# Forecasting Australian Beer Production using Time Series

*Ruben Solis* *

*12 December 2019*

## Contents



*rsolis5@miners.utep.edu

**Abstract**

This project will invovle analyzing a time series data set. In particular we will analyze Australian beer production volume from 1956(1) to 1994(4). Furthermore, investigate stationarity, spectral density estimation, ACF/PACF, and finally forecasting to predict how the sales will grow/decline in the next 5 years.

**Keywords:** arima, forecasting, spectral density, stationary, time series, trend.

# 1 Introduction

Trying to predict the future is a tall task for anyone. As statisticians we have tools that, although may not predict the future to the dot, can give us an insight to what to expect with some certainty. As is often the case with a time series data sets, linear regression models are often unsatisfactory for explaining all of the dynamics of a time series. Therefore, as a result, we will use an ARIMA model to help us analyze this data and forecast.

## 1.1 Objective

The primary objective of this project is to forecast Australian beer production using the data set and see whether sales will continue to increase or decrease. We will analyze the data set and then build a ARIMA model to forecast the beer production.Along the way we will also look at the stationarity, ACF/PACF and the spectral density function.

# 2 Dataset

The data set is available in multiple places on the web e.g., github. It is also available in the`fpp` package. For this project, however, we will be working the data in the `monthly-beer-production-in-austr` file instead and not the one in the package. The reason being is that the package `fpp`, masks many of the functions in `astsa` masked, which is the primary package used in this project to perform the analysis. For that reason, we will work with the text file and then convert it to a time series data set. The code used to accomplish this task is given in the appendix.

# 3 Methodology

As with any time series it is important to plot the data. Moreover, it also may a good idea to aggregate monthly production volume into quarterly and yearly volume. Depending on the business questions we try to answer, different time scales can be very useful. In this project we will consider the analysis of the quarterly volume. After plotting the data we will see

if differencing or taking a transformation is appropriate depending on how the data looks. Afterwards, we will look at the ACF and PACF to see what kind of ARIMA model would be appropriate for this data. We will also plot the spectral density to see what kind of AR model it recommends. The coding used to accomplish many of these tasks is given in the Appendix.

# 4 Analysis and Results

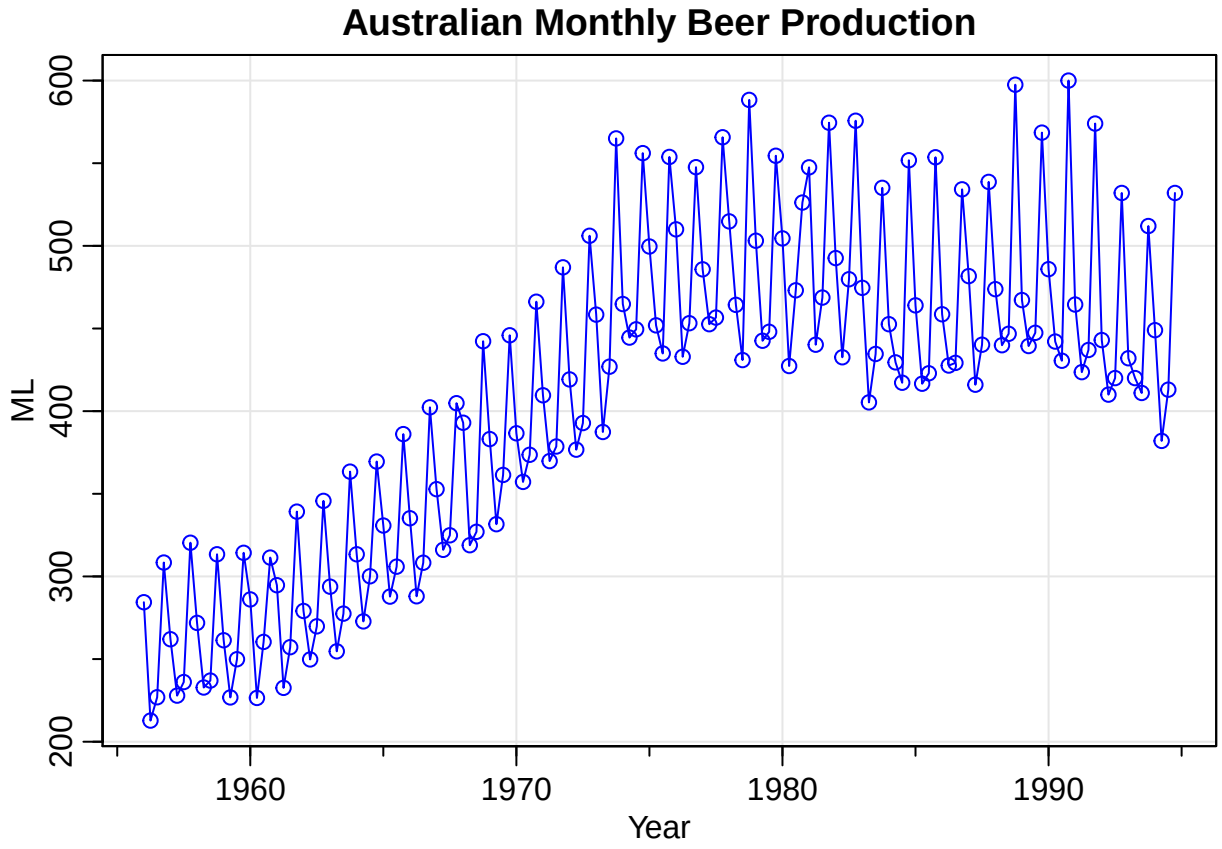
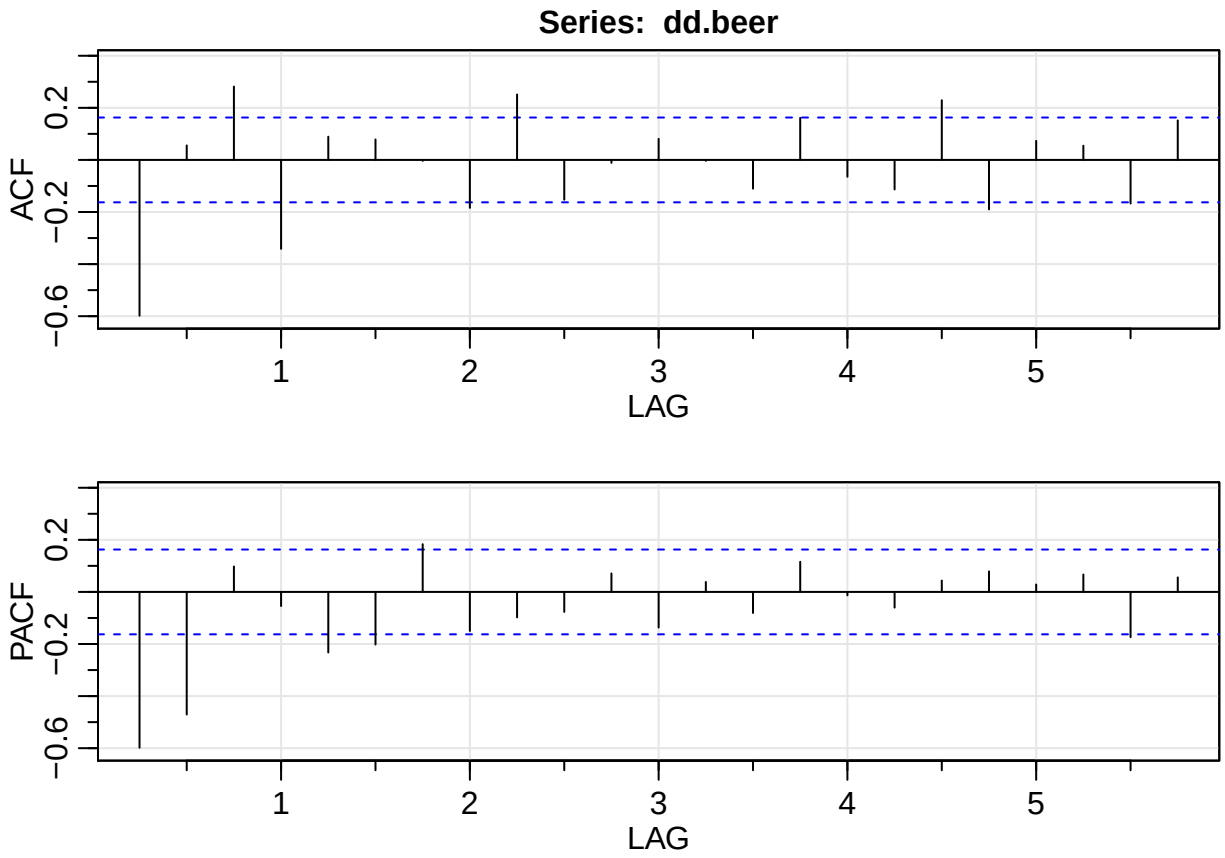



