

Final Project

Hannah Wilder and Chathura Gunasekara

April 9, 2016

Notes: Possible source of population data: <http://www.abs.gov.au/AUSSTATS/abs@.nsf/DetailsPage/3105.0.65.0012014?OpenDocument>

Change working directory here

Load data (assumes file is in working directory)

```
#load the data
beerData<-read.csv("monthly-beer-production-in-austr.csv")

#cut off the last row which is NA
beerData<-beerData[-nrow(beerData),]
colnames(beerData)<-c("Month", "Production")

#turn into time series also hold back the last two years of data for forecasting
beerTS<-ts(beerData[1:(nrow(beerData)-24),2], frequency=12, start=c(1956,1))
beer_forecast<-ts(beerData[(nrow(beerData)-23):nrow(beerData), 2], start=c(1993,9), frequency=12)
```

Investigate possible deterministic relationship with population data

```
#load population data
library(reshape)

## Warning: package 'reshape' was built under R version 3.2.5

library(ggplot2)

## Warning: package 'ggplot2' was built under R version 3.2.5

#Clean up population data
pop_totalData<-t(read.csv("Pop_total.csv", row.names=1))
dropCols<-colnames(pop_totalData) %in% c("Unspecified","Period not indicated")
rownames(pop_totalData)<-c(1921:2011)
pop_totalData<-pop_totalData[paste(1956:1995),!dropCols]

#Aggregate beer data
beerYear<-seq(from=1956, to=1996, by=1)
beerYear<-rep(beerYear, each=12)
beerYear<-beerYear[1:nrow(beerData)]
beerAg<-aggregate(beerData[,2], FUN=mean, by=list(year=beerYear))

#Attach to beer data
beerPop<-data.frame(cbind(beer=beerAg[,2],pop_totalData))
beerPopScale<-scale(beerPop)

beerPopRes<-melt(beerPopScale, variable.name="series")
colnames(beerPopRes)<-c("time", "series", "stddev")

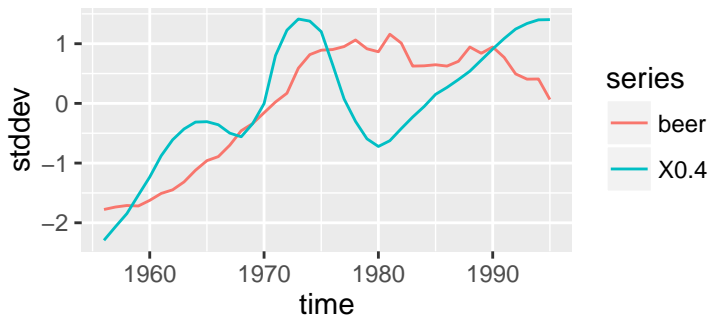
allNames<-colnames(beerPop)[2:length(colnames(beerPop))]
```

```

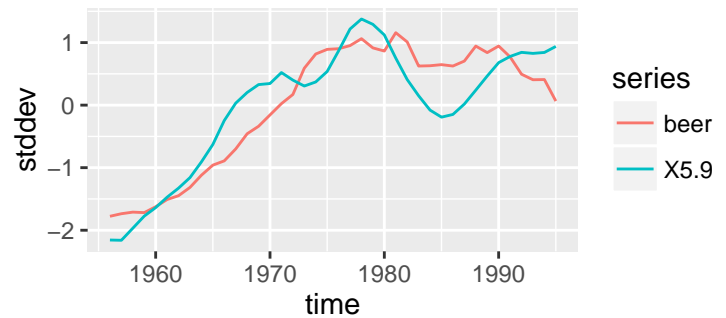
par(mfrow=c(2,2))
for (name in allNames) {
  subset_data<-subset(beerPopRes, beerPopRes$series%in%c("beer", name))
  newPlot<-ggplot(subset_data, aes(time,stddev)) + geom_line(aes(colour = series)) +ggtitle(paste("Pop (age gr", name, ") and Beer Prod"))
  print(newPlot)
}

