plot(decomp)
```



Looking at the decomposition of U_t above, it shows seasonality and a linear trend. Looking at the part labeled "random" which shows whats left after removing trend and seasonality, we see that variance looks similar throughout. All in all, this confirms my observations and tells me that I just need to remove trend and seasonality to make the data stationary.

To remove trend and seasonality, I will difference the data. First I will remove seasonality since that occasionally removes trend as well. If after differencing at lag 12 there is still a trend, I will then difference at lag 1 to remove it. However, I need to first determine at which lag I need to difference at to remove seasonality. To do this, I examine the ACF of U_t:

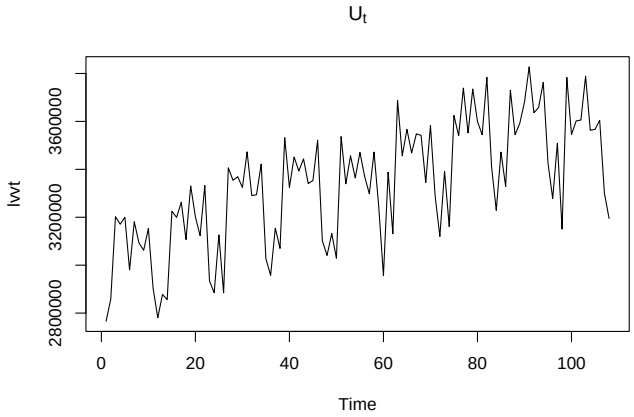
```
acf(lvvt,lag.max=40, main = expression("U"[t]))
```



The ACF of U_t shows peaks at multiples of 12. Therefore, I need to difference at lag 12.

As a reminder, here is the plot of Ut before differencing. Both seasonality and trend are very apparent.

```
plot.ts(lvvt, main = expression("U"[t]))
```




```
var(lvvt) # 6.388e+10
```

```
## [1] 63881794951
```

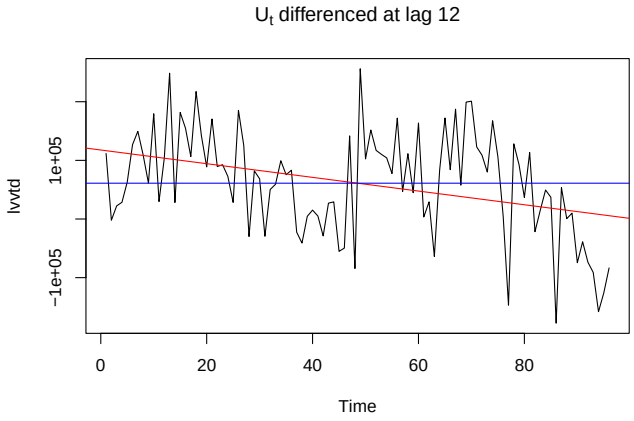
Variance of U_t before differencing: 6.388e+10

I then difference at lag 12 to remove seasonality.

```
lvvtd <- diff(lvvt, lag=12)
plot.ts(lvvtd, main=expression("U"[t]*" differenced at lag 12"))
var(lvvtd) # 8.087e+09
```

```
## [1] 8087403593
```

```
fit <- lm(lvvtd ~ time(lvvtd)); abline(fit, col="red")
# mean(lvvtd)
abline(h=mean(lvvtd), col="blue")
```



Variance of U_t differenced at lag. 12: 8.087e+09

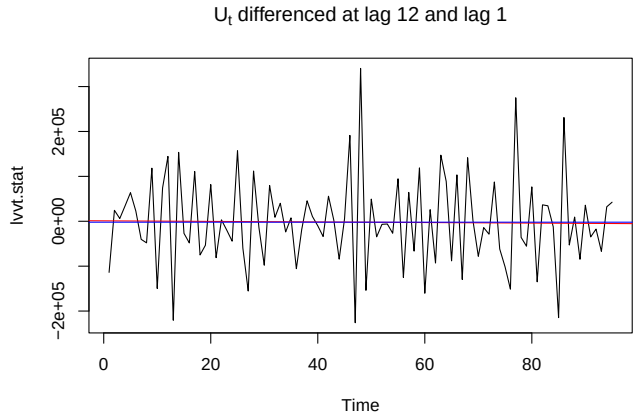
After differencing at lag 12, seasonality seems to have been eliminated. However, the data still looks non-stationary since I still observe a trend.

I then difference at lag 1 to remove trend.

```
lvvt.stat <- diff(lvvtd, lag=1)
plot.ts(lvvt.stat, main=expression("U"[t]*" differenced at lag 12 and lag 1"))
var(lvvt.stat) # 1.046e+10
```

```
## [1] 10459616465
```

```r
fit <- lm(lvvt.stat ~ time(lvvt.stat)); abline(fit, col="red")
# mean(lvvt.stat)
abline(h=mean(lvvt.stat), col="blue")
```

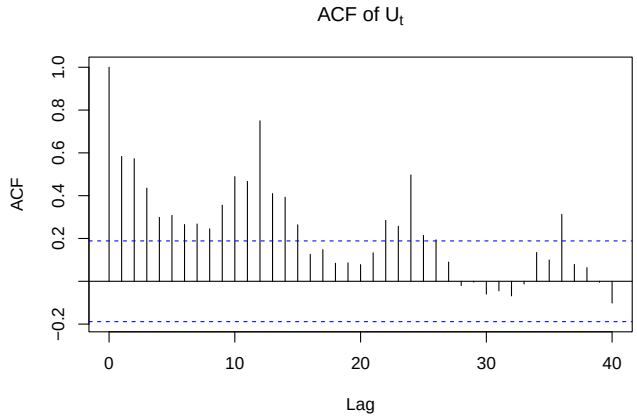


Variance of U_t differenced at lag 12 and lag 1: 1.046e+10

After differencing again at lag one, the trend is almost completely eliminated and the data looks stationary now. Looking at the variances, I see that differencing at lag 12 decreased the variance significantly. However, differencing again at lag 1 increased the variance slightly. This is not ideal and could be a sign of overdifferencing, but looking at the plots, it seems like the second differencing was necessary.

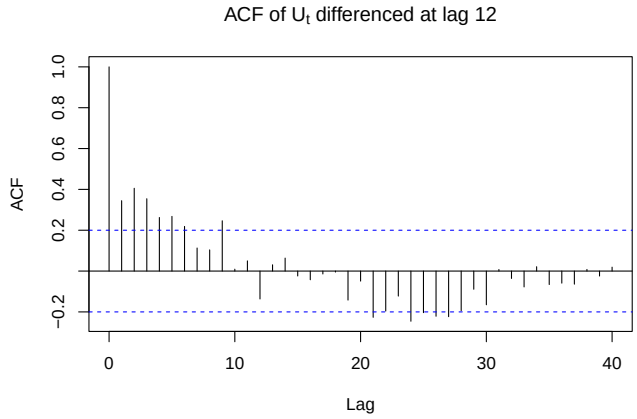
To getter a better sense, I will compare the ACFs.

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



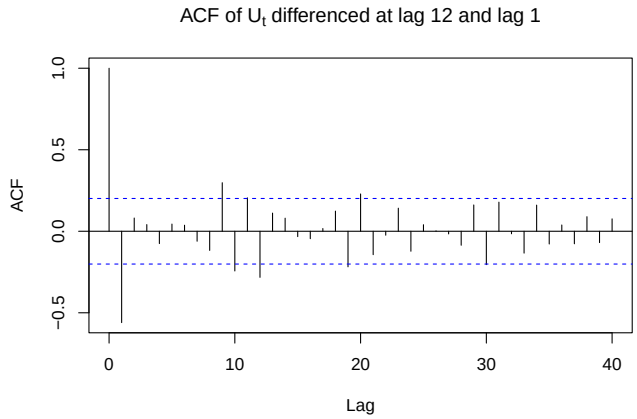
The ACF of U_t before differencing shows slow decay and periodicity, indicating non-stationarity.

