PSTAT174 Final Project 2

Week 9

Ryan Yee

5/29/2023

Contents

```r
#install.packages("devtools")
#devtools::install_github("FinYang/tsdl")
#install.packages("astsa")

library(tsdl)
library(astsa)
```

Source: U.S. Bureau of Economic Analysis

Release: Gross Domestic Product

Units: Billions of Chained 2012 Dollars, Not Seasonally Adjusted

Frequency: Quarterly

BEA Account Code: ND000334

U.S. Bureau of Economic Analysis, Real Gross Domestic Product [ND000334Q], retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/ND000334Q, May 30, 2023.

```r
gdp.csv = read.table("ND000334Q.csv", sep=",", header=FALSE, skip=1, nrows=85)
head(gdp.csv)
```

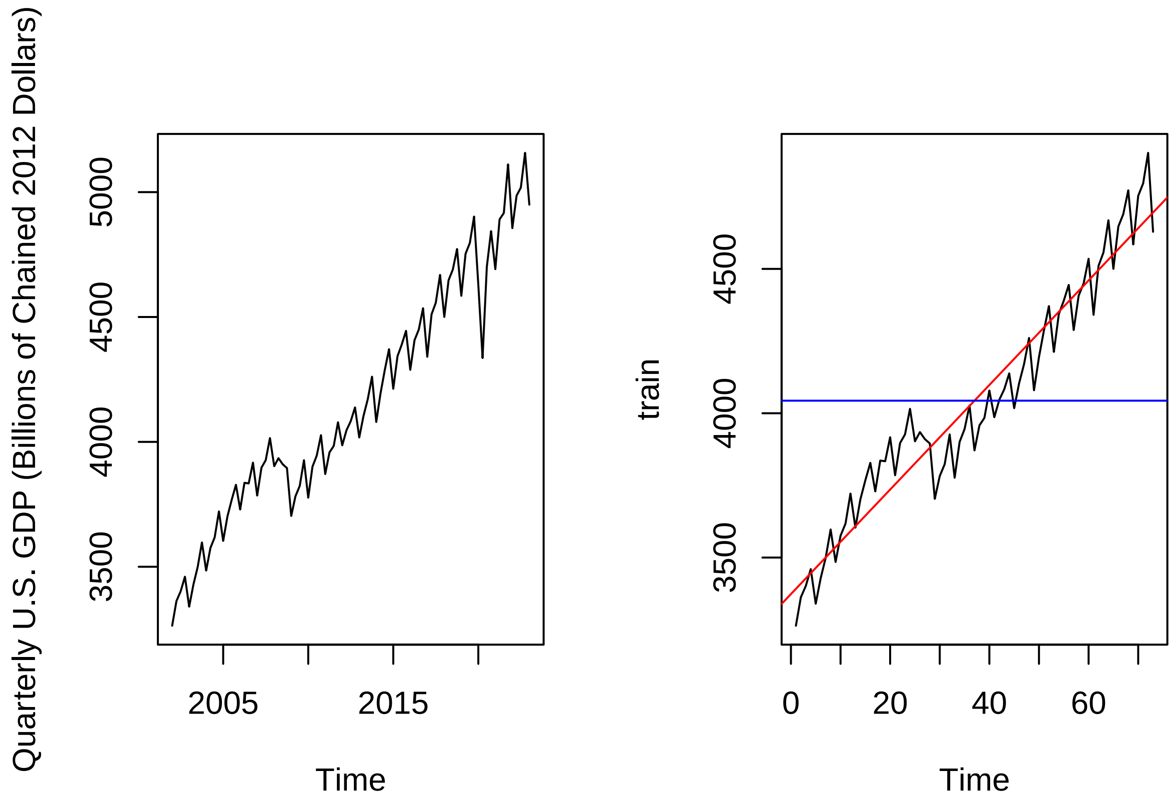
```
##           V1       V2
## 1 2002-01-01 3263.869
## 2 2002-04-01 3362.508
## 3 2002-07-01 3401.820
## 4 2002-10-01 3460.159
## 5 2003-01-01 3340.163
## 6 2003-04-01 3429.079
```

```r
tail(gdp.csv)
```

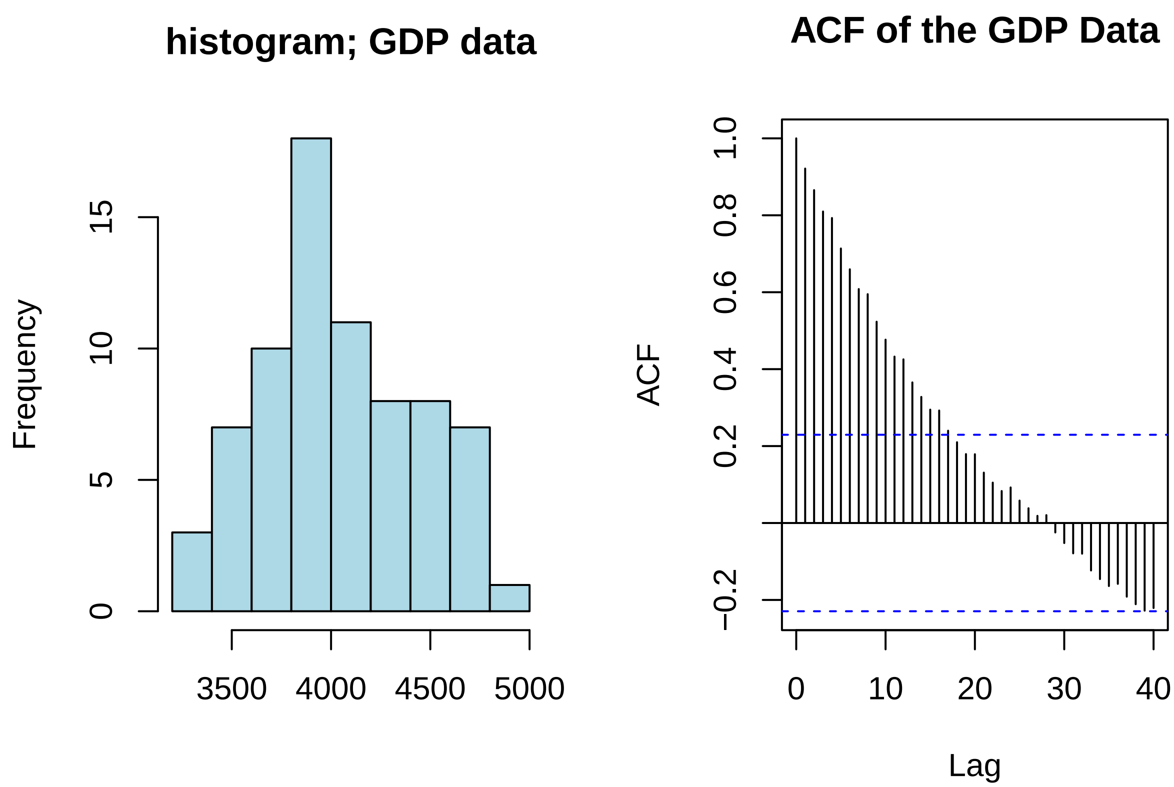
```
##            V1       V2
## 80 2021-10-01 5110.951
## 81 2022-01-01 4855.857
## 82 2022-04-01 4985.795
## 83 2022-07-01 5018.093
## 84 2022-10-01 5157.178
## 85 2023-01-01 4949.655
```

DISCLAIMER: The following data set is not gaussian distributed due to volatility during the Great Recession (December 2007 – June 2009) and the Pandemic (April 2020). The forecasting model does not pass the Shapiro-Wilk test!

```r
gdp = ts(gdp.csv[,2], start = c(2002,1), frequency = 4)
train <- gdp[1:73]
test <- gdp[74:85]
par(mfrow=c(1,2))
ts.plot(gdp, ylab = "Quarterly U.S. GDP (Billions of Chained 2012 Dollars)")
plot.ts(train)

fit <- lm(train ~ as.numeric(1:length(train))); abline(fit, col="red")
abline(h=mean(train), col="blue")
```



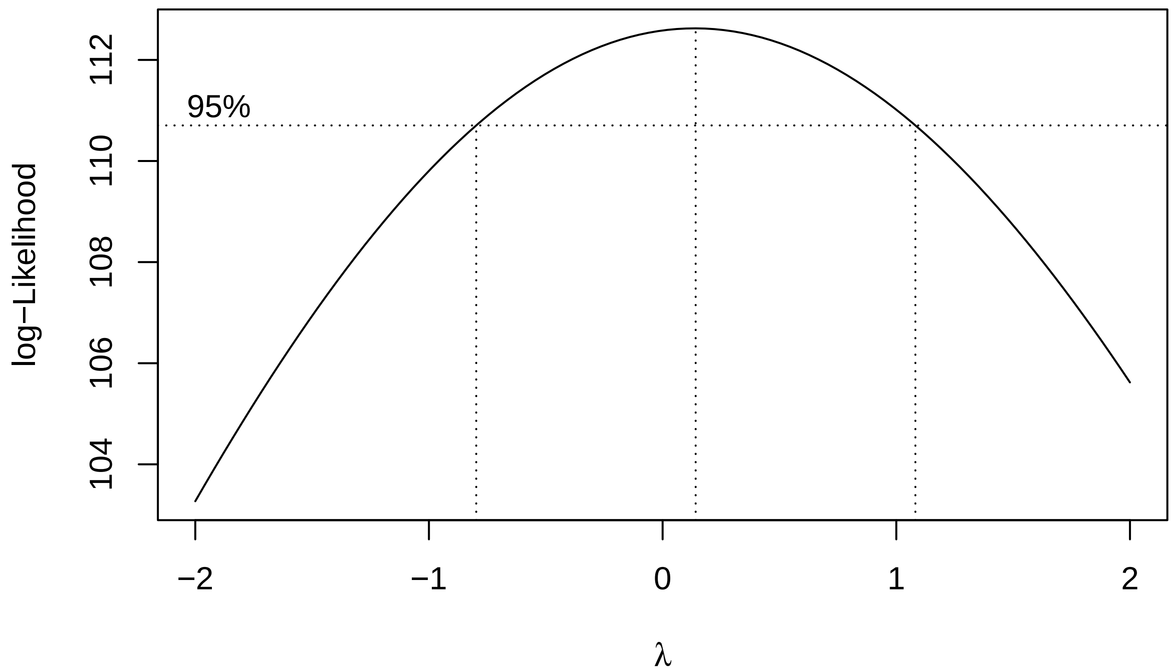

```
hist(train, col="light blue", xlab="", main="histogram; GDP data")
acf(train,lag.max=40, main="ACF of the GDP Data")
```



Histogram appear to be symmetric or bell-shaped (However, it is not gaussian). One might argue it is a little bit skewed right. Acf remains large.

Compare transformations

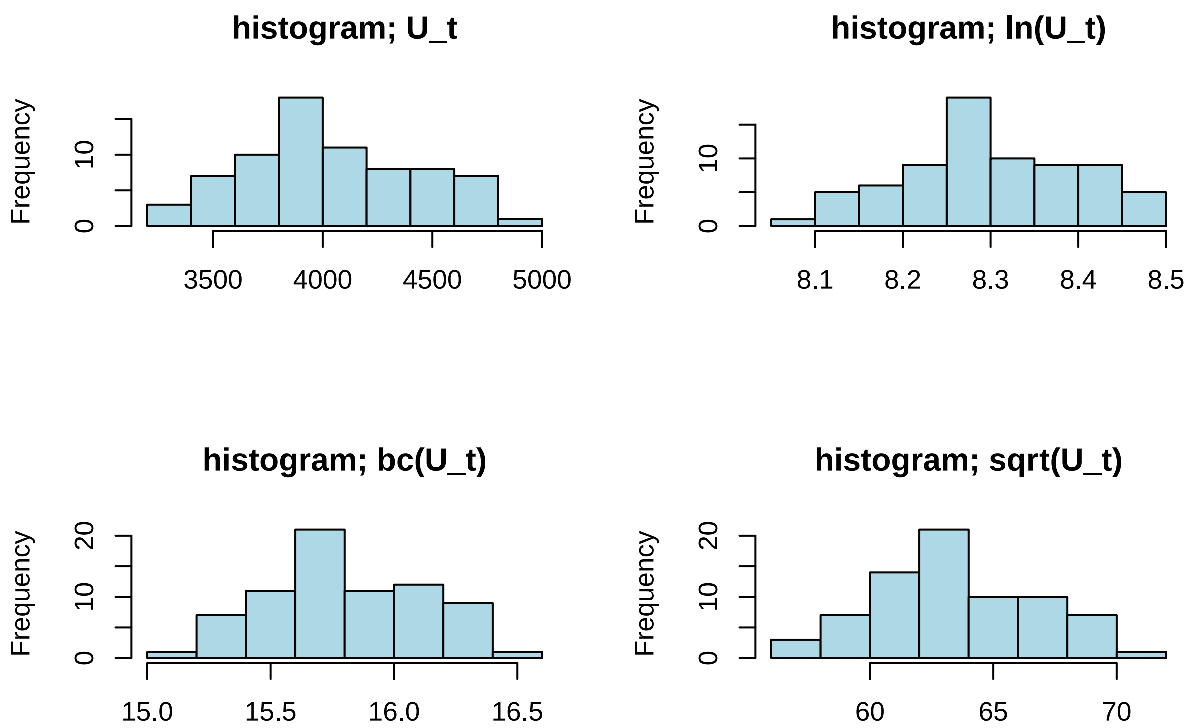
```
library(MASS)
bcTransform <- boxcox(train~ as.numeric(1:length(train)))
```