```

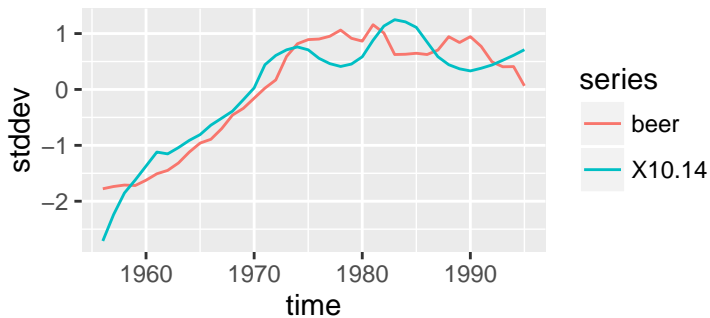
Pop (age group X0.4) and Beer Prod



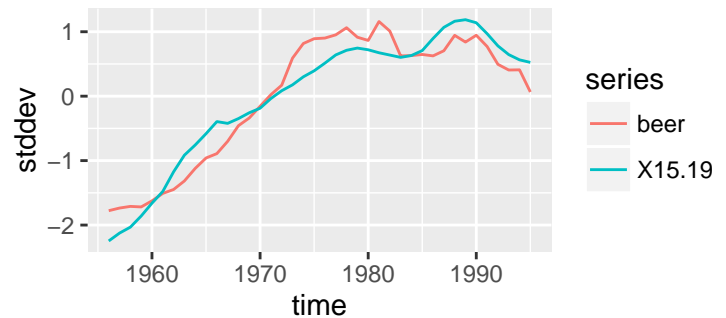
Pop (age group X5.9) and Beer Prod



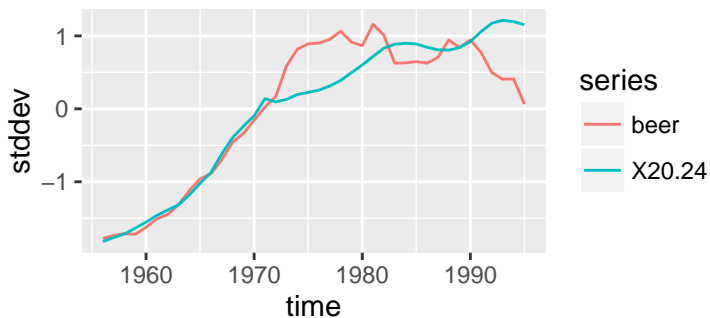
Pop (age group X10.14) and Beer Prod



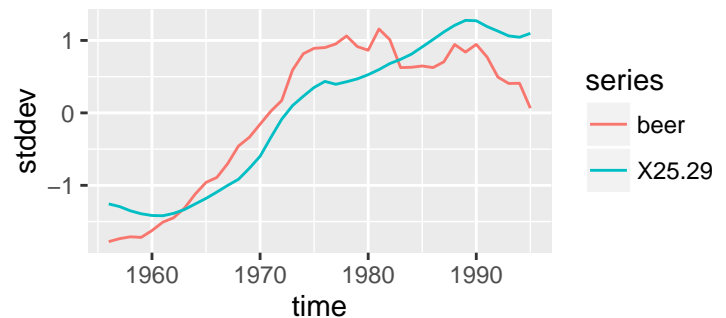
Pop (age group X15.19) and Beer Prod



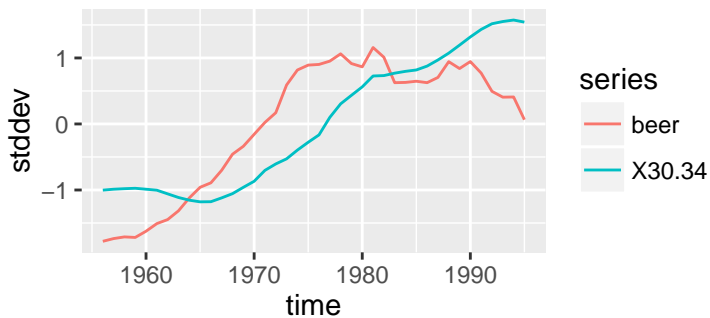
Pop (age group X20.24) and Beer Prod



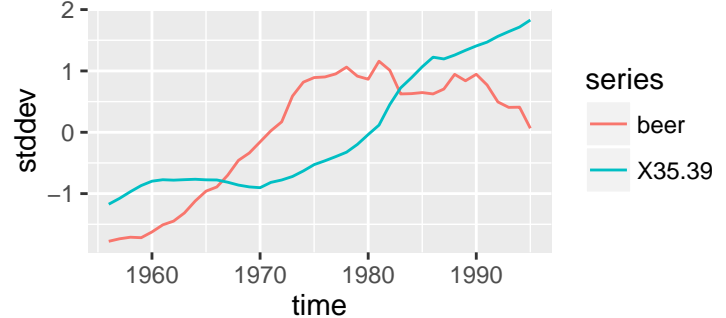
Pop (age group X25.29) and Beer Prod



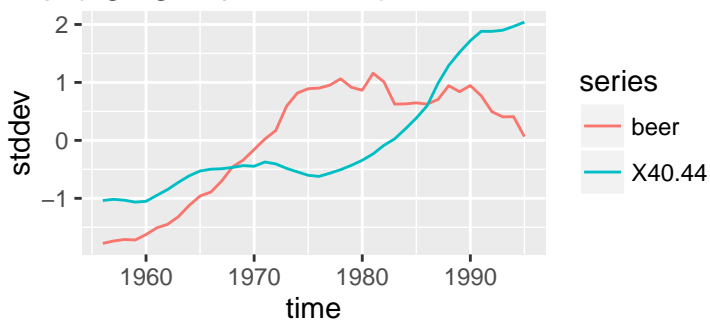
Pop (age group X30.34) and Beer Prod



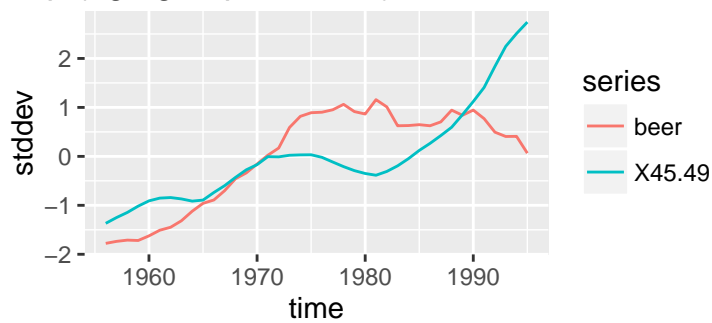
Pop (age group X35.39) and Beer Prod



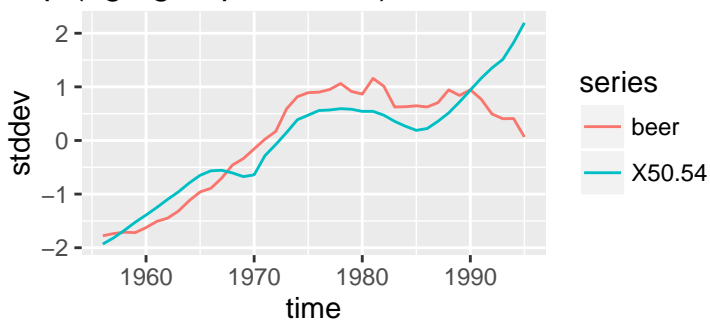
Pop (age group X40.44) and Beer Prod



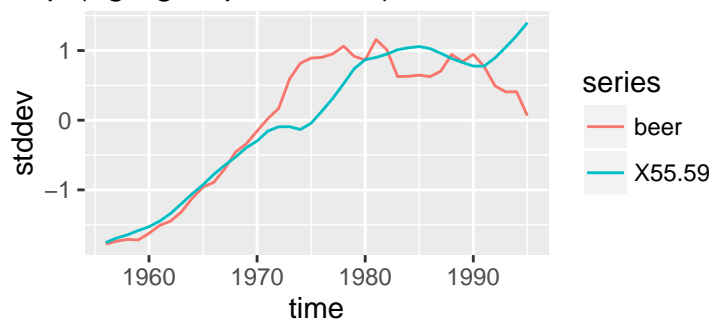
Pop (age group X45.49) and Beer Prod



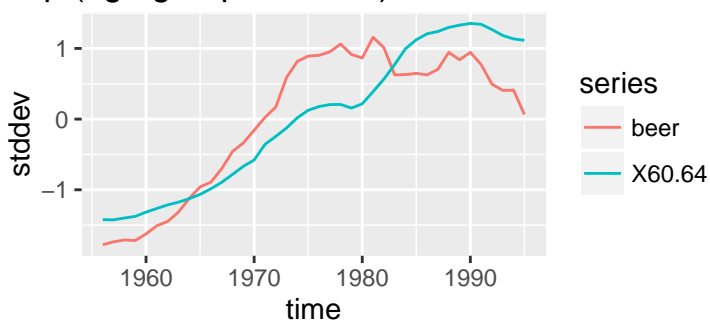
Pop (age group X50.54) and Beer Prod



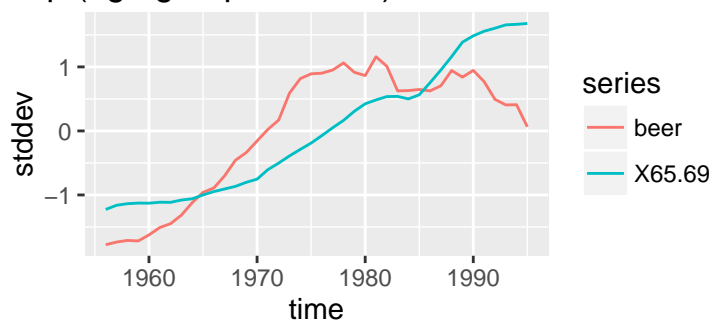
Pop (age group X55.59) and Beer Prod



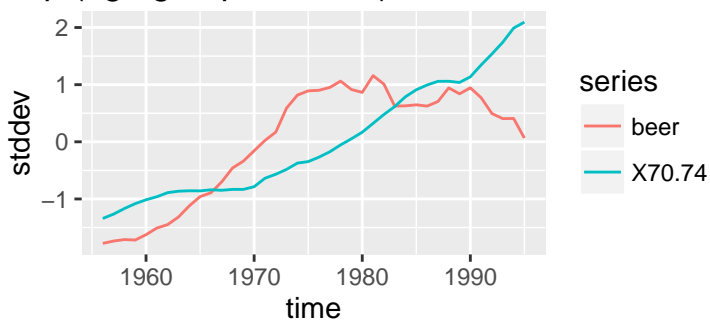
Pop (age group X60.64) and Beer Prod



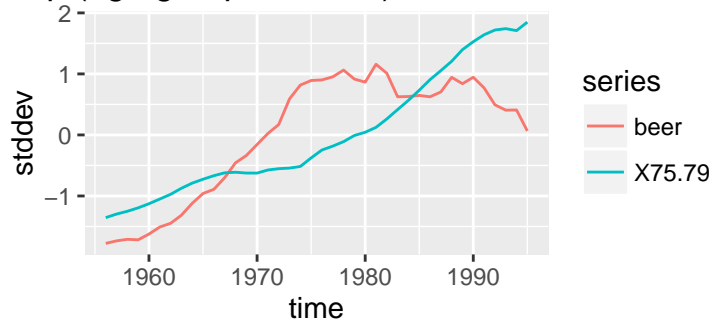
Pop (age group X65.69) and Beer Prod



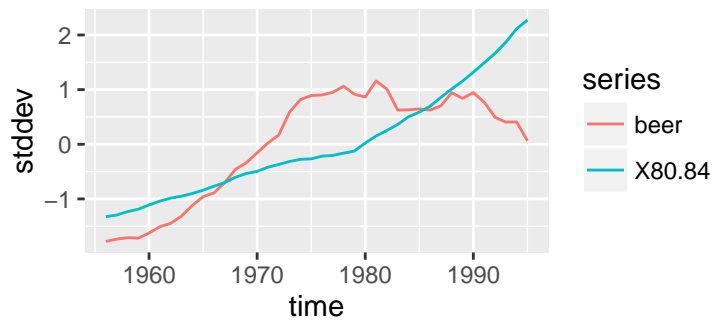
Pop (age group X70.74) and Beer Prod



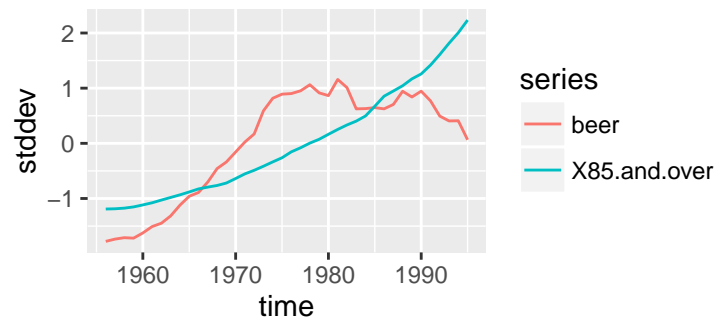
Pop (age group X75.79) and Beer Prod



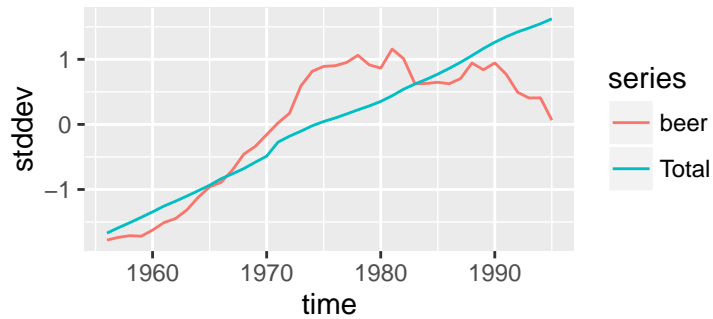
Pop (age group X80.84) and Beer Prod



(age group X85.and.over) and Beer Prod



Pop (age group Total) and Beer Prod



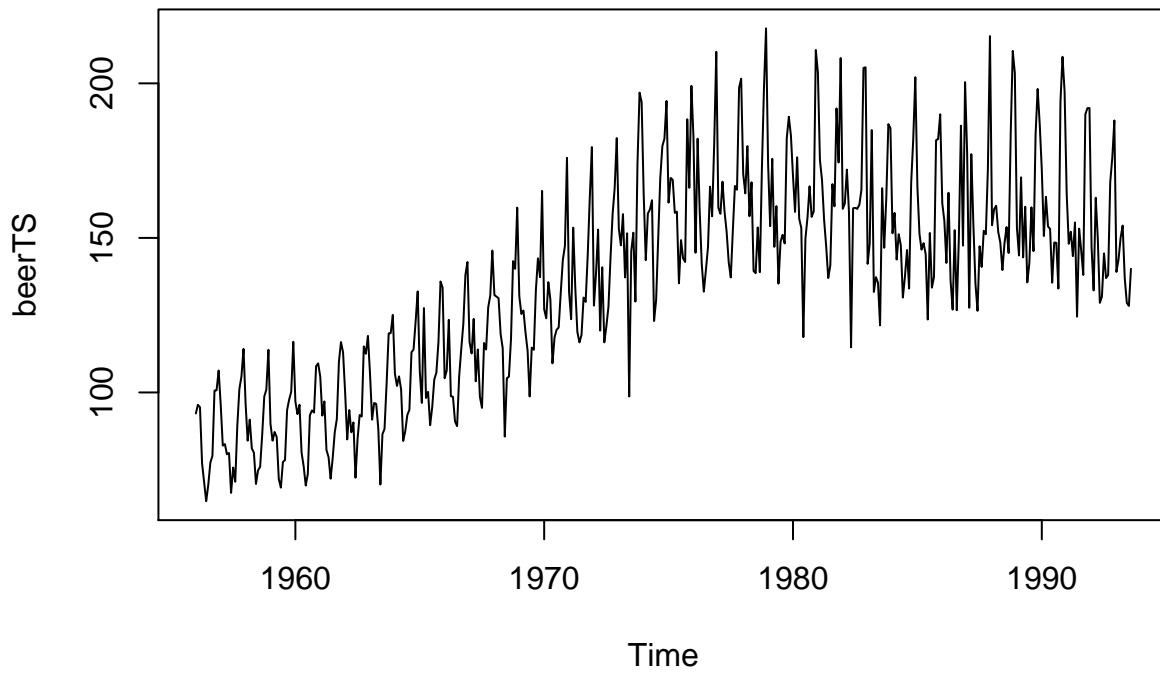
```
par(mfrow=c(1,1))
```

It appears that there may be a relationship with the 15-19 age block, which makes sense since the legal drinking age is 18. We may be able to use this as our deterministic trend.