```r
acf(lvvtd, lag.max=40, main=expression("ACF of U"[t]*" differenced at lag 12"))
```



The ACF of U_t after differencing at lag 12 made seasonality no longer apparent, but the ACF still decays slowly indicating non-stationarity.

```
acf(lvvt.stat, lag.max=40, main=expression("ACF of U"[t]*" differenced at lag 12 and lag 1"))
```

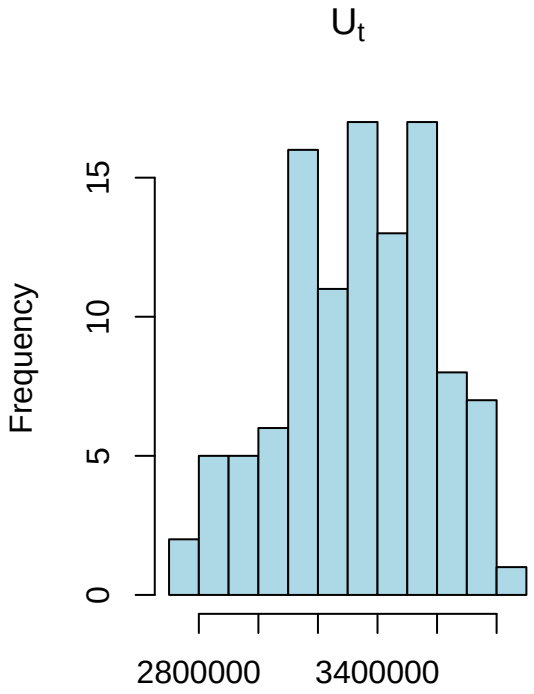


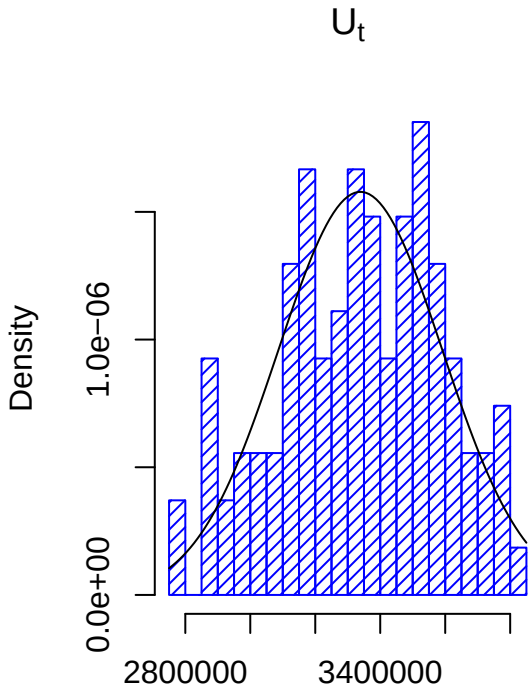
The ACF of U_t after differencing at lag 12 and lag 1 made both seasonality and trend no longer apparent and looks in line with a stationary process.

Overall, both the plots and the ACFs seem to indicate that differencing at both lags 12 and 1 is the right move.

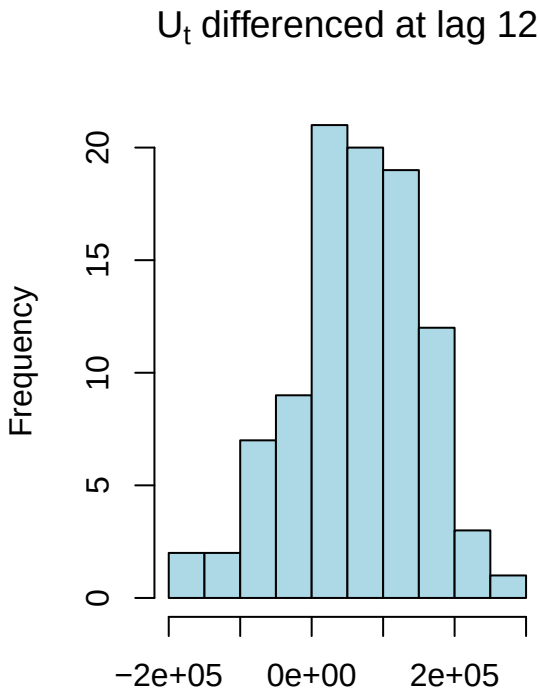
As one last check, I will compare the historgrams.

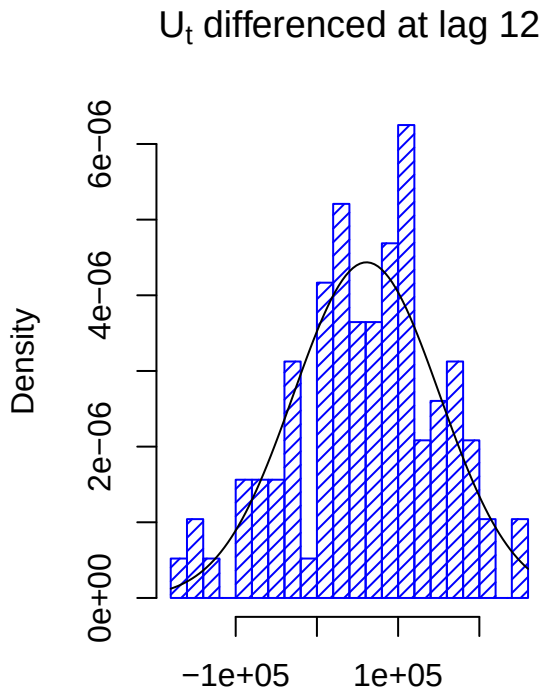
```
par(mfrow=c(1,2))
hist(lvvt, col="light blue", xlab="", main=expression("U"[t]))
hist(lvvt, density=20,breaks=20, col="blue", xlab="", main=expression("U"[t]), prob=TRUE)
m<-mean(lvvt)
std<- sqrt(var(lvvt))
curve( dnorm(x,m,std), add=TRUE )
```





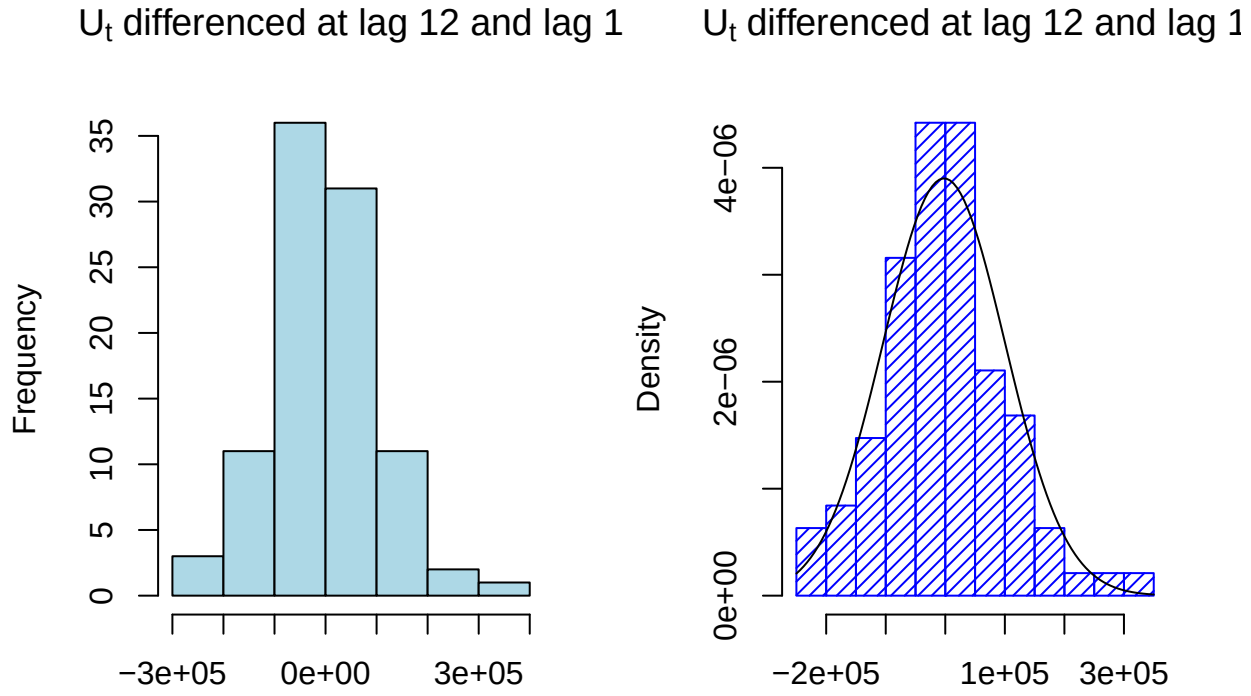

