Forecasting NBA 3 Point Shot Percentage

```r
# Load packages:

library(MASS)
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```r
library(tseries)
library(astsa)
```

```
##
## Attaching package: 'astsa'

## The following object is masked from 'package:forecast':
##
##     gas
```

```r
library(dse)
```

```
## Loading required package: tfplot

## Loading required package: tframe

##
## Attaching package: 'dse'

## The following objects are masked from 'package:forecast':
##
##     forecast, is.forecast

## The following objects are masked from 'package:stats':
##
##     acf, simulate
```

```r
library(knitr)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dse':
##
##     combine
```

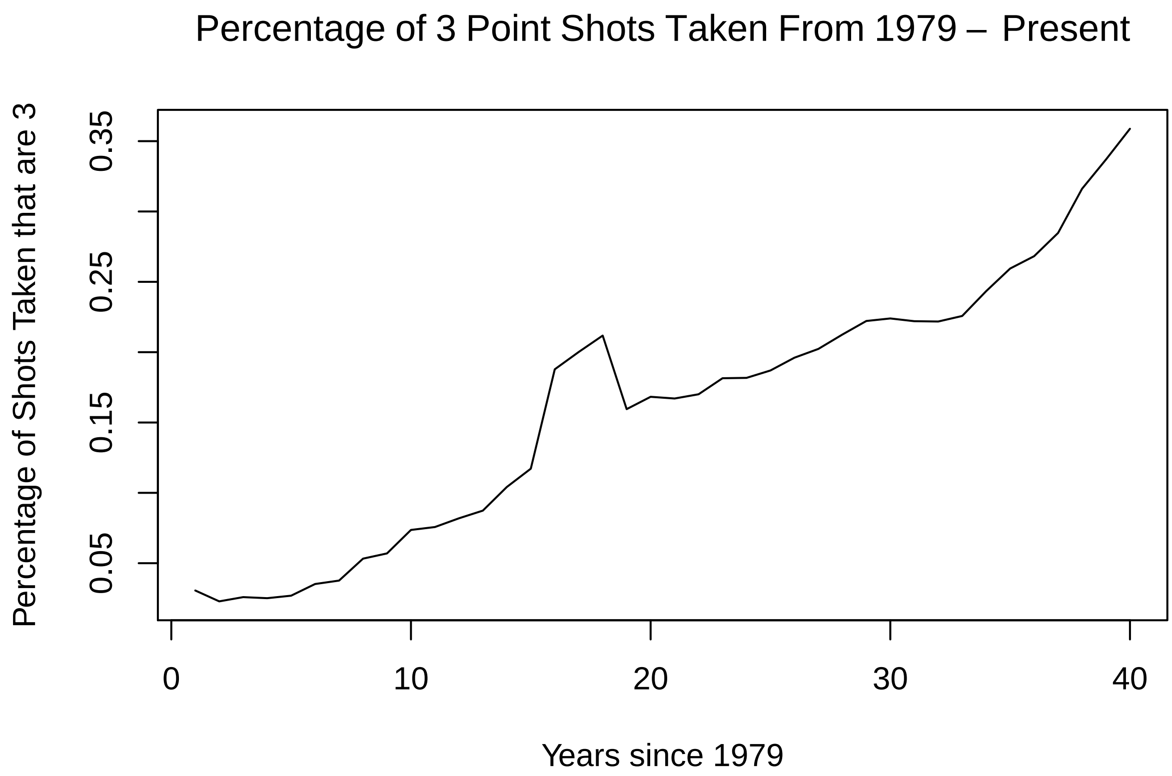
```r
library(grid)
library(tidyverse)
```

```
## -- Attaching packages -----------------------------------------------------------------------

## v ggplot2 3.3.1     v purrr   0.3.4
## v tibble  3.0.1     v dplyr   1.0.0
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts --------------------------------------------------------------------------------
## x dplyr::combine() masks gridExtra::combine(), dse::combine()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## x dplyr::select()  masks MASS::select()
## x purrr::splice()  masks tframe::splice()
```

```r
library(MAPA)
```

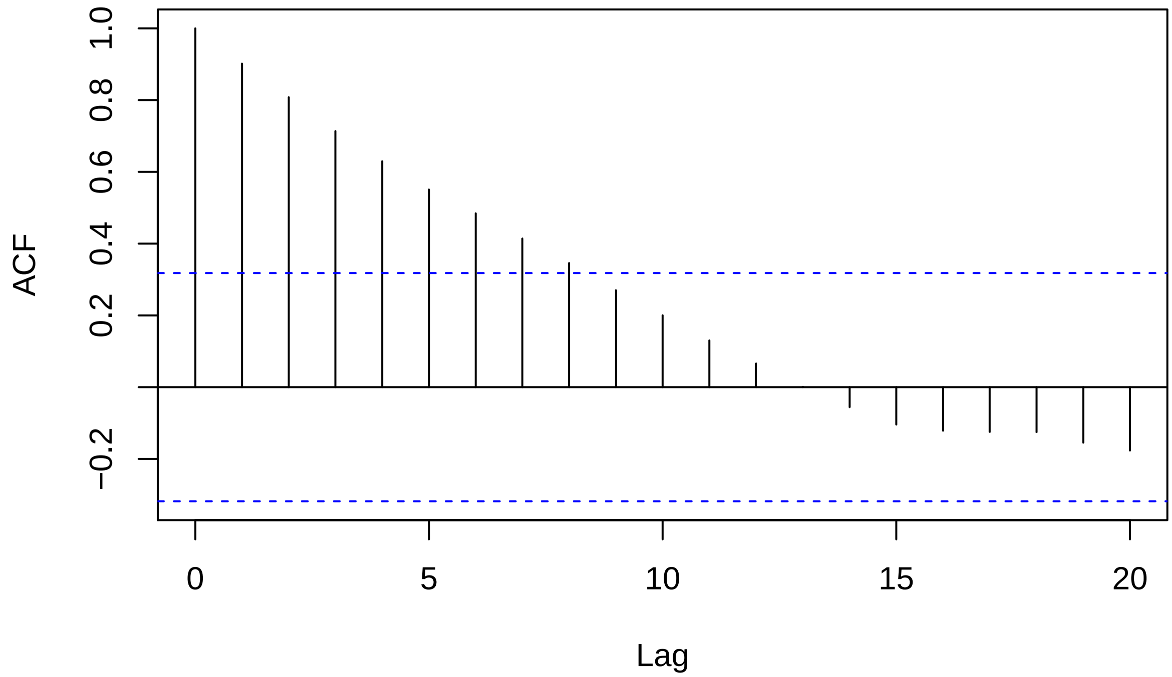
```
## Loading required package: parallel

## Loading required package: RColorBrewer

## Loading required package: smooth

## Loading required package: greybox

## Package "greybox", v0.6.0 loaded.

##
## Attaching package: 'greybox'

## The following object is masked from 'package:tidyr':
##
##     spread

## The following objects are masked from 'package:dse':
##
##     forecast, polyprod

## This is package "smooth", v2.6.0

##
## Attaching package: 'smooth'

## The following object is masked from 'package:dse':
##
##     forecast
```

```r
lbjmvp<-read.csv(file = 'sportsref_download.csv',
                 header = TRUE, sep = ',', skip = 1)

#Remove last few entries for training.
trainset <- lbjmvp %>%
  select(Season, X3PA, FGA) %>%
  mutate(percentageof3=X3PA/FGA) %>%
  filter(Season!='2019-20',Season!='2018-19',Season!='2017-18') %>%
  filter(!is.na(X3PA)) %>%
  arrange(-row_number())
#length(trainset)

#Full data. Subtract 2019-2020 season, since it's not over yet (at time
#of project!)
testset <- lbjmvp %>%
  select(Season, X3PA, FGA) %>%
  mutate(percentageof3=X3PA/FGA) %>%
  filter(Season!='2019-20') %>%
  filter(!is.na(X3PA)) %>%
  arrange(-row_number())

trainset<-trainset[,4]
#trainset

testset<-testset[,4]
#testset

ts.plot(testset, ylab = "Percentage of Shots Taken that are 3",
        xlab = "Years since 1979")
title(expression(Percentage~of~3~Point~Shots~Taken~From~1979~-~Present))
```