Plot data

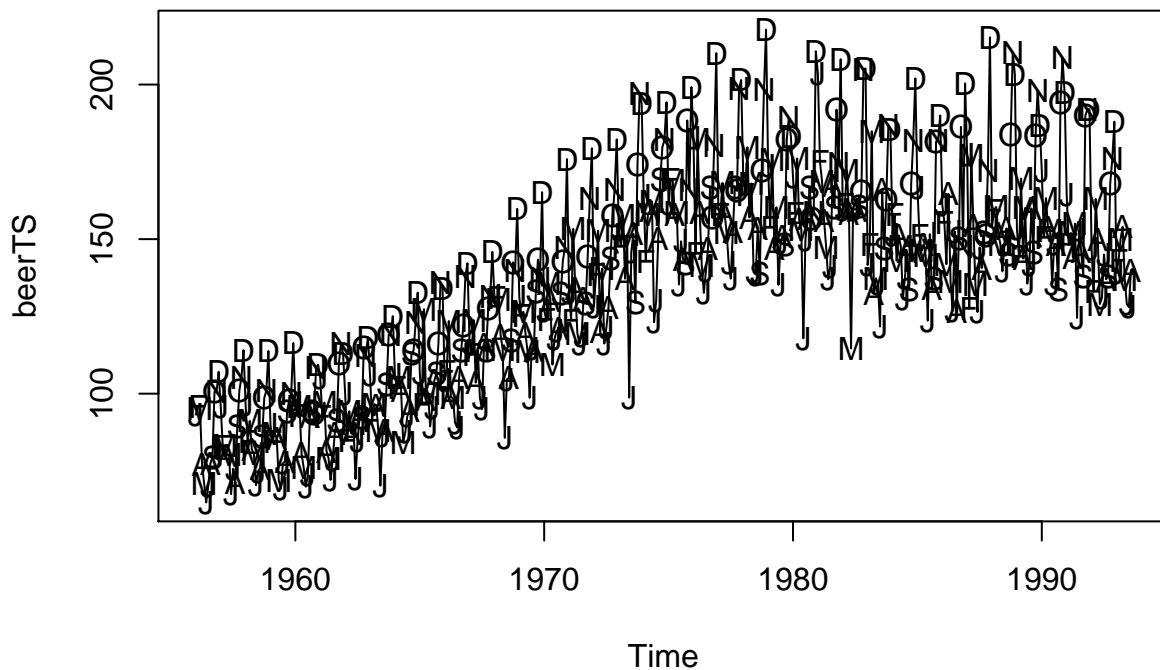
```
par(mfrow=c(1,1))
plot(beerTS, main="Beer Production in Australia by Month")
```

Beer Production in Australia by Month



```
plot(beerTS, main="Beer Production in Australia by Month (seasons marked)", type="l")
points(y=beerTS, x=time(beerTS), pch=as.vector(season(beerTS)))
```

Beer Production in Australia by Month (seasons marked)



Another plot to show seasonality

```
require(fpp)
```

```
## Loading required package: fpp
```

```
## Loading required package: forecast
```

```
## Loading required package: zoo
```

```
##
```

```
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
```

```
##
```

```
##      as.Date, as.Date.numeric
```

```
## Loading required package: timeDate
```

```
##
```

```
## Attaching package: 'timeDate'
```

```
## The following objects are masked from 'package:TSA':
```

```
##
```

```
##      kurtosis, skewness
```

```
## This is forecast 7.0
```

```
##
```

```
## Attaching package: 'forecast'
```

```
## The following objects are masked from 'package:TSA':
```

```
##
```

```
##      fitted.Arima, plot.Arima
```

```
## The following object is masked from 'package:nlme':
```

```
##
```

```
##      getResponse
```

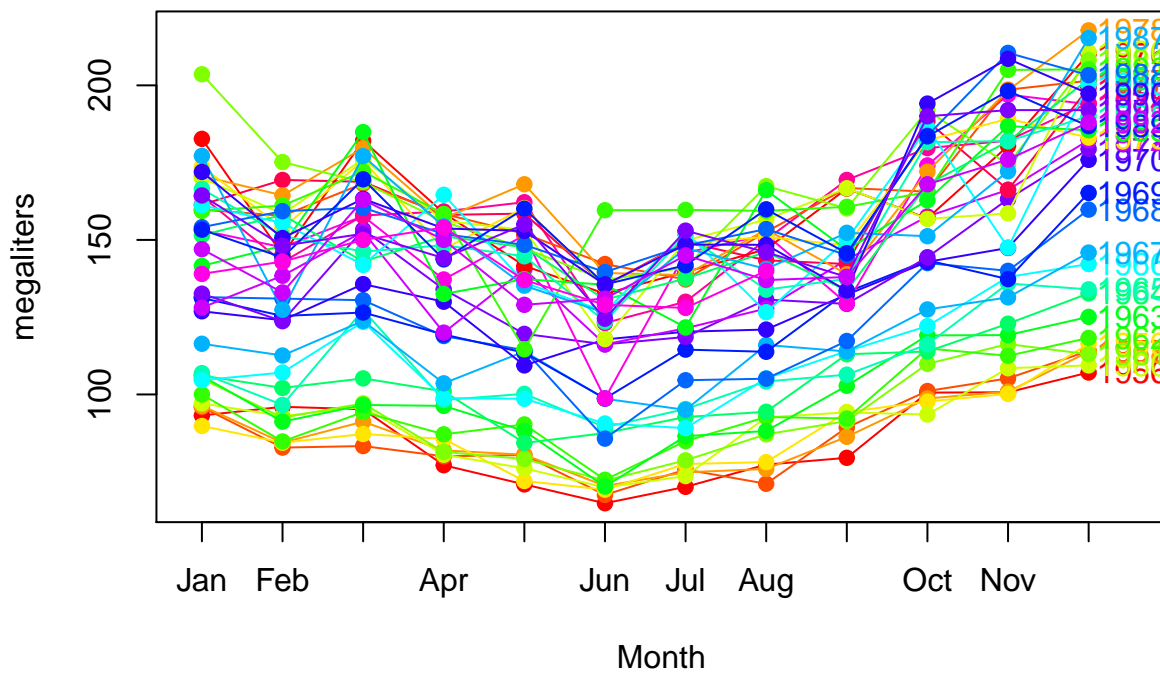
```
## Loading required package: fma
```

```
## Loading required package: expsmooth
```

```
## Loading required package: lmtest
```

```
seasonplot(beerTS,year.labels=TRUE,ylab="megaliters",main="Seasonal plot: quarterly beer production", col=rain)
```

Seasonal plot: quarterly beer production

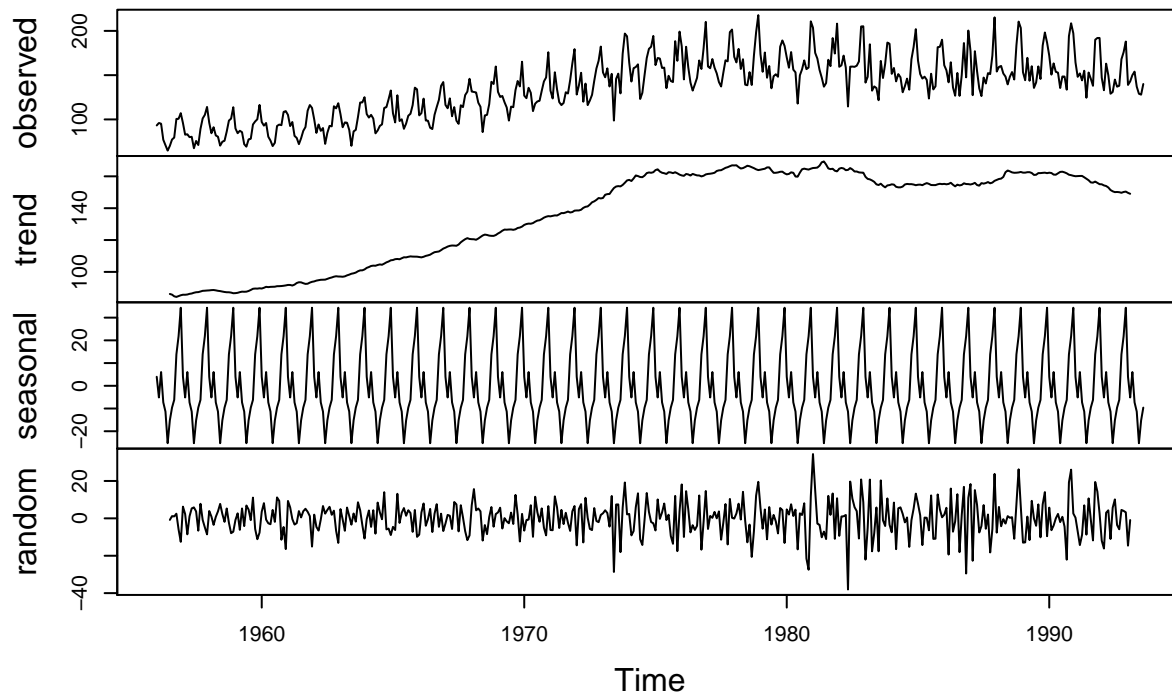


In the plot we see obvious seasonality with higher production in November and December and lower production in June and July. There is a trend which may be difficult to fit as it doesn't appear to be a “well known” function like a linear or quadratic function, so we'll have to experiment. It also looks like the variance of the data is larger in the middle, so we will probably want to take the log of our data to correct that variance issue.

Decompsing the time series to see trends and patterns

```
decompbeer = decompose (beerTS, type="additive")  
plot (decompbeer)
```

Decomposition of additive time series



by looking at the decomposed figures, i was wondering what if we plot a harmonic function with a quadratic polynomial... like imposing a sine curve with 2nd order poly ?

Try to figure out deterministic trend

```
t<-1:length(beerTS)
t2<-t^2
t3<-t^3
t4<-t^4
t5<-t^5

quadFit<-lm(beerTS~t+t2)
summary(quadFit)
```

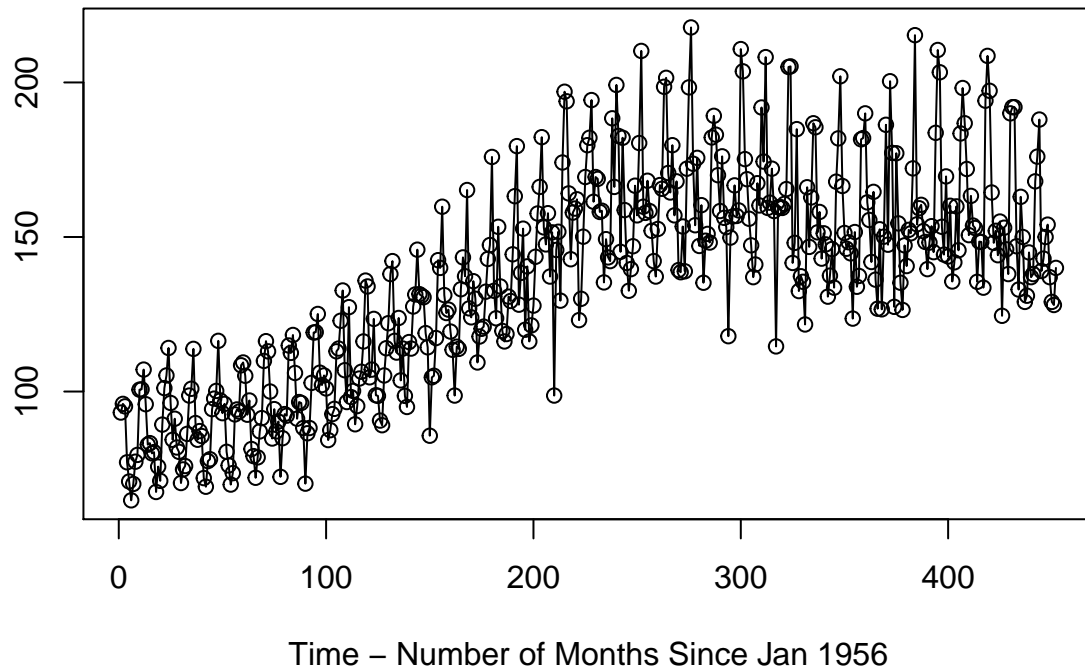
```
##
## Call:
## lm(formula = beerTS ~ t + t2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -46.861 -14.133  -1.991  11.937  61.174
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.560e+01  2.828e+00   23.20  <2e-16 ***
## t            5.429e-01  2.883e-02   18.83  <2e-16 ***
## t2          -7.721e-04  6.163e-05  -12.53  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```