```
par(mfrow=c(1,2))
hist(lvvtd, col="light blue", xlab="", main=expression("U"[t]*" differenced at lag 12"))
hist(lvvtd, density=20,breaks=20, col="blue", xlab="", main=expression("U"[t]*" differenced at lag 12")
m<-mean(lvvtd)
std<- sqrt(var(lvvtd))
curve( dnorm(x,m,std), add=TRUE )
```






```
par(mfrow=c(1,2))
hist(lvvt.stat, col="light blue", xlab="", main=expression("U"[t]*" differenced at lag 12 and lag 1"))
hist(lvvt.stat, density=20,breaks=20, col="blue", main=expression("U"[t]*" differenced at lag 12 and lag
```

```
m<-mean(lvvt.stat)
std<- sqrt(var(lvvt.stat))
curve( dnorm(x,m,std), add=TRUE )
```



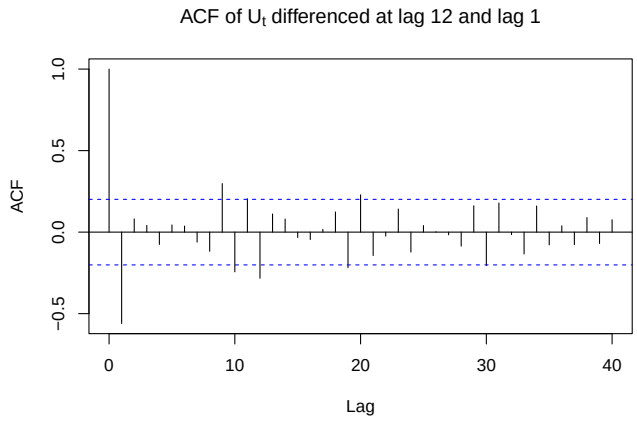
Looking at the histograms above, I see that U_t differenced at lag 12 and lag 1 fits the normal curve the best and also looks the most symmetrical, which is ideal for modeling.

To conclude, after comparing the different plots, ACFs, and histograms, I will move forward with U_t differenced at both lags 12 and 1, despite the small increase in variance. To be sure I did not over-difference, I will check the unit roots of my pure MA models later.

Model Identification

Next, I plot and analyze the ACF and PACF of my now stationary data to preliminary identify my model(s):

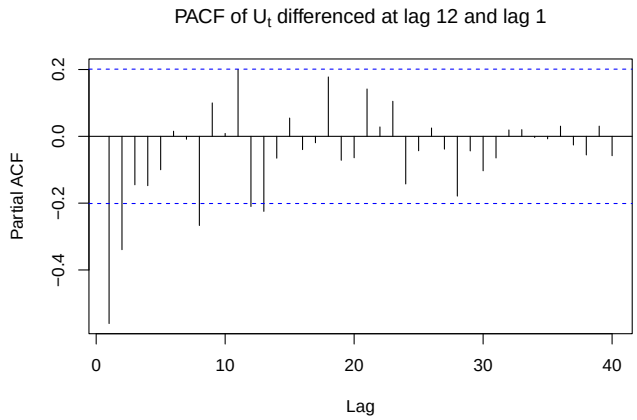
```
acf(lvvt.stat, lag.max=40, main=expression("ACF of U"[t]*" differenced at lag 12 and lag 1"))
```



The ACF is outside the confidence intervals at Lags 1, 9, maybe 10, 12, and maybe 20. Since the ACF at lags 10 and 20 are so close to the border, the Bartlett's formula tells me that I can disregard them. So I have

Q = 1 since there is a significant lag at only lag 12 and no other multiples of 12. Looking at the ACFs within the period (lag 1 to lag 11), I have q = 1 or 3 since ACF is significant at lag 1 and lag 9 and 12 - 9 = 3.