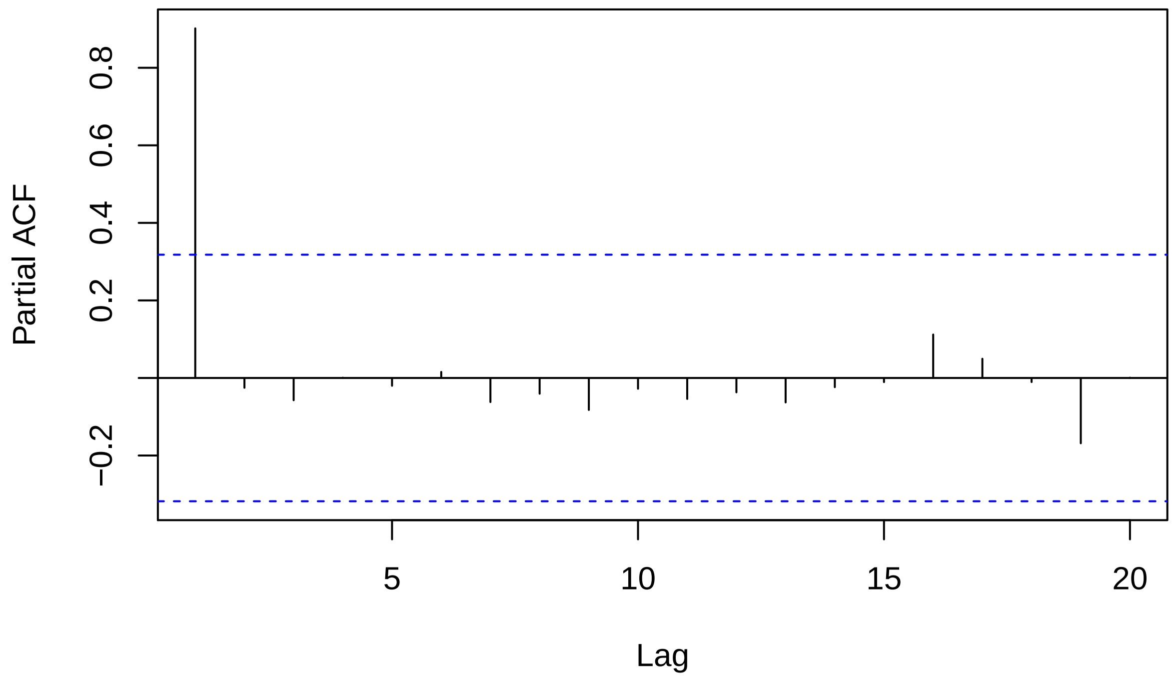



```
#Data does not appear to display seasonality.
```

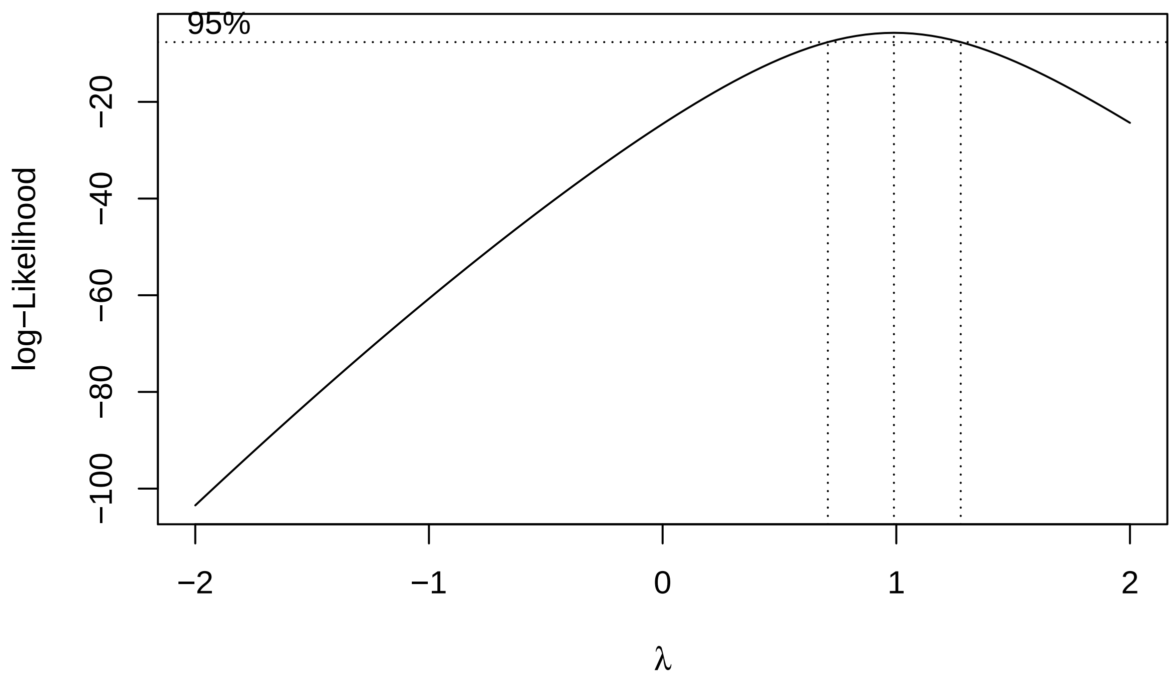
```
trainset.ts <- ts(trainset, frequency = 1)
testset.ts <- ts(testset, frequency = 1)
#Initial look at ACF and PACF plots.
acf(trainset.ts, lag.max =20,main = "" )
```




```
pacf(trainset.ts, lag.max=20, main = "")
```



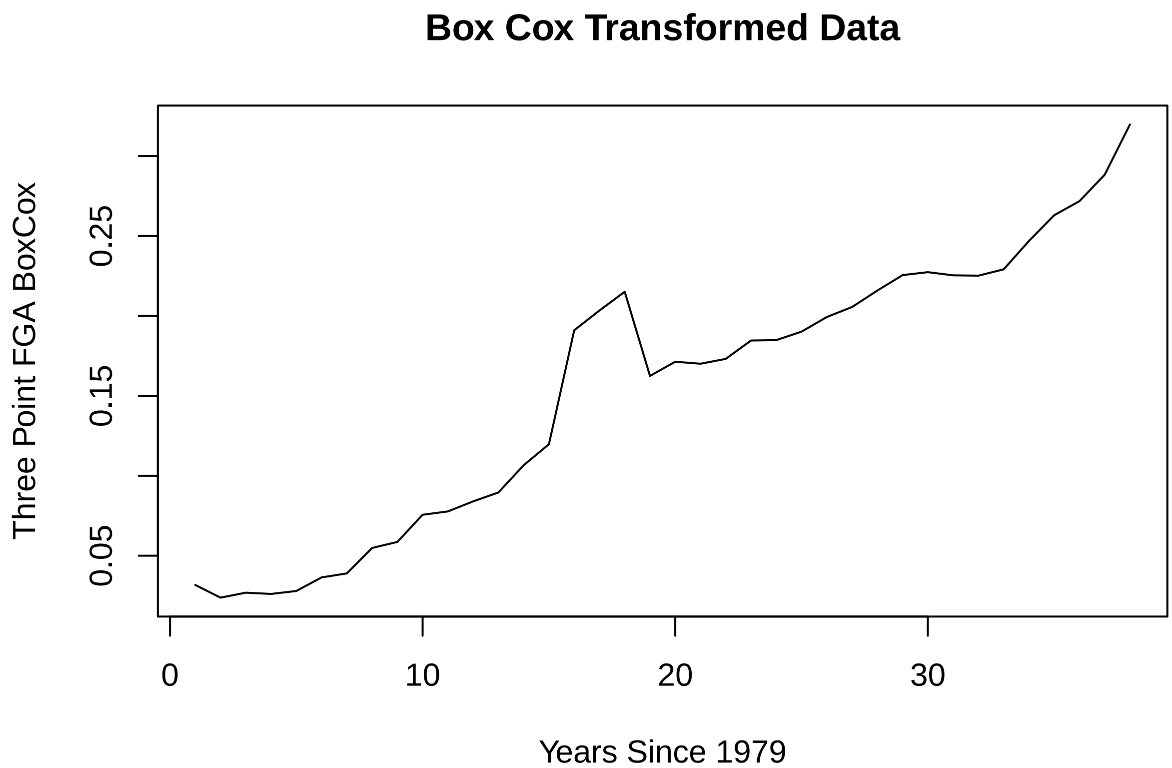

```
#Perhaps variance needs stabilizing?
bctrainset <- boxcox(trainset.ts~as.numeric(1:length(trainset)))
```




```
lambda1 <- bctrainset$x[which.max(bctrainset$y)]
lambda1
```

```
## [1] 0.989899
```

```
#Lambda is 0.989899
trainset.tr <- trainset.ts^(0.989899)
ts.plot(trainset.tr, ylab = "Three Point FGA BoxCox",
        xlab = "Years Since 1979",
        main = "Box Cox Transformed Data")
```




```
var(trainset.ts)
```

```
## [1] 0.007339486
```

```
#0.007339486
var(trainset.tr)
```

```
## [1] 0.007496046
```

```
#0.007496046
#Slight increase in variance, but we will allow this for now.
```

```
#There appears to be no seasonality, so we will difference to remove trend.

trainsetdiff1 <- diff(trainset.tr, lag =1) #Difference once and observe.
var(trainsetdiff1)
```