Figure one shows a plot of the data, say, $y_t$. As we can see there was a strong growth from 1950 to 1975 then the production was slowing declining with higher volatility. We can also see very strong seasonality which is obvious for the product like beer. We can also see that there is some trend that may obscure other effects, so it may be in our interest to take the difference of a log, say, $x_t = \nabla \log(y_t)$. In time series analysis, by applying log transformation, it will help us stabilize the variance. Moreover, the first differencing will help remove the trend. After first differences, we can still see some seasonality affect unfortunately. Therefore, we perform another differencing with lag 12. The second differencing now removed the seasonality. The time series now should be now stationary. ACF charts confirms our results.




```
##          ACF  PACF
##  [1,] -0.60 -0.60
##  [2,]  0.06 -0.47
##  [3,]  0.28  0.10
##  [4,] -0.34 -0.05
##  [5,]  0.09 -0.23
##  [6,]  0.08 -0.20
##  [7,]  0.00  0.18
##  [8,] -0.18 -0.15
##  [9,]  0.25 -0.10
## [10,] -0.15 -0.08
## [11,] -0.01  0.07
## [12,]  0.08 -0.14
## [13,]  0.00  0.04
## [14,] -0.11 -0.08
## [15,]  0.16  0.12
## [16,] -0.06 -0.01
## [17,] -0.11 -0.06
## [18,]  0.23  0.04
## [19,] -0.19  0.08
## [20,]  0.07  0.03
```

```
## [21,]  0.05  0.07
## [22,] -0.17 -0.17
## [23,]  0.15  0.06
```

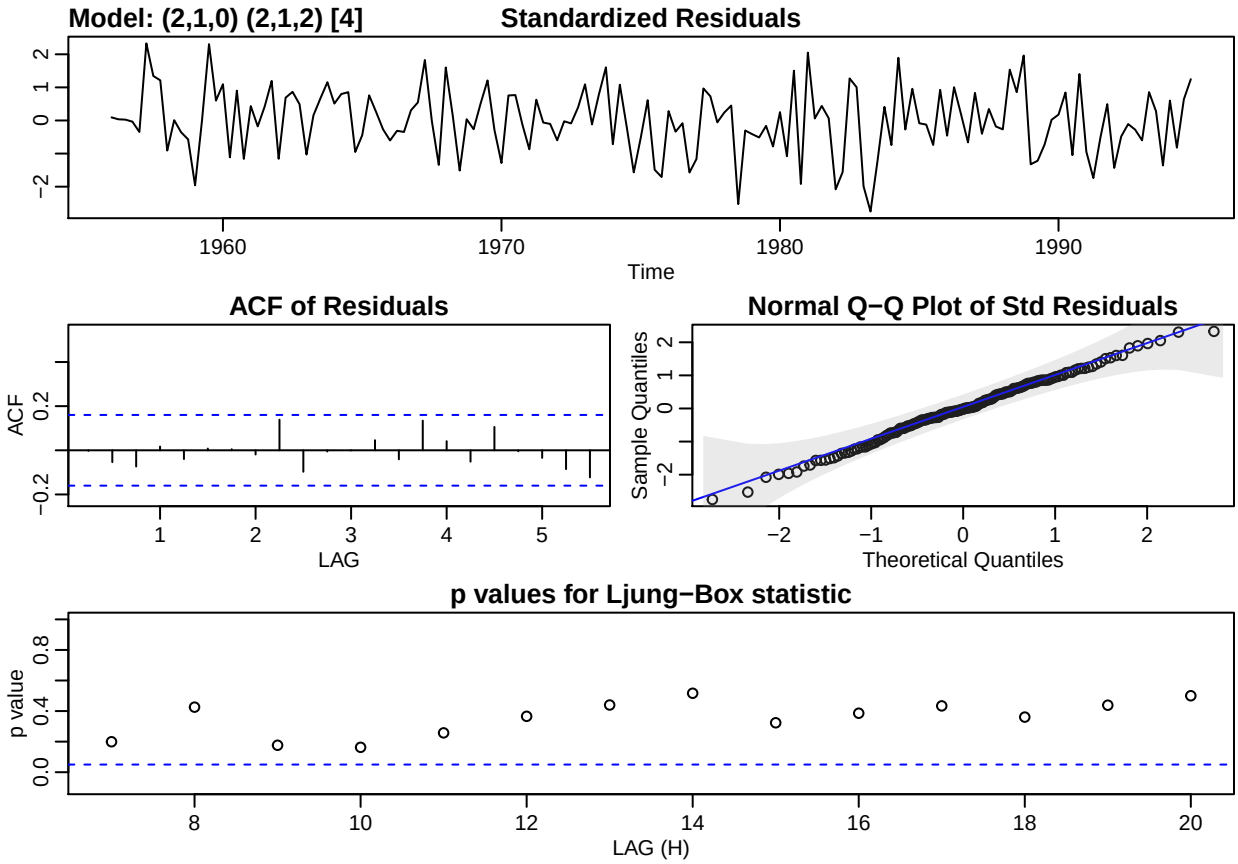
By analyzing the ACF and PACF charts after the differencing, we can try to fit the model by choosing $p$, $d$, $q$, $P$, $D$, $Q$, and $s$ parameters in ARIMA$(p,d,q) \times (P,D,Q)_s$ function. We can now pick the best $p$, $d$, $q$, $P$, $D$, $Q$, and $s$ parameters from the ACF and PACF tables presented in the textbook.

- **Seasonal:** It appears that at the seasons, the ACF and the PACF appear to tail off at lag 2 which would suggest a seasonal SARMA(2,2), $P = 2$, $Q = 2$.
- **Non-seasonal:** Inspecting the sample ACF and PACF at the first few lags, it appears as though the ACF tails off at lag 2 and the PACF cuts off at lag 2. This suggest an AR(2) within the seasons, $p = 2$,and $q = 0$.

Now we will fit an ARIMA$(2,1,0) \times (2,1,2)_4$ model and inspect the diagnostics. fitted sarima model here

```
## initial  value -2.769149
## iter   2 value -3.236267
## iter   3 value -3.278656
## iter   4 value -3.351355
## iter   5 value -3.360188
## iter   6 value -3.363008
## iter   7 value -3.364298
## iter   8 value -3.364630
## iter   9 value -3.364781
## iter  10 value -3.364976
## iter  11 value -3.365238
## iter  12 value -3.365458
## iter  13 value -3.365538
## iter  14 value -3.365551
## iter  15 value -3.365551
## iter  16 value -3.365551
## iter  16 value -3.365551
## iter  16 value -3.365551
## final  value -3.365551
## converged
## initial  value -3.313568
## iter   2 value -3.314306
## iter   3 value -3.315213
## iter   4 value -3.315255
## iter   5 value -3.315292
## iter   6 value -3.315329
## iter   7 value -3.315350
## iter   8 value -3.315489
```

```
## iter   9 value -3.315712
## iter  10 value -3.316074
## iter  11 value -3.317461
## iter  12 value -3.317660
## iter  13 value -3.317874
## iter  14 value -3.318035
## iter  15 value -3.318280
## iter  16 value -3.319794
## iter  17 value -3.321195
## iter  18 value -3.323621
## iter  19 value -3.324176
## iter  20 value -3.324506
## iter  21 value -3.324648
## iter  22 value -3.324775
## iter  23 value -3.324817
## iter  24 value -3.324836
## iter  25 value -3.324837
## iter  26 value -3.324837
## iter  26 value -3.324837
## iter  26 value -3.324837
## final  value -3.324837
## converged
```




```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), include.mean = !no.constant, transform.pars = trans, fixed = fix
##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))
##
## Coefficients:
##           ar1      ar2     sar1    sar2    sma1     sma2
##       -0.9110  -0.4395  -0.7453  0.1027  0.2159  -0.7841
## s.e.   0.0779   0.0814   0.1183  0.1220  0.0883   0.0836
##
## sigma^2 estimated as 0.001217:  log likelihood = 287.79,  aic = -561.58
##
## $degrees_of_freedom
## [1] 145
##
## $ttable
##      Estimate     SE  t.value p.value
## ar1   -0.9110 0.0779 -11.6929  0.0000
## ar2   -0.4395 0.0814  -5.3993  0.0000
```

```
## sar1  -0.7453 0.1183  -6.3011  0.0000
## sar2   0.1027 0.1220   0.8414  0.4015
## sma1   0.2159 0.0883   2.4461  0.0156
## sma2  -0.7841 0.0836  -9.3846  0.0000
##
## $AIC
## [1] -3.646633
##
## $AICc
## [1] -3.642922
##
## $BIC
## [1] -3.509484
```

The output from the `sarima` function shows that most of estimates are significant except `sar2`. `sma1` may or may not be significant depending on the level of significance. Inspection of the time plot of the standardized residuals shows no obvious patterns. The ACF of the residuals shows no apparent departure from the model assumptions since they are all within the 95% confidence interval. The normal Q-Q plot of the residuals suggests that the assumption of normality is not unreasonable, however there may be an outlier.