```
pacf(lvvt.stat, lag.max=40, main=expression("PACF of U"[t]*" differenced at lag 12 and lag 1"))
```



The PACF is outside the confidence intervals at Lags 1, 2, 8, maybe 12, and maybe 13. So I have P = 0 or 1 since there is a significant lag at only lag 12 and no other multiples of 12 and since that lag is very close to the border. Looking at the PACFs within the period (lag 1 to lag 11), I have p = 1, 2, or 4 since the PACF is significant at lags 1, 2, and 8 and because 12 - 8 = 4.

Since I differenced once at lag 12 and once at lag 1, D = 1 and d = 1.

Model Estimation and Selection

To summarize, here is my list of candidate models to try: SARIMA for U_t: s=12, D=1, d=1; Q = 1; P = 0 or 1; q = 1 or 3, p= 1, 2, or 4

I tried different models and selected the best few based on having the lowest AICc. I started with pure MA models and made sure to check that there were no unit roots so I could confirm I did not over difference. All the roots were outside of the unit circle, confirming that I did not over-difference the data and could continue. Then I tried SARIMA models. I started by adding P=1 into the models and saw it kept increasing the AICc. I also tried pure AR models, but the AICcs also were much larger than the pure MA models. Therefore, I decided to keep P=0 moving forward. I then tried just adding p = 1, 2, and 4. A couple of these models produced lower AICcs than the pure MA models.

Here are some pure MA models I tried:

SMA: Q = 1 and q = 1

```
library(qpcR)
```

```
## Loading required package: minpack.lm
```

```
## Warning: package 'minpack.lm' was built under R version 4.1.2
```

```
## Loading required package: rgl
```

```
## Warning: package 'rgl' was built under R version 4.1.2
```

```
## Loading required package: robustbase
```

```
## Warning: package 'robustbase' was built under R version 4.1.2
```

```
## Loading required package: Matrix
```

```
m1 <- arima(lvvt, order=c(0,1,1), seasonal = list(order = c(0,1,1), period = 12), method="ML")
m1
```

```
##
## Call:
## arima(x = lvvt, order = c(0, 1, 1), seasonal = list(order = c(0, 1, 1), period = 12),
##     method = "ML")
##
## Coefficients:
##           ma1     sma1
##       -0.7267  -0.5613
## s.e.   0.0639   0.1236
##
## sigma^2 estimated as 4.934e+09:  log likelihood = -1197.63,  aic = 2401.25
```

```
AICc(m1)
```

```
## [1] 2401.367
```

Since the absolute value of the coefficient for ma1 is less than 1 ($|-0.727| < 1$), the root is outside of the unit circle. Therefore the model is invertible.

SMA: Q = 1 and q = 3

```
m2 <- arima(lvvt, order=c(0,1,3), seasonal = list(order = c(0,1,1), period = 12), method="ML")
m2
```

```
##
## Call:
## arima(x = lvvt, order = c(0, 1, 3), seasonal = list(order = c(0, 1, 1), period = 12),
##     method = "ML")
##
## Coefficients:
##           ma1     ma2      ma3     sma1
##       -0.8630  0.2370  -0.0511  -0.5697
## s.e.   0.1066  0.1556   0.1176   0.1241
##
## sigma^2 estimated as 4.761e+09:  log likelihood = -1196.03,  aic = 2402.05
```

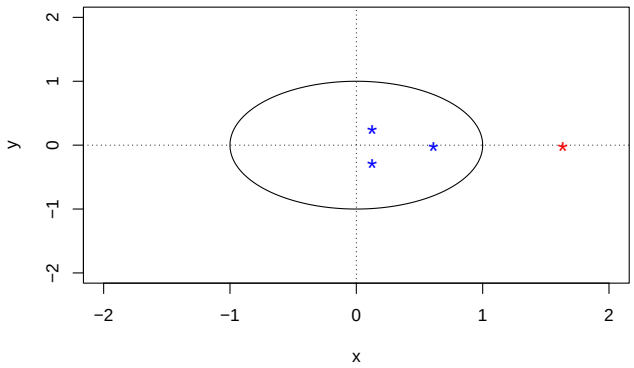
```
AICc(m2)
```

```
## [1] 2402.443
```

```
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="d
      if(!is.null(special))
        {lines(Re(special),Im(special),lwd=2)}
      if(!is.null(sqecial))
        {lines(Re(sqecial),Im(sqecial),lwd=2)}}

plot.roots(NULL, polyroot(c(1, -0.863,  0.237,  -0.051)))
```



The roots are all outside the unit circle. 0 is in the confidence interval for ma3, so I tried fixing that to zero which is equivalent to just having q=2:

SMA: Q = 1 and q = 2

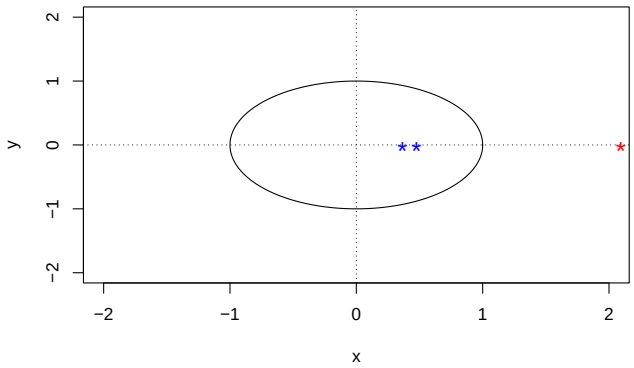
```
m3 <- arima(lvvt, order=c(0,1,2), seasonal = list(order = c(0,1,1), period = 12), method="ML")
m3
```

```
##
## Call:
## arima(x = lvvt, order = c(0, 1, 2), seasonal = list(order = c(0, 1, 1), period = 12),
##     method = "ML")
##
## Coefficients:
##           ma1     ma2     sma1
##       -0.8439  0.1755  -0.5601
## s.e.   0.0980  0.1050   0.1220
##
## sigma^2 estimated as 4.781e+09:  log likelihood = -1196.13,  aic = 2400.26
```

```
AICc(m3)
```

```
## [1] 2400.489
```

```
plot.roots(NULL, polyroot(c(1, -0.844,  0.175)))
```



The AICc has decreased from the prior model and the roots are all outside the unit circle.

Since the roots for pure MA are all outside the unit circle, my model is not over-differenced and I may continue. Now we try adding in P and see if the AICc decreases:

SARMA: Q = 1, q = 1, P = 1

```
m10 <- arima(lvvt, order=c(0,1,1), seasonal = list(order = c(1,1,1), period = 12), method="ML")
m10
```