```
##
## Residual standard error: 19.95 on 449 degrees of freedom
## Multiple R-squared:  0.6616, Adjusted R-squared:  0.6601
## F-statistic: 439 on 2 and 449 DF,  p-value: < 2.2e-16

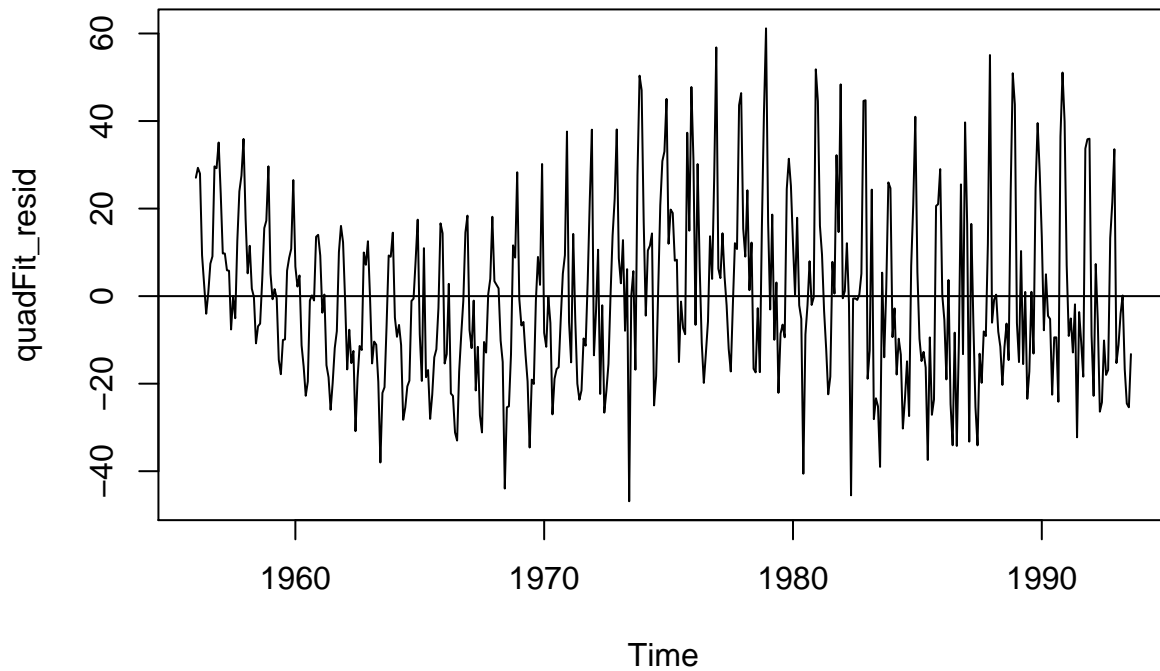
#### plot the data and the fitted quadratic trend function
plot(x=1:length(beerTS),y=beerTS,type='o',ylab="",xlab="Time - Number of Months Since Jan 1956",main="Quadratic Fit on Beer Production Data",col="black",lty=1,add = TRUE, cex=1.5)
curve(expr = coef(quadFit)[1]+coef(quadFit)[2]*x+coef(quadFit)[3]*x^2+coef(quadFit)[4]*x^3,lty=1,add = TRUE, col="black",lty=1,cex=1.5)
```

Quadratic Fit on Beer Production Data



```
quadFit_resid<-ts(residuals(quadFit),frequency=12, start=c(1956,1))
plot(quadFit_resid, main="Residuals from a Quadratic Trend Fit")
abline(h=0)
```

Residuals from a Quadratic Trend Fit



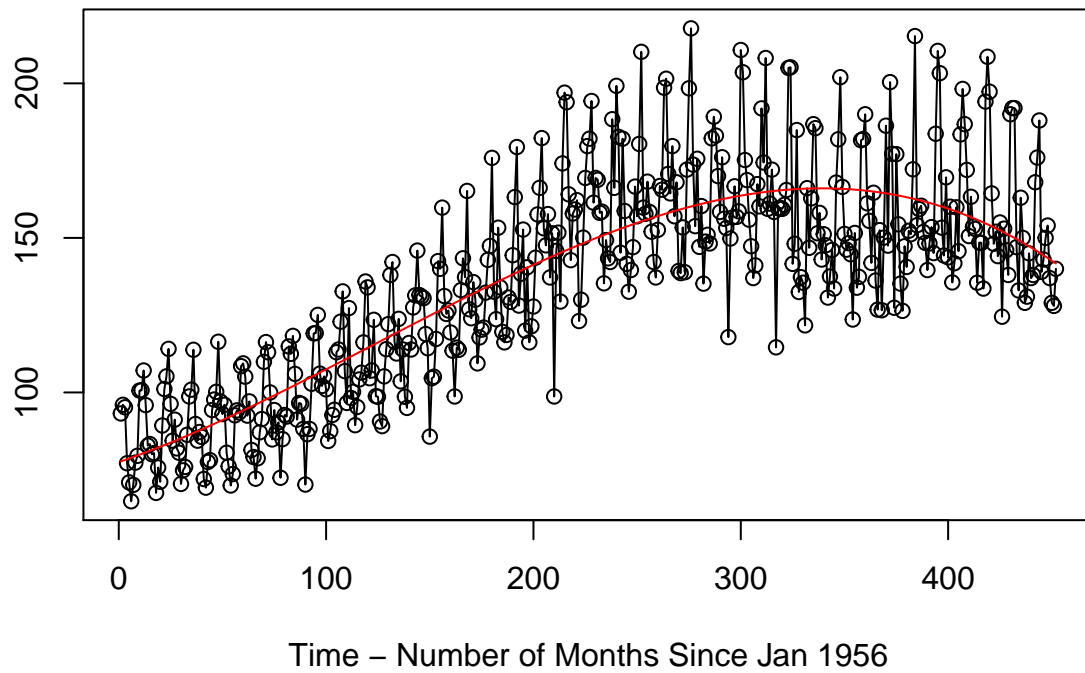
```
cubicFit<-lm(beerTS~t+t2+t3)
summary(cubicFit)
```

```
##
## Call:
## lm(formula = beerTS ~ t + t2 + t3)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -50.660 -13.783  -2.601  12.434  57.639
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.745e+01  3.695e+00  20.963  < 2e-16 ***
## t            2.307e-01  7.056e-02   3.270  0.00116 **
## t2           9.490e-04  3.617e-04   2.624  0.00900 **
## t3          -2.533e-06  5.249e-07  -4.826  1.92e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.47 on 448 degrees of freedom
## Multiple R-squared:  0.6784, Adjusted R-squared:  0.6762
## F-statistic: 315 on 3 and 448 DF, p-value: < 2.2e-16
```

```
#### plot the data and the fitted quadratic trend function
```

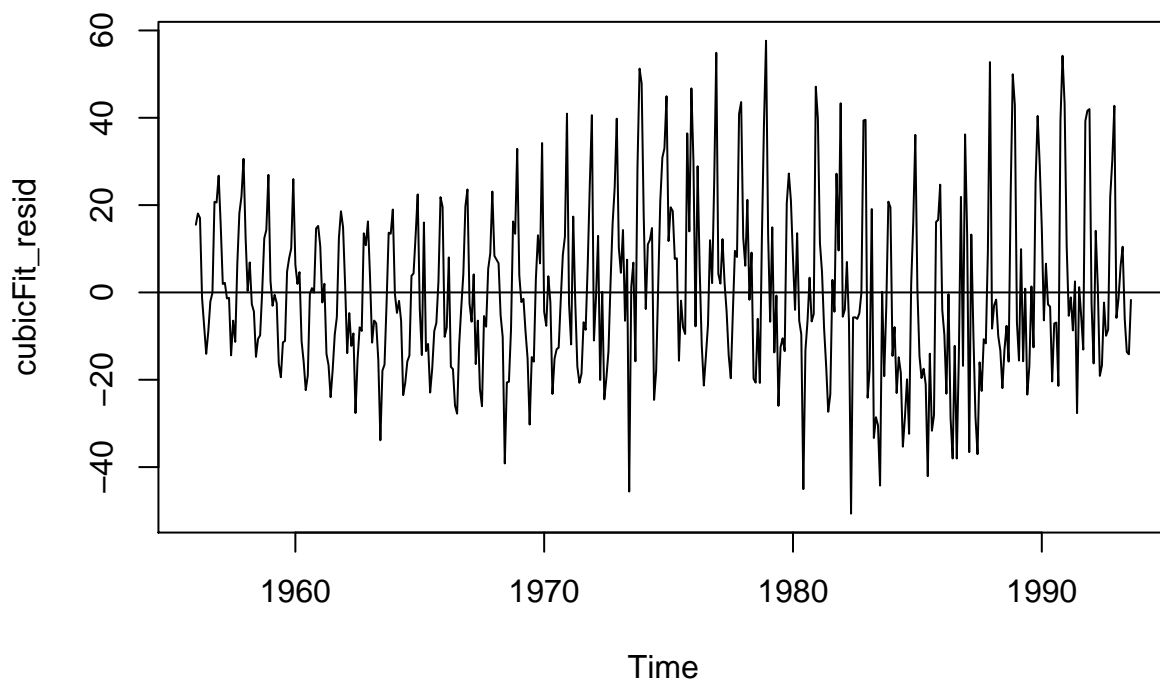
```
plot(x=1:length(beerTS),y=beerTS,type='o',ylab="",xlab="Time - Number of Months Since Jan 1956",main="Cubic Fit",
curve(expr = coef(cubicFit)[1]+coef(cubicFit)[2]*x+coef(cubicFit)[3]*x^2+coef(cubicFit)[4]*x^3,lty=1,add = TRUE))
```

Cubic Fit on Beer Production Data