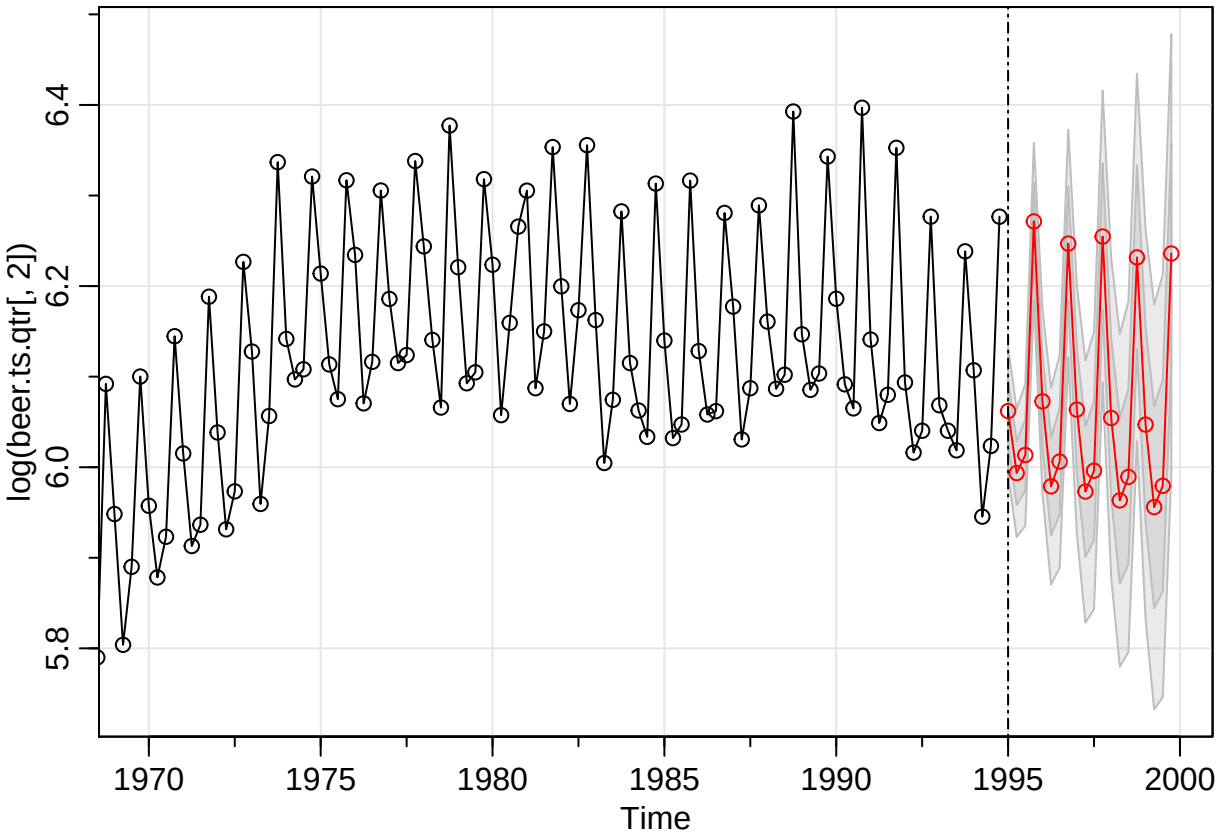
Now we will consider the Q-statistic. The dashed horizontal line on the bottom indicates the .05 level. All of the p-values exceed .05, so we can feel comfortable not rejecting the null hypothesis that the residuals are white.

We can now use this ARIMA model model to forecast.

```
sarima.for(log(beer.ts.qtr[, 2]), n.ahead = 20, 2, 1, 0, 2, 1, 2, 4)
```

```
## $pred
##          Qtr1     Qtr2     Qtr3     Qtr4
## 1995 6.061954 5.993552 6.013512 6.271407
## 1996 6.072721 5.978885 6.006192 6.246907
## 1997 6.063613 5.973136 5.996161 6.254705
## 1998 6.054300 5.963346 5.989293 6.231679
## 1999 6.047064 5.955970 5.979425 6.235940
##
## $se
##            Qtr1       Qtr2       Qtr3       Qtr4
## 1995 0.03521422 0.03535290 0.03917595 0.04330579
## 1996 0.05096980 0.05422742 0.05868545 0.06286605
## 1997 0.06863517 0.07236821 0.07654338 0.08056637
## 1998 0.08811523 0.09186759 0.09670069 0.10143437
## 1999 0.10746435 0.11180983 0.11656098 0.12116494
```

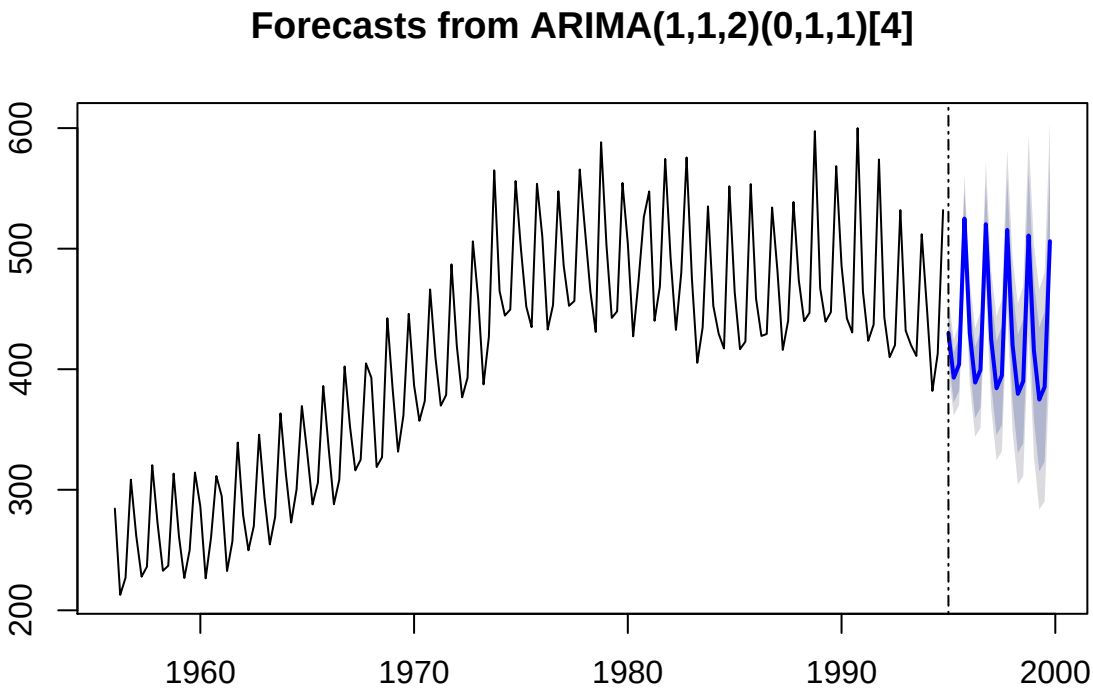
```
abline(v = 1995, lty = 6)
```




Instead of just focusing on one particular model we can take a look at another one. This time, we will use the `auto.arima` function in `R` from the package `fpp` that will select the best estimated ARIMA model for us with the lowest AIC score.

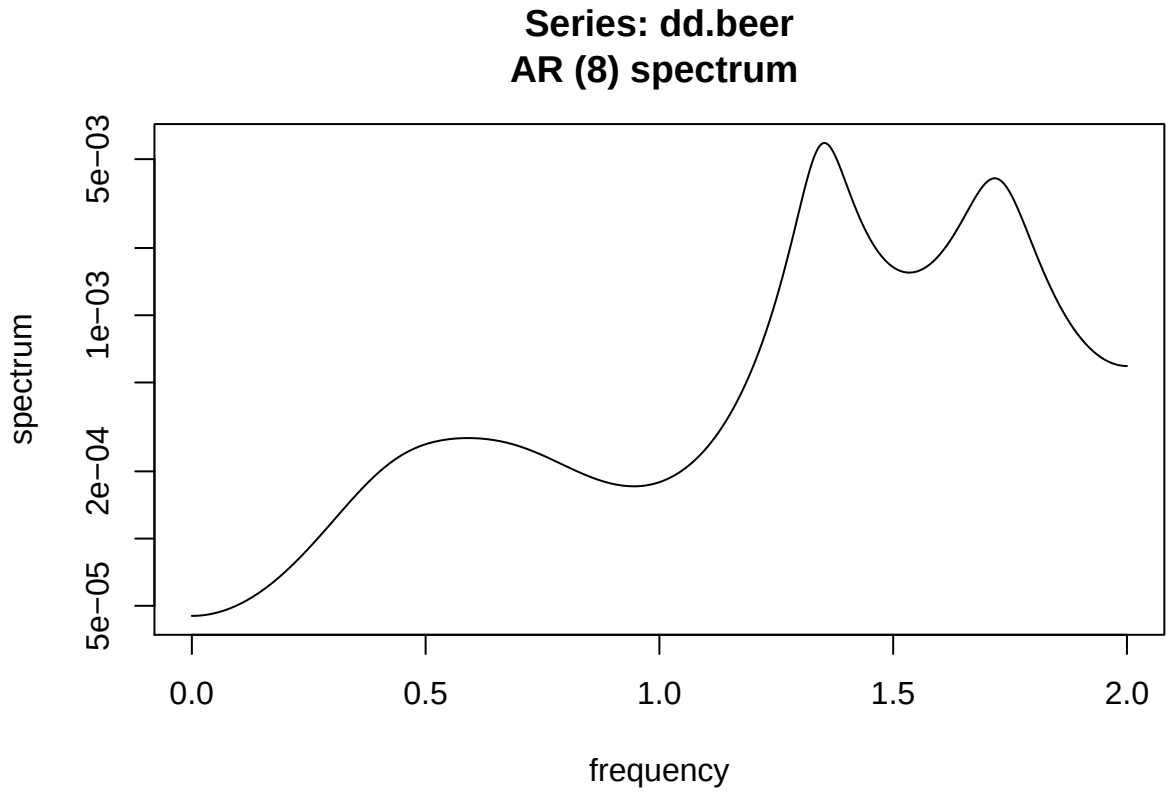
```
## Series: beer.ts.qtr[, 2]
## ARIMA(1,1,2)(0,1,1)[4]
##
## Coefficients:
##          ar1      ma1     ma2     sma1
##       0.1416  -1.1102  0.5077  -0.7807
## s.e.  0.2153   0.1943  0.1520   0.0538
##
## sigma^2 estimated as 256.2:  log likelihood=-633.03
## AIC=1276.06   AICc=1276.47   BIC=1291.15
```

So we now compare these two ARIMA models.




As we can see that they produced very similar results.

Lastly, we calculate the estimated spectral density to see what kind AR it suggest.




The model suggested is a AR(8).

