Time Series Analysis in R - Australia Beer Production

About the Data: The data is monthly beer production (megalitres) in Australia. It includes ale and stout. Does not include beverages with alcohol percentage less than 1.15. It is a time series data from Jan 1956 to Aug 1995.

Analysis Carried Out:

I have used time series forecasting models in R to analyze Australia beer production data.

Load packages

```
library(tseries)
library(forecast)
library(TSA)
library(fpp)
library(astsa)
library(DT)
library(dygraphs)
```

Load Australia beer production dataset

```
beer <- read.csv("monthly-beer-production-in-austr.csv")
head(beer)
```

```
##     Month Monthly.beer.production.in.Australia
## 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)
```

```
##       Month Monthly.beer.production.in.Australia
## 471 1995-03                                  152
## 472 1995-04                                  127
## 473 1995-05                                  151
```

```
## 474 1995-06                130
## 475 1995-07                119
## 476 1995-08                153
```

Convert data frame into time series object

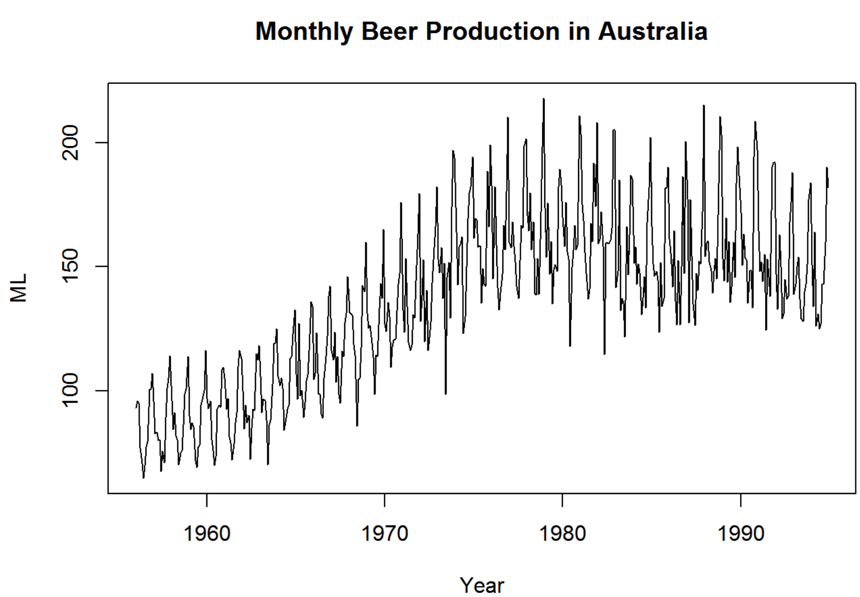
```
beer.ts <- ts(beer, frequency = 12, start = c(1956,1), end = c(1994,12))
```