```
cubicFit_resid<-ts(residuals(cubicFit),frequency=12, start=c(1956,1))  
plot(cubicFit_resid, main="Residuals from a Cubic Trend Fit")  
abline(h=0)
```

Residuals from a Cubic Trend Fit



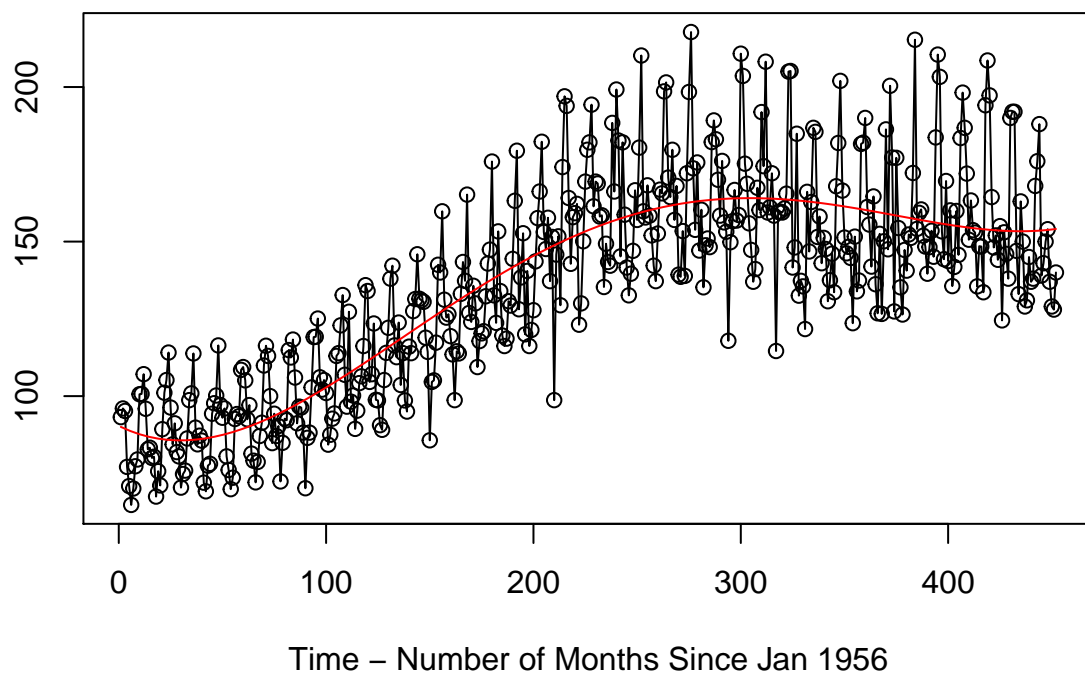
```
order4polyFit<-lm(beerTS~t+t2+t3+t4)
summary(order4polyFit)
```

```
##
## Call:
## lm(formula = beerTS ~ t + t2 + t3 + t4)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -50.079 -12.721  -3.199  10.135  57.983
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.037e+01  4.536e+00  19.924 < 2e-16 ***
## t           -3.341e-01  1.384e-01  -2.414  0.0162 *
## t2           6.545e-03  1.241e-03   5.276 2.07e-07 ***
## t3          -2.173e-05  4.113e-06  -5.285 1.97e-07 ***
## t4           2.119e-08  4.504e-09   4.706 3.38e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.03 on 447 degrees of freedom
## Multiple R-squared:  0.6935, Adjusted R-squared:  0.6908
## F-statistic: 252.9 on 4 and 447 DF,  p-value: < 2.2e-16
```

```
#### plot the data and the fitted 4th order polynomial trend function
```

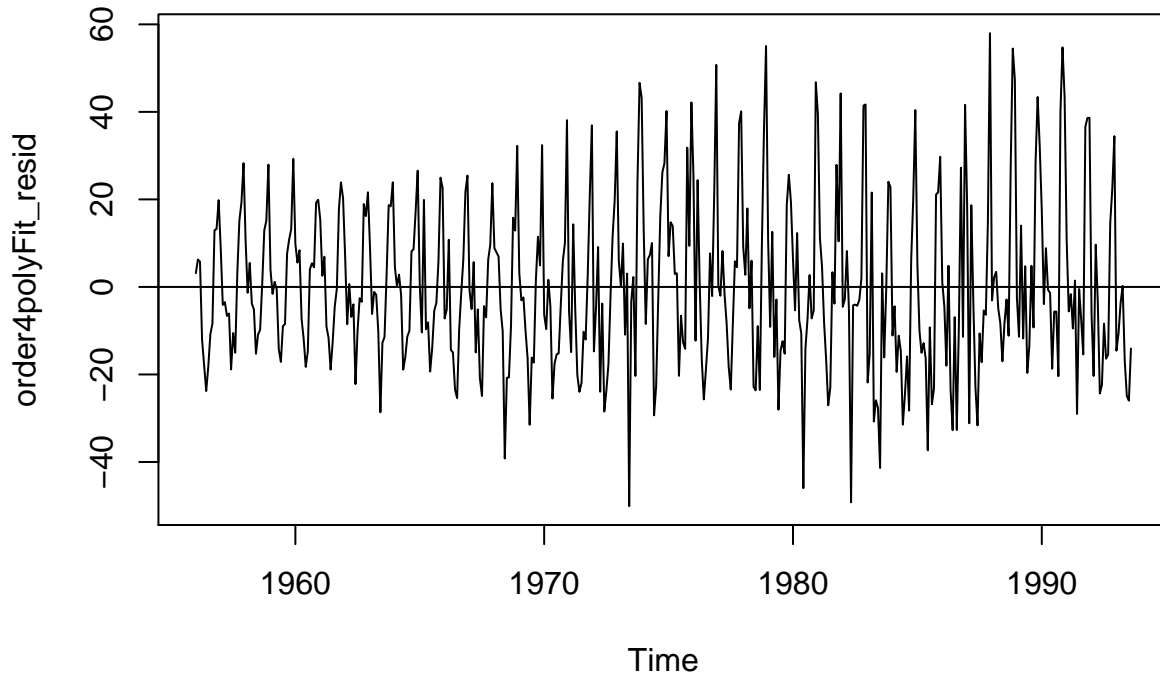
```
plot(x=1:length(beerTS),y=beerTS,type='o',ylab="",xlab="Time - Number of Months Since Jan 1956",main="order4poly
curve(expr = coef(order4polyFit)[1]+coef(order4polyFit)[2]*x+coef(order4polyFit)[3]*x^2+coef(order4polyFit)[4]
```

order4poly Fit on Beer Production Data



```
order4polyFit_resid<-ts(residuals(order4polyFit),frequency=12, start=c(1956,1))
plot(order4polyFit_resid, main="Residuals from a order4poly Trend Fit")
abline(h=0)
```

Residuals from a order4poly Trend Fit



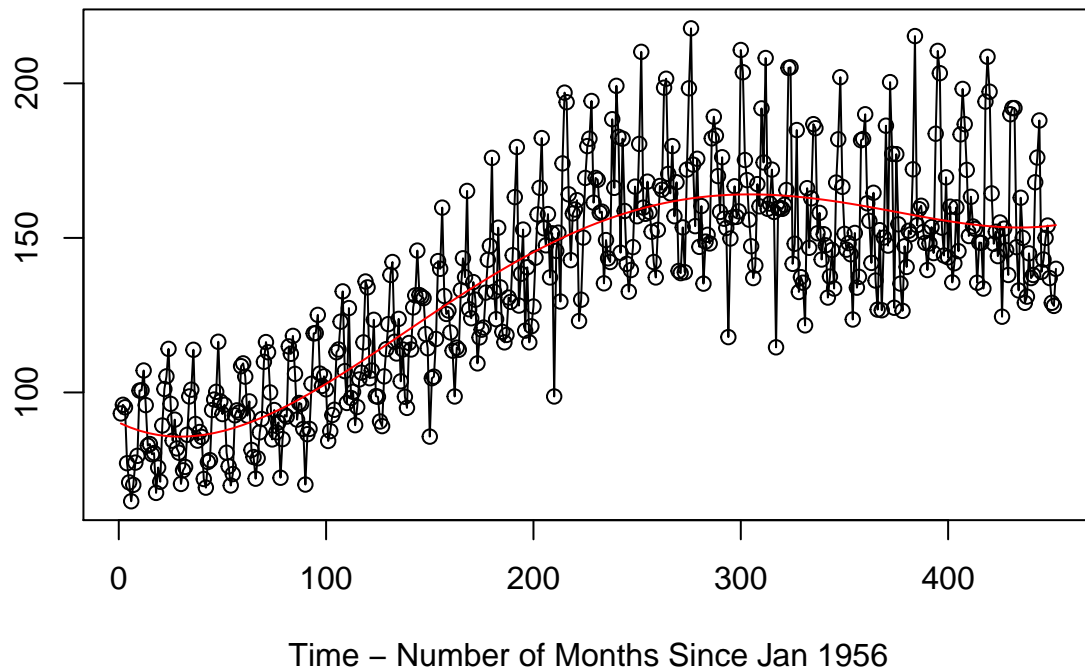
```
order5polyFit<-lm(beerTS~t+t2+t3+t4+t5)
summary(order5polyFit)
```

```
##
## Call:
## lm(formula = beerTS ~ t + t2 + t3 + t4 + t5)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -50.069 -12.729  -3.179  10.132  58.012
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.029e+01  5.483e+00  16.469  <2e-16 ***
## t            -3.288e-01  2.436e-01  -1.350   0.1778
## t2             6.463e-03  3.323e-03   1.945   0.0524 .
## t3            -2.126e-05  1.858e-05  -1.144   0.2532
## t4             2.000e-08  4.519e-08   0.443   0.6582
## t5             1.049e-12  3.970e-11   0.026   0.9789
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.05 on 446 degrees of freedom
## Multiple R-squared:  0.6935, Adjusted R-squared:  0.6901
## F-statistic: 201.9 on 5 and 446 DF,  p-value: < 2.2e-16
```