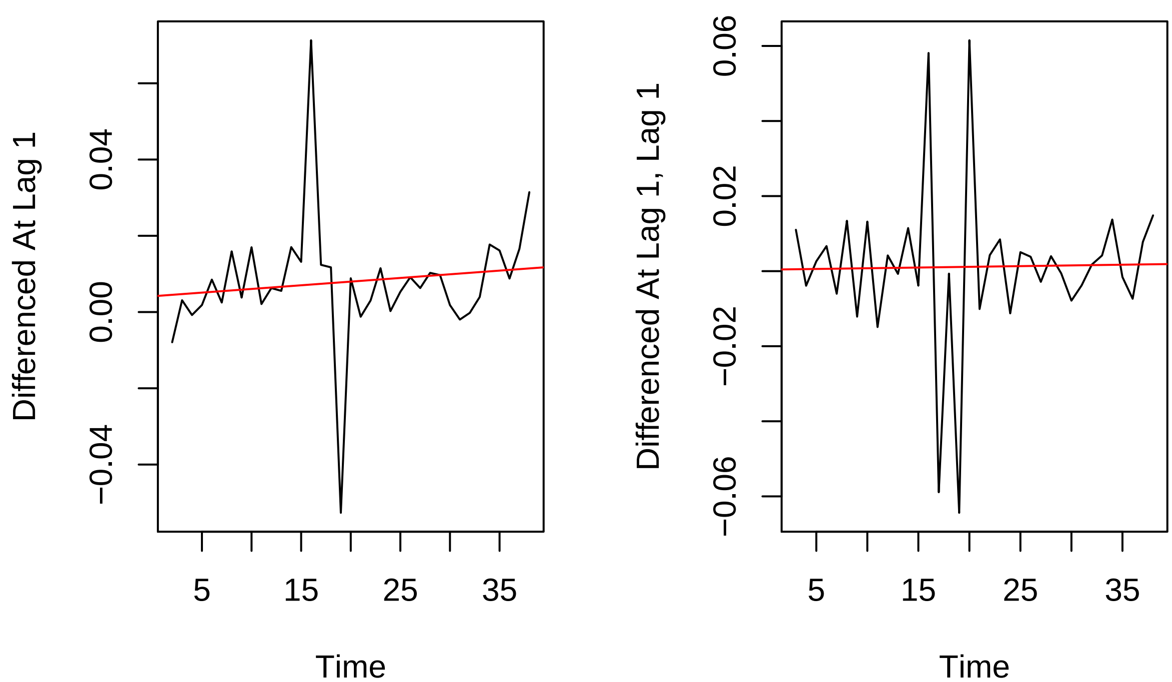
```
## [1] 0.0002683378
```

```
#0.0002683378
#ts.plot(trainsetdiff1, ylab = "Differenced At Lag 1")
#We want it to resemble white noise.
#abline(lm(trainsetdiff1~as.numeric(1:length(trainsetdiff1))),
```

```r
#       col ="red")

trainsetdiff1diff1 <- diff(trainsetdiff1, lag =1)
var(trainsetdiff1diff1)
```

```
## [1] 0.0004814831
```

```r
#0.0004814831
#ts.plot(trainsetdiff1diff1, ylab = "Differenced At Lag 1")
#abline(lm(trainsetdiff1diff1~as.numeric(1:length(trainsetdiff1diff1))),
#       col ="red")

par(mfrow=c(1,2))
ts.plot(trainsetdiff1, ylab = "Differenced At Lag 1")
abline(lm(trainsetdiff1~as.numeric(1:length(trainsetdiff1))),
       col ="red")
ts.plot(trainsetdiff1diff1, ylab = "Differenced At Lag 1, Lag 1")
abline(lm(trainsetdiff1diff1~as.numeric(1:length(trainsetdiff1diff1))),
       col ="red")
```



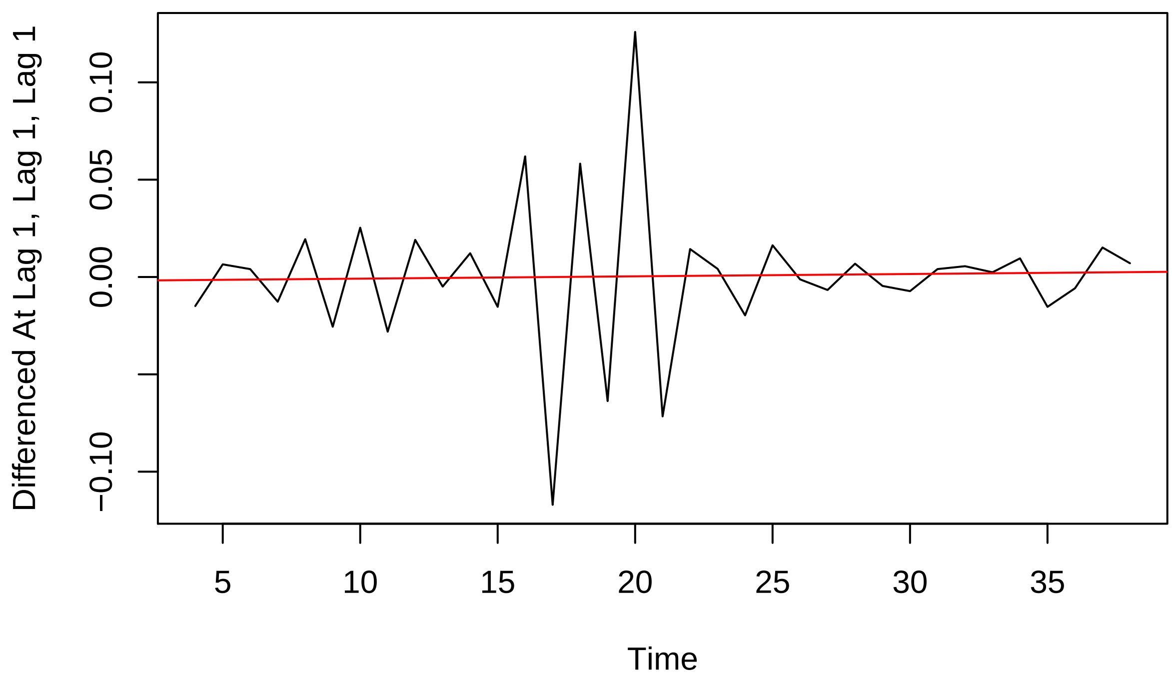

```r
#Differencing twice may work!
```

```r
#Perhaps more differencing may be useful?
#In general, it's best to not overdifference...
```

```
#but this may provide useful insight!

trainsetdiff1diff1diff1 <- diff(trainsetdiff1diff1, lag =1)
var(trainsetdiff1diff1diff1)
```

```
## [1] 0.001511063
```