```
##
## Call:
## arima(x = lvvt, order = c(0, 1, 1), seasonal = list(order = c(1, 1, 1), period = 12),
##     method = "ML")
##
## Coefficients:
##           ma1    sar1     sma1
##       -0.7299  0.1747  -0.6838
## s.e.   0.0639  0.1761   0.1580
##
## sigma^2 estimated as 4.864e+09:  log likelihood = -1197.12,  aic = 2402.23
```

```
AICc(m10)
```

```
## [1] 2402.463
```

AICc is higher than the pure MA model.

SARMA: Q = 1, q = 3, P = 1

```
m20 <- arima(lvvt, order=c(0,1,3), seasonal = list(order = c(1,1,1), period = 12), method="ML")
m20
```

```
##
## Call:
## arima(x = lvvt, order = c(0, 1, 3), seasonal = list(order = c(1, 1, 1), period = 12),
##     method = "ML")
##
## Coefficients:
##           ma1     ma2      ma3    sar1     sma1
##       -0.8596  0.2255  -0.0510  0.1553  -0.6796
## s.e.   0.1054  0.1508   0.1151  0.1782   0.1617
##
## sigma^2 estimated as 4.705e+09:  log likelihood = -1195.64,  aic = 2403.27
```

```
AICc(m20)
```

```
## [1] 2403.859
```

AICc is higher than the pure MA model.

Since adding in P has shown increases in AICc, I will stop working with it from here on out.

Next I tried adding in p to see if the AICc decreases. Of the models I tried, the following gave me the lowest AICcs:

SARMA: Q = 1, q = 1, p = 1

```
m4 <- arima(lvvt, order=c(1,1,1), seasonal = list(order = c(0,1,1), period = 12), method="ML")
m4
```

```
##
```

```
## Call:
## arima(x = lvvt, order = c(1, 1, 1), seasonal = list(order = c(0, 1, 1), period = 12),
##     method = "ML")
##
## Coefficients:
##           ar1      ma1     sma1
##       -0.2464  -0.6034  -0.5685
## s.e.   0.1393   0.1131   0.1229
##
## sigma^2 estimated as 4.776e+09:  log likelihood = -1196.17,  aic = 2400.33
```

```
AICc(m4)
```

```
## [1] 2400.565
```

SARMA: Q = 1, q = 3, p = 2

```
m8 <- arima(lvvt, order=c(2,1,3), seasonal = list(order = c(0,1,1), period = 12), method="ML")
m8
```

```
##
## Call:
## arima(x = lvvt, order = c(2, 1, 3), seasonal = list(order = c(0, 1, 1), period = 12),
##     method = "ML")
##
## Coefficients:
##           ar1      ar2     ma1     ma2      ma3     sma1
##       -1.1693  -0.9980  0.4773  0.2337  -0.6687  -0.5665
## s.e.   0.0126   0.0059  0.0827  0.0948   0.0789   0.1078
##
## sigma^2 estimated as 3.998e+09:  log likelihood = -1190.32,  aic = 2394.64
```

```
AICc(m8)
```

```
## [1] 2395.474
```

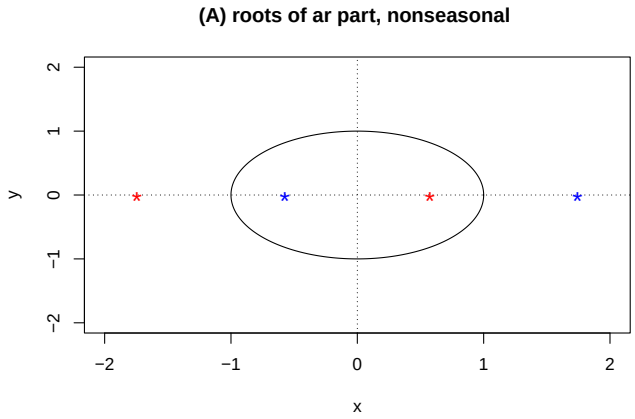
I also tried AR models with P=1 and p = 1, 2 or 4, but they produced much higher AICcs than the previous models.

Overall the three models I identified as having with the lowest AICcs (in order) were: Model (A). SARMA: Q = 1, q = 3, p = 2 ; AICc = 2395 Model (B). SMA: Q = 1 and q = 2; AICc = 2400 Model (C). SMA: Q = 1 and q = 1; AICc = 2401 * Since two models essentially tied for third lowest, I chose the one with the least coefficients following the Principle of Parsimony.

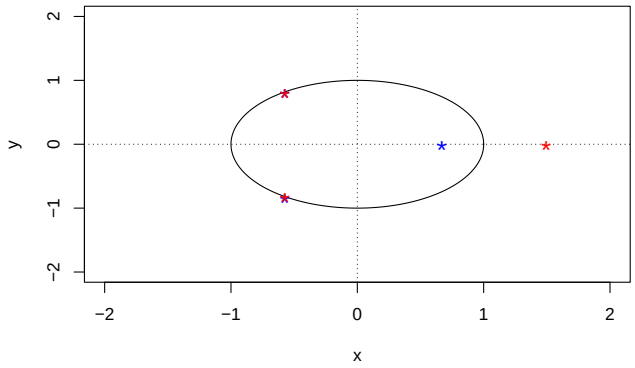
I checked these models to see if they were stationary and invertible and therefore could be used before proceeding:

For models (B) and (C), they are both pure MA and thus both stationary; I also showed earlier that their roots are outside of the unit circle,so they are also both invertible and can therefore be used. Model (A) however, is not stationary and invertible since it has roots in the unit circle for the ar part and ma part as shown in the following root plots:

```
plot.roots(NULL, polyroot(c(1, -1.169,  -0.998)), main="(A) roots of ar part, nonseasonal ") # ar roots
```