```
#### plot the data and the fitted 5th order polynomial trend function
```

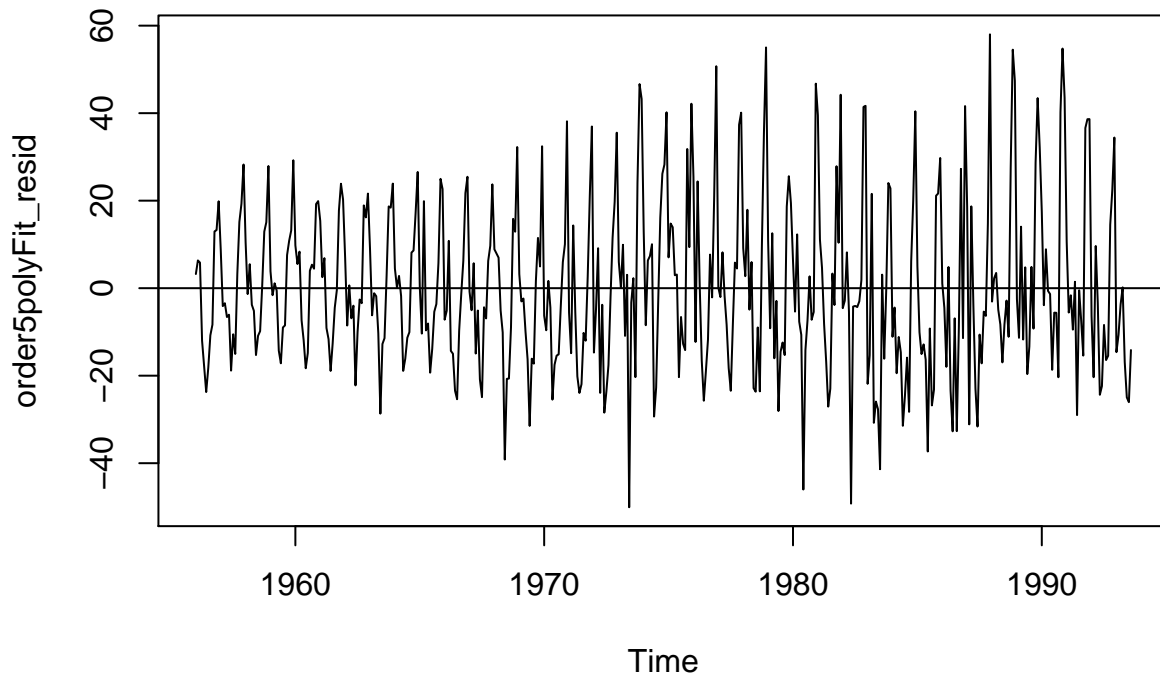
```
plot(x=1:length(beerTS),y=beerTS,type='o',ylab="",xlab="Time - Number of Months Since Jan 1956",main="order5poly  
curve(expr = coef(order5polyFit)[1]+coef(order5polyFit)[2]*x+coef(order5polyFit)[3]*x^2+coef(order5polyFit)[4]
```

order5poly Fit on Beer Production Data



```
order5polyFit_resid<-ts(residuals(order5polyFit),frequency=12, start=c(1956,1))  
plot(order5polyFit_resid, main="Residuals from a order5poly Trend Fit")  
abline(h=0)
```

Residuals from a order5poly Trend Fit



It looks like a 4th order polynomial might take care of the worst of it, the question is are we okay with using a 4th order polynomial or should we drop it down to a cubic function and just deal with it? I found population data and I would be interested to see if we can find a good correlation there (total population won't work, I already looked at that, but maybe a specific age group?)

Assume we go with the 4th order polynomial for now. Let's see what we can do about the seasonality with a seasonal means model

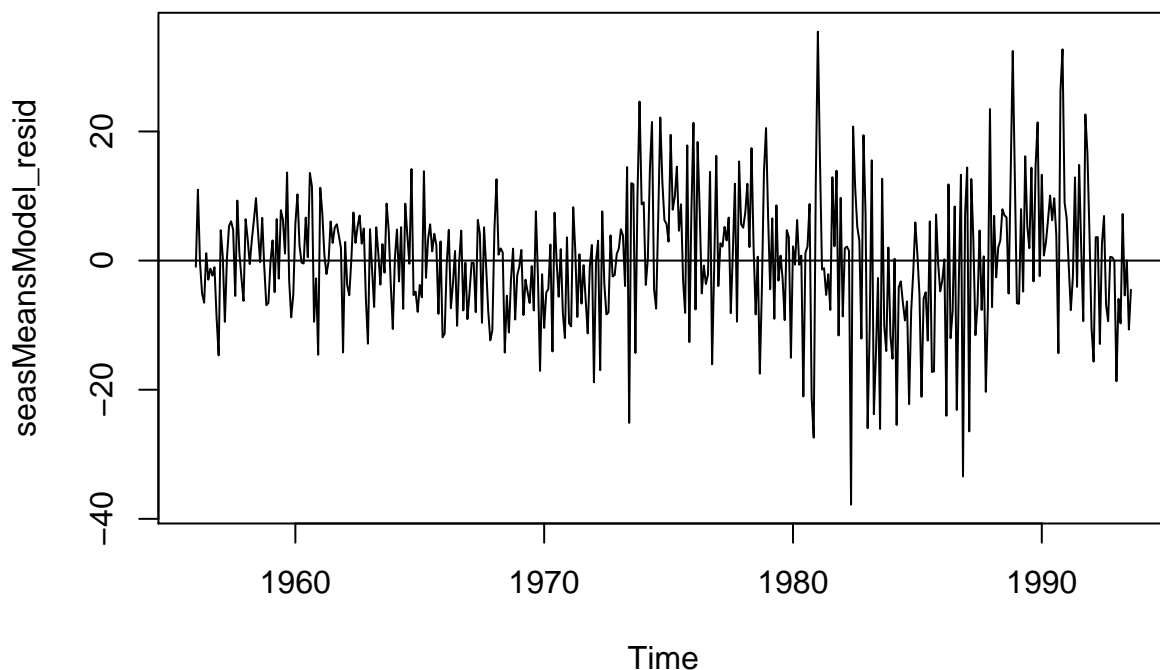
```
library(TSA)
month=season(order4polyFit_resid)
seasMeansModel<-lm(order4polyFit_resid~month)
summary(seasMeansModel)
```

```
##
## Call:
## lm(formula = order4polyFit_resid ~ month)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -37.789  -6.263   0.327   6.128  35.453
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)      4.108      1.652   2.487 0.013268 *
## monthFebruary   -8.810      2.336  -3.771 0.000185 ***
## monthMarch       1.929      2.336   0.826 0.409372
## monthApril     -11.085      2.336  -4.744 2.83e-06 ***
## monthMay       -15.540      2.336  -6.651 8.65e-11 ***
## monthJune      -29.051      2.336 -12.434 < 2e-16 ***
## monthJuly      -19.403      2.336  -8.304 1.25e-15 ***
## monthAugust    -13.661      2.336  -5.847 9.78e-09 ***
## monthSeptember -10.151      2.352  -4.316 1.97e-05 ***
```

```
## monthOctober      9.861      2.352   4.192 3.34e-05 ***
## monthNovember    17.904      2.352   7.612 1.66e-13 ***
## monthDecember     30.406      2.352  12.927 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.18 on 440 degrees of freedom
## Multiple R-squared:  0.7181, Adjusted R-squared:  0.7111
## F-statistic: 101.9 on 11 and 440 DF,  p-value: < 2.2e-16
```

```
seasMeansModel_resid<-ts(residuals(seasMeansModel),frequency=12, start=c(1956,1))
plot(seasMeansModel_resid, main="Residuals from Seasonal Means Model \n(after fitting 4th order polynomial)")
abline(h=0)
```

Residuals from Seasonal Means Model (after fitting 4th order polynomial)



With an adjusted R-squared value of 71%, this is looking pretty good, but in the residual plot you can still see the variance increasing over time. In addition, there is a noticeable “wave” in the residuals that starts around 1970, but I’m not sure what to do about that yet. For now, let’s go back, log the data, and apply both the 4th order polynomial and the seasonal means model at the same time.

```
logBeer<-log(beerTS)
t<-1:length(logBeer)
t2<-t^2
t3<-t^3
t4<-t^4
month<-season(logBeer)

logSeasPoly<-lm(logBeer~t+t2+t3+t4+month)
summary(logSeasPoly)
```

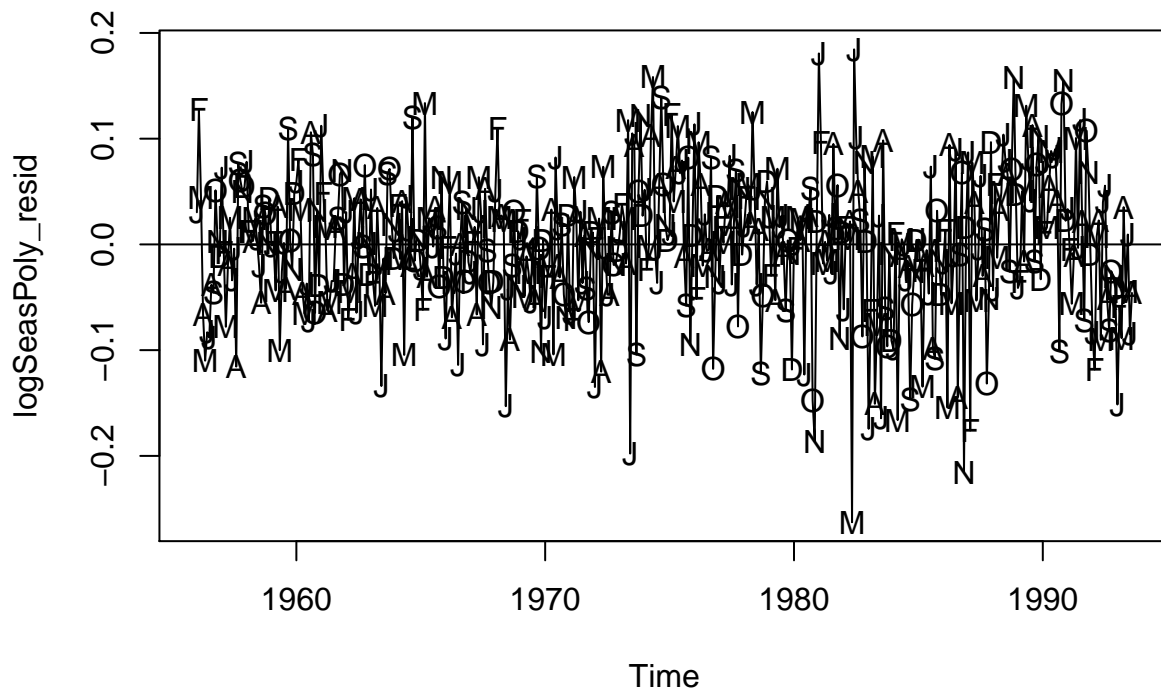
```
##
## Call:
```



```
## lm(formula = logBeer ~ t + t2 + t3 + t4 + month)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.262750 -0.039816  0.003297  0.043475  0.184483
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   4.506e+00  1.945e-02 231.675 < 2e-16 ***
## t             -1.796e-03  5.014e-04  -3.583 0.000378 ***
## t2            4.926e-05  4.494e-06  10.962 < 2e-16 ***
## t3            -1.745e-07  1.490e-08 -11.713 < 2e-16 ***
## t4            1.785e-10  1.631e-11  10.941 < 2e-16 ***
## monthFebruary -6.602e-02  1.580e-02  -4.178 3.56e-05 ***
## monthMarch     1.069e-02  1.580e-02   0.676 0.499268
## monthApril    -8.834e-02  1.581e-02  -5.590 4.02e-08 ***
## monthMay      -1.271e-01  1.581e-02  -8.042 8.39e-15 ***
## monthJune     -2.427e-01  1.581e-02 -15.354 < 2e-16 ***
## monthJuly     -1.578e-01  1.581e-02  -9.984 < 2e-16 ***
## monthAugust   -1.089e-01  1.581e-02  -6.890 1.96e-11 ***
## monthSeptember -7.261e-02  1.591e-02  -4.563 6.57e-06 ***
## monthOctober   6.706e-02  1.591e-02   4.213 3.06e-05 ***
## monthNovember  1.172e-01  1.592e-02   7.363 9.03e-13 ***
## monthDecember  1.936e-01  1.592e-02  12.164 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06889 on 436 degrees of freedom
## Multiple R-squared:  0.9365, Adjusted R-squared:  0.9344
## F-statistic: 429 on 15 and 436 DF, p-value: < 2.2e-16
```