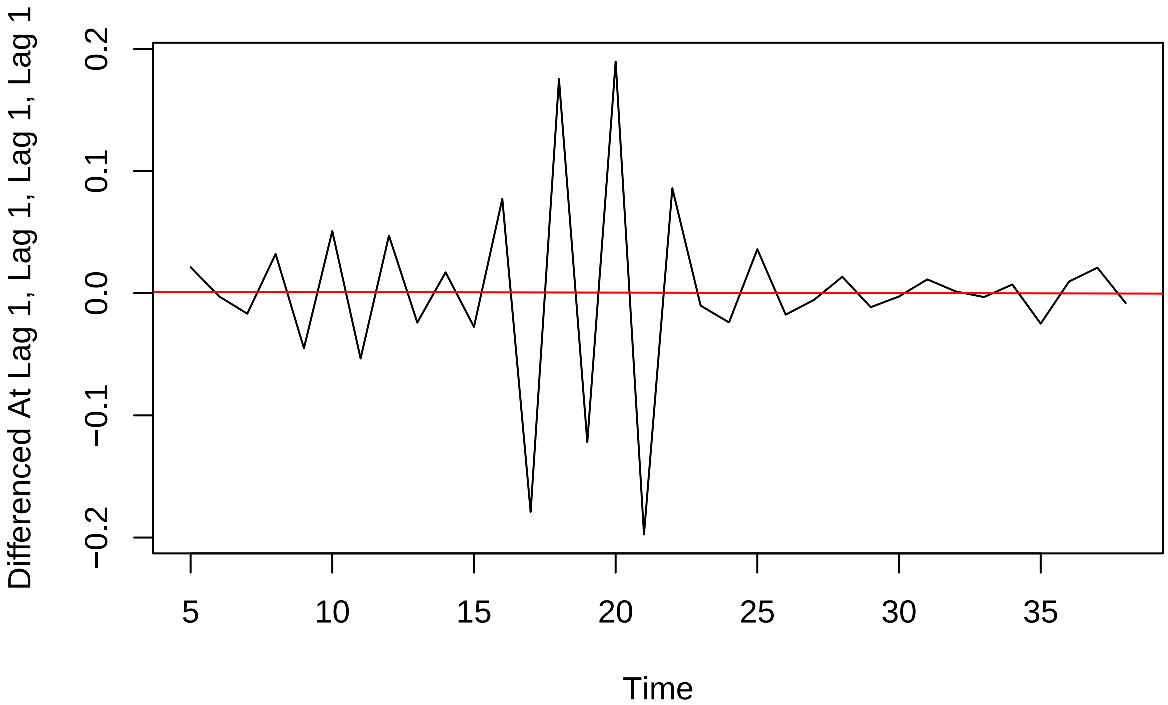
```
#0.001511063  Too high?
ts.plot(trainsetdiff1diff1diff1,
        ylab = "Differenced At Lag 1, Lag 1, Lag 1")
abline(lm(trainsetdiff1diff1diff1~
            as.numeric(1:length(trainsetdiff1diff1diff1))),
       col ="red")
```




```
trainsetdiff1diff1diff1diff1 <- diff(trainsetdiff1diff1diff1, lag =1)
var(trainsetdiff1diff1diff1diff1)
```

```
## [1] 0.005547701
```

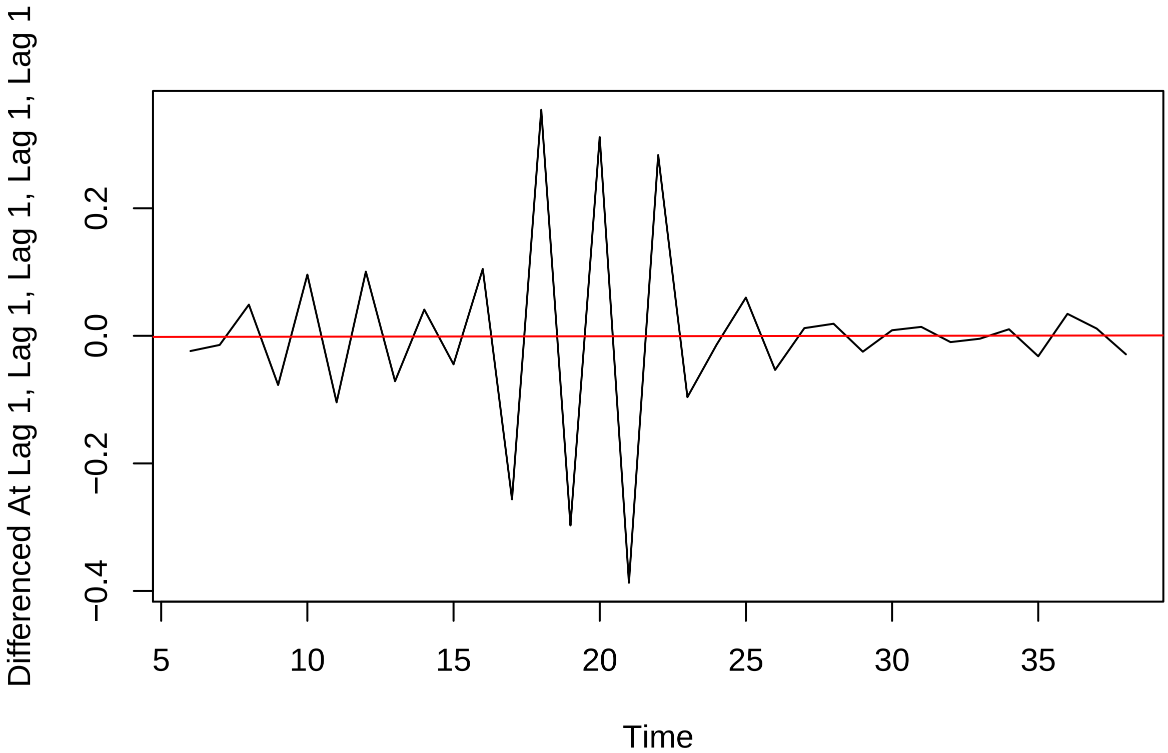
```
#0.005547701  Too high?
ts.plot(trainsetdiff1diff1diff1diff1,
        ylab = "Differenced At Lag 1, Lag 1, Lag 1, Lag 1")
abline(lm(trainsetdiff1diff1diff1diff1~
            as.numeric(1:length(trainsetdiff1diff1diff1diff1))),
       col ="red")
```




```r
trainsetdiff1diff1diff1diff1diff1 <- diff(trainsetdiff1diff1diff1diff1, lag =1)
var(trainsetdiff1diff1diff1diff1diff1)
```

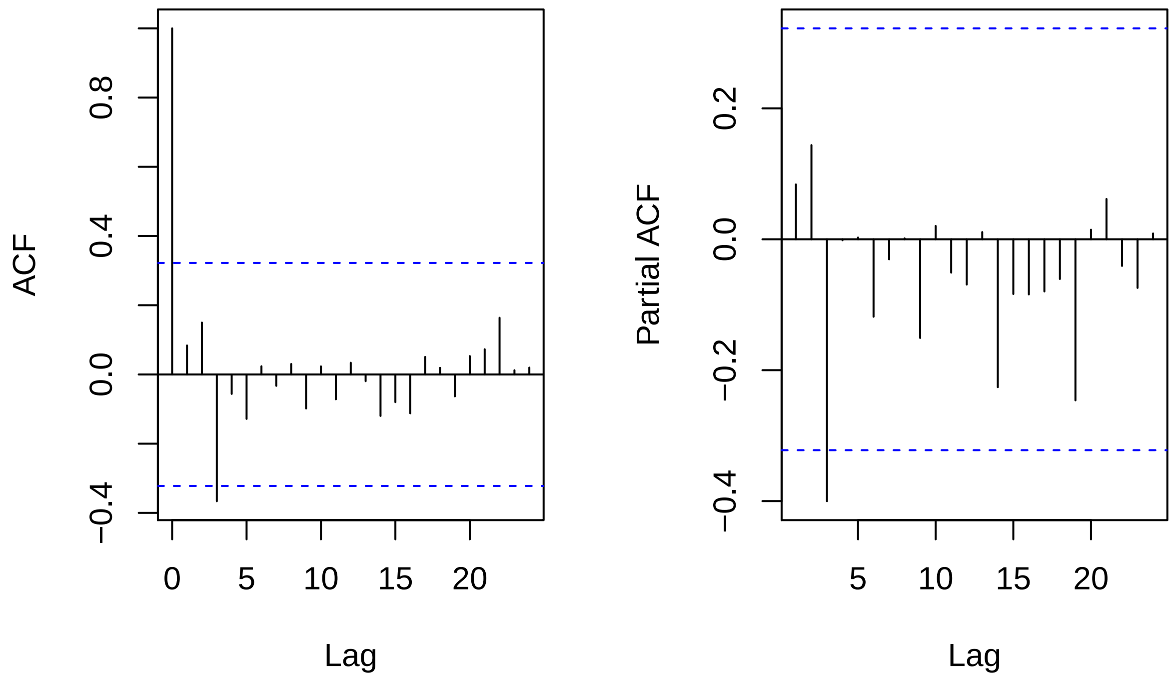
```
## [1] 0.02145416
```

```r
#0.02145416  Too high!
ts.plot(trainsetdiff1diff1diff1diff1diff1,
        ylab = "Differenced At Lag 1, Lag 1, Lag 1, Lag 1, Lag 1")
abline(lm(trainsetdiff1diff1diff1diff1diff1~
            as.numeric(1:length(trainsetdiff1diff1diff1diff1diff1))),
       col ="red")
```