```
plot.roots(NULL, polyroot(c(1, 0.477, 0.234, -0.669)), main="(A) roots of ma part, nonseasonal ") # ma
```



Therefore, the final two models I decided to proceed with were:
Model B:
$\nabla_1\nabla_{12}U_t = (1 - 0.844_{(0.098)}B - 0.175_{(0.105)}B^2)(1 - 0.56_{(0.122)}B^{12})Z_t \ ; \ \sigma_Z^2 = 4.78e + 09$

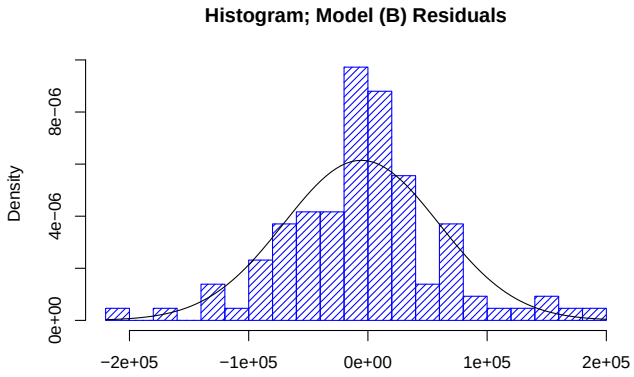
Model C:
$\nabla_1\nabla_{12}U_t = (1 - 0.727_{(0.064)}B)(1 - 0.561_{(0.124)}B^{12})Z_t \ ; \ \sigma_Z^2 = 4.93e + 09$

Model diagnostics

Next, I performed diagnostic checking on my final two models to see which, if any, I could use for forecasting.

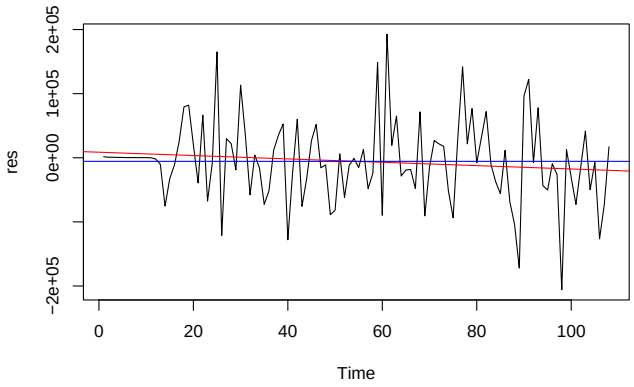
Model B Diagnostic Checking:

```
fit <- arima(lvvt, order=c(0,1,2), seasonal = list(order = c(0,1,1), period = 12), method="ML")
res <- residuals(fit)
hist(res,density=20,breaks=20, col="blue", xlab="", prob=TRUE, main = "Histogram; Model (B) Residuals")
m <- mean(res)
std <- sqrt(var(res))
curve( dnorm(x,m,std), add=TRUE )
```



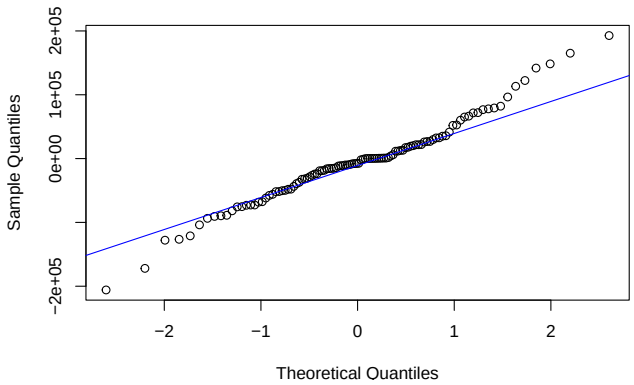
The histogram of the residuals looks pretty good. It seems to follow the normal line well enough.

```
plot.ts(res, main = "Model (B) Residuals")
fitt <- lm(res ~ as.numeric(1:length(res))); abline(fitt, col="red")
abline(h=mean(res), col="blue")
```



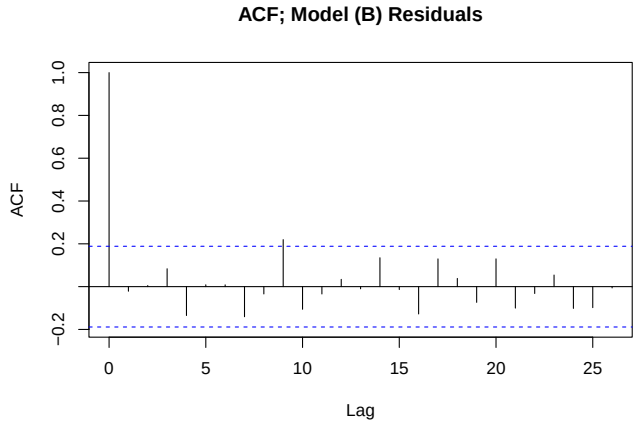
Looking at the plot of the residuals, I see there is a tiny downward trend, however it is practically invisible. I also see no seasonality or visible change of variance.

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

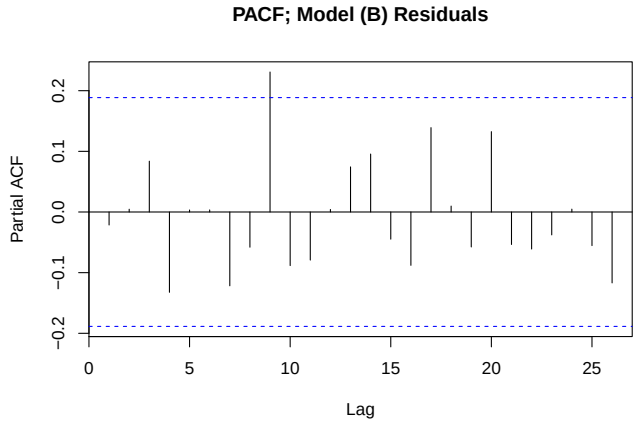


The Q-Q plot also looks pretty good. There are some deviations, but there are mostly at the tails.

```
acf(res, lag.max=26, main = "ACF; Model (B) Residuals")
```




```
pacf(res, lag.max=26, main = "PACF; Model (B) Residuals")
```



The ACF of the residuals also looks good. The ACF does fall out of the confidence interval at lag 9, however, only just barely. The ACF still looks good since it's only 1 out of 26 lags that falls outside the confidence interval. The PACF of the residuals also looks pretty good. It does fall out of the confidence interval at lag 9 as well, however, since it's only 1 out of 26 lags that falls outside, it still looks fine. Therefore, the ACF and PACF look good enough to be considered white noise.

Running the Shapiro-Wilk test of normality, I get a p-value less than 0.05. Therefore, model (B) does not pass the test and rejects the null hypothesis of normality.

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

```
##
##  Shapiro-Wilk normality test
##
## data:  res
## W = 0.97457, p-value = 0.03625
```

I then run Box-Pierce and Ljung-Box tests and the McLeod Li test. I set the lag equal to 10 because lag h approximately equals the square root of the sample size and the square root of 108 is about 10 (lag is used to find degrees of freedom, so it needs to be an integer and so I round to the nearest whole number). I set fitdf = 3 because fitdf is equal to the number of estimated coefficients for the model. The degree of freedom is equal to lag minus fitdf, so the df for Box-Pierce and Ljung-Box tests are 10 - 3 = 7. However, for the McLeod Li test, degrees of freedom is equal to lag H so fitdf must be set to 0. All three tests give p-values larger than 0.05, which indicates that our model fits the data.