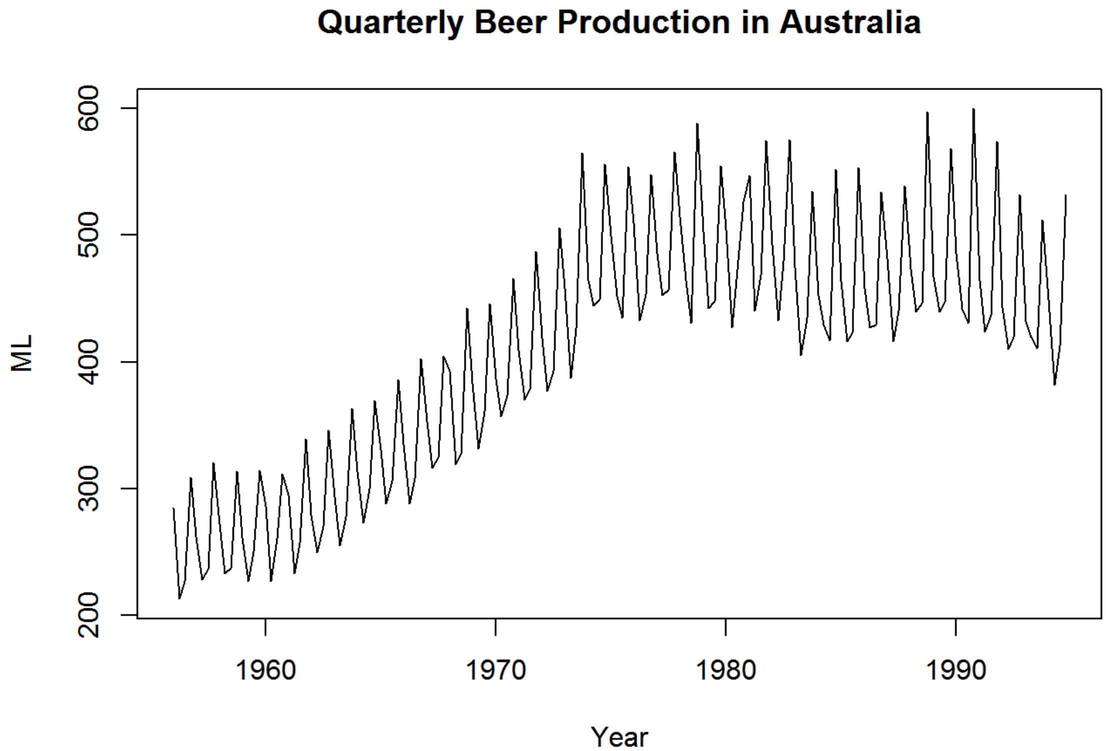
Time Series Plots

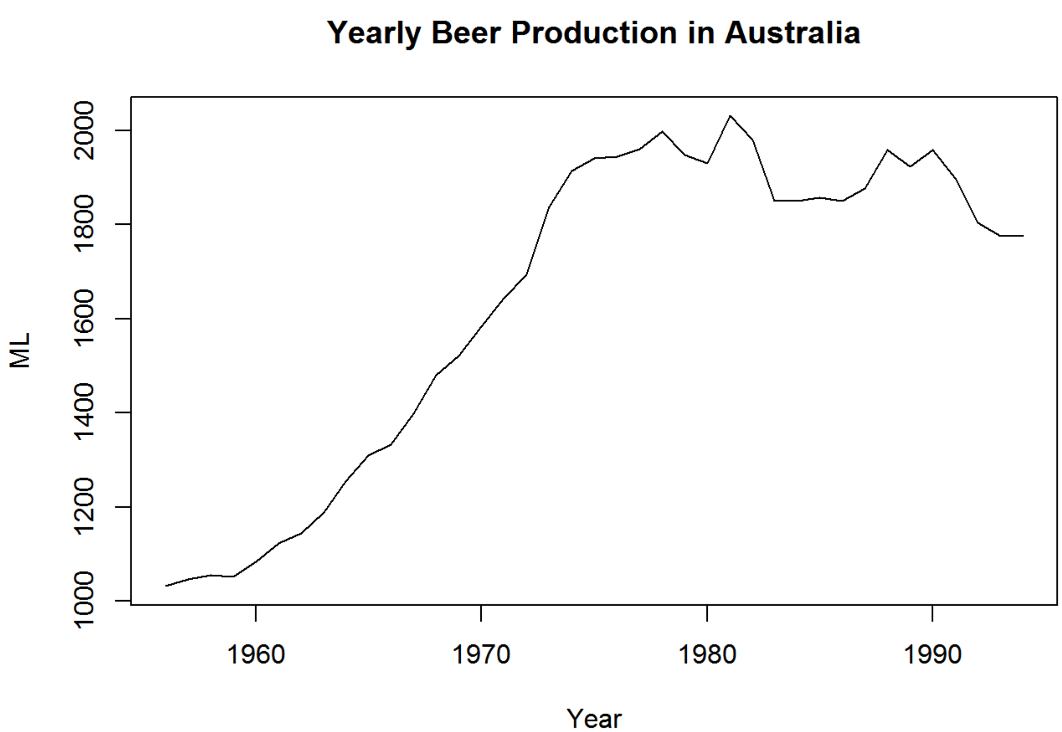
The first thing to do with any time series analysis is to plot the charts.

It is also a good idea to aggregate monthly production volume into quarterly and yearly volume.

```
beer.ts.qtr <- aggregate(beer.ts, nfrequency=4)
beer.ts.yr <- aggregate(beer.ts, nfrequency=1)
plot.ts(beer.ts[,2], main = "Monthly Beer Production in Australia", xlab = "Year", ylab = "ML")
plot.ts(beer.ts.qtr[,2], main = "Quarterly Beer Production in Australia", xlab = "Year", ylab = "M
L")
plot.ts(beer.ts.yr[,2], main = "Yearly Beer Production in Australia", xlab = "Year", ylab = "ML")
```



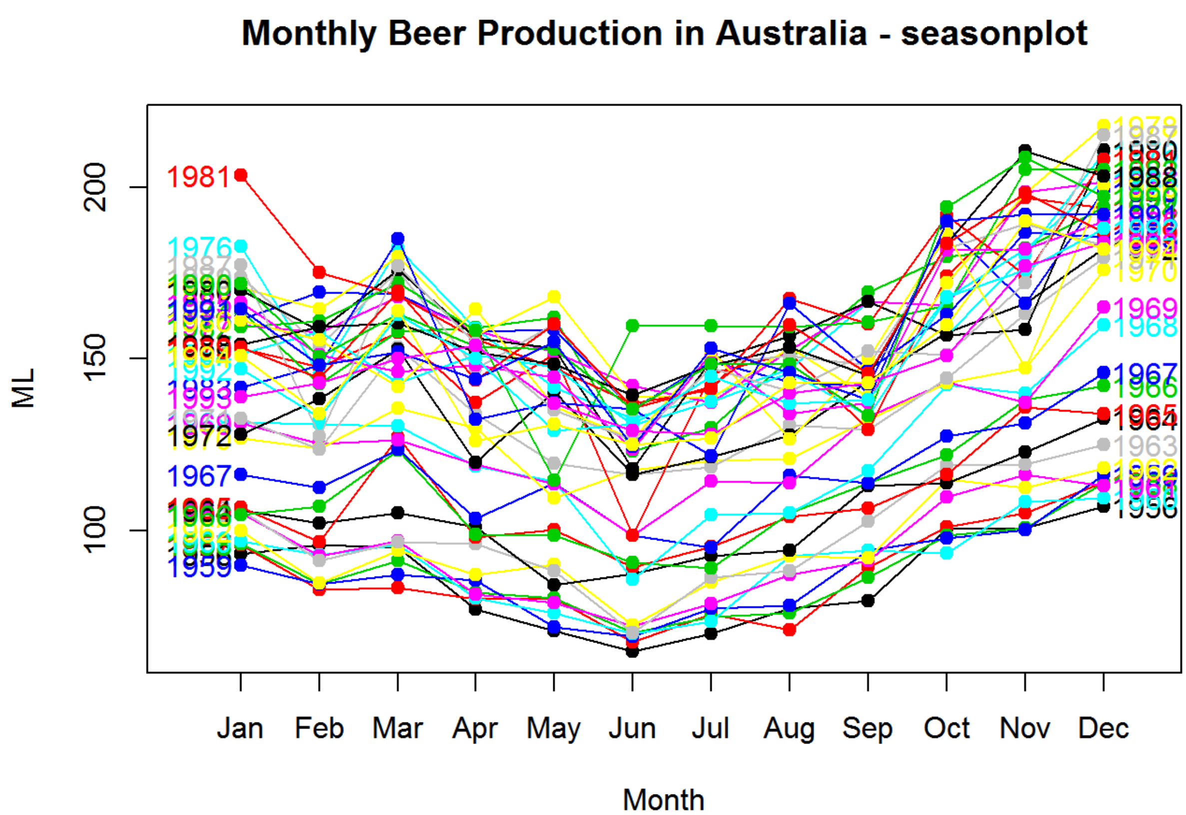

Quarterly Beer Production in Australia
ML
200
300
400
500
600
1960
1970
1980
1990
Year


Yearly Beer Production in Australia
ML
1000
1200
1400
1600
1800
2000
1960
1970
1980
1990
Year

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.