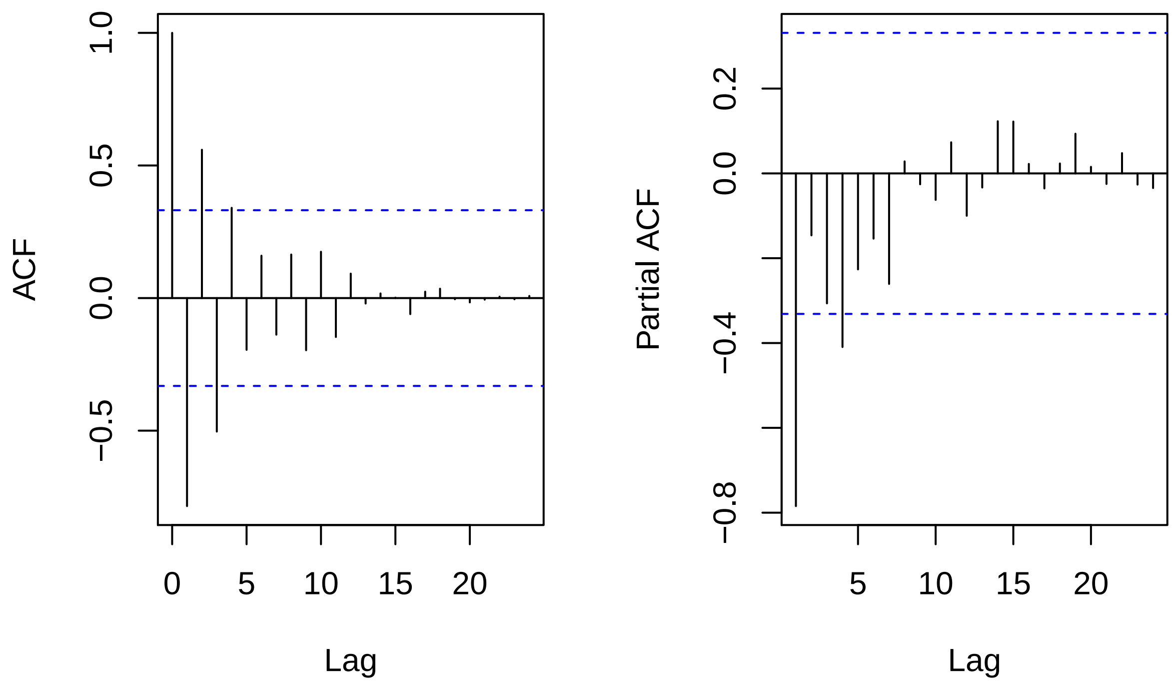



```
#The data may be overdifferenced, but it clearly resembles white noise!
#More analysis is needed.
```

```
# Differenced at lag 1 PACF AND ACF
par(mfrow=c(1,2))
acf(trainsetdiff1, lag.max = 24, main = "")
pacf(trainsetdiff1, lag.max = 24, main = "")
```




```r
# DIFFERENCED at lag 1 and lag 1 and lag 1 PACF AND ACF
par(mfrow=c(1,2))
acf(trainsetdiff1diff1diff1, lag.max=24, main = "")
pacf(trainsetdiff1diff1diff1, lag.max=24, main = "")
```




```
# S = 0, no seasonality.
# D = 0, d = 3, differenced 3x to remove trend
# Examine at lag = 1,2,3,4,5,...
# ACF plot tails off(q = 0)
# and PACF cuts off after lag 4 (p = 4) ----> ARIMA(4,3,0)?
# Or, perhaps ACF cuts off at lag 4 (q = 4)
# and PACF tails off (p = 0) -> ARIMA(0,3,4)?
# More analysis is needed.
# We will consider multiple models and choose the best one.
```

```
#We want the lowest possible AICc values.

mod <- Arima(trainset.tr, order = c(4,3,0)) #AICc ARIMA(4,3,0) -160.6
mod
```

```
## Series: trainset.tr
## ARIMA(4,3,0)
##
## Coefficients:
##           ar1      ar2     ar3      ar4
##       -1.0734  -0.6138  -0.692  -0.4005
## s.e.   0.1508   0.2064   0.199   0.1456
##
## sigma^2 estimated as 0.0004503:  log likelihood=86.32
## AIC=-162.64   AICc=-160.57   BIC=-154.86
```

```r
mod2 <- Arima(trainset.tr, order = c(0,3,4)) #AICc ARIMA(0,3,4) -169.5
mod2
```

```
## Series: trainset.tr
## ARIMA(0,3,4)
##
## Coefficients:
##           ma1     ma2      ma3     ma4
##       -1.4818  0.6229  -0.5113  0.4982
## s.e.   0.2273  0.2872   0.2288  0.1685
##
## sigma^2 estimated as 0.0003048:  log likelihood=90.77
## AIC=-171.54   AICc=-169.47   BIC=-163.76
```

```r
#Slight adjust to consider more models.

mod3 <- Arima(trainset.tr, order = c(4,2,0)) #AICc ARIMA(4,2,0) -182.1
mod3
```

```
## Series: trainset.tr
## ARIMA(4,2,0)
##
## Coefficients:
##           ar1      ar2      ar3      ar4
##       -0.5714  -0.1501  -0.4958  -0.2705
## s.e.   0.1602   0.1664   0.1600   0.1545
##
## sigma^2 estimated as 0.0002915:  log likelihood=97.03
## AIC=-184.05   AICc=-182.05   BIC=-176.14
```

```r
mod4 <- Arima(trainset.tr, order = c(0,2,4)) #AICc ARIMA(0,2,4) -186.2
mod4
```

```
## Series: trainset.tr
## ARIMA(0,2,4)
##
## Coefficients:
##           ma1     ma2      ma3     ma4
##       -0.7965  0.0086  -0.5853  0.3733
## s.e.   0.2193  0.2015   0.1677  0.2042
##
## sigma^2 estimated as 0.0002378:  log likelihood=99.11
## AIC=-188.21   AICc=-186.21   BIC=-180.3
```

```r
#We have multiple models to consider. But the auto.arima() function
#may give us an even better recommendation.
```