# 5 Conclusion

In this project we analyzed the the Australian beer production volume by using time series techniques. We made the data stationary by differencing and using a log transform. Next, we fit one ARIMA model by inspecting the ACF and PACF. Next we fit another model using the `auto.arima` function in the `fpp` package to compare. We saw that they produced very similar results. Lastly we saw that the estimated spectral density function suggested an AR(8) based off of AIC. In future work I would like to use regression methods to forecast and perhaps using a neural network autoregression to compare forecasts predictions.

# 6 Appendix

Below is the code used to perform the analysis in this project.

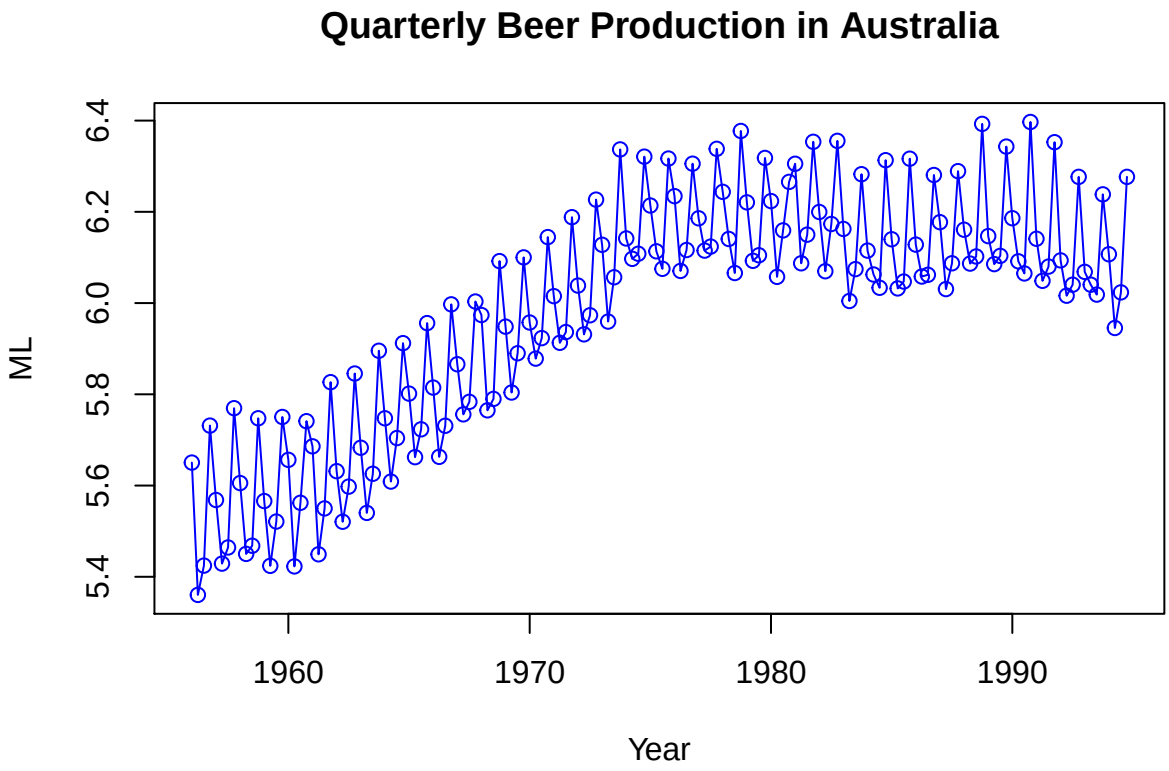
```
library(astsa)
beer <- read.delim("monthly-beer-production-in-austr.txt", header = FALSE, sep = ",")
head(beer)
```

```
##        V1   V2
## 1 1956-01 93.2
## 2 1956-02 96.0
## 3 1956-03 95.2
## 4 1956-04 77.1
## 5 1956-05 70.9
## 6 1956-06 64.8
```

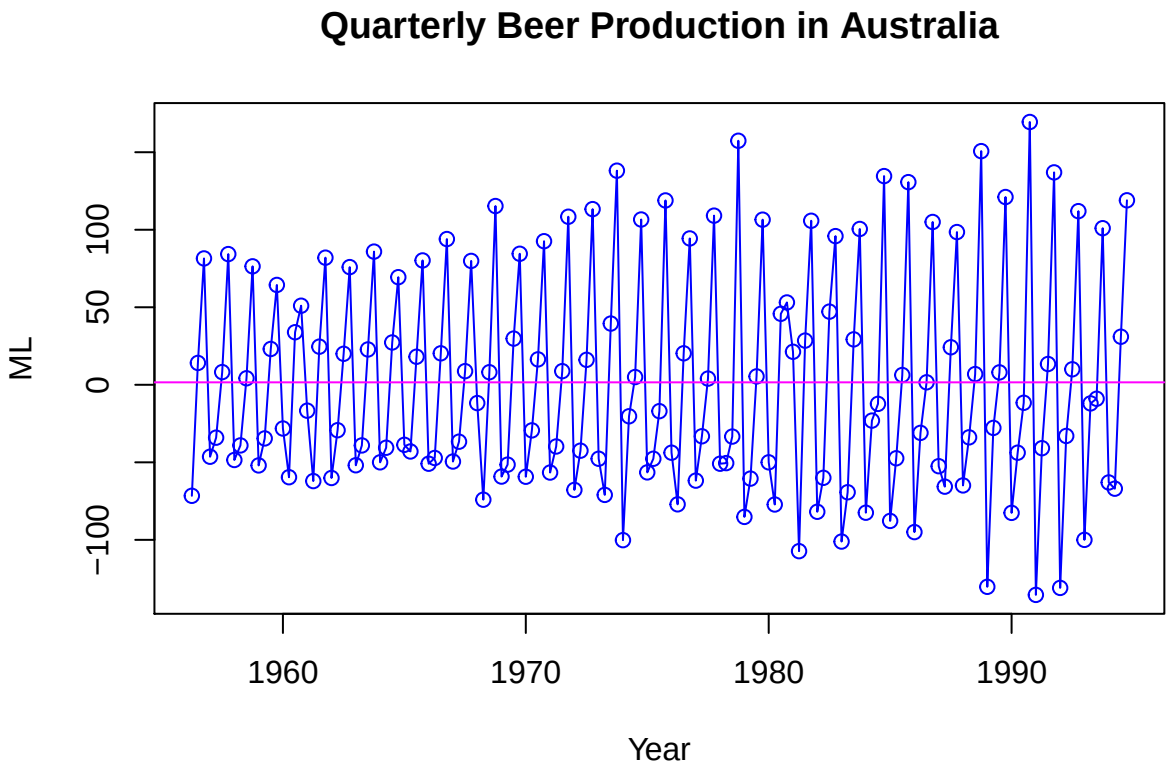
```
tail(beer)
```

```
##          V1  V2
## 471 1995-03 152
## 472 1995-04 127
## 473 1995-05 151
## 474 1995-06 130
## 475 1995-07 119
## 476 1995-08 153
```

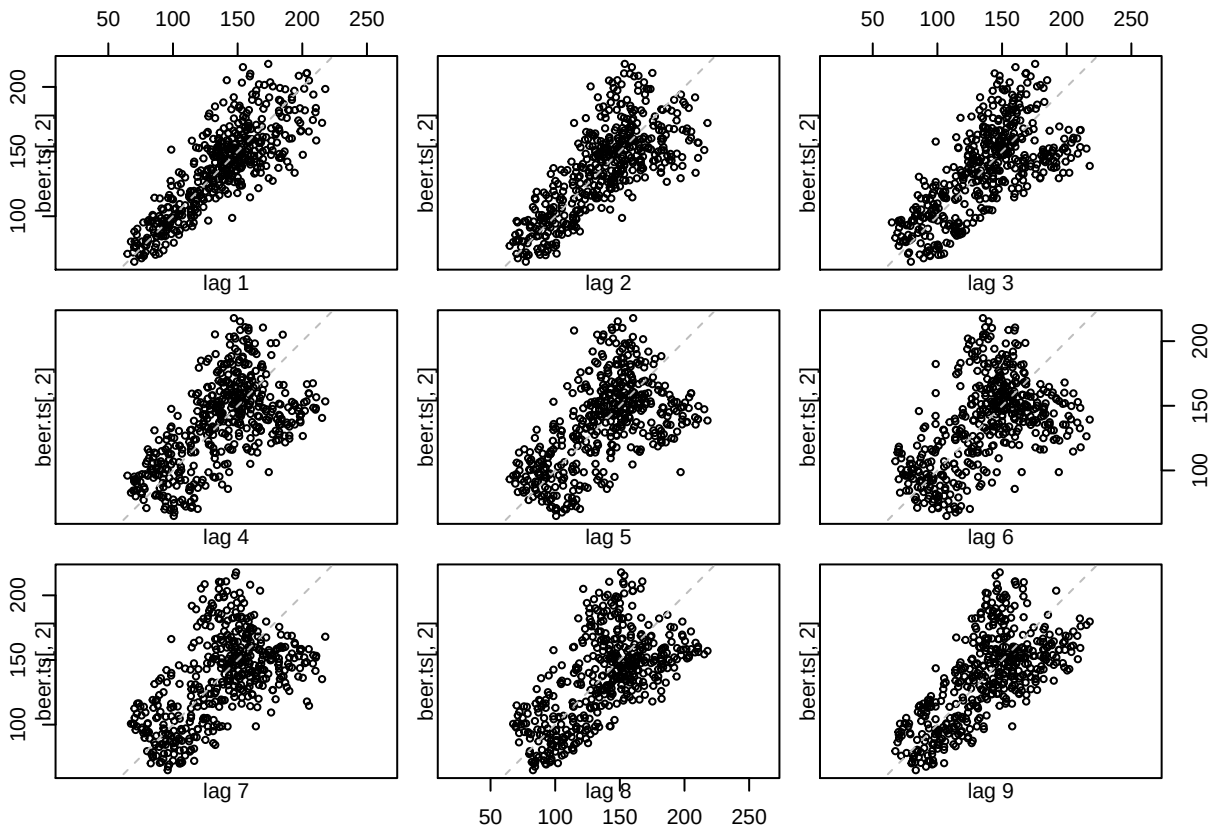
```
# make the data a time series

beer.ts <- ts(beer, frequency = 12, start = c(1956, 1), end = c(1994, 12))
beer.ts.qtr <- aggregate(beer.ts, nfrequency = 4)
beer.ts.yr <- aggregate(beer.ts, nfrequency = 1)
plot.ts(log(beer.ts.qtr[,2]), main = "Quarterly Beer Production in Australia",
        xlab = "Year", ylab = "ML", col = 4, type = "o")
```