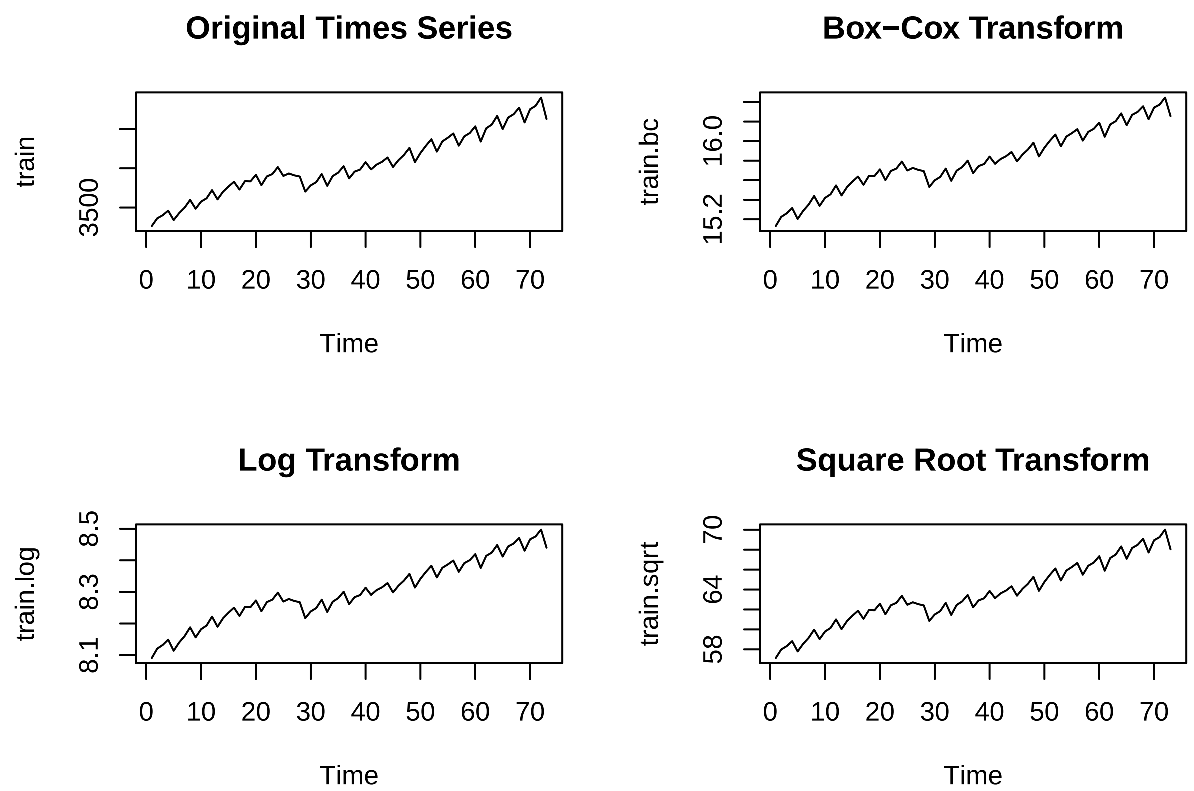

```
bcTransform$x[which(bcTransform$y == max(bcTransform$y))]
```

```
## [1] 0.1414141
```

```
lambda=bcTransform$x[which(bcTransform$y == max(bcTransform$y))]
train.bc = (1/lambda)*(train^lambda-1)
test.bc = (1/lambda)*(test^lambda-1)
train.log <- log(train)
train.sqrt = sqrt(train)

op= par(mfrow=c(2,2))
hist(train, col="light blue", xlab="", main="histogram; U_t")
hist(train.log, col="light blue", xlab="", main="histogram; ln(U_t)")
hist(train.bc, col="light blue", xlab="", main="histogram; bc(U_t)")
hist(train.sqrt, col="light blue", xlab="", main="histogram; sqrt(U_t)")
```



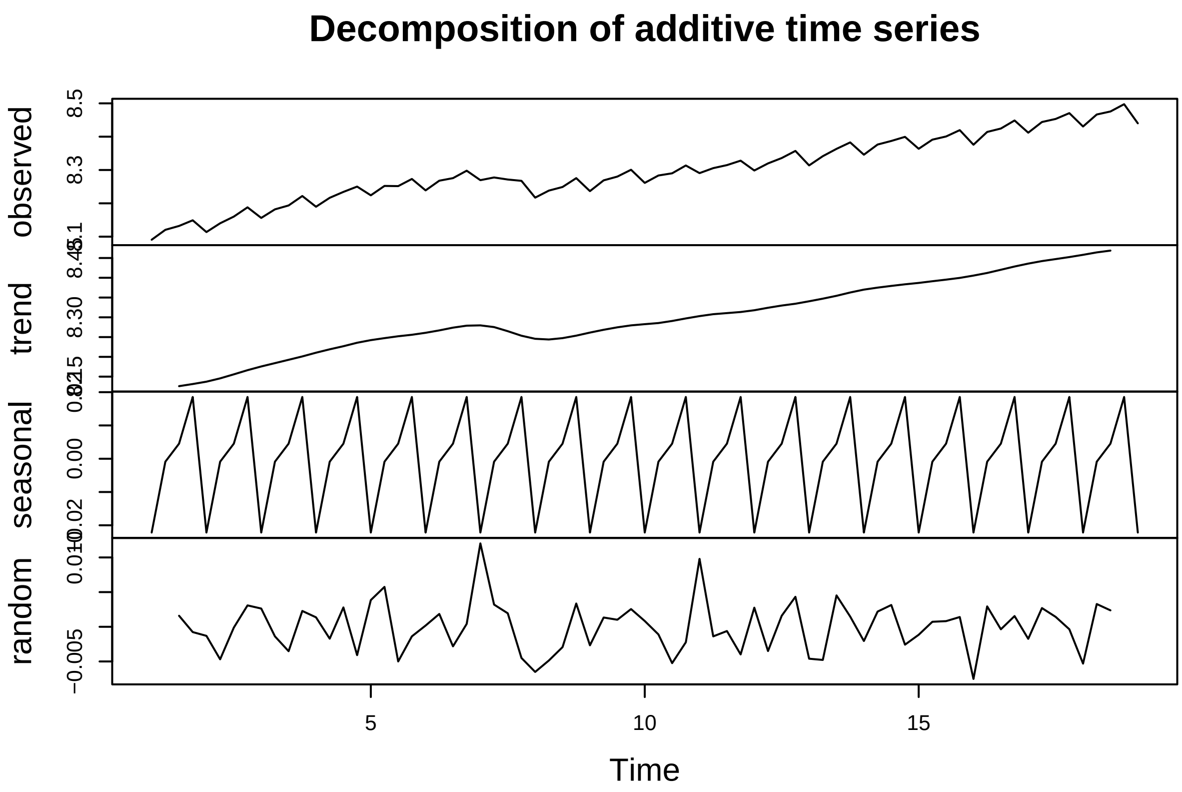

```r
#Compare transforms
op= par(mfrow=c(2,2))
ts.plot(train, main = "Original Times Series")
ts.plot(train.bc, main = "Box-Cox Transform")
ts.plot(train.log, main = "Log Transform")
ts.plot(train.sqrt, main = "Square Root Transform")
```



There isn't much difference between transformations. One might argue that a transformation is unnecessary. Choose log(U_t) because it is more symmetric and seems to have more even variance.

Decomposition of $ln(U_t)$ shows seasonality and almost linear trend

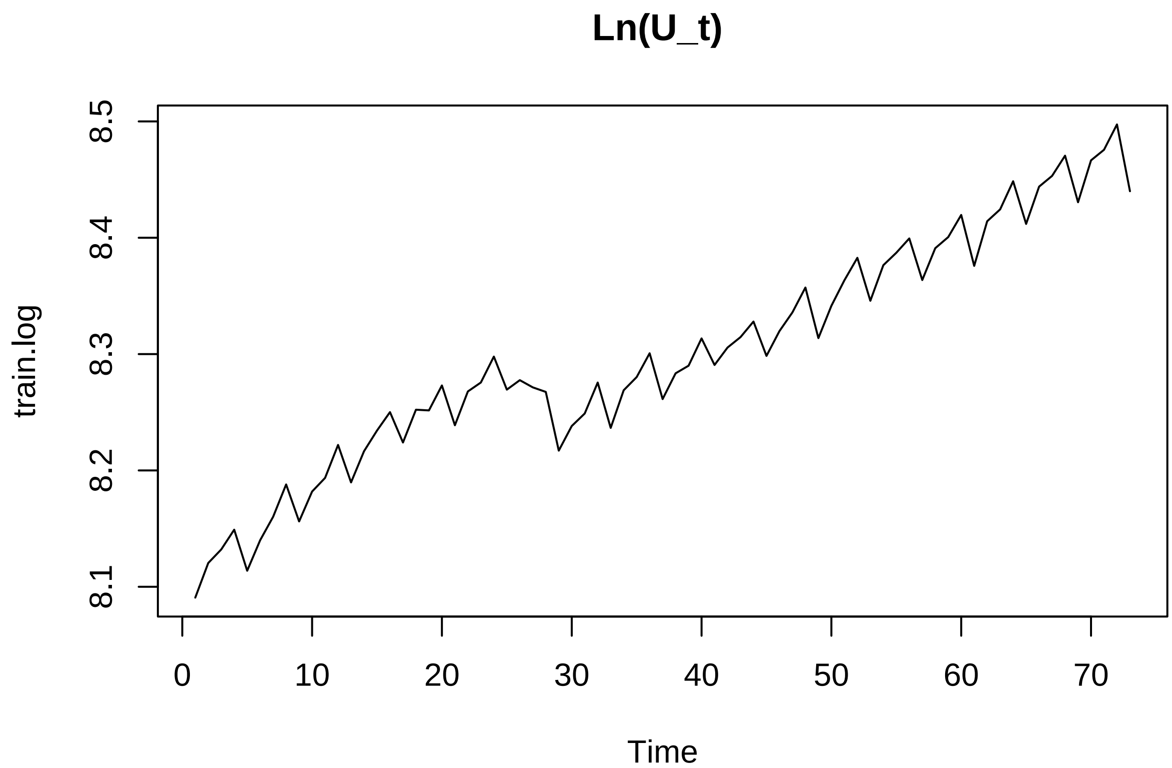
```
#install.packages("ggplot2")
#install.packages("ggfortify")
library(ggplot2)
library(ggfortify)

y <- ts(as.ts(train.log), frequency = 4)
decomp <- decompose(y)
plot(decomp)
```



```
var(train.log)
```

```
## [1] 0.009664501
```

```
plot.ts(train.log, main="Ln(U_t) ")
```



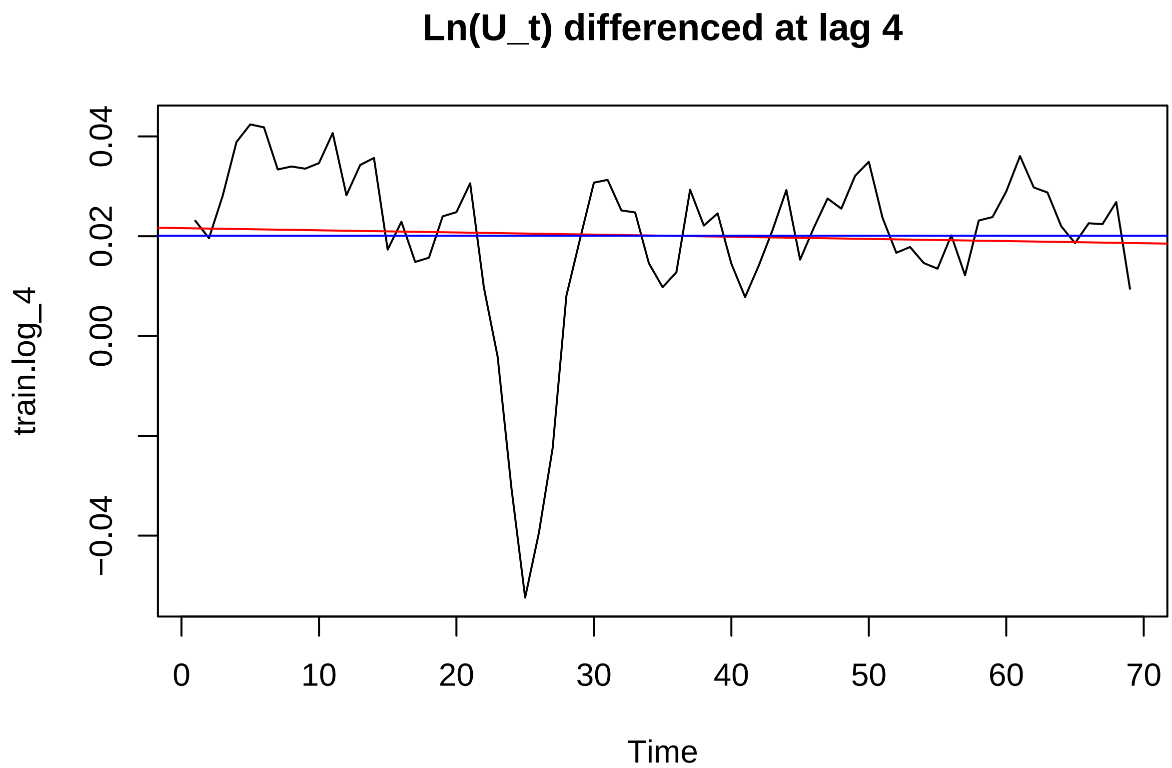

```
train.log_4 <- diff(train.log, lag=4)
plot.ts(train.log_4, main="Ln(U_t) differenced at lag 4")
var(train.log_4)
```

```
## [1] 0.0002872528
```

```
fit <- lm(train.log_4 ~ as.numeric(1:length(train.log_4))); abline(fit, col="red")
mean(train.log_4)
```

```
## [1] 0.02010507
```

```
abline(h=mean(train.log_4), col="blue")
```