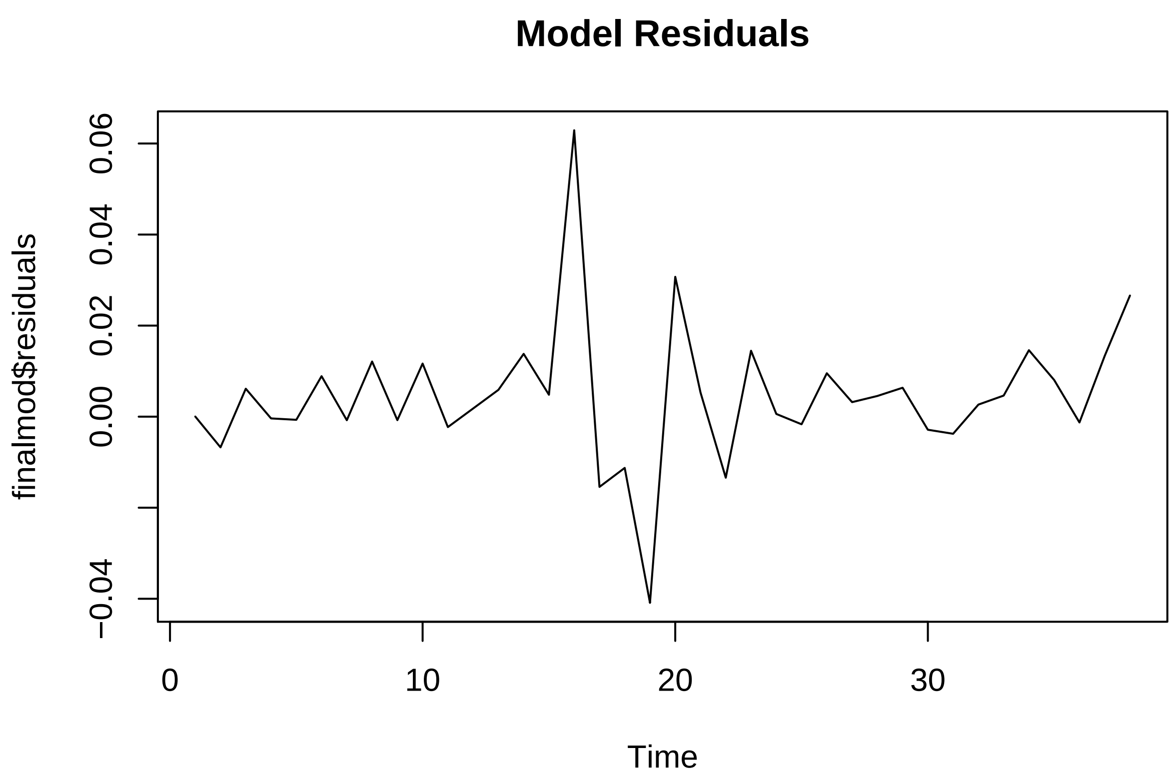
```r
modauto<-auto.arima(trainset.tr,seasonal = FALSE,
                    stepwise = FALSE,
                    approximation = FALSE,
                    allowdrift = FALSE)
modauto #AICc ARIMA(0,1,2) -193.8 Lowest one yet!
```

```
## Series: trainset.tr
## ARIMA(0,1,2)
##
## Coefficients:
##          ma1     ma2
##       0.4066  0.4651
## s.e.  0.1826  0.1579
##
## sigma^2 estimated as 0.0002698:  log likelihood=100.27
## AIC=-194.55   AICc=-193.82   BIC=-189.72
```

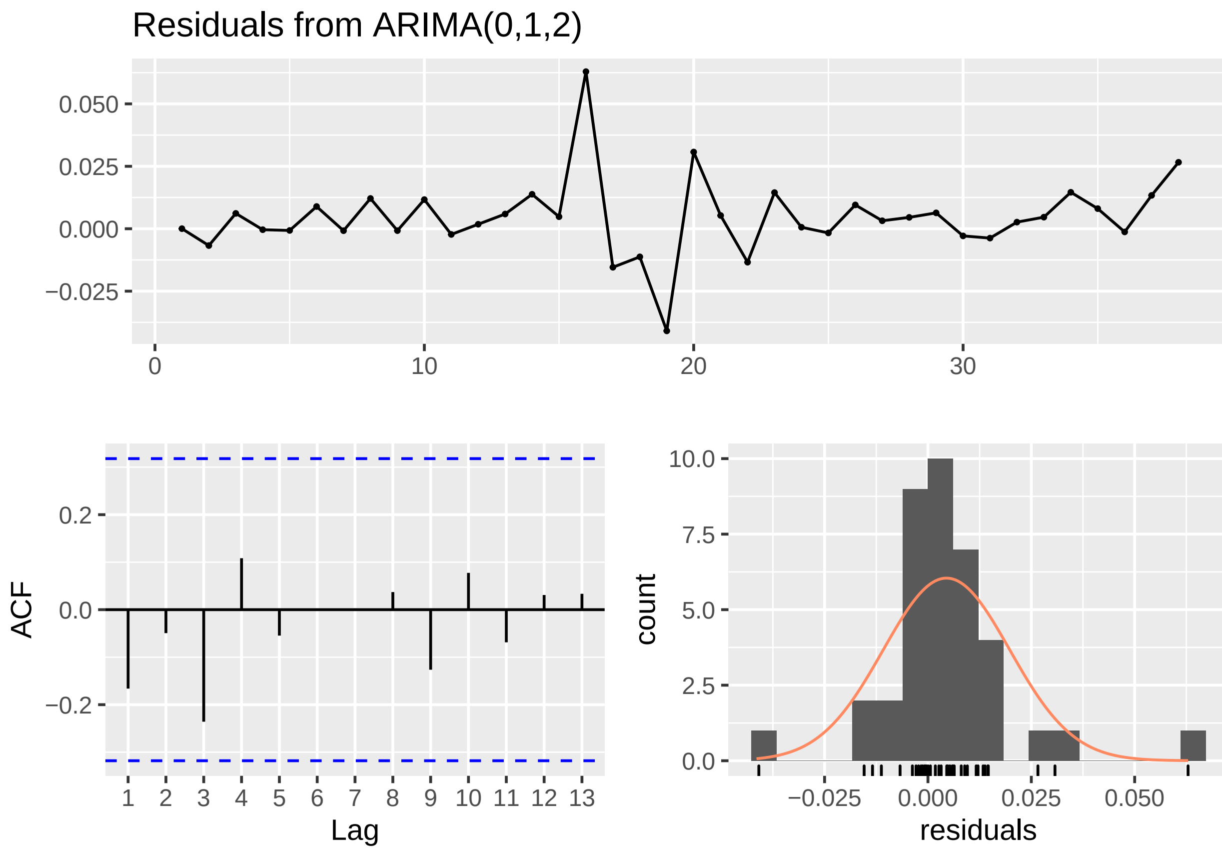
```
finalmod <- Arima(trainset.tr, order = c(0,1,2))
finalmod
```


```
## Series: trainset.tr
## ARIMA(0,1,2)
##
## Coefficients:
##          ma1     ma2
##       0.4066  0.4651
## s.e.  0.1826  0.1579
##
## sigma^2 estimated as 0.0002698:  log likelihood=100.27
## AIC=-194.55   AICc=-193.82   BIC=-189.72
```


```
ts.plot(finalmod$residuals, main = "Model Residuals")
```

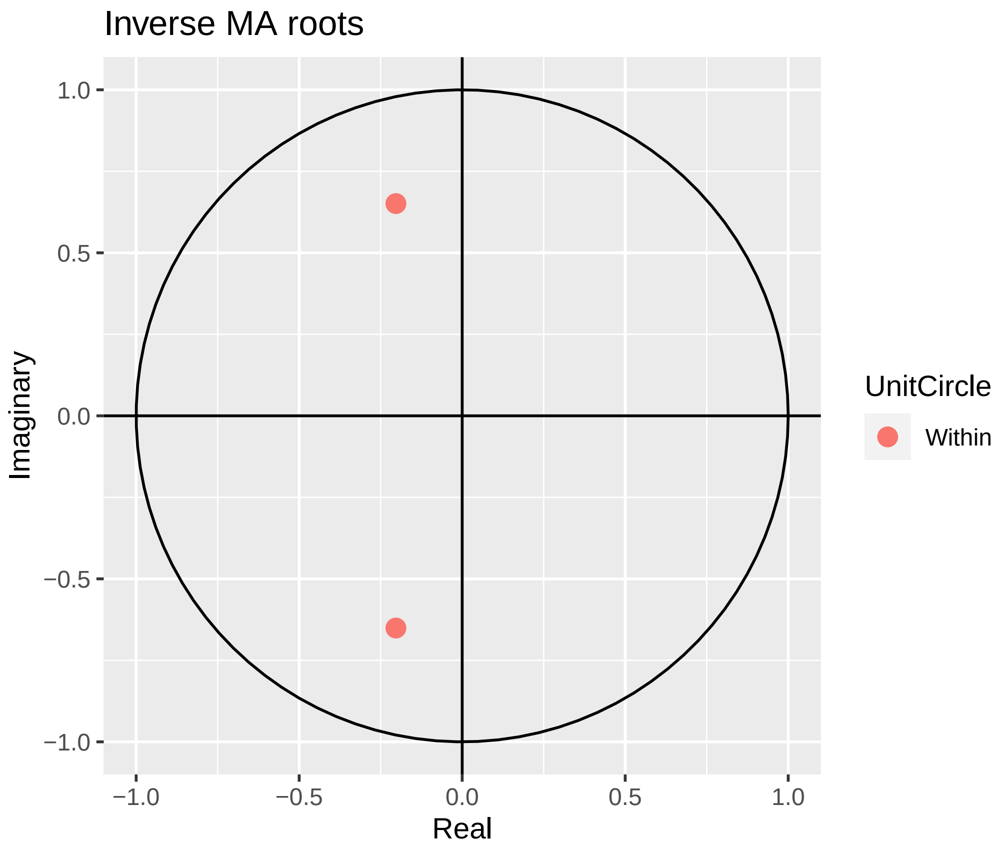


```r
#Resembles white noise.
checkresiduals(finalmod)
```