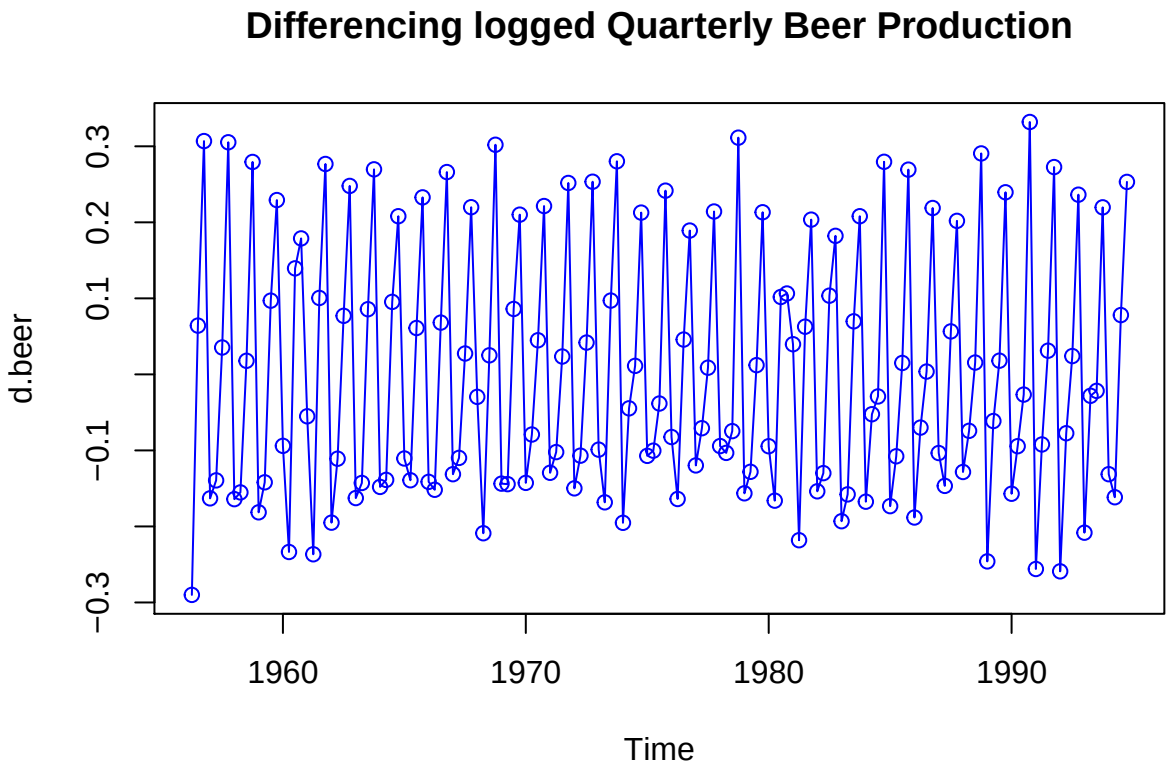

```
plot.ts(diff(beer.ts.qtr[,2]), main = "Quarterly Beer Production in Australia",
        xlab = "Year", ylab = "ML", col = 4, type = "o")
abline(h = mean(diff(beer.ts.qtr[, 2])), col = 6)
```



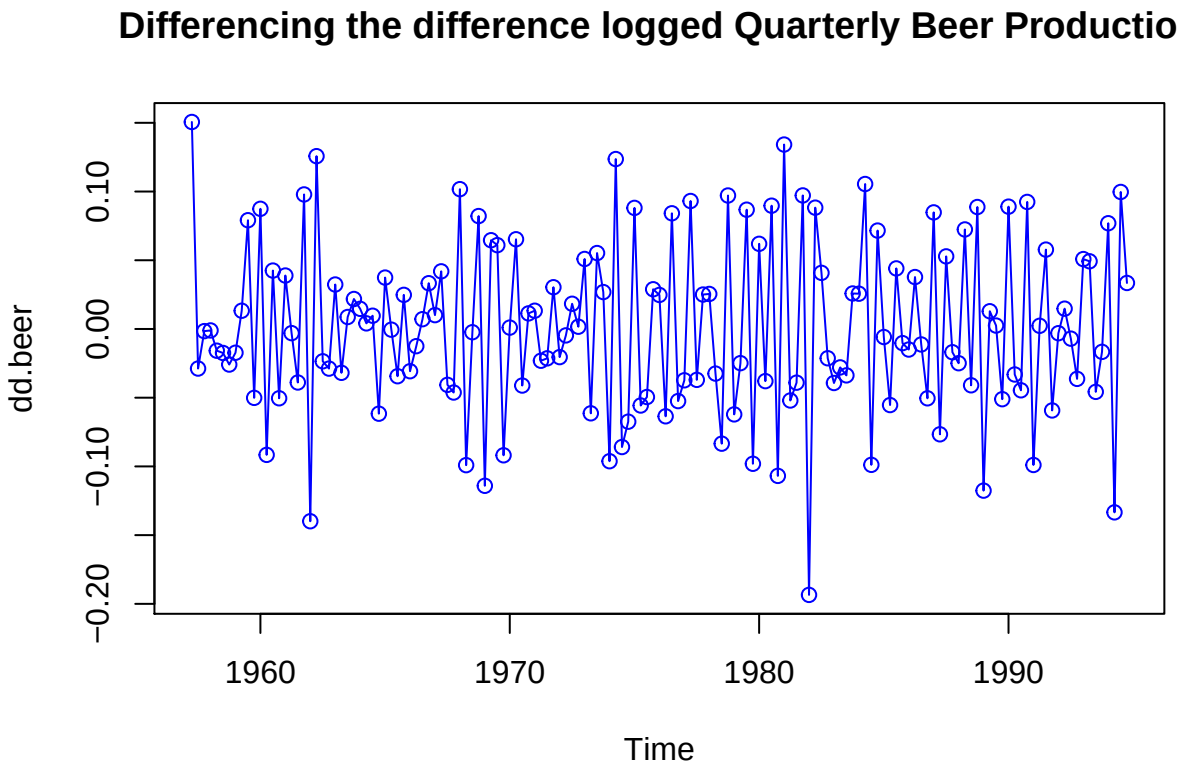

```
# Autocorrelation
lag.plot(beer.ts[,2], lags = 9, do.lines = FALSE )
```



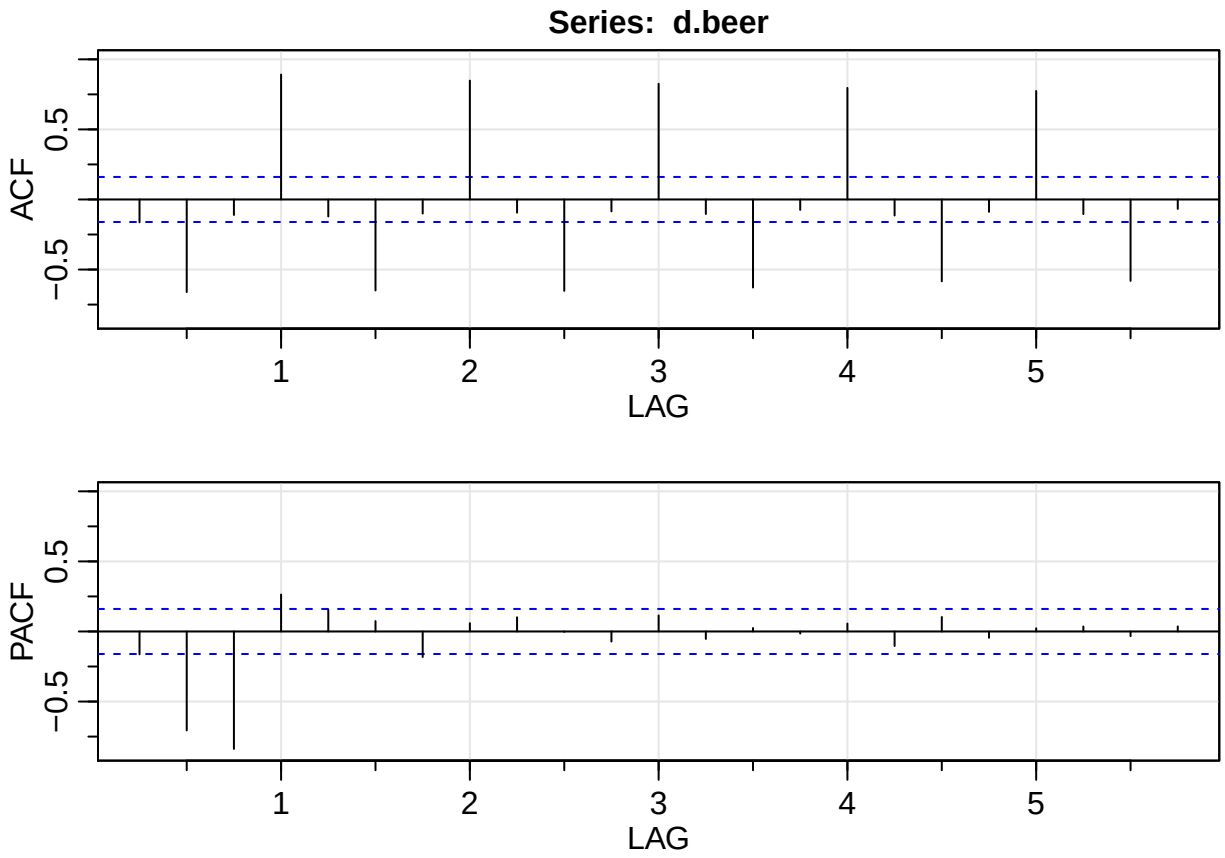

```r
# Differencing
d.beer <- diff(log(beer.ts.qtr[,2]))
plot(d.beer, main = "Differencing logged Quarterly Beer Production",
     type="o", col = 4)
```




```
dd.beer <- diff(d.beer, lag = 4)
plot(dd.beer, main = "Differencing the difference logged Quarterly Beer Production",
     type="o", col=4)
```