```
logSeasPoly_resid<-ts(residuals(logSeasPoly),frequency=12, start=c(1956,1))
plot(logSeasPoly_resid, main="Residuals from Logged Beer\nseasonal Means and 4th order poly fit at same time",
points(y=logSeasPoly_resid, x=time(logSeasPoly_resid), pch=as.vector(season(logSeasPoly_resid)))
abline(h=0)
```

Residuals from Logged Beer seasonal Means and 4th order poly fit at same time



Let's take a look and see if we have a stationary series yet

```
# d
adf.test(logSeasPoly_resid)
```

```
## Warning in adf.test(logSeasPoly_resid): p-value smaller than printed p-
## value
```

```
##
## Augmented Dickey-Fuller Test
##
## data: logSeasPoly_resid
## Dickey-Fuller = -5.3999, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

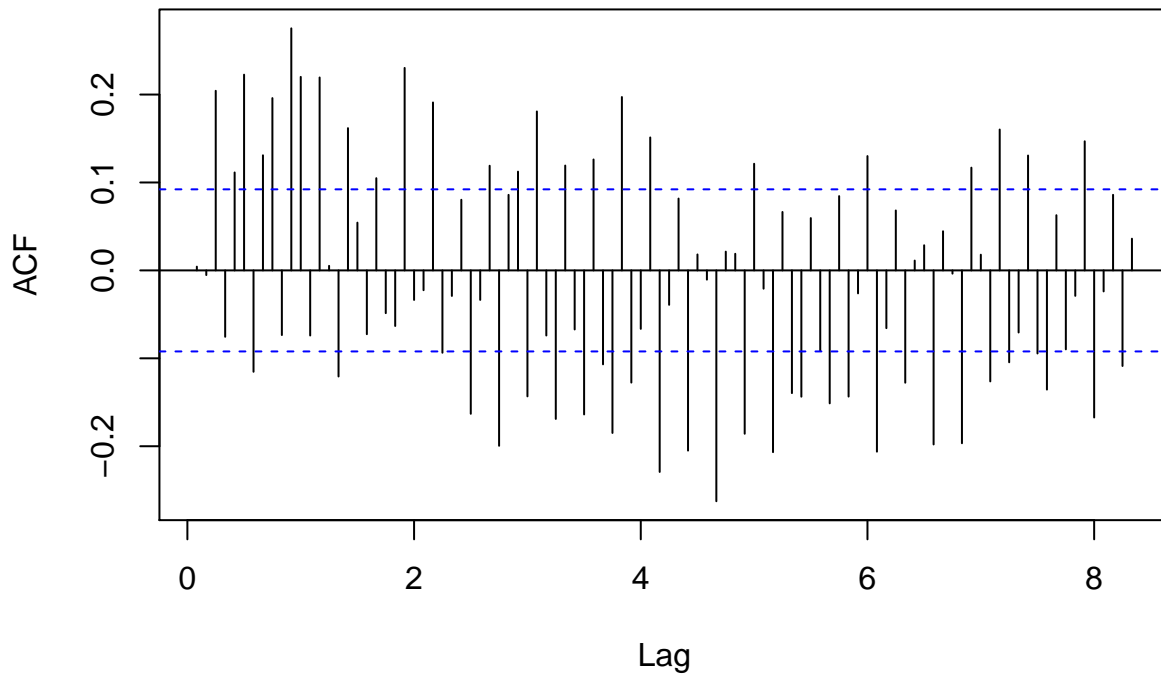
```
pp.test(logSeasPoly_resid)
```

```
## Warning in pp.test(logSeasPoly_resid): p-value smaller than printed p-value
```

```
##
## Phillips-Perron Unit Root Test
##
## data: logSeasPoly_resid
## Dickey-Fuller Z(alpha) = -489.81, Truncation lag parameter = 5,
## p-value = 0.01
## alternative hypothesis: stationary
```

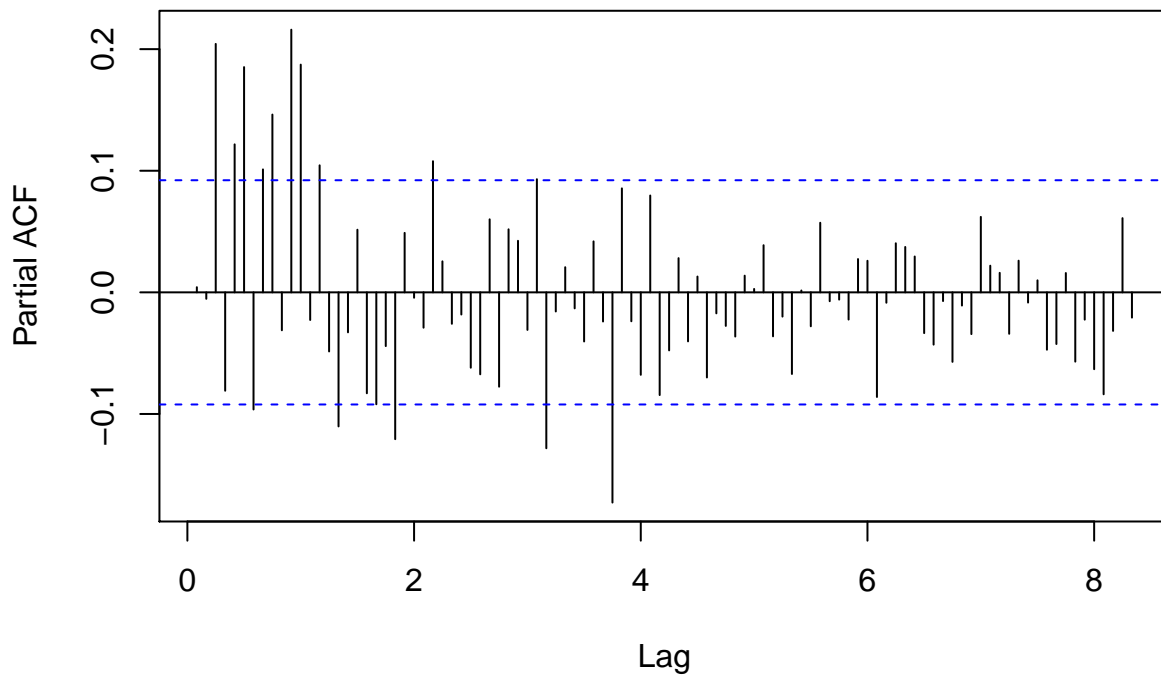
```
# p & q
par(mfrow=c(1,1))
acf(logSeasPoly_resid, lag.max=100)
```

Series logSeasPoly_resid



```
pacf(logSeasPoly_resid, lag.max=100)
```

Series logSeasPoly_resid



```
par(mfrow=c(1,1))
eacf(logSeasPoly_resid)
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 o o x o x x x x x o x x o x
## 1 x o x o o x o o x o x x o x
## 2 o o x o x x o x x x x x x x
## 3 x x x o o o o o o o x x x o
## 4 x x x o o o o o o o x o o
## 5 x x x x x o o o o o o x o x
## 6 x x x x x o o o o o o o o x
## 7 x x x x x o o o o o o o o x
```

Try an AR(12) model and examine residuals

```
#Set up external regressors and dummy vars
library(forecast)
monthDummies<-seasonaldummy(logBeer)
externReg<-data.frame(t, t2, t3, t4, monthDummies)

ar12_poly<-arima(logBeer, order=c(12,0,0), xreg=externReg)
ar12_poly
```

```
##
## Call:
## arima(x = logBeer, order = c(12, 0, 0), xreg = externReg)
##
## Coefficients:
##      ar1      ar2      ar3      ar4      ar5      ar6      ar7      ar8
##    -0.0185 -0.0373  0.0592 -0.0600  0.0852  0.1086 -0.0873  0.0867
## s.e.   0.0460  0.0449  0.0449  0.0441  0.0443  0.0442  0.0443  0.0438
##      ar9      ar10      ar11      ar12 intercept      t      t2      t3      t4
##    0.1421 -0.0077  0.2301  0.2099      4.6792 -0.0009  0e+00  0  0
## s.e.  0.0440  0.0440  0.0440  0.0453      0.0781  0.0014  2e-04  0  0
##      Jan      Feb      Mar      Apr      May      Jun      Jul
##    -0.1923 -0.2566 -0.1788 -0.2787 -0.3181 -0.4344 -0.3506
## s.e.   0.0154  0.0170  0.0156  0.0165  0.0167  0.0156  0.0167
##      Aug      Sep      Oct      Nov
##    -0.3012 -0.2658 -0.1259 -0.0756
## s.e.   0.0165  0.0157  0.0171  0.0155
##
## sigma^2 estimated as 0.003593:  log likelihood = 629.73,  aic = -1203.47
```