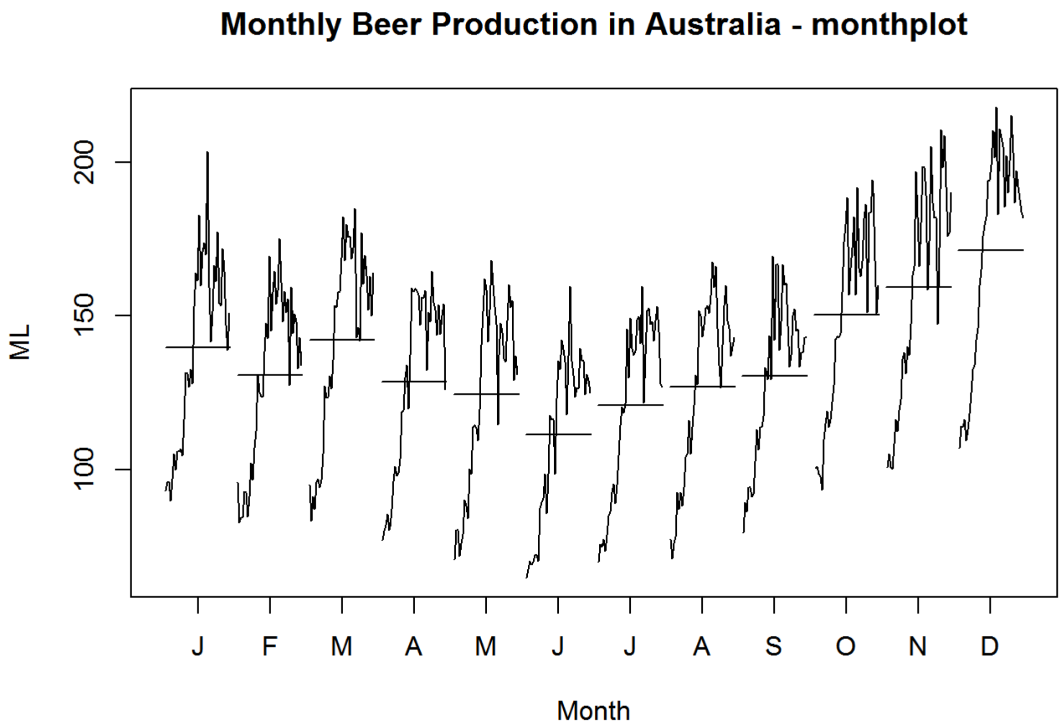
Next step we want to take a look at seasonality in more detail.

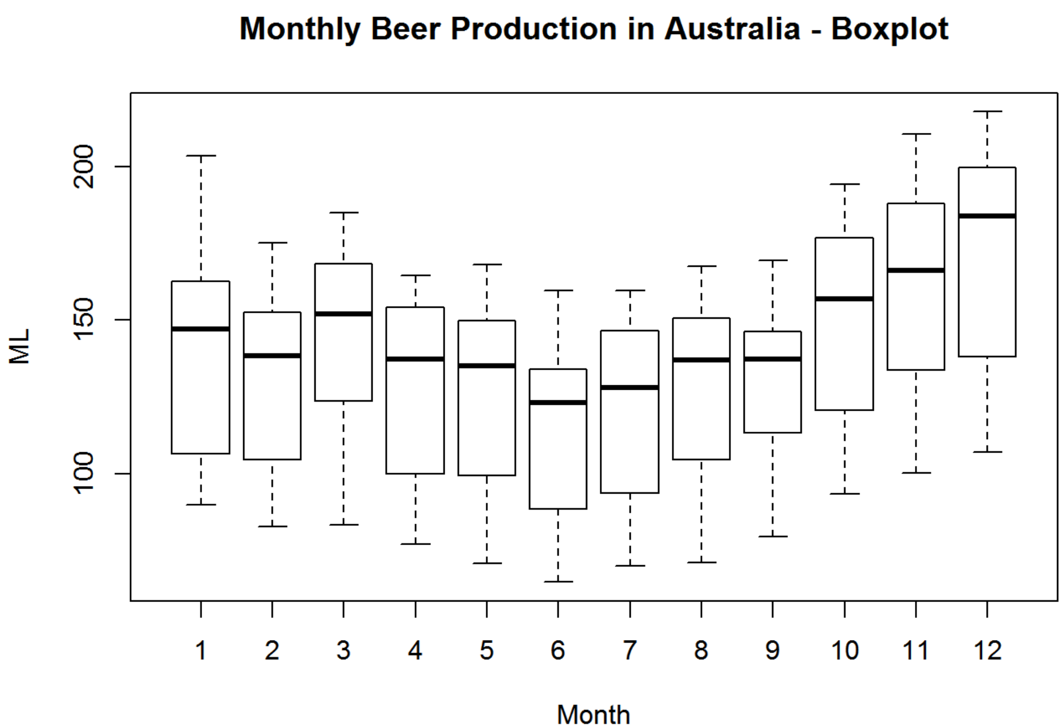
```
seasonplot(beer.ts[,2], year.labels = TRUE, year.labels.left=TRUE, col=1:40, pch=19, main = "
Monthly Beer Production in Australia - seasonplot", xlab = "Month", ylab = "ML")
```




```
monthplot(beer.ts[,2], main = "Monthly Beer Production in Australia - monthplot", xlab = "Mont
h", ylab = "ML")

boxplot(beer.ts[,2] ~ cycle(beer.ts[,2]), xlab = "Month", ylab = "ML", main = "Monthly Beer Pro
duction in Australia - Boxplot")
```