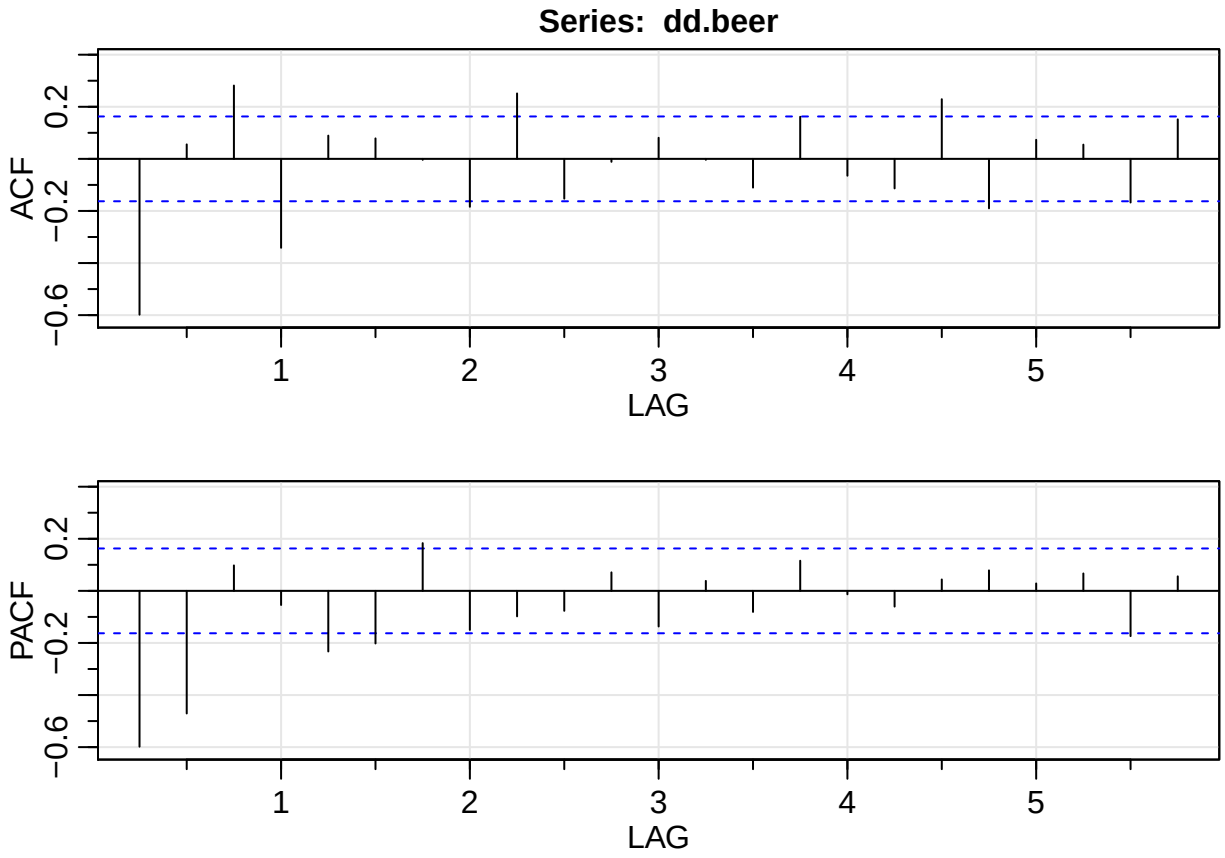

```
# acf
acf2(d.beer)
```




```
##          ACF  PACF
##  [1,] -0.16 -0.16
##  [2,] -0.66 -0.71
##  [3,] -0.11 -0.84
##  [4,]  0.89  0.26
##  [5,] -0.12  0.16
##  [6,] -0.65  0.07
##  [7,] -0.10 -0.18
##  [8,]  0.85  0.06
##  [9,] -0.09  0.10
## [10,] -0.65  0.00
## [11,] -0.08 -0.07
## [12,]  0.83  0.11
## [13,] -0.10 -0.05
## [14,] -0.63  0.02
## [15,] -0.07 -0.01
## [16,]  0.80  0.06
## [17,] -0.11 -0.10
## [18,] -0.58  0.10
## [19,] -0.09 -0.05
## [20,]  0.77  0.02
```

```
## [21,] -0.10  0.04
## [22,] -0.58 -0.03
## [23,] -0.07  0.04
```

```
acf2(dd.beer)
```



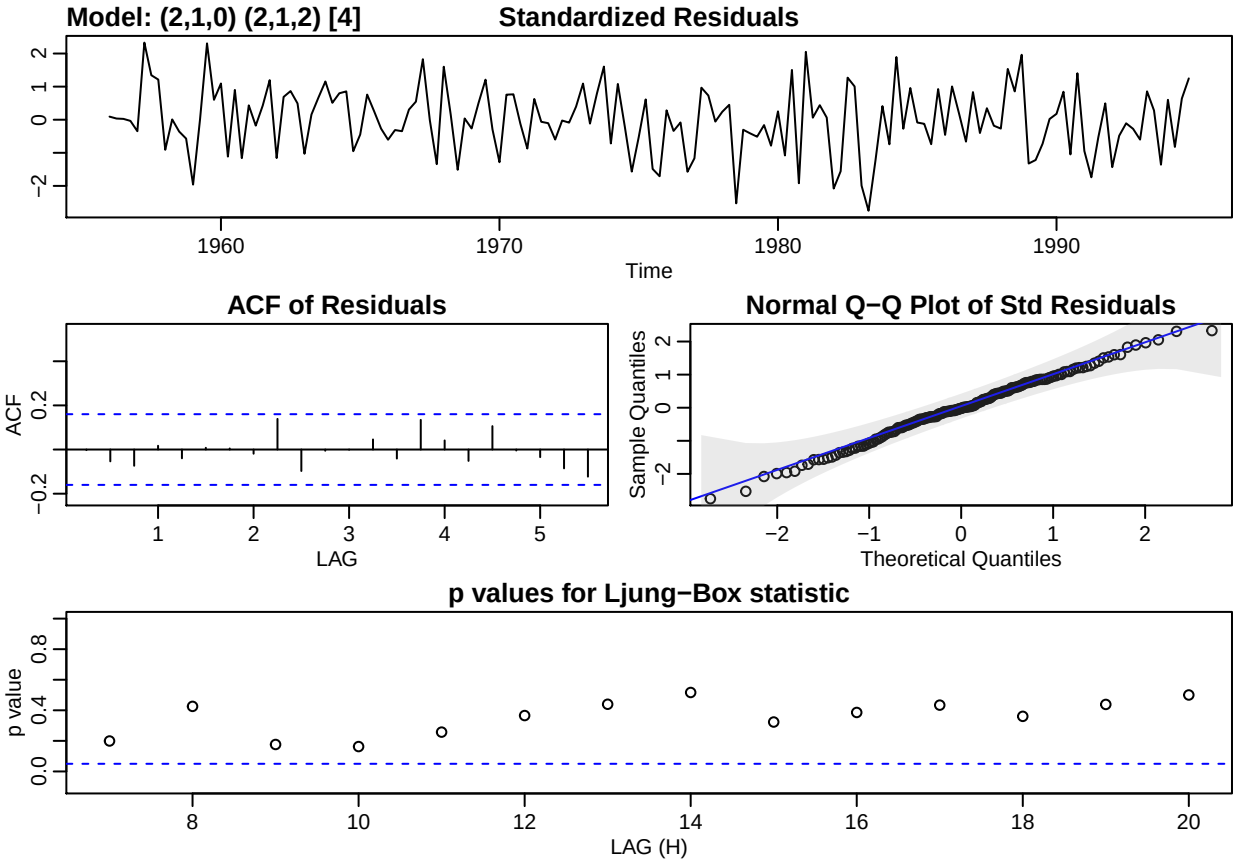

```
##          ACF  PACF
##  [1,] -0.60 -0.60
##  [2,]  0.06 -0.47
##  [3,]  0.28  0.10
##  [4,] -0.34 -0.05
##  [5,]  0.09 -0.23
##  [6,]  0.08 -0.20
##  [7,]  0.00  0.18
##  [8,] -0.18 -0.15
##  [9,]  0.25 -0.10
## [10,] -0.15 -0.08
## [11,] -0.01  0.07
## [12,]  0.08 -0.14
## [13,]  0.00  0.04
## [14,] -0.11 -0.08
## [15,]  0.16  0.12
```

```
## [16,] -0.06 -0.01
## [17,] -0.11 -0.06
## [18,]  0.23  0.04
## [19,] -0.19  0.08
## [20,]  0.07  0.03
## [21,]  0.05  0.07
## [22,] -0.17 -0.17
## [23,]  0.15  0.06
```

```r
# ARIMA model
sarima(log(beer.ts.qtr[, 2]), 2, 1, 0, 2, 1, 2, 4)
```

```
## initial  value -2.769149
## iter   2 value -3.236267
## iter   3 value -3.278656
## iter   4 value -3.351355
## iter   5 value -3.360188
## iter   6 value -3.363008
## iter   7 value -3.364298
## iter   8 value -3.364630
## iter   9 value -3.364781
## iter  10 value -3.364976
## iter  11 value -3.365238
## iter  12 value -3.365458
## iter  13 value -3.365538
## iter  14 value -3.365551
## iter  15 value -3.365551
## iter  16 value -3.365551
## iter  16 value -3.365551
## iter  16 value -3.365551
## final  value -3.365551
## converged
## initial  value -3.313568
## iter   2 value -3.314306
## iter   3 value -3.315213
## iter   4 value -3.315255
## iter   5 value -3.315292
## iter   6 value -3.315329
## iter   7 value -3.315350
## iter   8 value -3.315489
## iter   9 value -3.315712
## iter  10 value -3.316074
## iter  11 value -3.317461
## iter  12 value -3.317660
## iter  13 value -3.317874
```