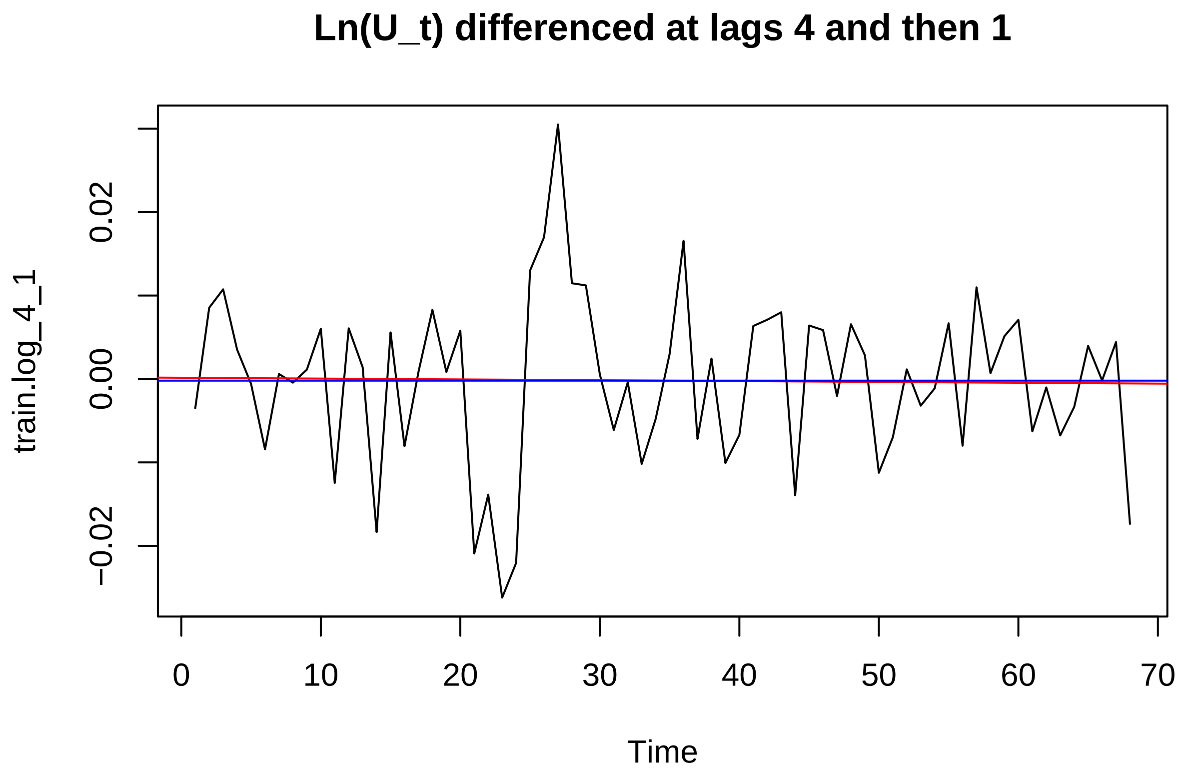

```r
train.log_4_1 <- diff(train.log_4, lag=1)
plot.ts(train.log_4_1, main="Ln(U_t) differenced at lags 4 and then 1")
var(train.log_4_1)
```

```
## [1] 9.788039e-05
```

```r
fit <- lm(train.log_4_1 ~ as.numeric(1:length(train.log_4_1))); abline(fit, col="red")
mean(train.log_4_1)
```

```
## [1] -0.0002004654
```

```r
abline(h=mean(train.log_4_1), col="blue")
```



Plot of $ln(U_t)$

- Seasonality
- Trend
- Variance: 0.009665

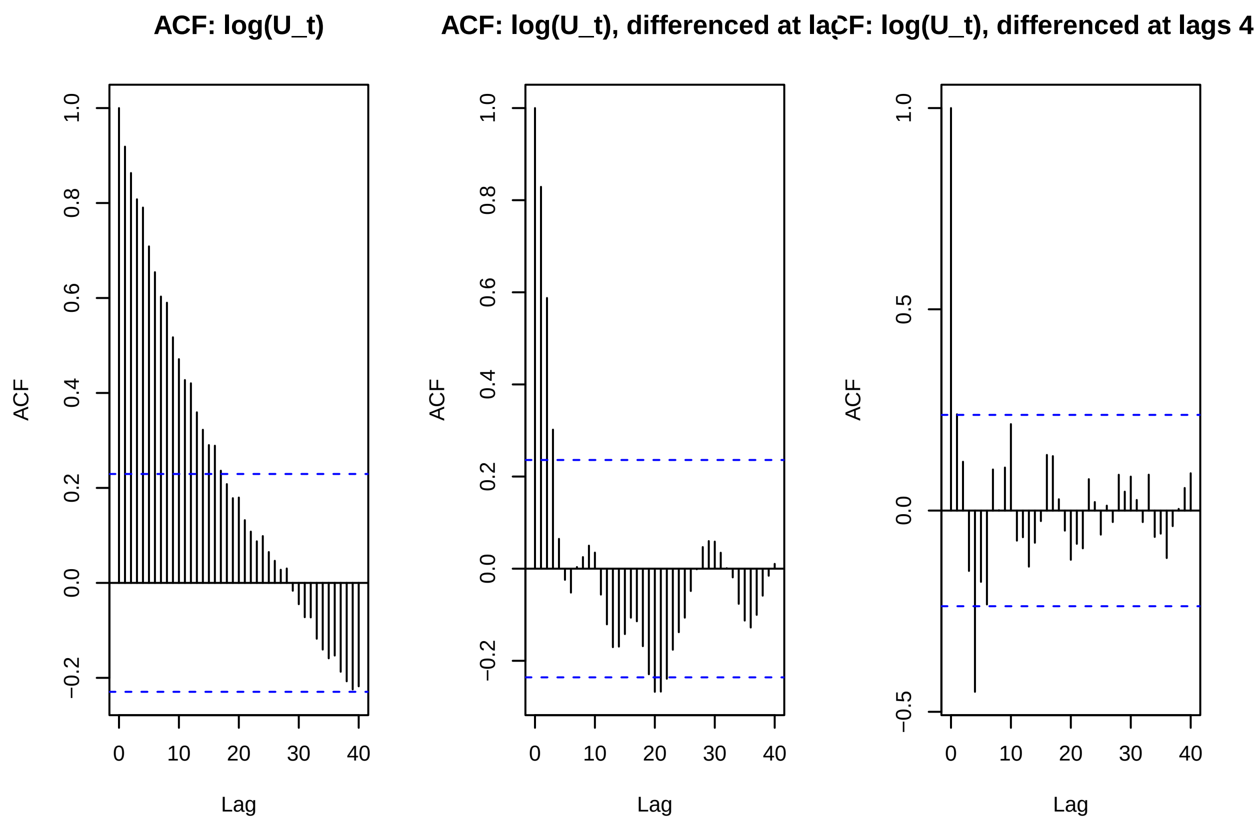
Plot of $ln(U_t)$ differenced at lag 4

- Seasonality no longer apparent
- Variance: 0.0002873 – lower!
- Trend is still here

Plot of $ln(U_t)$ differenced at lags 4 and then 1

- No Seasonality
- Variance: 9.788e-05 – even lower!
- No trend
- Data looks stationary, but check ACFs

```r
par(mfrow=c(1, 3))
acf(train.log, lag.max=40, main="ACF: log(U_t)")
acf(train.log_4, lag.max=40, main="ACF: log(U_t), differenced at lag 4")
acf(train.log_4_1, lag.max=40, main="ACF: log(U_t), differenced at lags 4 and 1")
```



Plot of ACF of $ln(U_t)$

- Slows decay indicated non-stationarity
- One sees seasonality

Plot of ACF of $ln(U_t)$ differenced at lag 4

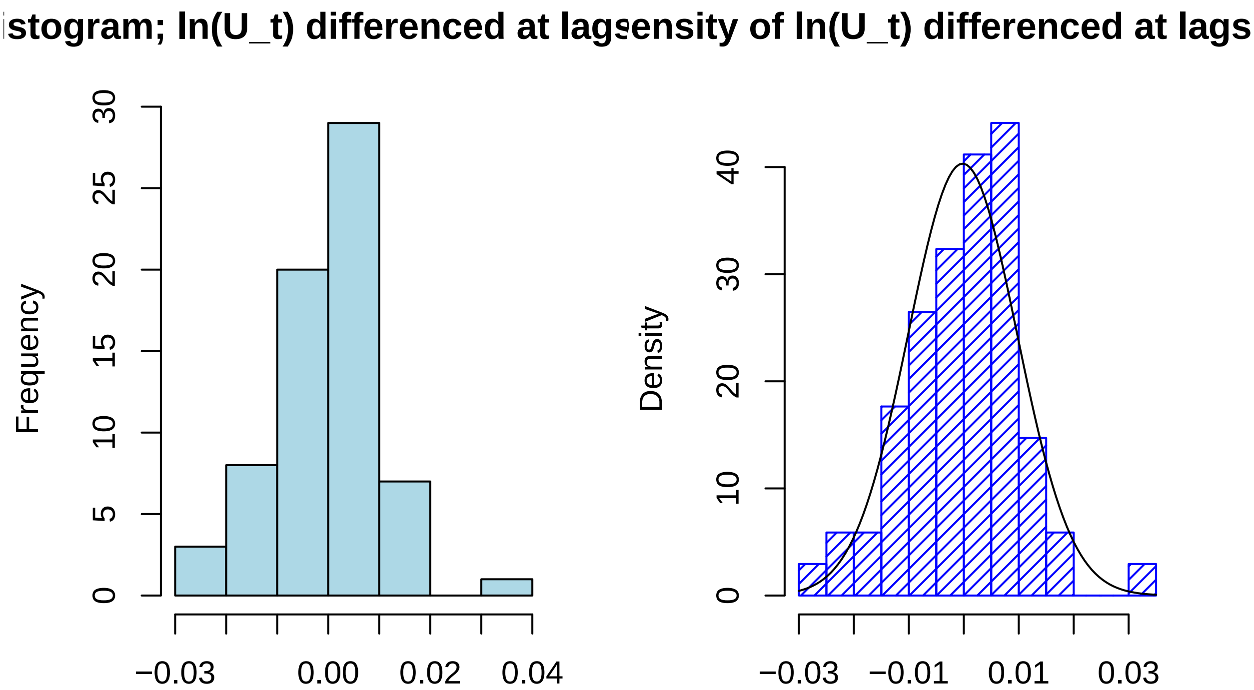
- Seasonality no longer apparent
- ACF decays slowly indicating non-stationarity

Plot of ACF of $ln(U_t)$ differenced at lags 4 & 1

- ACF decay corresponds to a stationary process Conclude: Work with data $ln(U_t)$ differenced at lags 4 & 1, U_t = the first 73 observations of the original data.