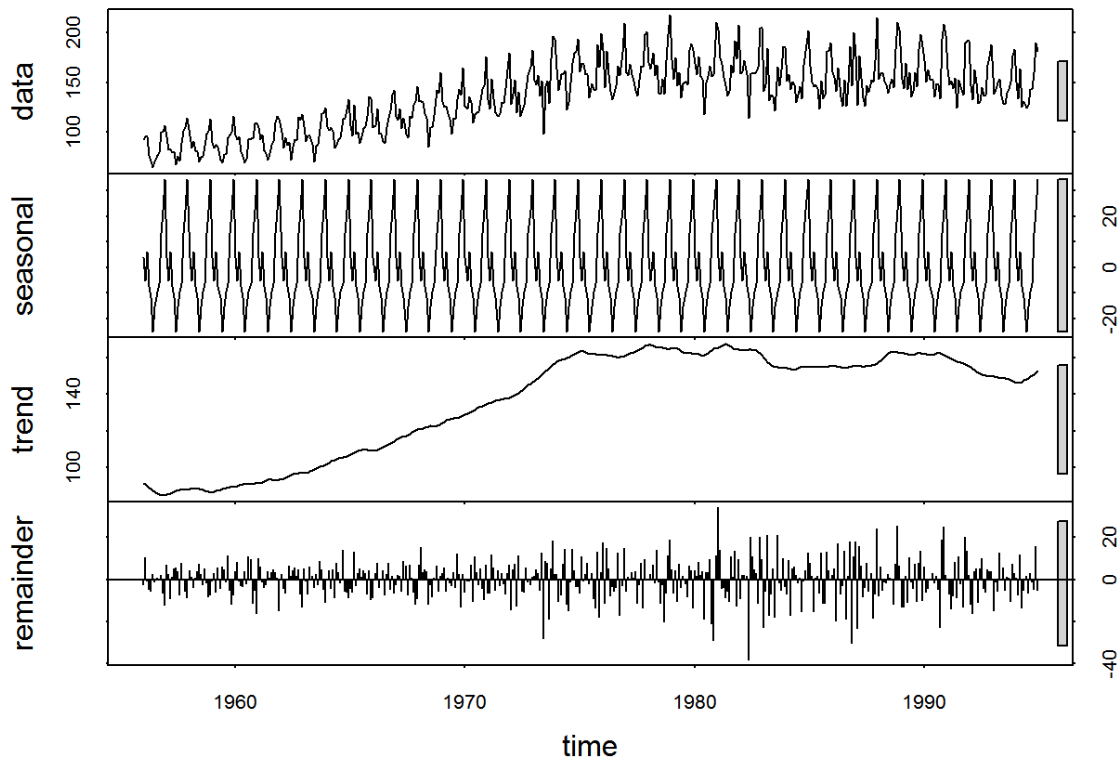
3 different charts all give us the level among different months, but also the range and variation. We can also see the variation and range within the same month.

Decomposition

After just looking at the different plots, we normally can get a good sense on how the time series behaves and the different components within the data. Decomposition is a tool that we can separate different components in a time series data so we can see trend, seasonality, and random noises individually.

STL Decomposition

```
plot(stl(beer.ts[,2], s.window="periodic"))
```



From this chart, we can see that the seasonality is strong but consistent. The trend is similar to what we saw when we aggerated the data into yearly which is that from 1950 to 1975, there was a strong growth and after that, the production slowly went down. Also, the noises went up started from 1975 as well.

Stationarity

I have used ADF test to check the stationarity of data in R.

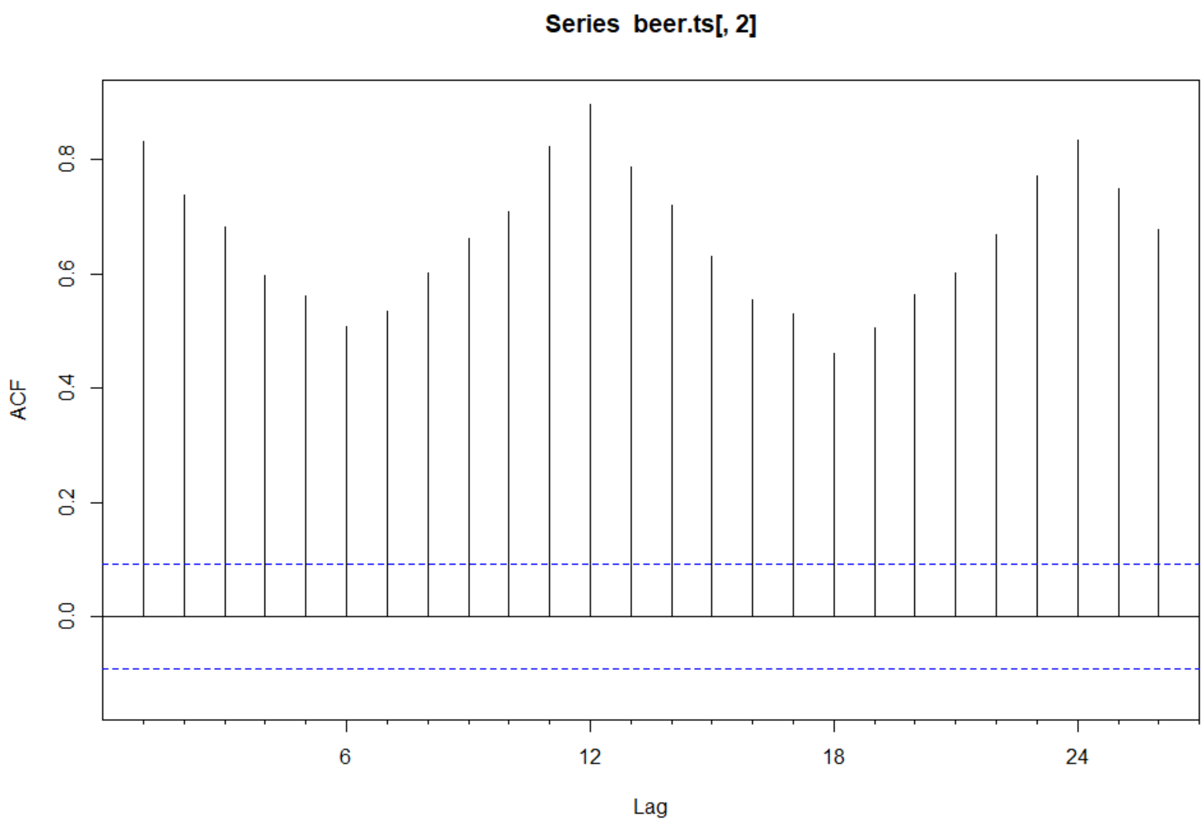
```
adf.test(beer.ts[,2])                    #To test the stationarity of data
```

```
Augmented Dickey-Fuller Test

data:  beer.ts[, 2]
Dickey-Fuller = -4.341, Lag order = 7, p-value = 0.01
alternative hypothesis: stationary
```