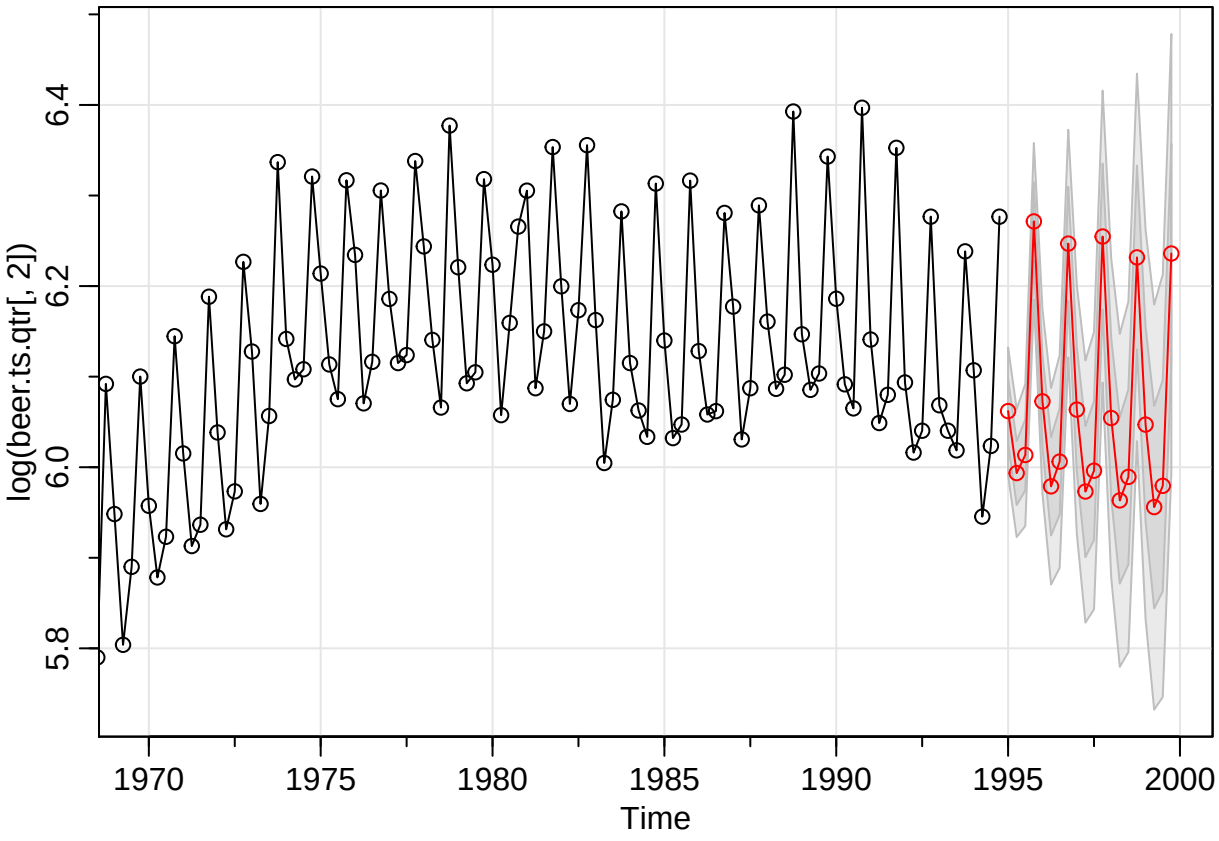
```
## iter  14 value -3.318035
## iter  15 value -3.318280
## iter  16 value -3.319794
## iter  17 value -3.321195
## iter  18 value -3.323621
## iter  19 value -3.324176
## iter  20 value -3.324506
## iter  21 value -3.324648
## iter  22 value -3.324775
## iter  23 value -3.324817
## iter  24 value -3.324836
## iter  25 value -3.324837
## iter  26 value -3.324837
## iter  26 value -3.324837
## iter  26 value -3.324837
## final  value -3.324837
## converged
```




```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
```

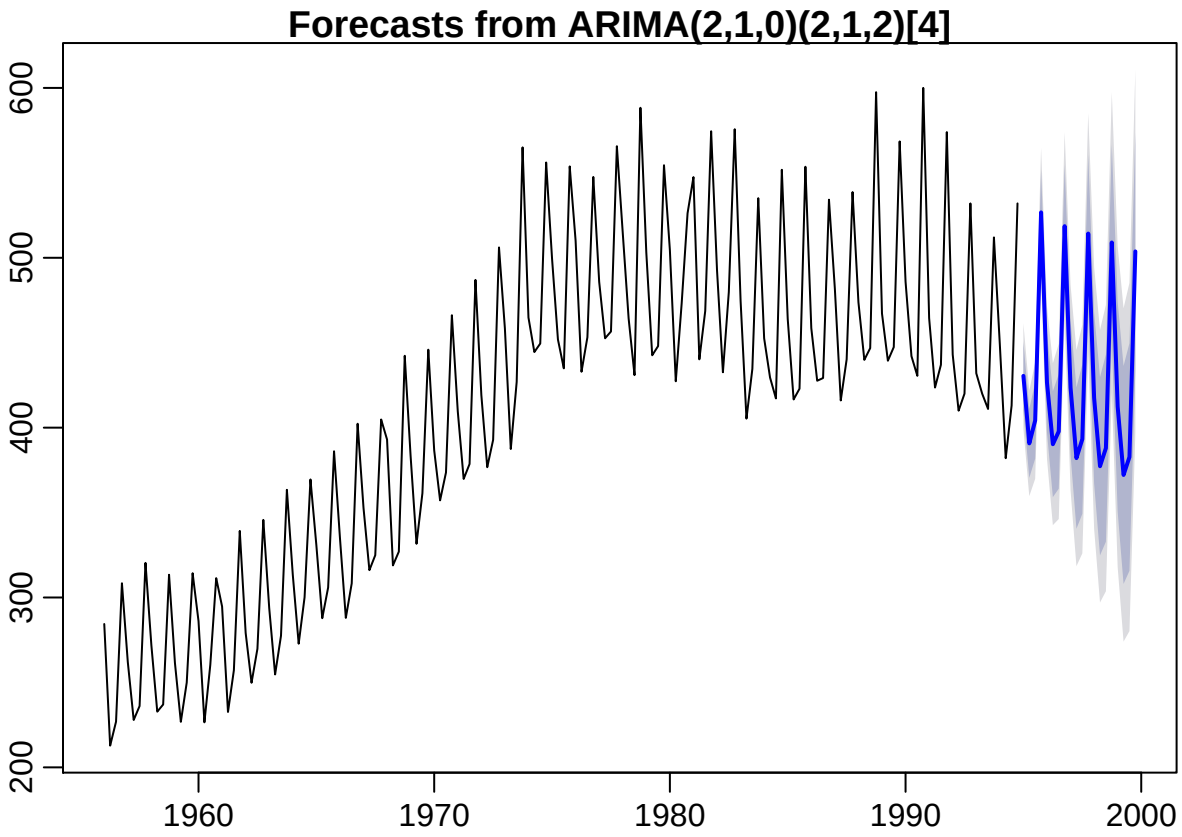
```
##     Q), period = S), include.mean = !no.constant, transform.pars = trans, fixed = fix
##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))
##
## Coefficients:
##           ar1      ar2     sar1    sar2    sma1     sma2
##       -0.9110  -0.4395  -0.7453  0.1027  0.2159  -0.7841
## s.e.   0.0779   0.0814   0.1183  0.1220  0.0883   0.0836
##
## sigma^2 estimated as 0.001217:  log likelihood = 287.79,  aic = -561.58
##
## $degrees_of_freedom
## [1] 145
##
## $ttable
##      Estimate     SE  t.value p.value
## ar1   -0.9110 0.0779 -11.6929  0.0000
## ar2   -0.4395 0.0814  -5.3993  0.0000
## sar1  -0.7453 0.1183  -6.3011  0.0000
## sar2   0.1027 0.1220   0.8414  0.4015
## sma1   0.2159 0.0883   2.4461  0.0156
## sma2  -0.7841 0.0836  -9.3846  0.0000
##
## $AIC
## [1] -3.646633
##
## $AICc
## [1] -3.642922
##
## $BIC
## [1] -3.509484
```

```
sarima.for(log(beer.ts.qtr[, 2]), n.ahead=20, 2, 1, 0, 2, 1, 2, 4)
```




```
## $pred
##          Qtr1     Qtr2     Qtr3     Qtr4
## 1995 6.061954 5.993552 6.013512 6.271407
## 1996 6.072721 5.978885 6.006192 6.246907
## 1997 6.063613 5.973136 5.996161 6.254705
## 1998 6.054300 5.963346 5.989293 6.231679
## 1999 6.047064 5.955970 5.979425 6.235940
##
## $se
##            Qtr1       Qtr2       Qtr3       Qtr4
## 1995 0.03521422 0.03535290 0.03917595 0.04330579
## 1996 0.05096980 0.05422742 0.05868545 0.06286605
## 1997 0.06863517 0.07236821 0.07654338 0.08056637
## 1998 0.08811523 0.09186759 0.09670069 0.10143437
## 1999 0.10746435 0.11180983 0.11656098 0.12116494
```