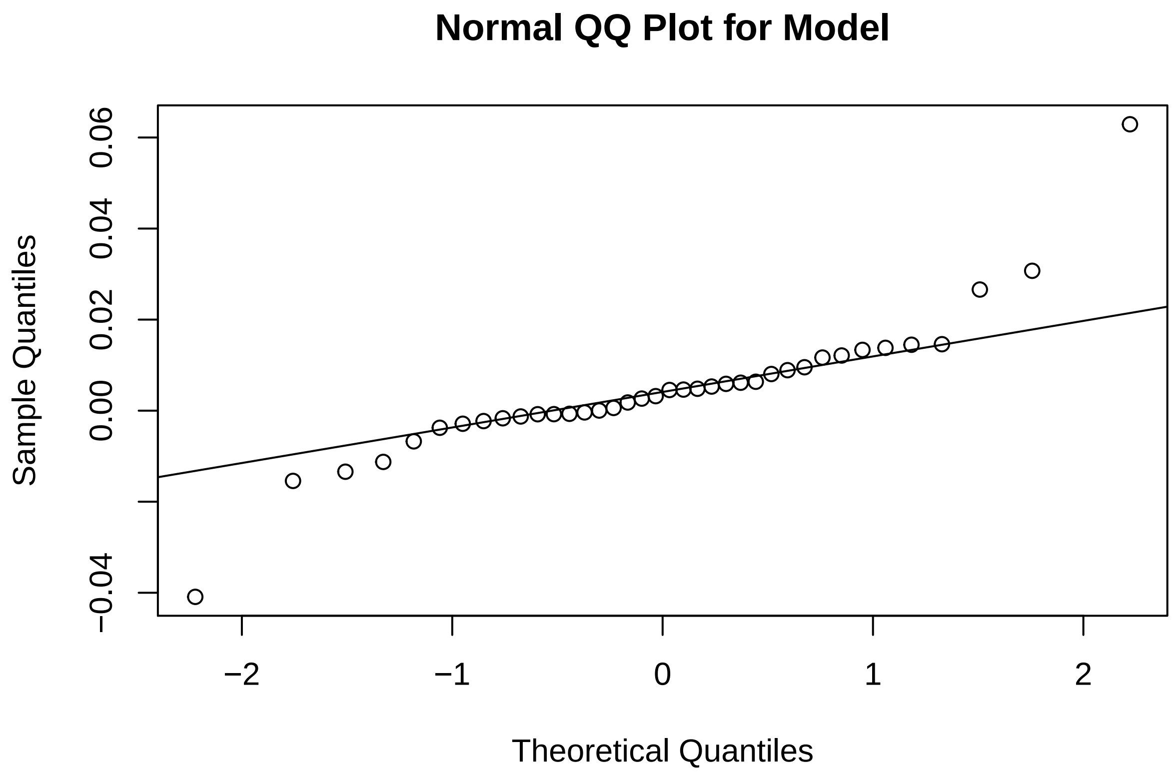



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,2)
## Q* = 4.3833, df = 6, p-value = 0.625
##
## Model df: 2.   Total lags used: 8
```

```r
#Residuals are normally distributed.
autoplot(finalmod)
```



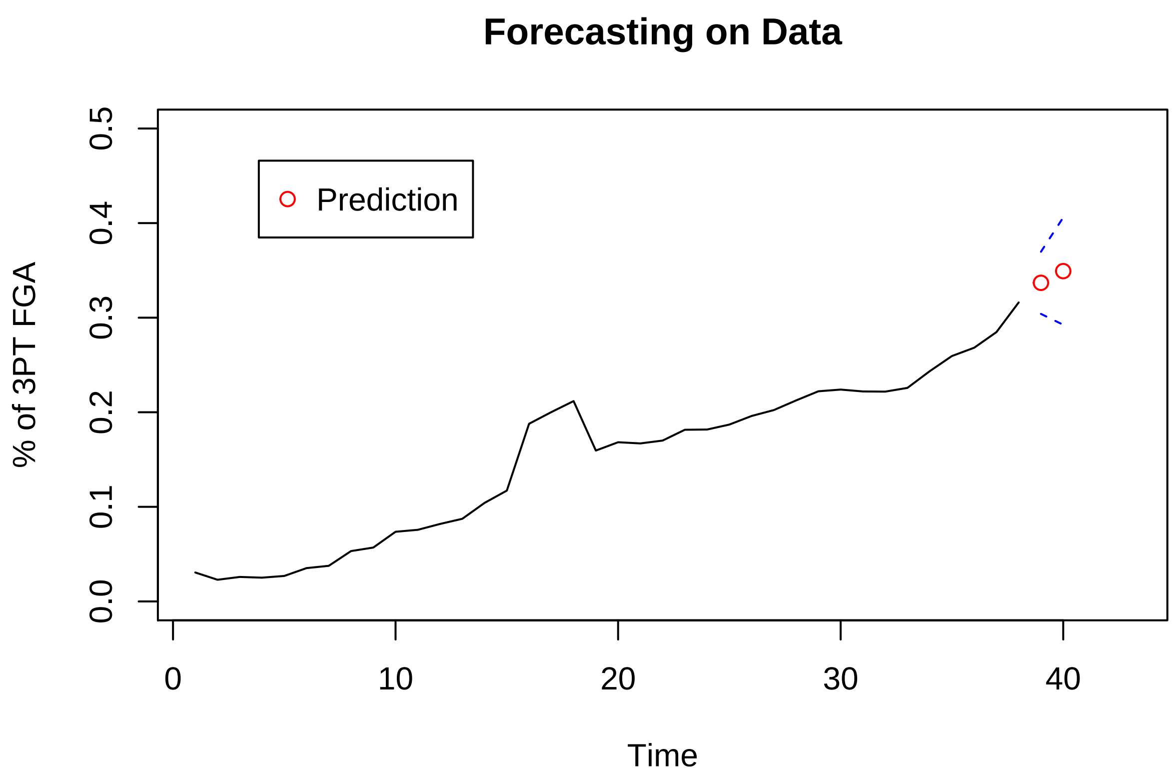

```
#Roots lie within unit circle. Implies stationarity and invertibility.
qqnorm(finalmod$residuals, main = "Normal QQ Plot for Model")
qqline(finalmod$residuals)
```