We can see that the p-value < 0.05 hence we reject the null hypothesis.
And the data is stationary.

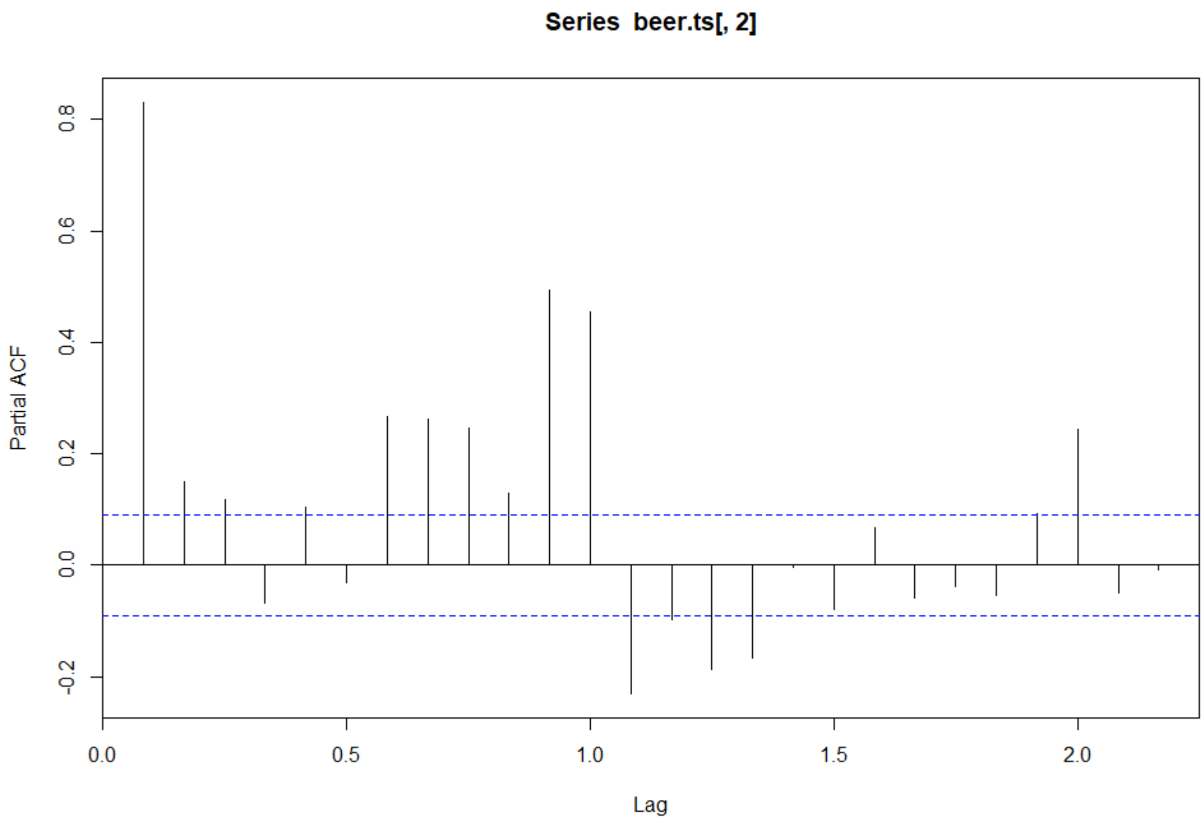
ACF Plots



The plot between ACF and lag is called **Correlogram.**

Another very useful tool to look at the correlation from lag is acf function. As we can see that there are very strong relationship from lag 1 to lag 9 and we can also see the seasonal effect as well.