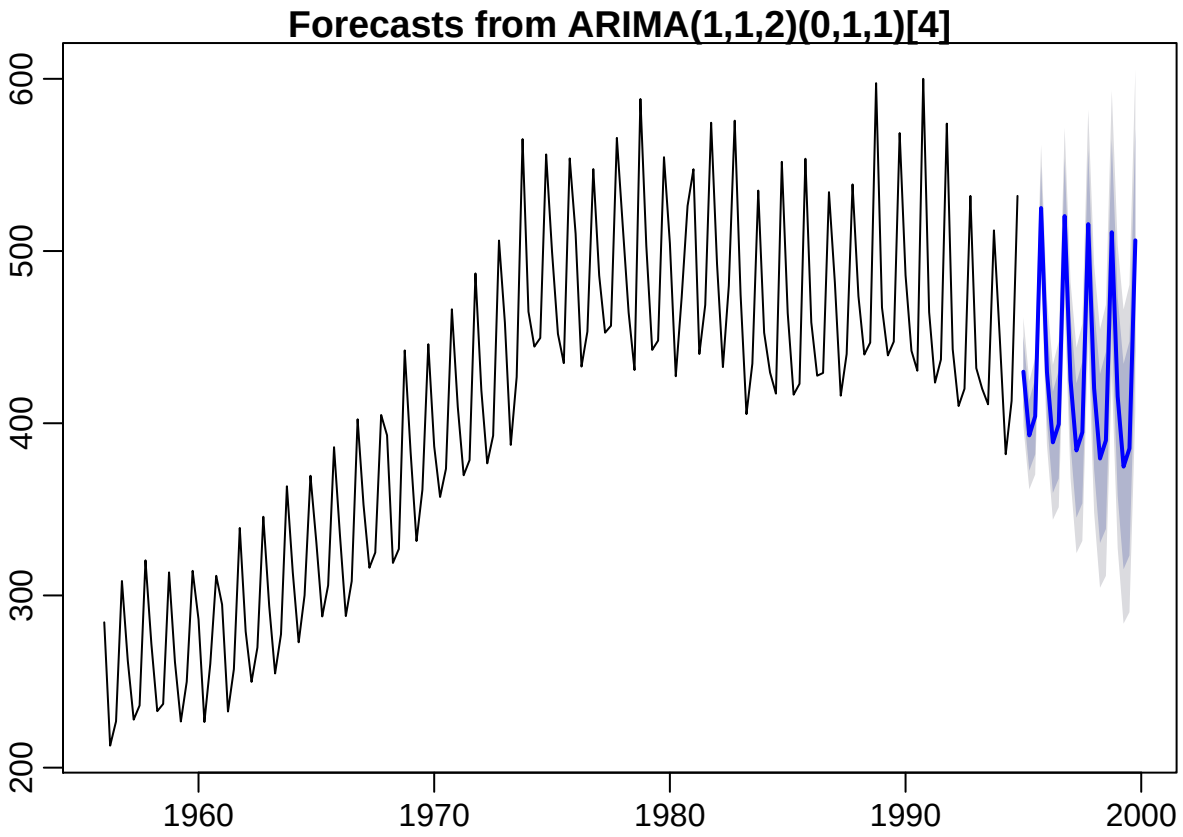
```
# forecasting
beer.fit <- arima(beer.ts.qtr[,2], order = c(2, 1, 0), seasonal = c(2, 1, 2))
fcst.arima <- forecast(beer.fit, h = 20)
plot(fcst.arima)
```



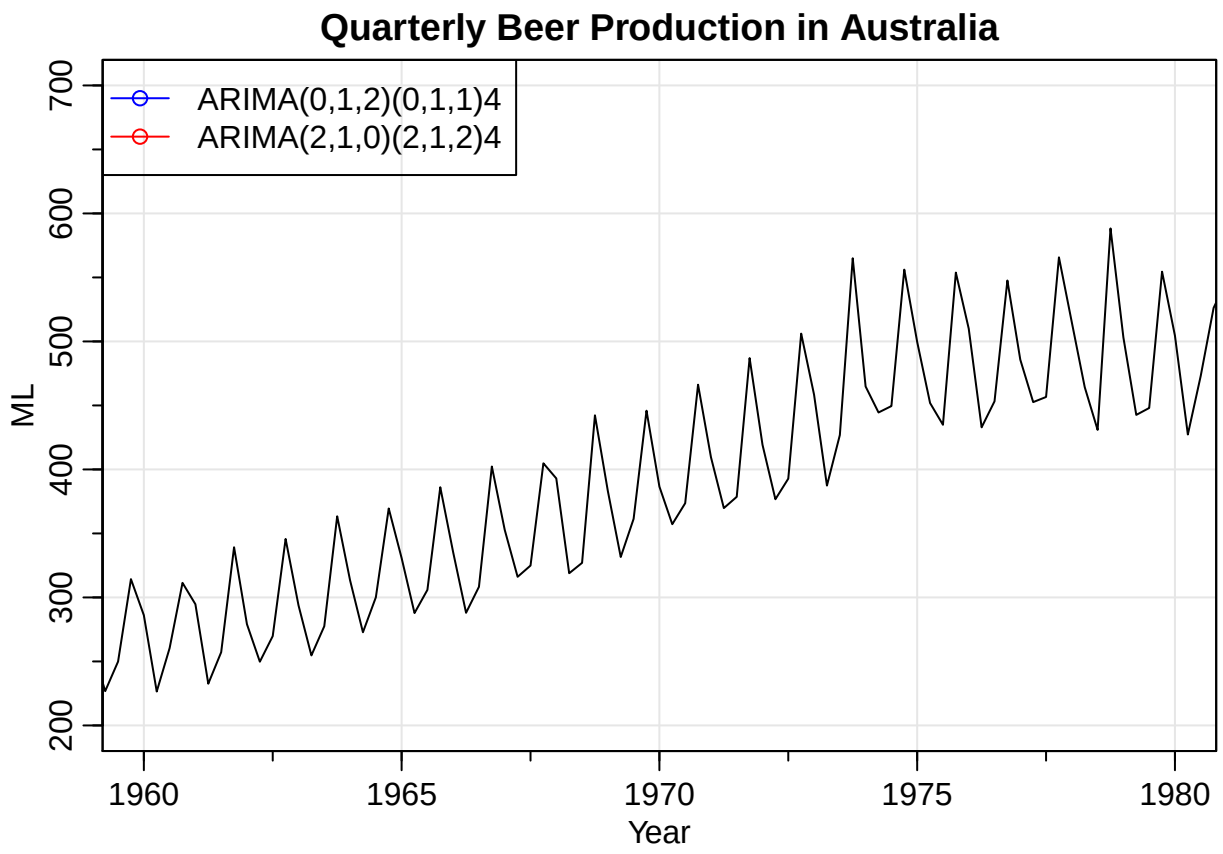

```
# auto arima
beer.fit.arima.auto <- auto.arima(beer.ts.qtr[,2])
beer.fit.arima.auto
```

```
## Series: beer.ts.qtr[, 2]
## ARIMA(1,1,2)(0,1,1)[4]
##
## Coefficients:
##          ar1      ma1     ma2     sma1
##       0.1416  -1.1102  0.5077  -0.7807
## s.e.  0.2153   0.1943  0.1520   0.0538
##
## sigma^2 estimated as 256.2:  log likelihood=-633.03
## AIC=1276.06   AICc=1276.47   BIC=1291.15
```

```
fcst.auto <- forecast(beer.fit.arima.auto, h = 20)
plot(fcst.auto)
```




```
# Compare Models
tsplot(beer.ts.qtr[,2], main = "Quarterly Beer Production in Australia", xlab = "Year",
lines(fcst.auto$mean, col = "blue", type = "o")
lines(fcst.arima$mean, col = "red", type = "o")
legend("topleft", lty = 1, pch = 1, col = c("blue", "red"),
       c("ARIMA(0,1,2)(0,1,1)4", "ARIMA(2,1,0)(2,1,2)4"))
```


Quarterly Beer Production in Australia
ARIMA(0,1,2)(0,1,1)4
ARIMA(2,1,0)(2,1,2)4
ML
200
300
400
500
600
700
1960
1965
1970
1975
1980
Year