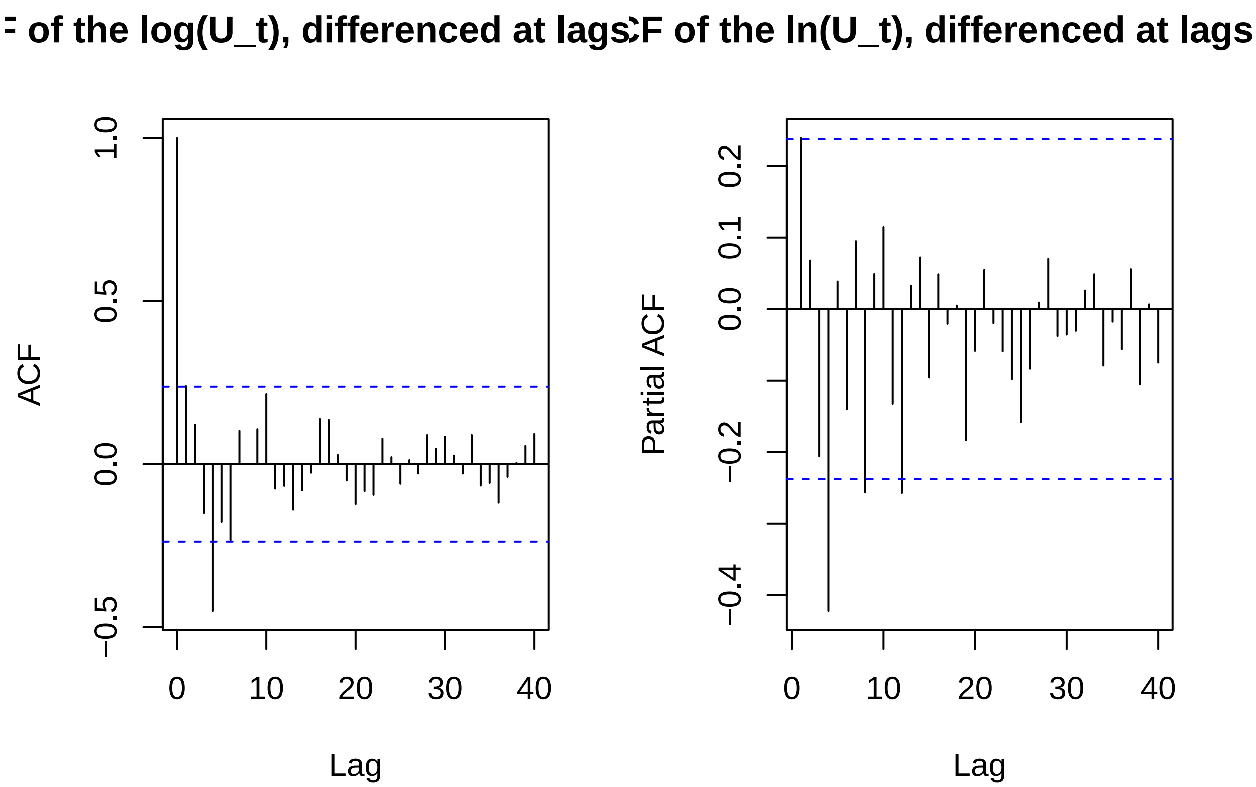
Histogram of $\nabla 1 \nabla 4 ln(U_t)$ looks symmetric and almost Gaussian.

```
par(mfrow=c(1, 2))
hist(train.log_4_1, col="light blue", xlab="", main="histogram; ln(U_t) differenced at l
hist(train.log_4_1, density=20,breaks=20, col="blue", xlab="", main="Density of ln(U_t)
m<-mean(train.log_4_1)
std<- sqrt(var(train.log_4_1))
curve( dnorm(x,m,std), add=TRUE )
```



ACF and PACF of train.log_4_1 $= \nabla 1 \nabla 4 ln(U_t)$,

```
par(mfrow=c(1, 2))
acf(train.log_4_1, lag.max=40, main="ACF of the log(U_t), differenced at lags 4 and 1")
pacf(train.log_4_1, lag.max=40, main="PACF of the ln(U_t), differenced at lags 4 and 1")
```



Determine possible candidate models SARIMA$(p, d, q) \times (P, D, Q)_s$ for the series $ln(U_t)$.

Modeling the seasonal part (P, D, Q): For this part, focus on the seasonal lags h = 1s, 2s, etc.

- We applied one seasonal differencing so D = 1 at lag s = 4.
- The ACF shows a strong peak at h = 1s and h = 4s.

A good choice for the MA part could be Q = 1 or Q = 4

- The PACF shows a peak at h = 1s, 4s, 8s and 12s. A good choice for the AR part could be P = 1 or P = 4.

Modeling the non-seasonal part (p , d, q): In this case focus on the within season lags, h = 1,. . . ,11.

- We applied one differencing to remove the trend: d = 1
- The ACF seems to be tailing off. Or perhaps cuts off at lag 1 or 4.

A good choice for the MA part could be q = 1, 4.

- The PACF cuts off at lag h=1 or 4.

A good choice for the AR part could be p = 1, 4.

As an illustration we fit the following model:

SARIMA (p = 1, d = 1, q = 1) × (P = 1, D = 1, Q = 1)s=4

SARIMA (p = 4, d = 1, q = 4) × (P = 4, D = 1, Q = 4)s=4

Try candidate models.

```
arima(train.log, order=c(4,1,4), seasonal = list(order = c(4,1,4), period = 4), method="
```

```
## Warning in arima(train.log, order = c(4, 1, 4), seasonal = list(order = c(4, :
## possible convergence problem: optim gave code = 1
```

```
##
## Call:
## arima(x = train.log, order = c(4, 1, 4), seasonal = list(order = c(4, 1, 4),
##     period = 4), method = "ML")
##
## Coefficients:
```

```
## Warning in sqrt(diag(x$var.coef)): NaNs produced
```

```
##           ar1     ar2     ar3      ar4     ma1      ma2      ma3     ma4
##       -0.1864  0.5091  0.0887  -0.8158  0.4534  -0.3167  -0.2387  0.4697
## s.e.      NaN  0.1544  0.1215   0.0829     NaN      NaN   0.2366     NaN
##          sar1     sar2    sar3  sar4    sma1    sma2     sma3    sma4
##       -0.4179  -0.2166  0.3249  0.09  0.0358  -0.342  -0.7007  0.2117
## s.e.      NaN      NaN     NaN   NaN     NaN     NaN   0.1359     NaN
##
## sigma^2 estimated as 4.667e-05:  log likelihood = 237.54,  aic = -441.08
```

```
arima(train.log, order=c(1,1,1), seasonal = list(order = c(1,1,1), period = 4), method="
```

```
##
## Call:
## arima(x = train.log, order = c(1, 1, 1), seasonal = list(order = c(1, 1, 1),
##     period = 4), method = "ML")
##
## Coefficients:
##          ar1      ma1    sar1     sma1
```

```
##       0.4739  -0.2655  0.0334  -0.8102
## s.e.  0.3534   0.3747  0.1762   0.1477
##
## sigma^2 estimated as 5.746e-05:  log likelihood = 233.48,  aic = -456.96
```

```
arima(train.log, order=c(1,1,1), seasonal = list(order = c(1,1,1), period = 4), fixed =
```

```
## Warning in arima(train.log, order = c(1, 1, 1), seasonal = list(order = c(1, :
## some AR parameters were fixed: setting transform.pars = FALSE
```

```
##
## Call:
## arima(x = train.log, order = c(1, 1, 1), seasonal = list(order = c(1, 1, 1),
##     period = 4), fixed = c(NA, 0, 0, NA), method = "ML")
##
## Coefficients:
##          ar1  ma1  sar1     sma1
##       0.2132    0     0  -0.7857
## s.e.  0.1246    0     0   0.1197
##
## sigma^2 estimated as 5.808e-05:  log likelihood = 233.2,  aic = -460.39
```

```
arima(train.log, order=c(1,1,0), seasonal = list(order = c(0,1,1), period = 4), method="
```

```
##
## Call:
## arima(x = train.log, order = c(1, 1, 0), seasonal = list(order = c(0, 1, 1),
##     period = 4), method = "ML")
##
## Coefficients:
##          ar1     sma1
##       0.2132  -0.7857
## s.e.  0.1246   0.1197
##
## sigma^2 estimated as 5.808e-05:  log likelihood = 233.2,  aic = -460.39
```

```
arima(train.log, order=c(0,1,4), seasonal = list(order = c(1,1,1), period = 4), method="
```

```
##
## Call:
## arima(x = train.log, order = c(0, 1, 4), seasonal = list(order = c(1, 1, 1),
##     period = 4), method = "ML")
```

```
##
## Coefficients:
##          ma1     ma2      ma3     ma4     sar1     sma1
##       0.1984  0.2131  -0.0709  0.8025  -0.6712  -0.7937
## s.e.  0.0990  0.0906   0.1167  0.1242   0.1412   0.1261
##
## sigma^2 estimated as 5.009e-05:  log likelihood = 236.7,  aic = -459.39
```

```r
arima(train.log, order=c(0,1,4), seasonal = list(order = c(1,1,1), period = 4), fixed =
```

```
##
## Call:
## arima(x = train.log, order = c(0, 1, 4), seasonal = list(order = c(1, 1, 1),
##     period = 4), fixed = c(NA, NA, 0, NA, NA, NA), method = "ML")
##
## Coefficients:
##          ma1     ma2  ma3     ma4     sar1     sma1
##       0.1811  0.2353    0  0.8422  -0.6987  -0.7934
## s.e.  0.0826  0.0782    0  0.1030   0.1465   0.1237
##
## sigma^2 estimated as 5.1e-05:  log likelihood = 236.48,  aic = -460.96
```

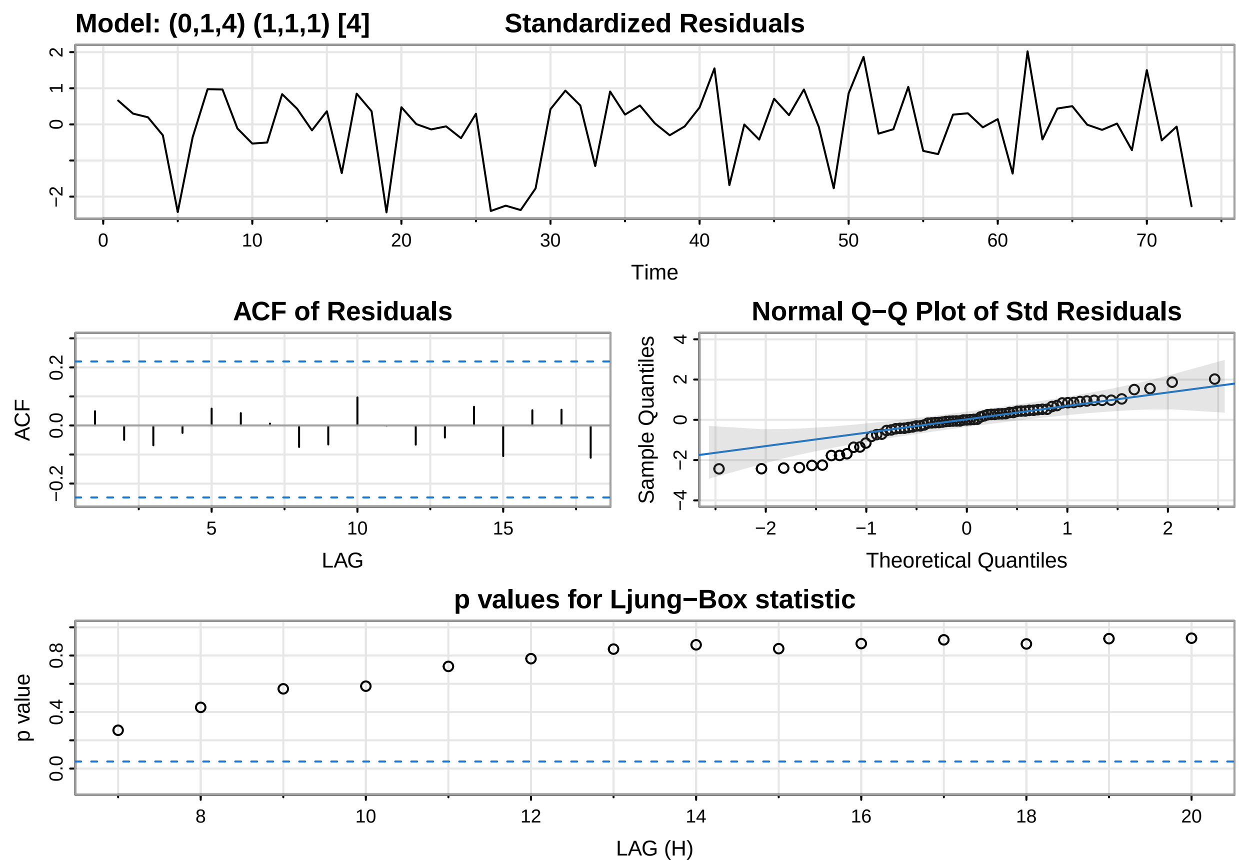
Our chosen model is: SARMA ($p = 0$, $d = 1$, $q = 4$) $\times$ ($P = 1$, $D = 1$, $Q = 1$)s=4

```r
fit.i <- sarima(xdata = train.log, p = 0, d = 1, q = 4, P = 1, D = 1, Q = 1, S = 4)
```

```
## initial  value -4.609268
## iter   2 value -4.642214
## iter   3 value -4.800009
## iter   4 value -4.811586
## iter   5 value -4.819757
## iter   6 value -4.822514
## iter   7 value -4.824085
## iter   8 value -4.826922
## iter   9 value -4.829141
## iter  10 value -4.830885
## iter  11 value -4.834626
## iter  12 value -4.842839
## iter  13 value -4.850116
## iter  14 value -4.857194
## iter  15 value -4.861821
```

```
## iter  16 value -4.867761
## iter  17 value -4.872671
## iter  18 value -4.877943
## iter  19 value -4.883747
## iter  20 value -4.890798
## iter  21 value -4.893435
## iter  22 value -4.893997
## iter  23 value -4.896854
## iter  24 value -4.897684
## iter  25 value -4.906952
## iter  26 value -4.910366
## iter  27 value -4.932993
## iter  28 value -4.950032
## iter  29 value -4.963518
## iter  30 value -4.974003
## iter  31 value -4.982583
## iter  32 value -4.983232
## iter  33 value -4.987585
## iter  34 value -4.991273
## iter  35 value -5.005516
## iter  36 value -5.015382
## iter  37 value -5.026687
## iter  38 value -5.032704
## iter  39 value -5.041045
## iter  40 value -5.043318
## iter  41 value -5.047523
## iter  42 value -5.054274
## iter  43 value -5.057815
## iter  44 value -5.060731
## iter  45 value -5.060856
## iter  46 value -5.065235
## iter  47 value -5.067495
## iter  48 value -5.070624
## iter  49 value -5.071428
## iter  50 value -5.075305
## iter  51 value -5.078713
## iter  52 value -5.081303
## iter  53 value -5.084384
## iter  54 value -5.086204
## iter  55 value -5.089075
## iter  56 value -5.089132
## iter  57 value -5.089195
## iter  58 value -5.089196
## iter  59 value -5.090696
## iter  59 value -5.090696
```

```
## iter  60 value -5.090790
## iter  61 value -5.090792
## iter  62 value -5.091655
## iter  63 value -5.092347
## iter  64 value -5.092487
## iter  65 value -5.093596
## iter  66 value -5.093611
## iter  67 value -5.093669
## iter  68 value -5.093670
## iter  69 value -5.093670
## iter  69 value -5.093670
## iter  70 value -5.094122
## iter  71 value -5.094122
## iter  71 value -5.094122
## iter  72 value -5.094234
## iter  72 value -5.094234
## iter  73 value -5.094237
## iter  73 value -5.094237
## iter  74 value -5.094238
## iter  74 value -5.094238
## iter  75 value -5.094239
## iter  75 value -5.094239
## iter  75 value -5.094239
## final  value -5.094239
## converged
## initial  value -4.898013
## iter   2 value -4.899022
## iter   3 value -4.899519
## iter   4 value -4.899755
## iter   5 value -4.899773
## iter   6 value -4.899778
## iter   7 value -4.899779
## iter   8 value -4.899780
## iter   9 value -4.899780
## iter   9 value -4.899780
## iter   9 value -4.899780
## final  value -4.899780
## converged
```




```r
print('Coefficients')
```

```
## [1] "Coefficients"
```

```r
fit.i$fit$coef
```

```
##         ma1         ma2         ma3         ma4        sar1        sma1
##  0.19838235  0.21305446 -0.07094011  0.80250441 -0.67129980 -0.79368654
```

```r
polyroot(c(1,0.19838,0.21305,-0.07094,0.80250))
```

```
## [1]  0.7261190+0.8478433i -0.6819196+0.7314374i -0.6819196-0.7314374i
## [4]  0.7261190-0.8478433i
```