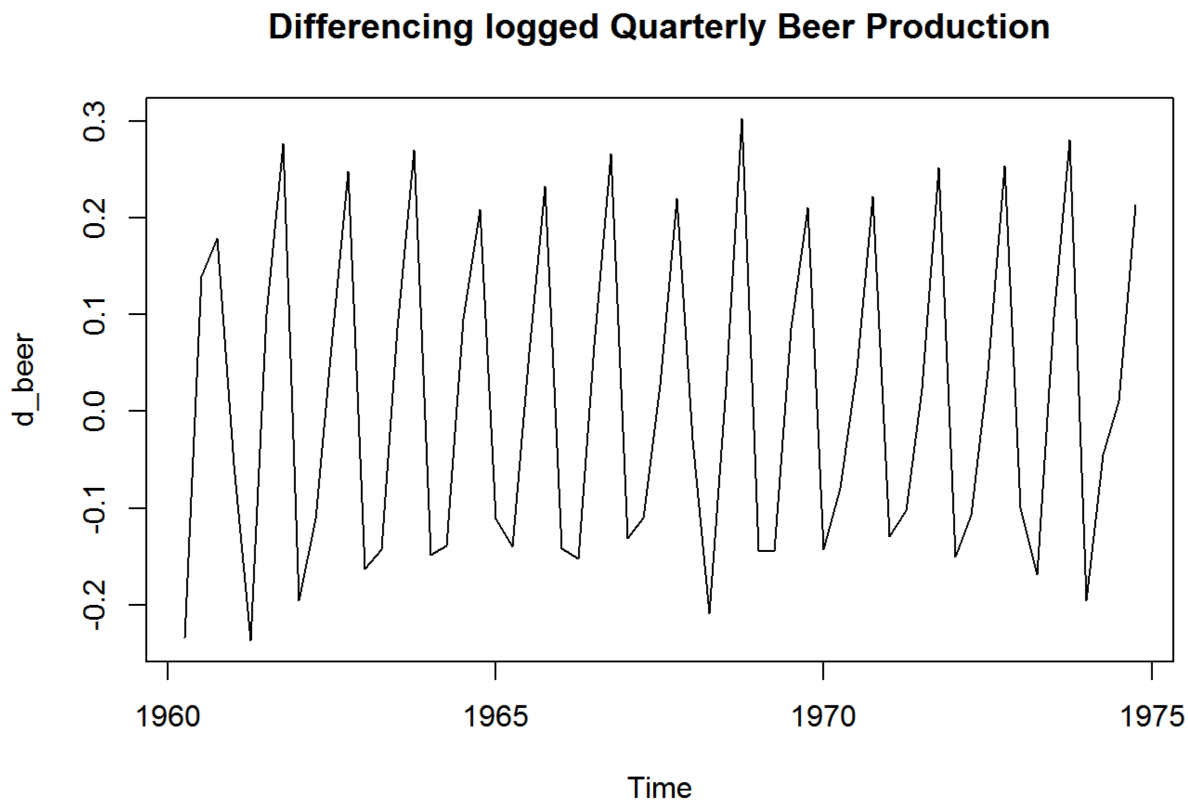
PACF Plots



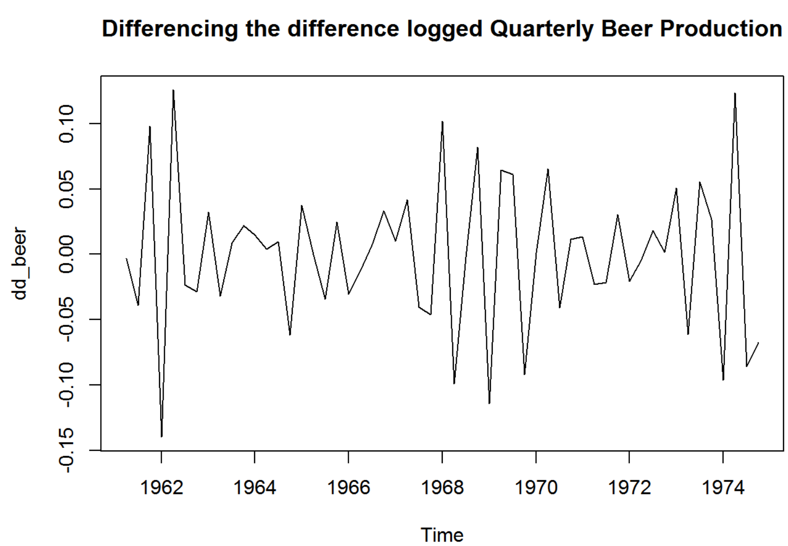
Differencing

Differencing computes the differences between consecutive observations. By differencing the time series data, we can remove the trend and seasonality.

```
d_beer <- diff(log(beer.ts.qtr[,2]))
plot(d_beer, main = "Differencing logged Quarterly Beer Production")
```

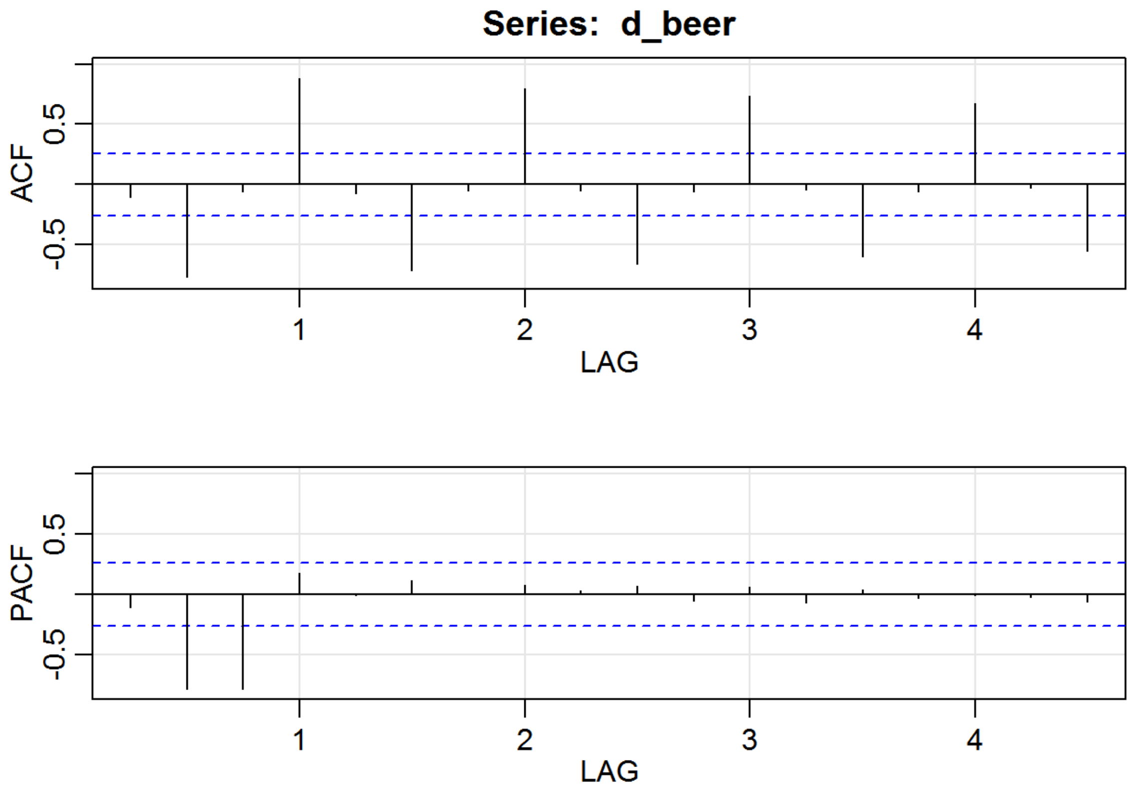



```
dd_beer <- diff(d_beer, lag = 4)
plot(dd_beer, main = "Differencing the difference logged Quarterly Beer Production")
```