```
Box.test(res, lag = 10, type = c("Box-Pierce"), fitdf = 3)
```

```
##
##  Box-Pierce test
##
## data:  res
## X-squared = 11.473, df = 7, p-value = 0.1193
```

```
Box.test(res, lag = 10, type = c("Ljung-Box"), fitdf = 3)
```

```
##
##  Box-Ljung test
##
## data:  res
## X-squared = 12.555, df = 7, p-value = 0.08372
```

```
Box.test(res^2, lag = 10, type = c("Ljung-Box"), fitdf = 0)
```

```
##
##  Box-Ljung test
##
## data:  res^2
## X-squared = 8.081, df = 10, p-value = 0.6209
```

I then check the residuals using the yule-walker method where I ask it to use AICc to automatically select an order. It fitted the residuals to AR(0), which is the same as white noise.

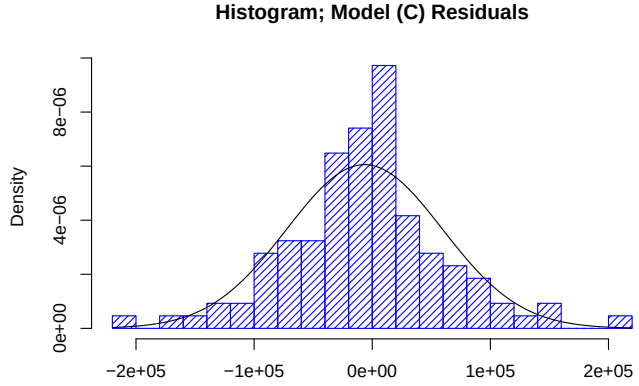
```
ar(res, aic = TRUE, order.max = NULL, method = c("yule-walker"))
```

```
##
## Call:
## ar(x = res, aic = TRUE, order.max = NULL, method = c("yule-walker"))
##
##
## Order selected 0  sigma^2 estimated as  4.215e+09
```

Therefore Model (B) passes all tests but the Shapiro-Wilk test.

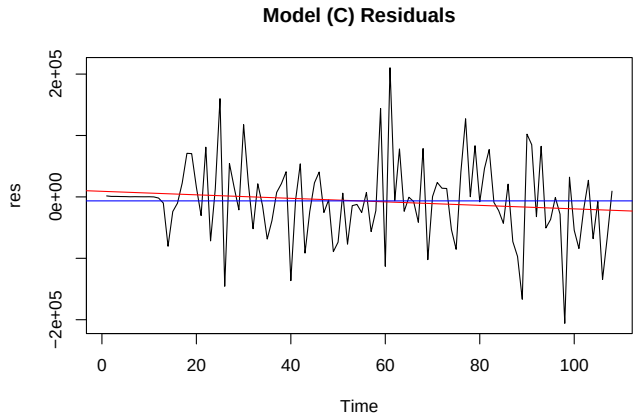
Model C Diagnostic Checking

```
fit <- arima(lvvt, order=c(0,1,1), seasonal = list(order = c(0,1,1), period = 12), method="ML")
res <- residuals(fit)
hist(res,density=20,breaks=20, col="blue", xlab="", prob=TRUE, main = "Histogram; Model (C) Residuals")
m <- mean(res)
std <- sqrt(var(res))
curve( dnorm(x,m,std), add=TRUE )
```



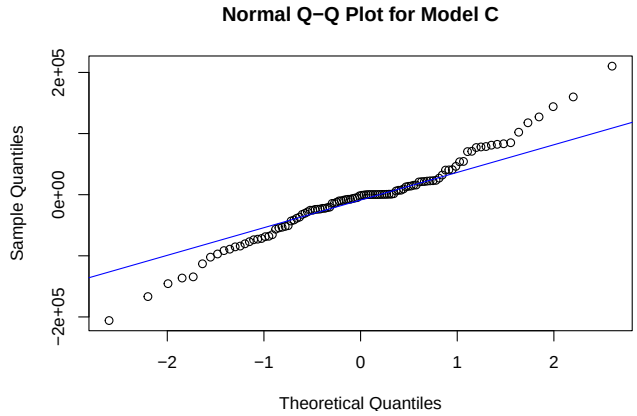
The histogram of the residuals also looks pretty good and again seems to follow the normal line well enough.

```
plot.ts(res, main = "Model (C) Residuals")
fitt <- lm(res ~ as.numeric(1:length(res))); abline(fitt, col="red")
abline(h=mean(res), col="blue")
```



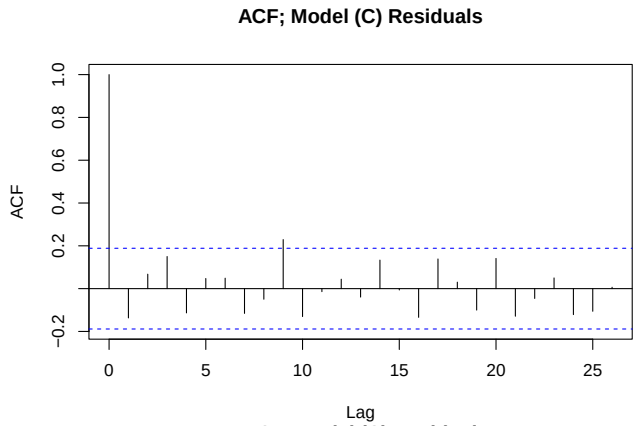
Looking at the plot of the residuals, I also see there is a tiny downward trend, however it is again practically invisible. I also see no seasonality or visible change of variance.

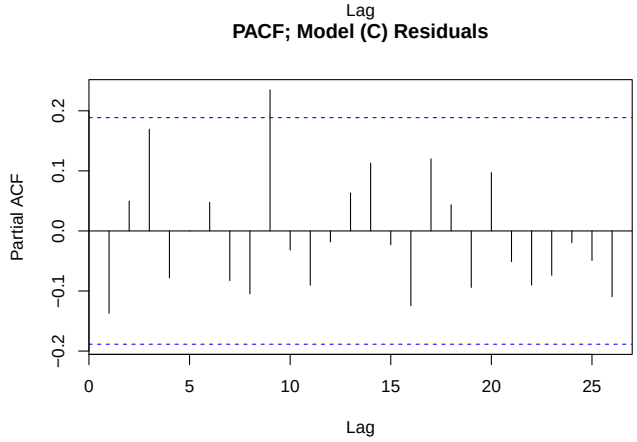
```
qqnorm(res,main= "Normal Q-Q Plot for Model C")
qqline(res,col="blue")
```



The Q-Q plot looks only okay. There are more pronounced deviations, though again, they are mostly at the tails.