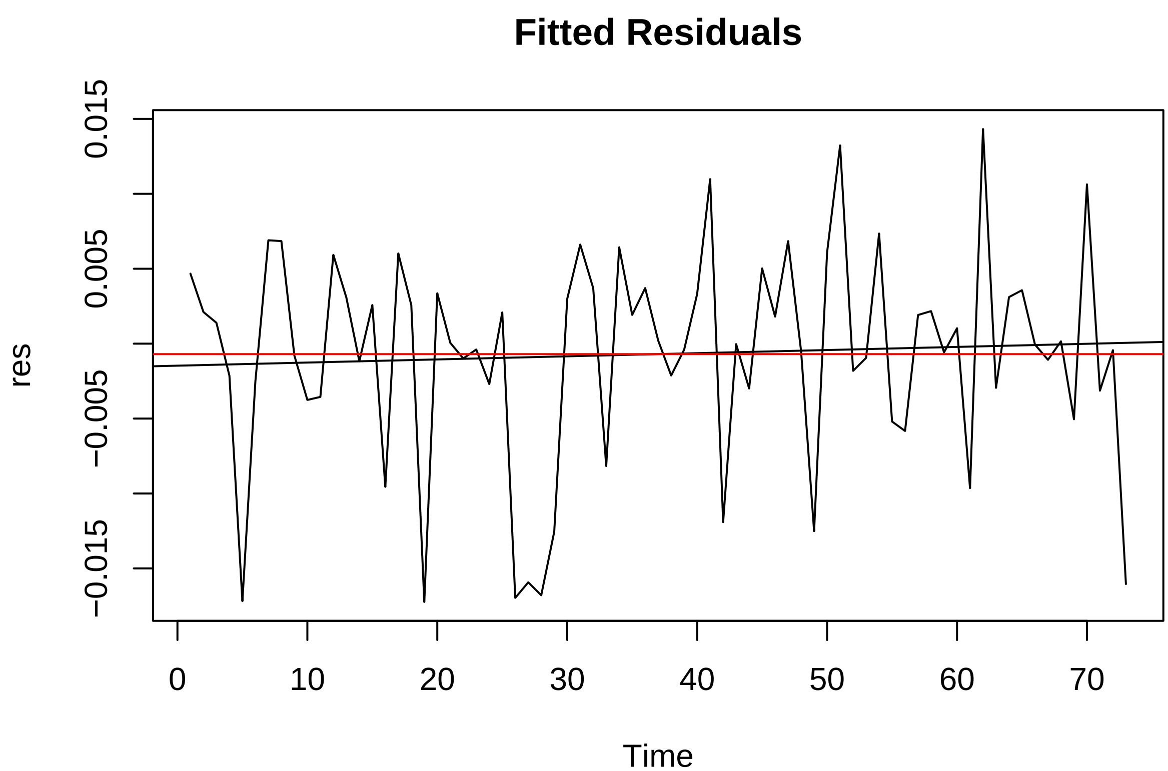
model is invertible.

```r
# Residual plots:
res <- fit.i$fit$residuals
mean(res); var(res)
```

```
## [1] -0.0006974634
```

```
## [1] 5.137446e-05
```

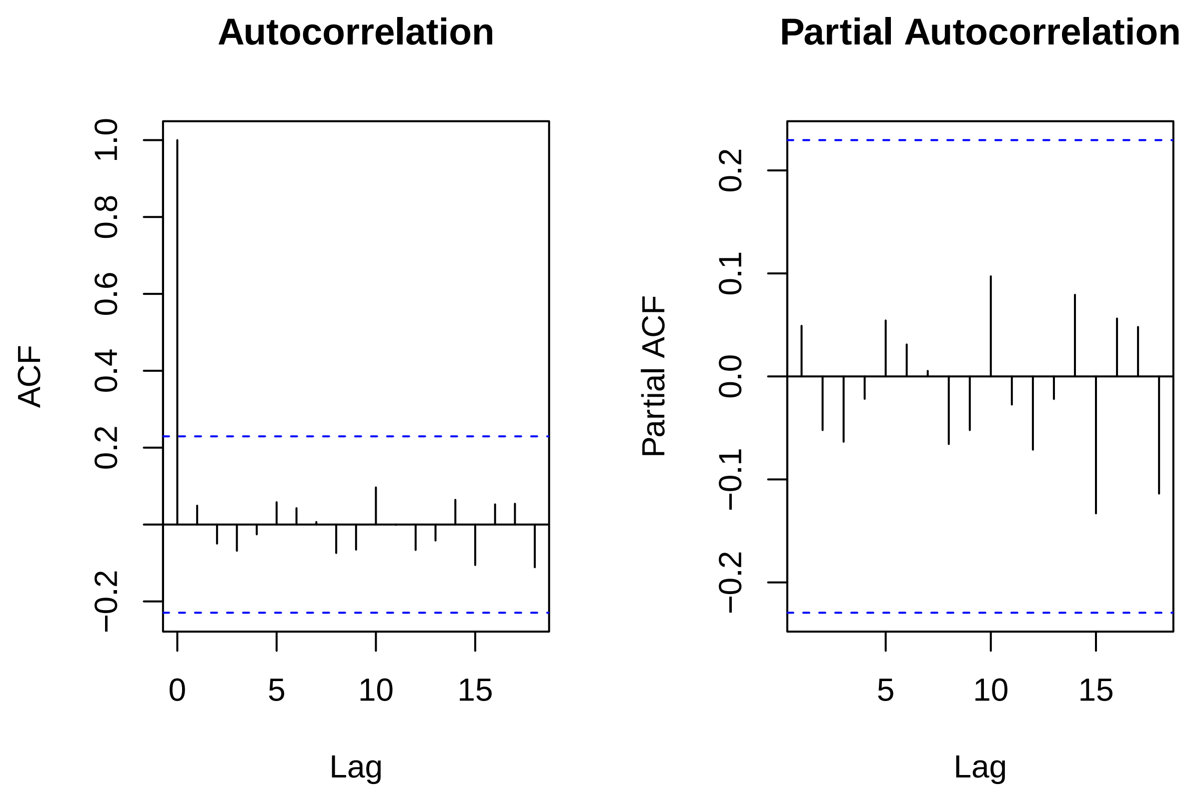
```r
# layout(matrix(c(1, 1, 2, 3), 2, 2, byrow=T))
par(mfrow=c(1, 1))
ts.plot(res, main="Fitted Residuals")
t <- 1:length(res)
fit.res = lm(res~ t)
abline(fit.res)
abline(h = mean(res), col = "red")
```




```r
var(fit.res$residuals)
```

```
## [1] 5.117826e-05
```

```r
# ACF and PACF:
par(mfrow=c(1, 2))
acf(res, main="Autocorrelation")
pacf(res, main="Partial Autocorrelation")
```



All acf and pacf of residuals are within confidence intervals and can be counted as zeros!

```r
# Test for independence of residuals
Box.test(res, lag = 9, type = c("Box-Pierce"), fitdf = 6)
```

```
##
##  Box-Pierce test
##
## data:  res
## X-squared = 1.8362, df = 3, p-value = 0.6071
```

```r
Box.test(res, lag = 9, type = c("Ljung-Box"), fitdf = 6)
```

```
##
##  Box-Ljung test
##
## data:  res
## X-squared = 2.0388, df = 3, p-value = 0.5644
```

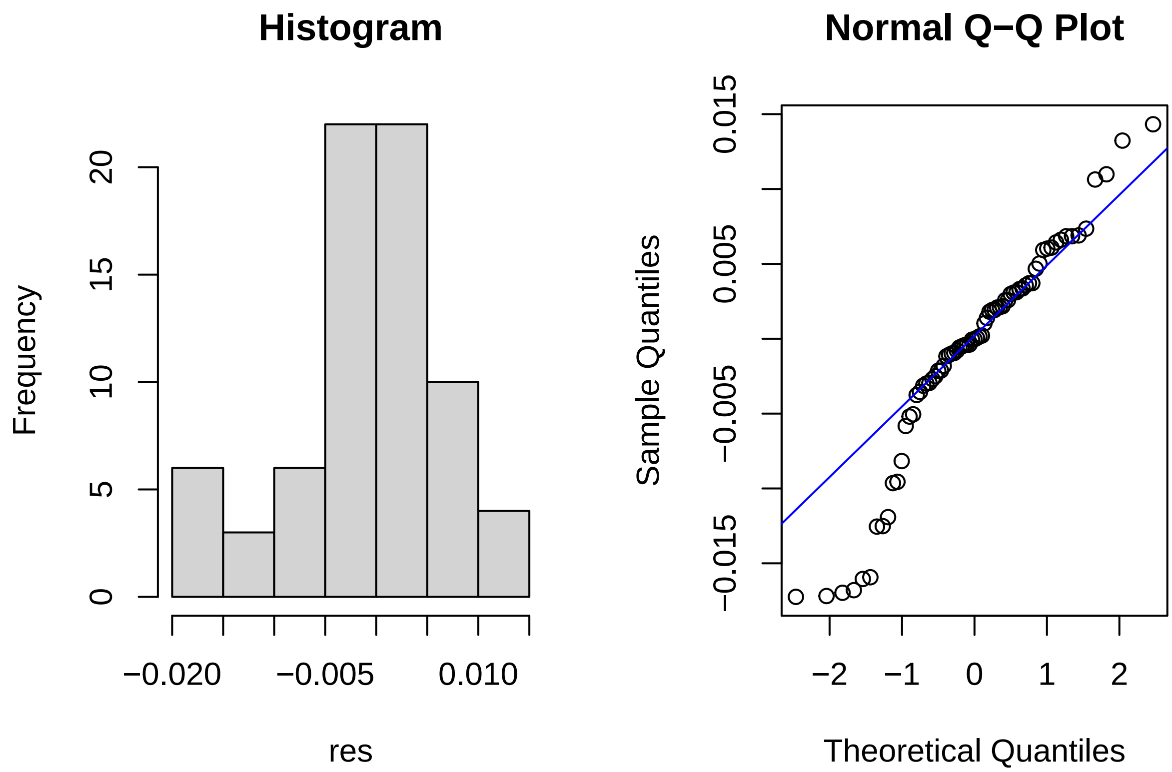
```r
Box.test(res^2, lag = 9, type = c("Ljung-Box"), fitdf = 0)
```

```
##
##  Box-Ljung test
##
## data:  res^2
## X-squared = 11.61, df = 9, p-value = 0.2362
```