The first differencing removed the trend but we can still see some seasonality effect. By doing another differencing with lag 12, we now removed the seasonality. Time series now should be now stationary. We can also see that from ACF charts

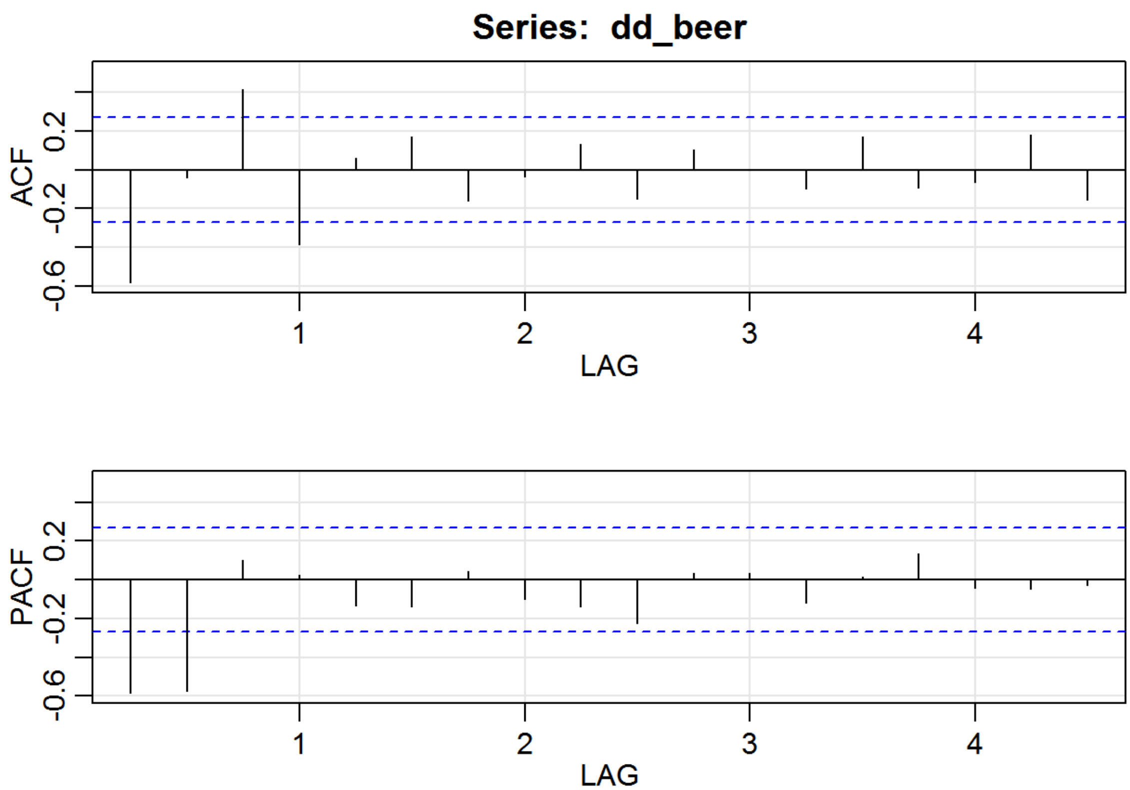
```
acf2(d_beer)
```




```
##        ACF  PACF
##  [1,] -0.11 -0.11
##  [2,] -0.77 -0.79
##  [3,] -0.06 -0.79
##  [4,]  0.88  0.18
##  [5,] -0.07 -0.01
##  [6,] -0.71  0.11
##  [7,] -0.05  0.00
##  [8,]  0.79  0.08
##  [9,] -0.05  0.03
## [10,] -0.66  0.07
```

```
## [11,] -0.06 -0.06
## [12,]  0.73  0.06
## [13,] -0.04 -0.07
## [14,] -0.60  0.04
## [15,] -0.06 -0.03
## [16,]  0.67 -0.01
## [17,] -0.03 -0.03
## [18,] -0.56 -0.06
```

```r
acf2(dd_beer)
```




```
##        ACF  PACF
##  [1,] -0.58 -0.58
##  [2,] -0.04 -0.57
##  [3,]  0.42  0.10
##  [4,] -0.39  0.03
##  [5,]  0.06 -0.13
```

```
##  [6,]  0.17 -0.14
##  [7,] -0.16  0.04
##  [8,] -0.04 -0.10
##  [9,]  0.13 -0.14
## [10,] -0.15 -0.23
## [11,]  0.10  0.04
## [12,]  0.00  0.04
## [13,] -0.10 -0.12
## [14,]  0.17  0.01
## [15,] -0.10  0.14
## [16,] -0.06 -0.04
## [17,]  0.18 -0.05
## [18,] -0.15 -0.03
```

ARMA & ARIMA Forecasting

```
t<-data.frame(matrix(NA, ncol = 3))
for(i in 0:5)
{
  for (j in 0:5)
  {
    a<-arima(data1[,2],order=c(i,0,j))
    t<-rbind(t,c(paste("ARMA(",i,",",j,")"),a$aic,BIC(a)))
  }
}
t<-t[-1, ]                    #first row is a NULL row
```

To check which ARMA model is best to forecast I created a null data frame in which I store model in one column in other two columns model’s AIC and BIC.