```
#Residuals are normally distributed.
```

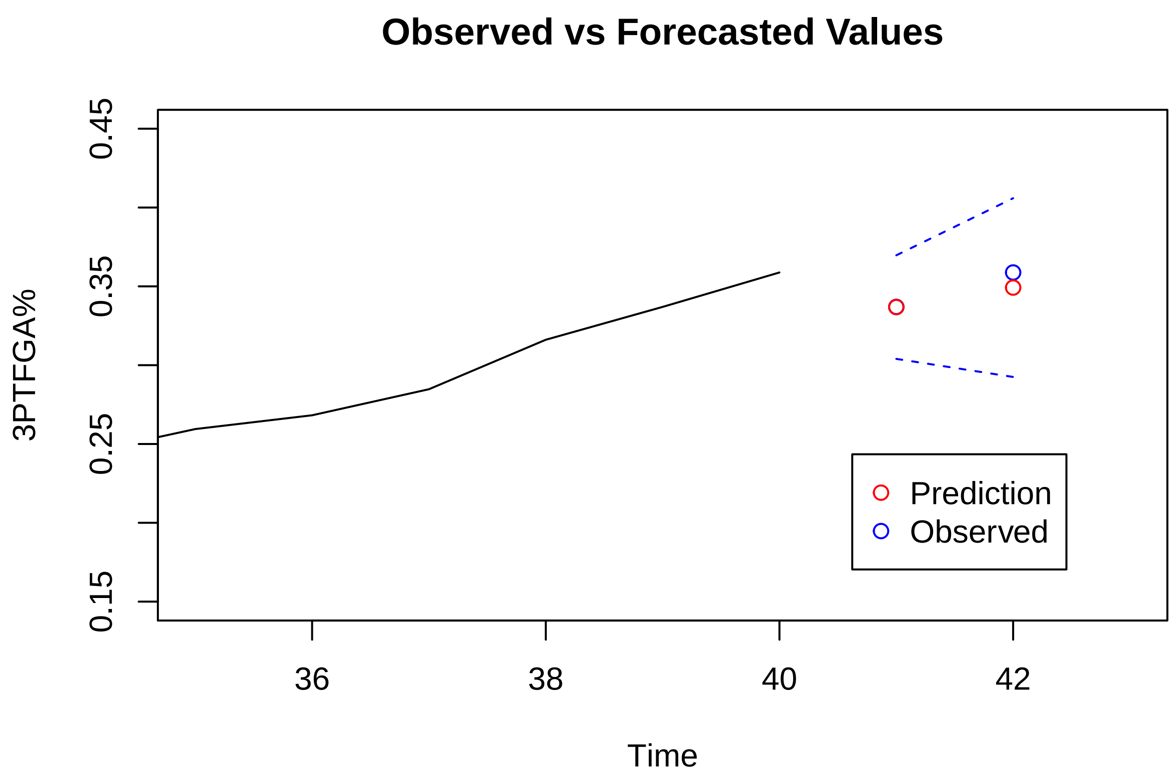
```
modelpredictions <- predict(finalmod, n.ahead=2)

upperbound <- modelpredictions$pred + 2*modelpredictions$se
lowerbound <- modelpredictions$pred - 2*modelpredictions$se

ts.plot(trainset,
        xlim=c(1, length(trainset)+5),
        ylim=c(0,0.5),
        main = "Forecasting on Data",
        ylab= "% of 3PT FGA")

lines(upperbound, col="blue", lty = "dashed")
lines(lowerbound, col="blue", lty = "dashed")

points((length(trainset)+1):(length(trainset)+2),
       modelpredictions$pred, col ="red")

# Legend:
legend("topleft",
  legend = c("Prediction"),
  col = c("red"),
  pch = 1,
  bty = "o",
  pt.cex = 1,
```

```
  cex = 1,
  text.col = "black",
  horiz = F ,
  inset = c(0.1, 0.1))
```




```
#Zoom in! Compare it to test set.

ts.plot(testset,
        xlim = c(length(testset)-5, length(testset)+3),
        ylim = c(0.15,0.45),
        main = "Observed vs Forecasted Values",
        ylab = "3PTFGA%")

# Points for observed data
points((length(testset)+1):(length(testset)+2),
       testset.ts[39:40], col ="blue")

# Points for forecasted data
points((length(testset)+1):(length(testset)+2),
       modelpredictions$pred, col ="red")

lines((length(testset)+1): (length(testset)+2),
      upperbound, lty=2, col = "blue")

lines((length(testset)+1): (length(testset)+2),
```

```r
      lowerbound, lty=2, col = "blue")

# Add a legend
legend("bottomright",
  legend = c("Prediction", "Observed"),
  col = c("red",
  "blue"),
  pch = 1,
  bty = "o",
  pt.cex = 1,
  cex = 1,
  text.col = "black",
  horiz = F ,
  inset = c(0.1, 0.1))
```

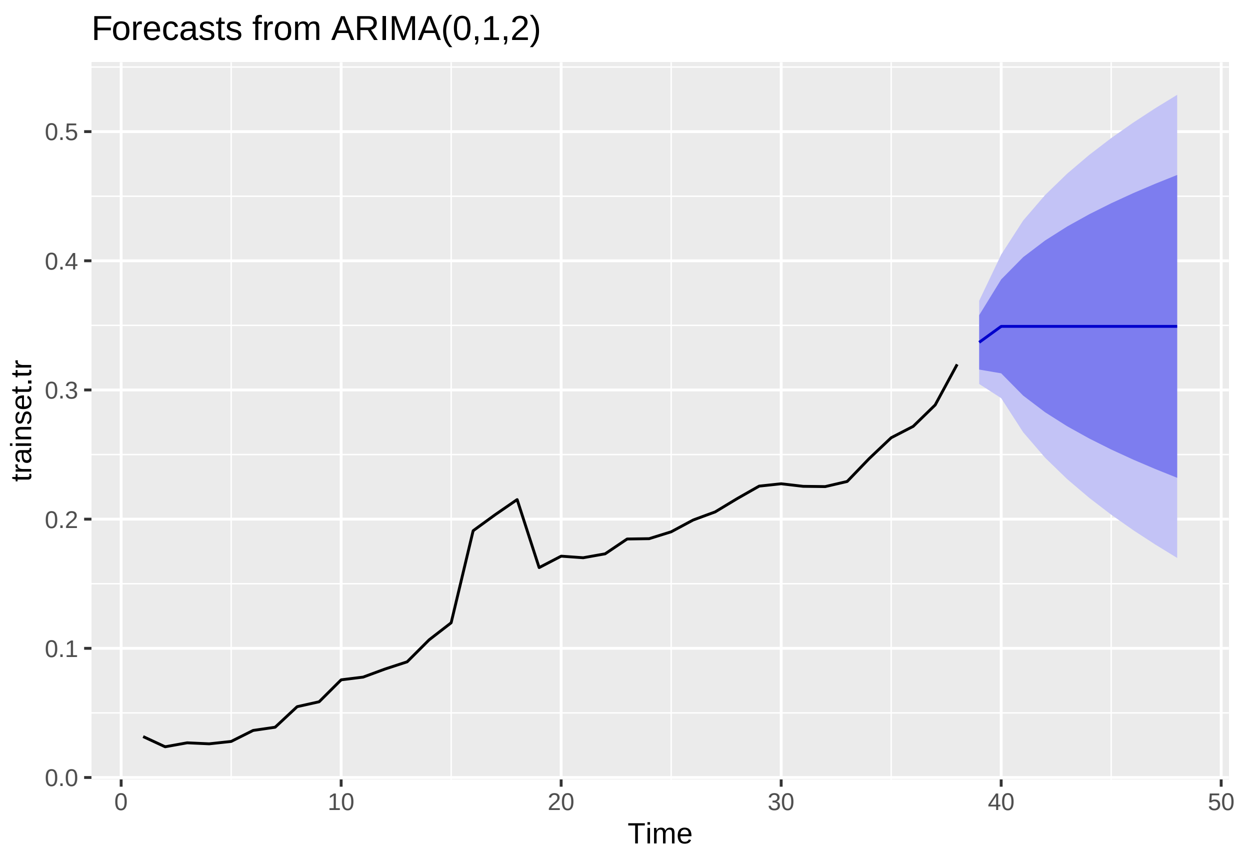



```r
predict(finalmod, n.ahead = 5) #Our ARIMA(0,1,2)
```

```
## $pred
## Time Series:
## Start = 39
## End = 43
## Frequency = 1
## [1] 0.3368486 0.3492225 0.3492225 0.3492225 0.3492225
##
## $se
```

```
## Time Series:
## Start = 39
## End = 43
## Frequency = 1
## [1] 0.01642527 0.02834745 0.04181777 0.05190255 0.06032432
```

```r
library(MAPA)
#install.packages('smooth')
library(smooth)
modelforecasts<-forecast(finalmod)
library(forecast)
autoplot(modelforecasts)
```




```r
#Our model suggests the data converges to a point! Namely, 35% 3PT FGA.
```