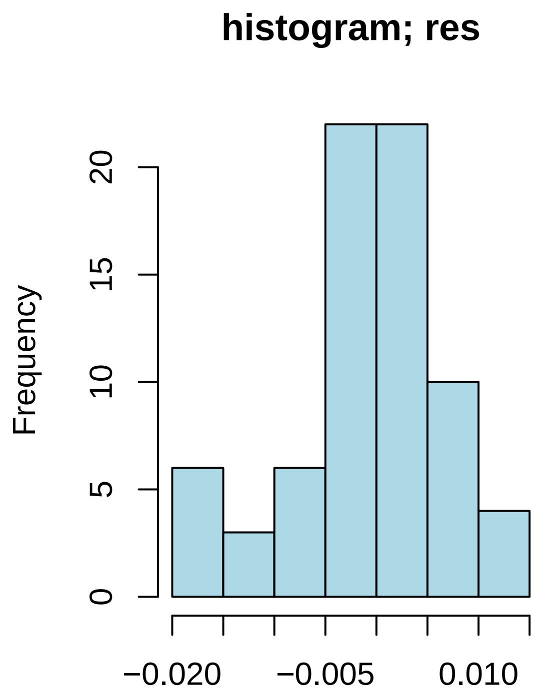
```r
# Test for normality of residuals
shapiro.test(res)
```

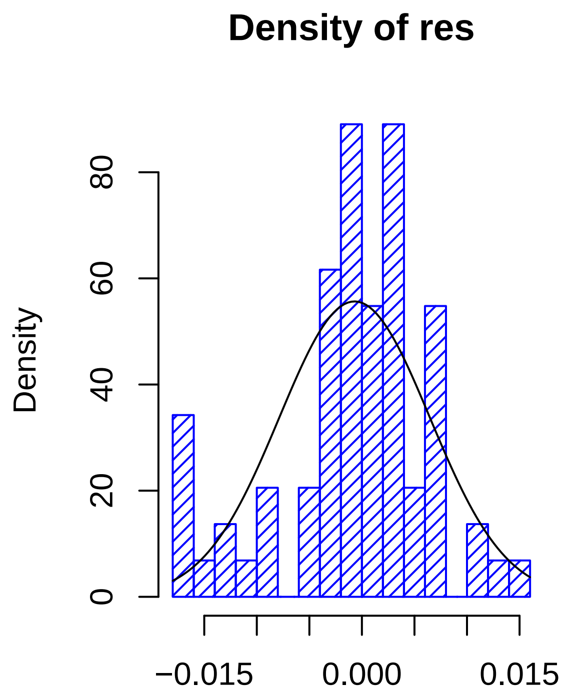
```
##
##  Shapiro-Wilk normality test
##
## data:  res
## W = 0.93546, p-value = 0.001031
```

```r
# Histogram and QQ-plot:
par(mfrow=c(1,2))
hist(res,main = "Histogram")
qqnorm(res)
qqline(res,col ="blue")
```




```
par(mfrow=c(1, 2))
hist(res, col="light blue", xlab="", main="histogram; res")
hist(res, density=20,breaks=20, col="blue", xlab="", main="Density of res", prob=TRUE)
m<-mean(res)
#0.001234
std<- sqrt(var(res))
curve( dnorm(x,m,std), add=TRUE )
```





```r
# Access the coefficients of the SARIMA model
coefficients <- coef(fit.i$fit)

# Print the coefficients
coefficients
```

```
##         ma1         ma2         ma3         ma4        sar1        sma1 
##  0.19838235  0.21305446 -0.07094011  0.80250441 -0.67129980 -0.79368654
```

```r
# Access the variance
var <- fit.i$fit$sigma2
var
```

```
## [1] 5.009395e-05
```

The residuals appear to be white noise and they are normally distributed since the histogram appears approximately symmetric and the QQ plot shows residuals aligning along the diagonal line.

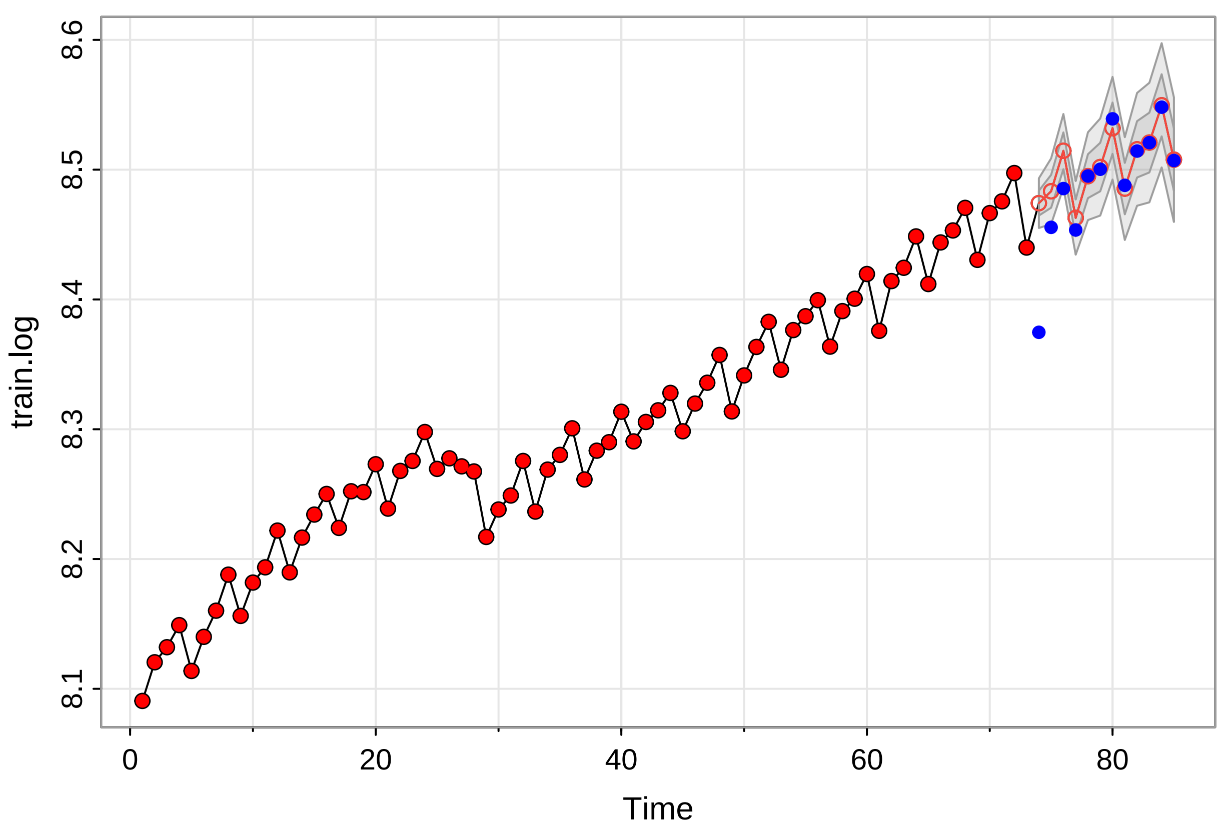
The SARIMA(0, 1, 0) × (1, 1, 1)12 model with coefficients (sar1 = -0.04274, sma1 = -0.40209) can be expressed as:

$$(1-(-0.04274))(1-B^{12})Y_t = (1-(-0.40209)B)(1-B)Z_t$$

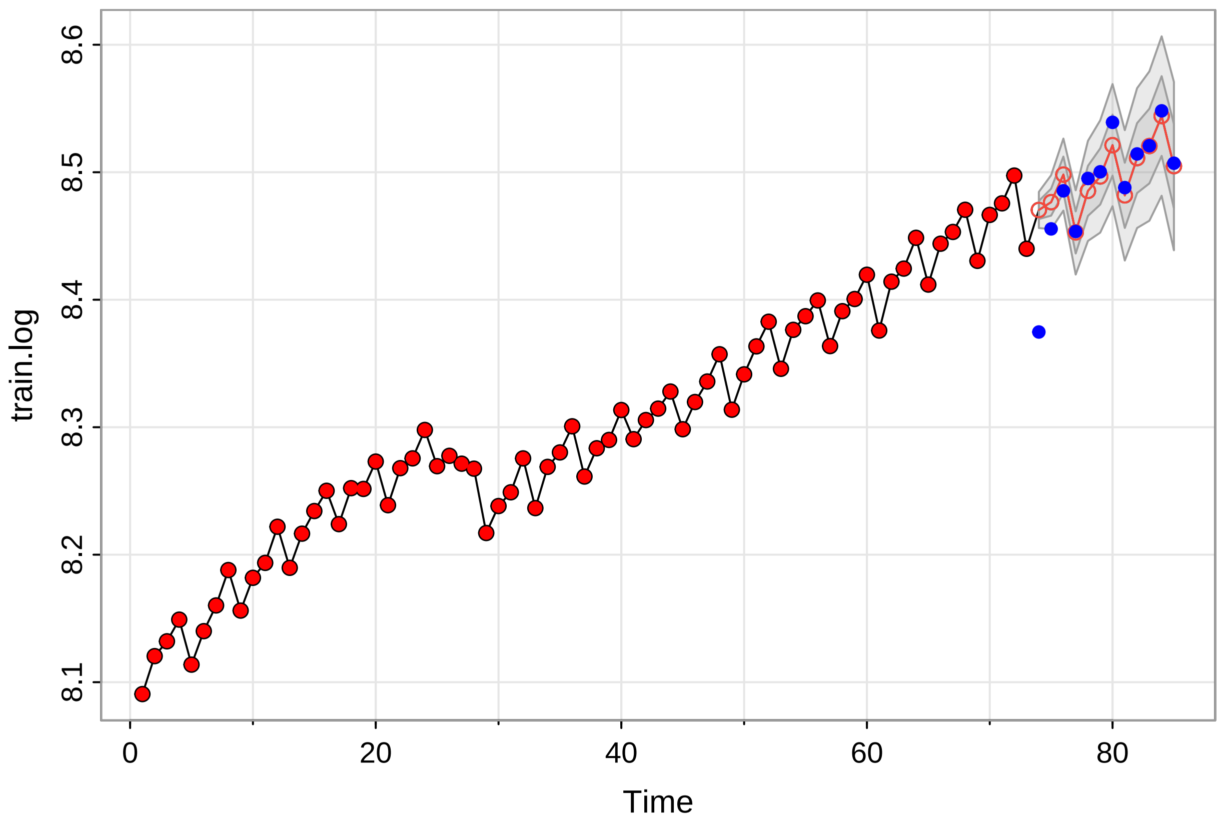
Here, Y_t represents the series "train.log_4_1" and Z_t is a normally distributed error term. The coefficients sar1 and sma1 correspond to the autoregressive (AR) and seasonal moving average (SMA) terms, respectively, in the SARIMA model.

Hence, the final model will be $(1+0.04274))(1-B^{12})Y_t = (1+0.40209)B)(1-B)Z_t$, where $Z_t \sim N(0, 1503)$.