	MODEL	AIC	BIC		MODEL	AIC	BIC
1	ARMA(0 , 0)	4631.515929	4639.812865	19	ARMA(3 , 0)	4062.711941	4083.454
2	ARMA(0 , 1)	4316.145392	4328.590797	20	ARMA(3 , 1)	4052.648651	4077.539
3	ARMA(0 , 2)	4246.180542	4262.774415	21	ARMA(3 , 2)	4054.163489	4083.203
4	ARMA(0 , 3)	4158.608337	4179.350679	22	ARMA(3 , 3)	4024.426223	4057.614
5	ARMA(0 , 4)	4122.966162	4147.856972	23	ARMA(3 , 4)	3968.536778	4005.873
6	ARMA(0 , 5)	4089.383223	4118.422501	24	ARMA(3 , 5)	4004.750585	4046.235
7	ARMA(1 , 0)	4076.966328	4089.411733	25	ARMA(4 , 0)	4062.545752	4087.437
8	ARMA(1 , 1)	4064.261819	4080.855692	26	ARMA(4 , 1)	4054.477076	4083.516
9	ARMA(1 , 2)	4020.005039	4040.747381	27	ARMA(4 , 2)	4003.834196	4037.022
10	ARMA(1 , 3)	4061.418405	4086.309215	28	ARMA(4 , 3)	3966.417005	4003.753
11	ARMA(1 , 4)	3995.376646	4024.415924	29	ARMA(4 , 4)	3968.454192	4009.939
12	ARMA(1 , 5)	4009.504275	4042.692022	30	ARMA(4 , 5)	3996.946897	4042.58
13	ARMA(2 , 0)	4067.331	4083.924873	31	ARMA(5 , 0)	4058.990235	4088.03
14	ARMA(2 , 1)	4007.54976	4028.292101	32	ARMA(5 , 1)	4054.474753	4087.662
15	ARMA(2 , 2)	4054.852663	4079.743473	**33**	**ARMA(5 , 2)**	**3818.59877**	**3855.935**
16	ARMA(2 , 3)	4056.130945	4085.170223	34	ARMA(5 , 3)	4025.226063	4066.711
17	ARMA(2 , 4)	3996.582505	4029.770252	35	ARMA(5 , 4)	3976.666046	4022.299
18	ARMA(2 , 5)	4008.140607	4045.476822	36	ARMA(5 , 5)	3982.543571	4032.325

```
t[which.min(t[,2]),]

t[which.min(t[,3]),]
```

```
                X1            X2              X3
33 ARMA( 5 , 2 ) 3818.5987704574 3855.93498512066
          X1            X2              X3
33 ARMA( 5 , 2 ) 3818.5987704574 3855.93498512066
```

We can see that the model with minimum AIC and BIC is ARMA(5,2). Hence we will select ARMA(5,2) model to forecast our time series data.

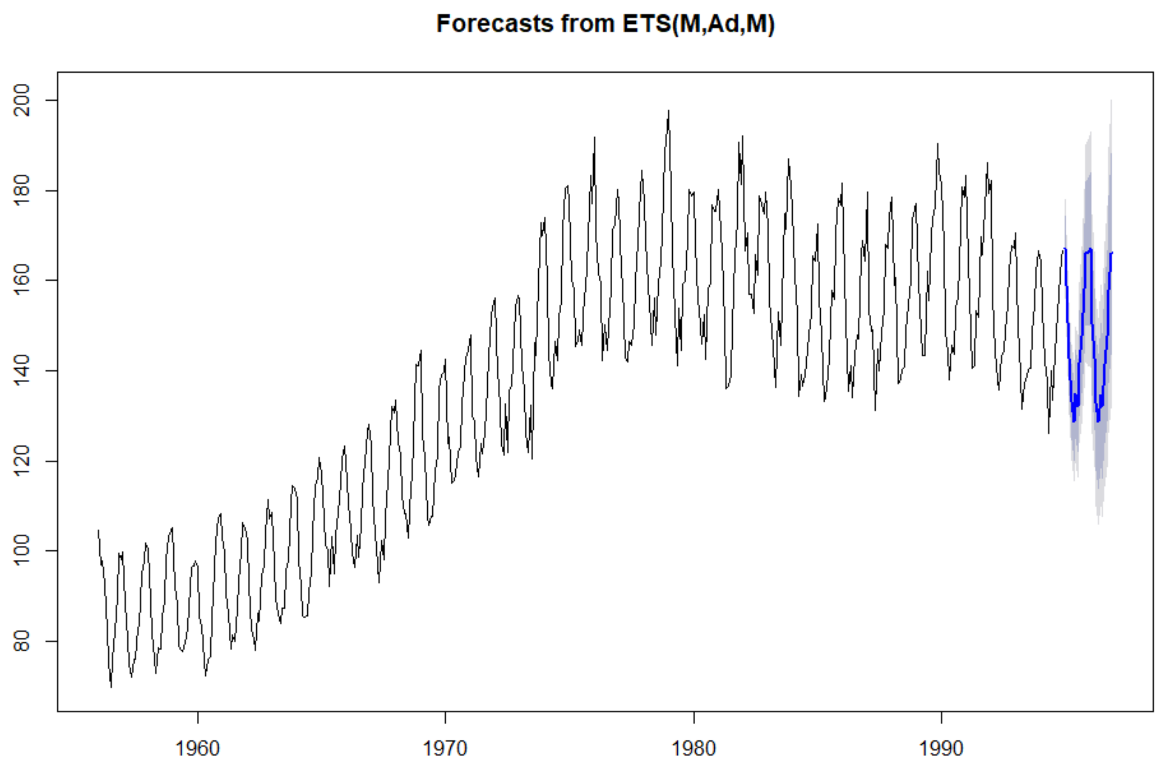
```
a<-arima(data1[,2],order=c(5,0,2))

fit<-fitted(a)

ab<-forecast(fit,h=24)

ab

plot(ab)
```



We can see here in the above graph the forecast for two years in blue lines.