```
acf(res, lag.max=26, main = "ACF; Model (C) Residuals")
pacf(res, lag.max=26, main = "PACF; Model (C) Residuals")
```





Again, the ACF of the residuals looks good. This model's ACF also falls out of the confidence interval at lag 9, however, again only just barely. So, the ACF still looks good since it's only 1 out of 26 lags that falls outside the confidence interval. And again, the PACF of the residuals look pretty good. Like the last model, it falla out of the confidence interval at lag 9, however, since it's only 1 out of 26 lags that falls outside, it still looks okay. Thus, the ACF and PACF look good enough to be considered white noise.

Running the Shapiro-Wilk test of normality, I get a p-value less than 0.05 and significantly lower than model (B). Therefore, model (C) also does not pass the test and rejects the null hypothesis of normality.

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

```
##
##  Shapiro-Wilk normality test
##
## data:  res
## W = 0.97175, p-value = 0.02117
```

I again run Box-Pierce and Ljung-Box tests and the McLeod Li test. The lag is still equal to 10, however, this time fitdf is set to 2 for Box-Pierce and Ljung-Box tests since this model only has two coefficients. All but the McLeod Li tests give p-values smaller than 0.05, which indicates that our model does not fit the data.

```
Box.test(res, lag = 10, type = c("Box-Pierce"), fitdf = 2)
```

```
##
##  Box-Pierce test
##
## data:  res
## X-squared = 16.069, df = 8, p-value = 0.0414
```

```r
Box.test(res, lag = 10, type = c("Ljung-Box"), fitdf = 2)
```

```
##
##  Box-Ljung test
##
## data:  res
## X-squared = 17.388, df = 8, p-value = 0.02631
```

```r
Box.test(res^2, lag = 10, type = c("Ljung-Box"), fitdf = 0)
```

```
##
##  Box-Ljung test
##
## data:  res^2
## X-squared = 8.8144, df = 10, p-value = 0.5498
```

I also again check the residuals using the yule-walker method where I ask it to use AICc to automatically select an order. It fitted the residuals to AR(1), which says the residuals do not resemble white noise.

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

```
##
## Call:
## ar(x = res, aic = TRUE, order.max = NULL, method = c("yule-walker"))
##
## Coefficients:
##      1
## -0.137
##
## Order selected 1  sigma^2 estimated as  4.298e+09
```

Therefore, Model (C) fails all tests but the McLeod Li tests.

Diagnostic Conclusion: Since Model (C) only passed one test whilst model (B) only failed one, I will move forward with model (B) as my final model for forecasting. So, the final model for U_t follows the SARIMA$(0,1,2)(0,1,1)_{12}$ model.

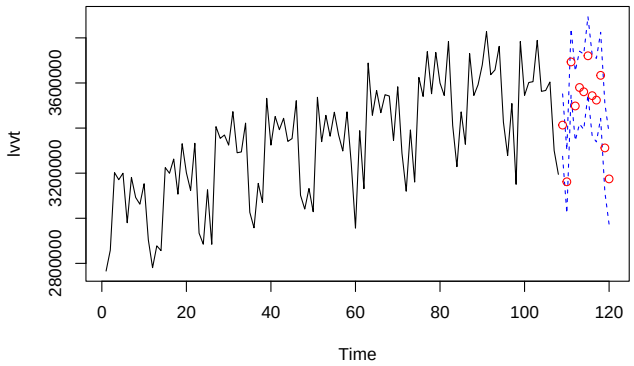
Model B: $\nabla_1\nabla_{12}U_t = (1 - 0.844_{(0.098)}B - 0.175_{(0.105)}B^2)(1 - 0.56_{(0.122)}B^{12})Z_t$; $\sigma_Z^2 = 4.78e+09$

Forecasting of Original Data

Using Model (B), I get the following forecast:

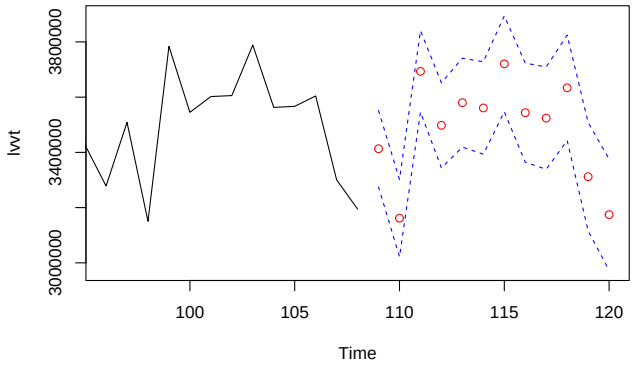
```r
#library(forecast)
# forecast next 12 obs of time series
fit.A <- m3
#forecast(fit.A) # prints forecasts with prediction bounds in a table

#To produce graph with 12 forecasts on transformed data:
pred <- predict(fit.A, n.ahead = 12)
#pred
U= pred$pred + 2*pred$se # upper bound of prediction interval
L= pred$pred - 2*pred$se # lower bound
ts.plot(lvvt, xlim=c(1,length(lvvt)+12), ylim = c(min(lvvt),max(U)))
lines(U, col="blue", lty="dashed")
lines(L, col="blue", lty="dashed")
points((length(lvvt)+1):(length(lvvt)+12), pred$pred, col="red")
```



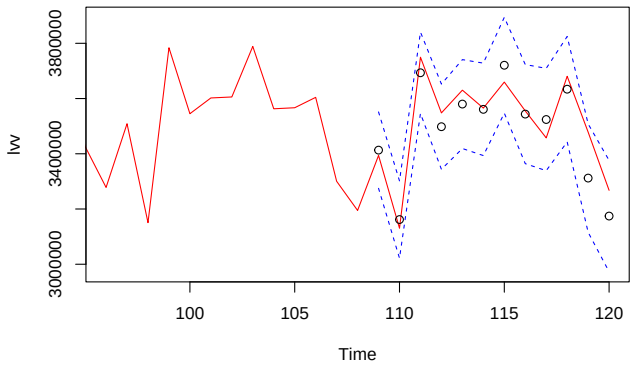
The blue lines are the confidence intervals and the red dots are the forecasted values. Here is a closer look at the forecast:

```
# to plot the last 10 values plus forecast:
ts.plot(lvvt, xlim=c(length(lvvt)-12,length(lvvt)+12), ylim = c(min(L),max(U)))
points((length(lvvt)+1):(length(lvvt)+12),pred$pred, col="red")
lines((length(lvvt)+1):(length(lvvt)+12),U, lty=2, col="blue")
lines((length(lvvt)+1):(length(lvvt)+12),L, lty=2, col="blue")
```



Lastly, I include the test set to see how well my model was able to forecast Last Vegas visitor volume:

```
# to plot zoomed forecasts and true values
ts.plot(lvv, xlim = c(length(lvvt)-12,length(lvvt)+12), ylim = c(min(L),max(U)), col="red")
lines(U, col="blue", lty="dashed")
lines(L, col="blue", lty="dashed")
points((length(lvvt)+1):(length(lvvt)+12), pred$pred, col="black")
```



As you can see above, the red line falls within the confidence intervals and even lines up with some of the specific forecasted values. Overall, I am happy with my results and would determine my forecast as a success.