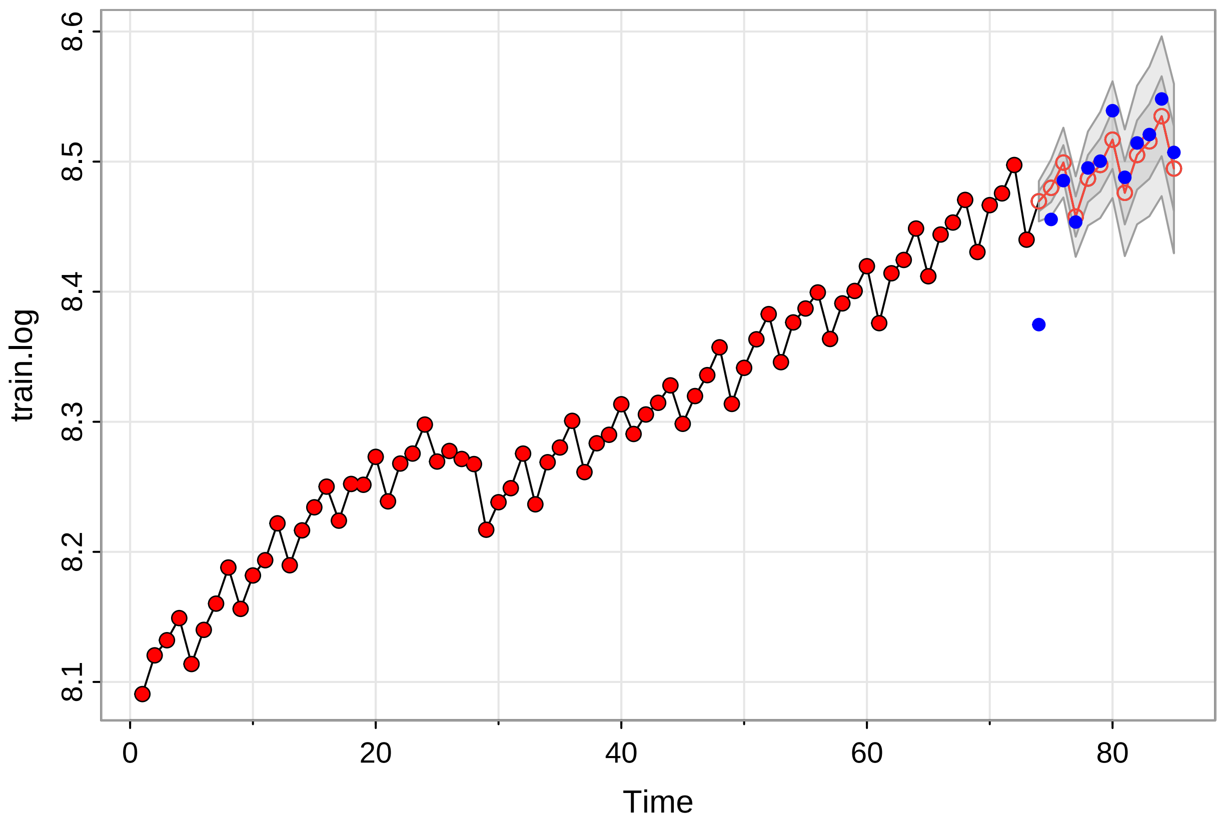
```
x <- sarima.for(xdata=train.log, n.ahead = 12, p=3, d=1, q=2, P=0, D=0, Q=0, S=4)
points(1:length(train), train.log, col = "red", pch = 19, cex = 0.8)
points(length(train) + 1:length(test), log(test), col = "blue", pch = 19, cex = 0.8)
```



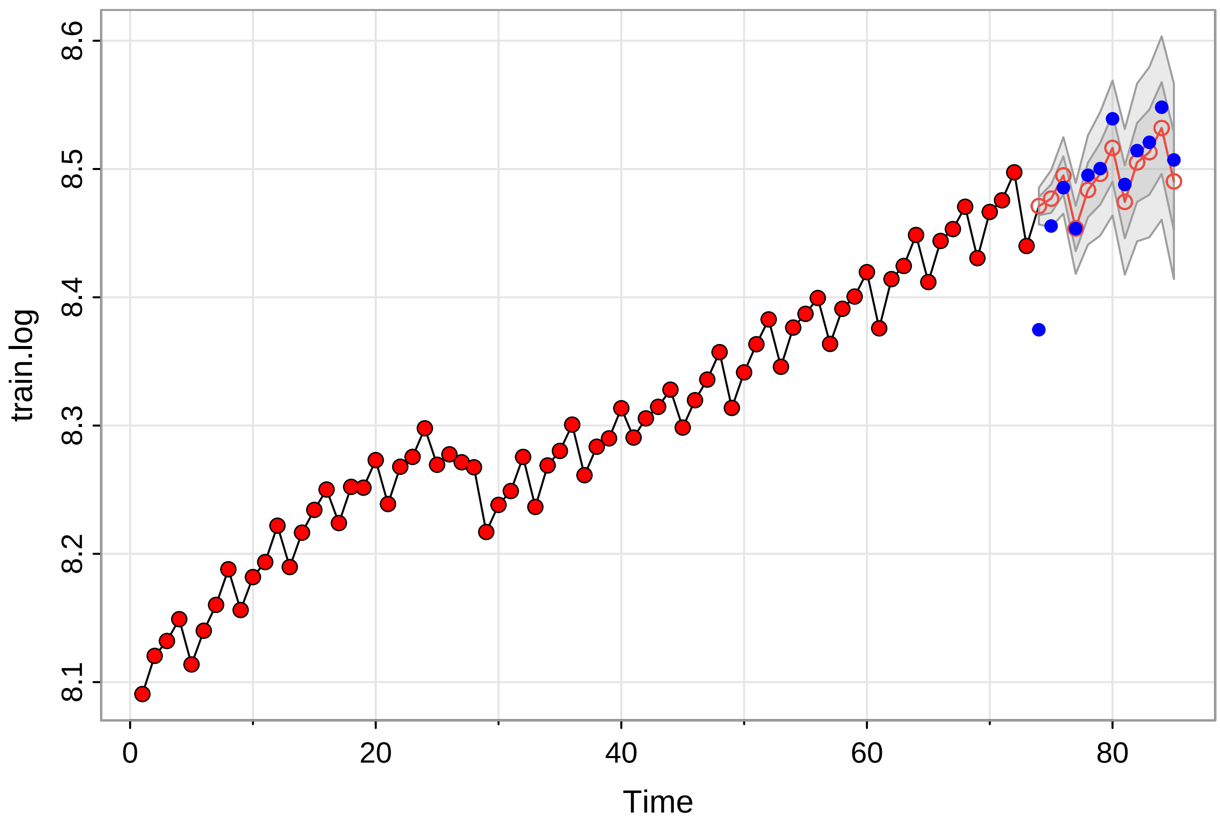

```
x <- sarima.for(xdata=train.log, n.ahead = 12, p=4, d=1, q=4, P=4, D=1, Q=4, S=4)
points(1:length(train), train.log, col = "red", pch = 19, cex = 0.8)
points(length(train) + 1:length(test), log(test), col = "blue", pch = 19, cex = 0.8)
```




```
x <- sarima.for(xdata=train.log, n.ahead = 12, p=0, d=1, q=0, P=0, D=1, Q=4, S=4)
points(1:length(train), train.log, col = "red", pch = 19, cex = 0.8)
points(length(train) + 1:length(test), log(test), col = "blue", pch = 19, cex = 0.8)
```




```
x <- sarima.for(xdata=train.log, n.ahead = 12, p=0, d=1, q=4, P=1, D=1, Q=1, S=4)
points(1:length(train), train.log, col = "red", pch = 19, cex = 0.8)
points(length(train) + 1:length(test), log(test), col = "blue", pch = 19, cex = 0.8)
```



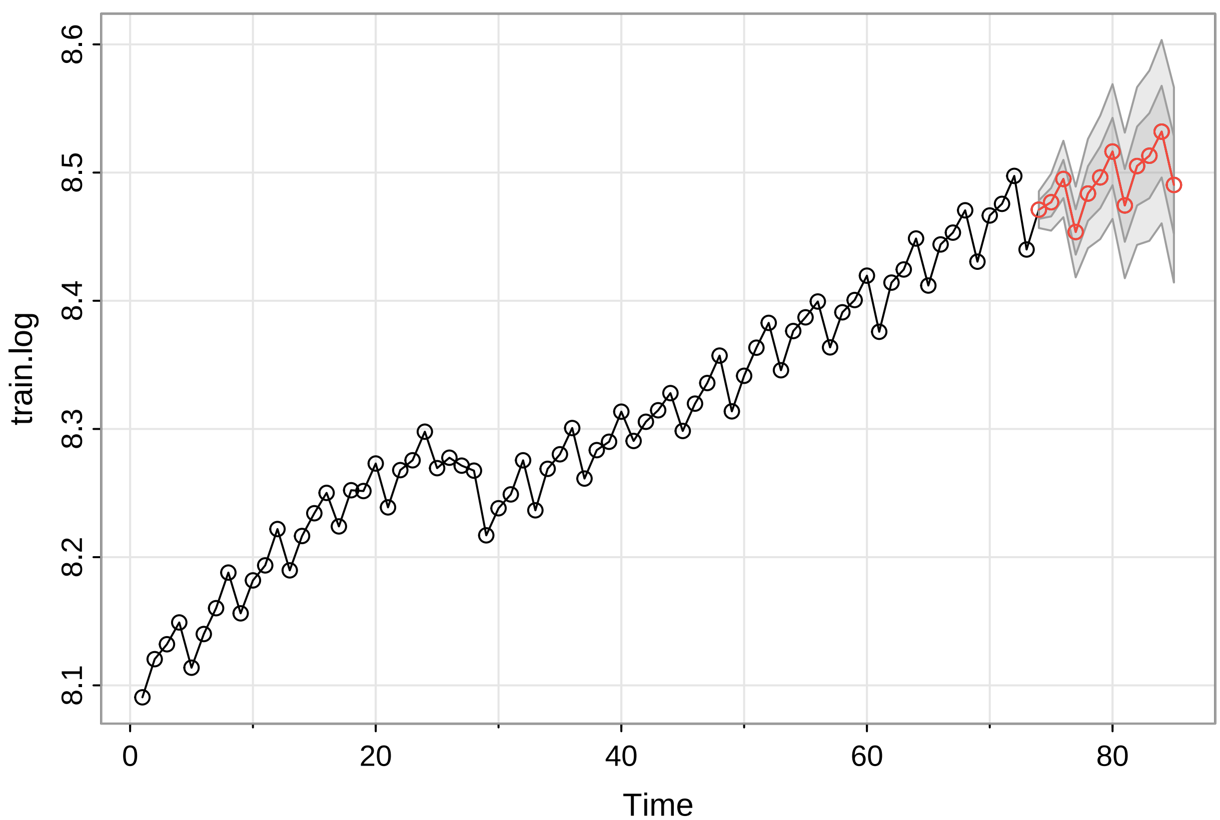
```r
#install.packages("forecast")  # Install the forecast package if not already installed
library(forecast)              # Load the forecast package
```

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

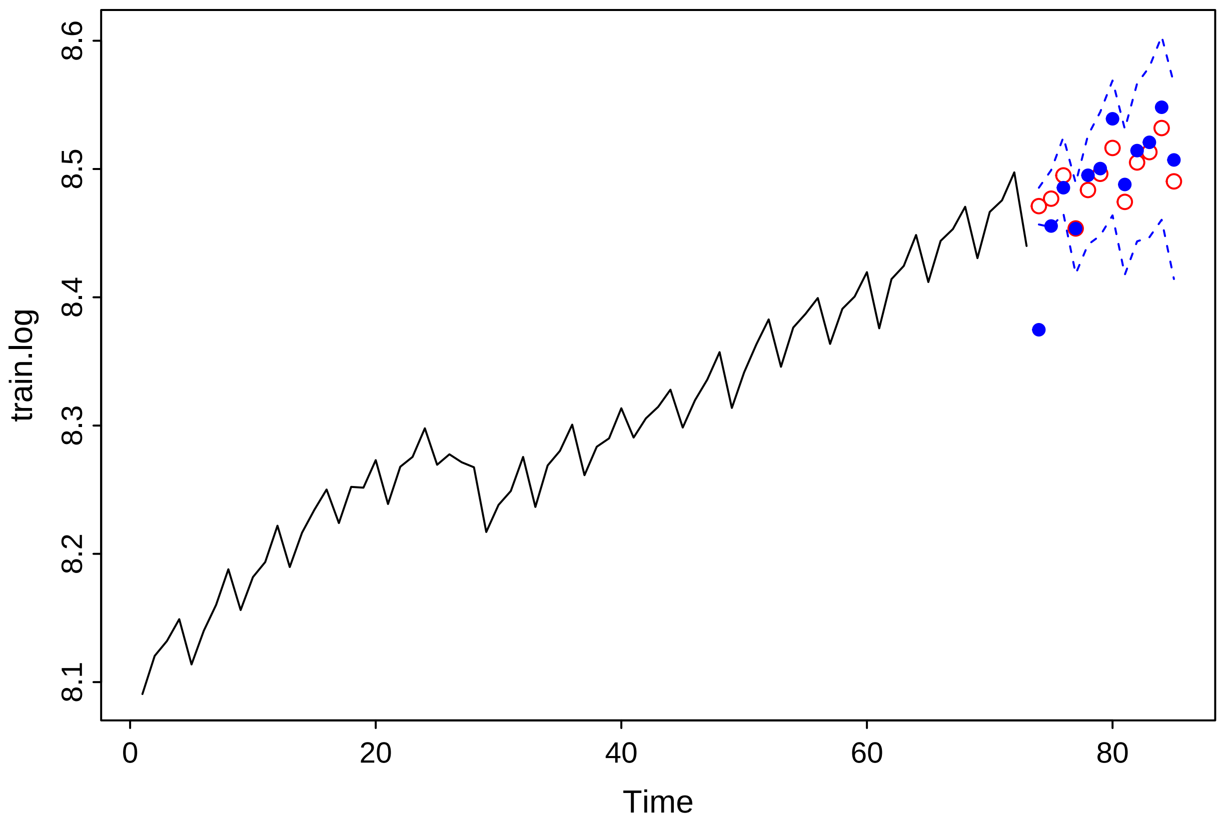
```
## Registered S3 methods overwritten by 'forecast':
##   method                 from
##   autoplot.Arima         ggfortify
##   autoplot.acf           ggfortify
##   autoplot.ar            ggfortify
##   autoplot.bats          ggfortify
##   autoplot.decomposed.ts ggfortify
##   autoplot.ets           ggfortify
##   autoplot.forecast      ggfortify
##   autoplot.stl           ggfortify
##   autoplot.ts            ggfortify
```

```
##   fitted.ar              ggfortify
##   fortify.ts             ggfortify
##   residuals.ar           ggfortify

##
## Attaching package: 'forecast'

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

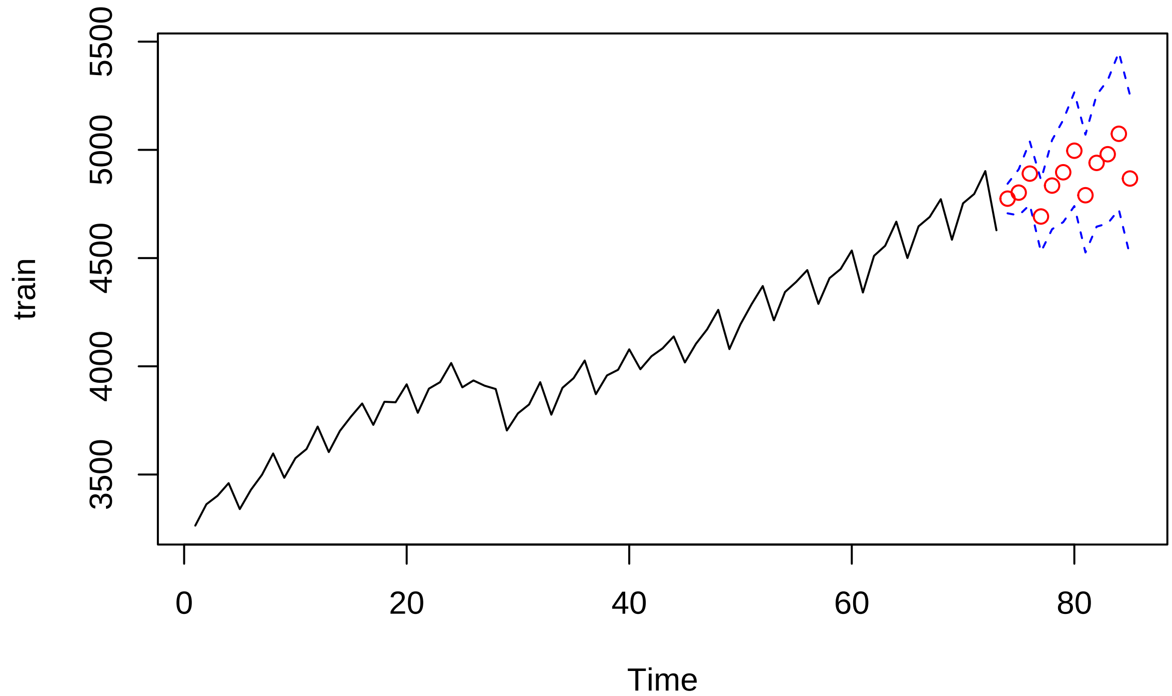
```
pred.tr <- sarima.for(xdata=train.log, n.ahead = 12, p=0, d=1, q=4, P=1, D=1, Q=1, S=4)
```



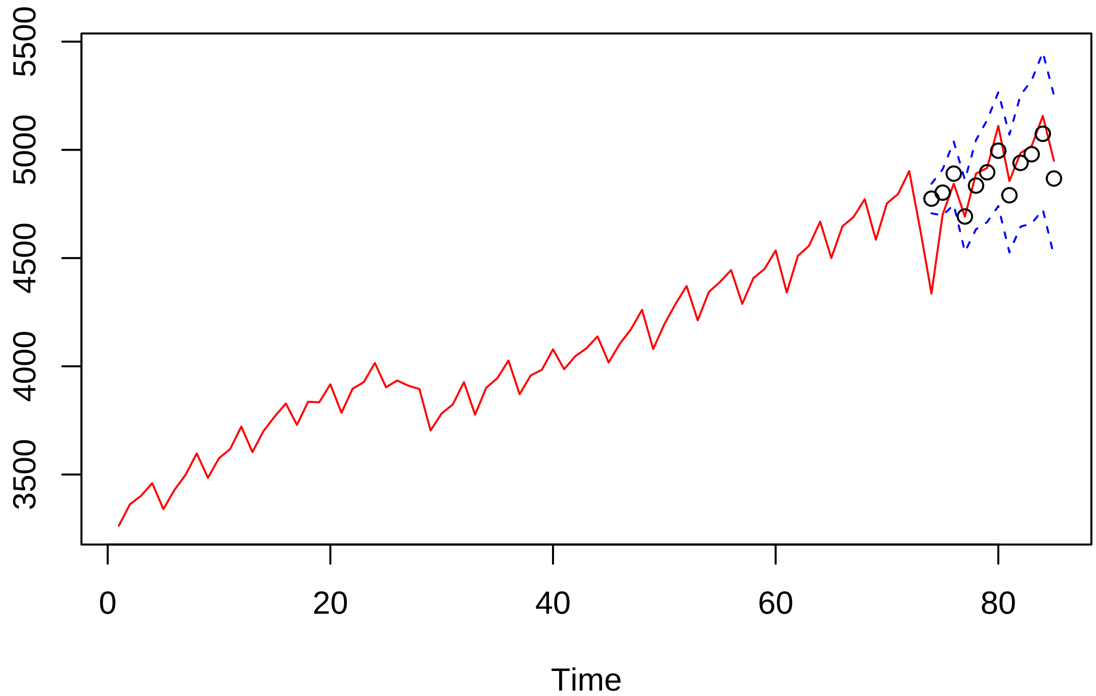

```
U.tr= pred.tr$pred + 2*pred.tr$se
L.tr= pred.tr$pred - 2*pred.tr$se
ts.plot(train.log, xlim=c(1,length(train.log)+12), ylim = c(min(train.log),max(U.tr)))
lines(U.tr, col="blue", lty="dashed")
lines(L.tr, col="blue", lty="dashed")
points((length(train.log)+1):(length(train.log)+12), pred.tr$pred, col="red")
points(length(train) + 1:length(test), log(test), col = "blue", pch = 19, cex = 0.8)
```



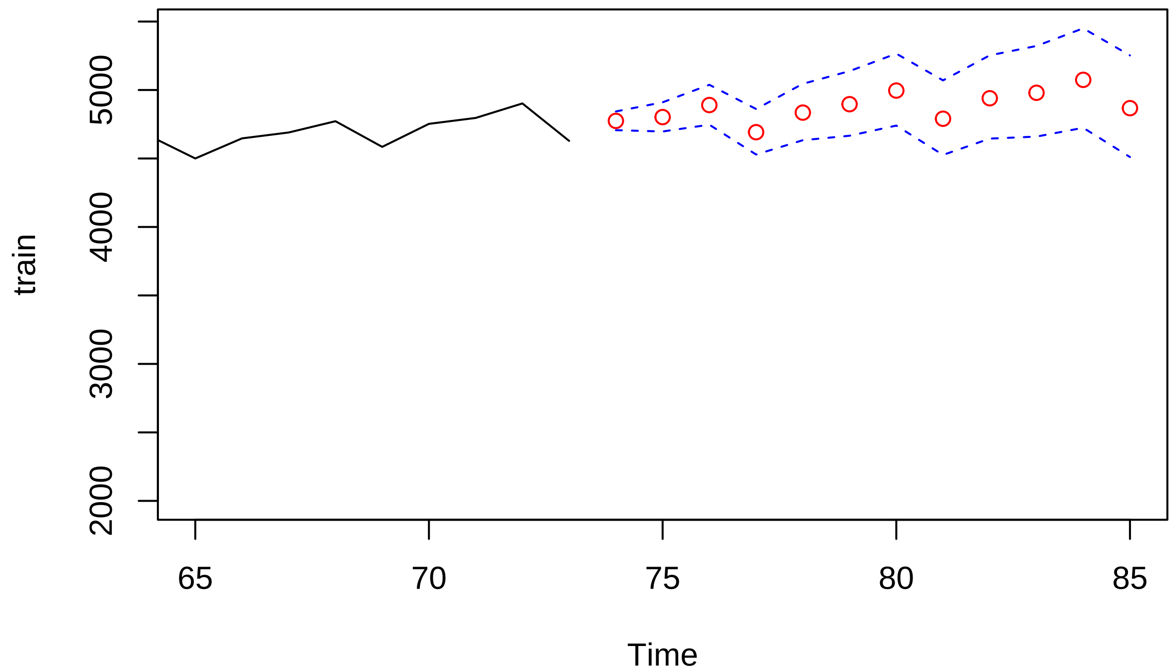

```
pred.orig <- exp(pred.tr$pred)
U= exp(U.tr)
L= exp(L.tr)
ts.plot(train, xlim=c(1,length(train)+12), ylim = c(min(train),max(U)))
lines(U, col="blue", lty="dashed")
lines(L, col="blue", lty="dashed")
points((length(train)+1):(length(train)+12), pred.orig, col="red")
```



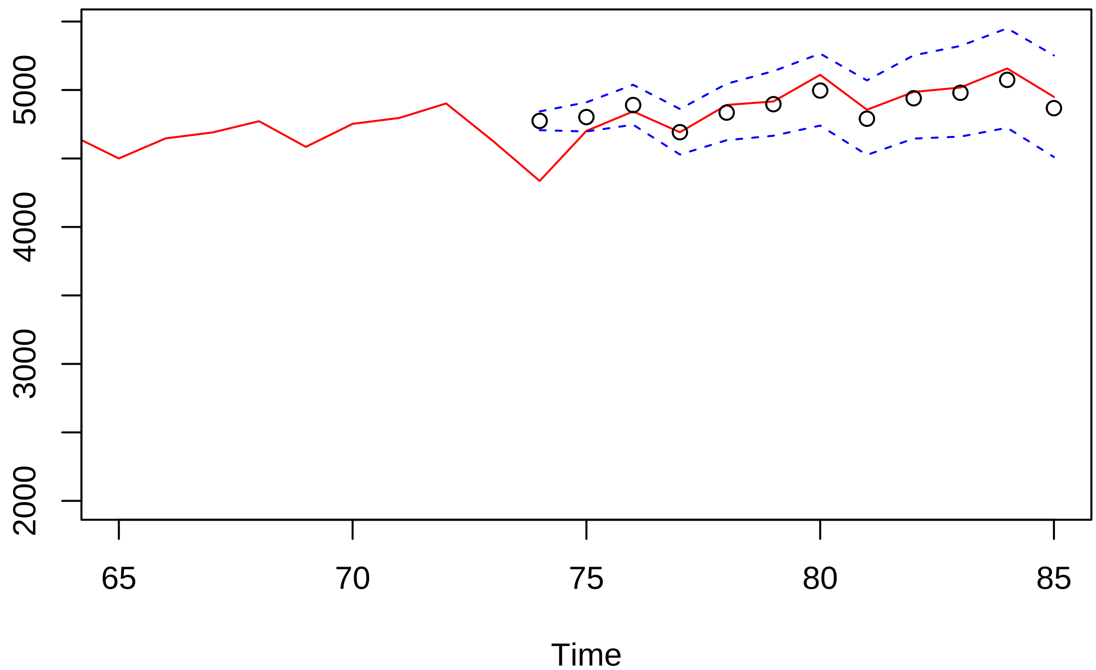

```
ts.plot(gdp.csv, xlim=c(1,length(train)+12), ylim = c(min(train),max(U)), col="red")
lines(U, col="blue", lty="dashed")
lines(L, col="blue", lty="dashed")
points((length(train)+1):(length(train)+12), pred.orig, col="black")
```




```
ts.plot(train, xlim = c(65,length(train)+12), ylim = c(2000,max(U)))
lines(U, col="blue", lty="dashed")
lines(L, col="blue", lty="dashed")
points((length(train)+1):(length(train)+12), pred.orig, col="red")
```




```
ts.plot(gdp.csv, xlim = c(65,length(train)+12), ylim = c(2000,max(U)), col="red")
lines(U, col="blue", lty="dashed")
lines(L, col="blue", lty="dashed")
points((length(train)+1):(length(train)+12), pred.orig, col="black")
```


5000
4000
3000
2000
65
70
75
80
85
Time