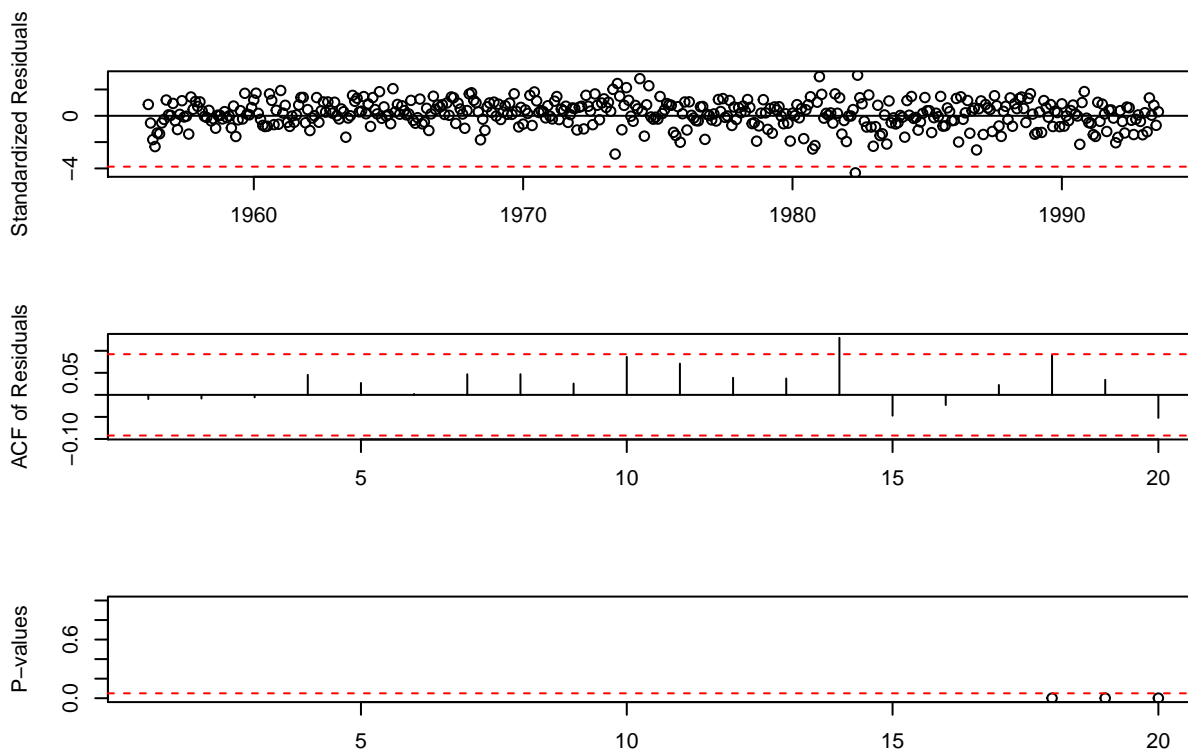
We seem to be having trouble getting fits for the trend line, ask about this Monday, try just using the month dummies.

```
ar12<-arima(logBeer, order=c(12,0,0), xreg=monthDummies)
ar12
```

```
##
## Call:
## arima(x = logBeer, order = c(12, 0, 0), xreg = monthDummies)
##
## Coefficients:
##      ar1      ar2      ar3      ar4      ar5      ar6      ar7      ar8
##    0.0631  0.0283  0.1255 -0.0127  0.1286  0.1447 -0.0733  0.0983
## s.e.   0.0464  0.0458  0.0458  0.0457  0.0458  0.0460  0.0461  0.0459
```

```
##          ar9      ar10      ar11      ar12  intercept      Jan      Feb
##      0.1446 -0.0278  0.2102  0.1670      4.9720 -0.1925 -0.2574
## s.e.  0.0461  0.0461  0.0461  0.0471      0.2644  0.0152  0.0169
##          Mar      Apr      May      Jun      Jul      Aug      Sep
##      -0.1798 -0.2797 -0.3188 -0.4349 -0.3507 -0.3012 -0.2657
## s.e.  0.0154  0.0165  0.0167  0.0153  0.0167  0.0165  0.0155
##          Oct      Nov
##      -0.126 -0.0757
## s.e.  0.017  0.0153
##
## sigma^2 estimated as 0.003946:  log likelihood = 606.68,  aic = -1165.36
```

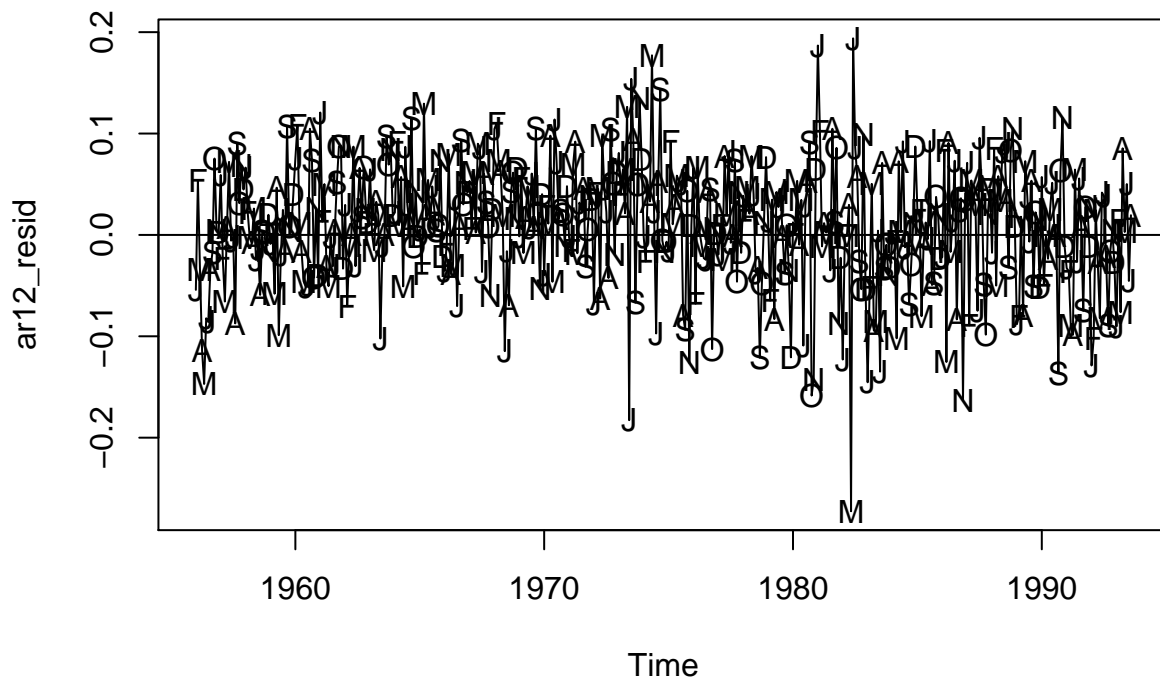
```
tsdiag(ar12, gof.lag=20)
```



```
#residuals
```

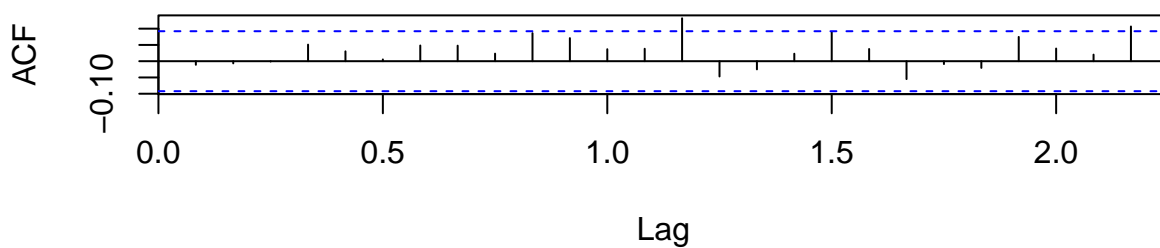
```
ar12_resid<-ts(residuals(ar12), frequency=12, start=c(1956,1))
plot(ar12_resid, main="AR 12 model Residuals from Logged Beer\nseasonal Means and 4th order poly fit at same t
points(y=ar12_resid, x=time(ar12_resid), pch=as.vector(season(ar12_resid)))
abline(h=0)
```

AR 12 model Residuals from Logged Beer seasonal Means and 4th order poly fit at same time

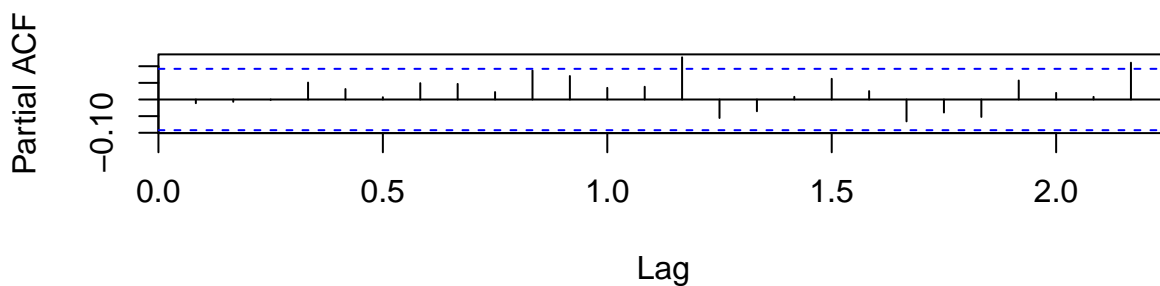


```
par(mfrow=c(2,1))
acf(ar12_resid)
pacf(ar12_resid)
```

Series ar12_resid



Series ar12_resid

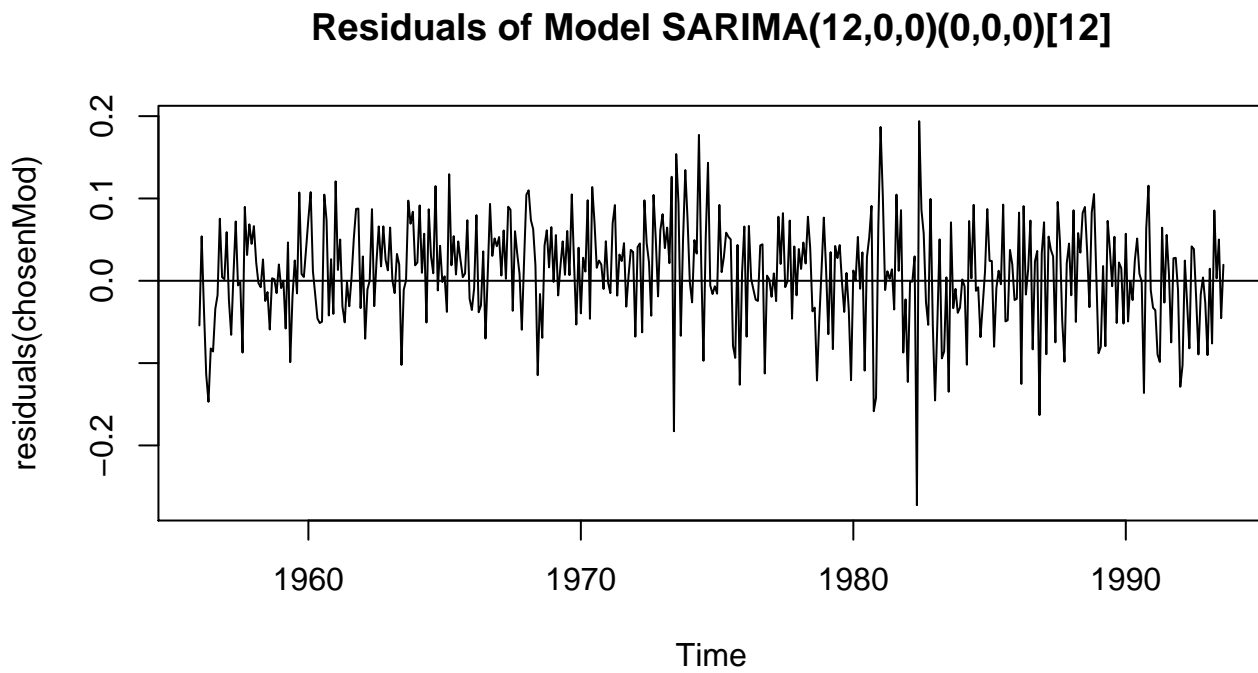


```
pacf_acf<-data.frame(acfVal=acf(ar12_resid, plot=FALSE)$acf, pacfVal=pacf(ar12_resid, plot=FALSE)$acf)
#print(pacf_acf)
```

After we choose a model, run all of the diagnostic tests

```
chosenMod<-ar12
modelString<-"SARIMA(12,0,0)(0,0,0)[12]"

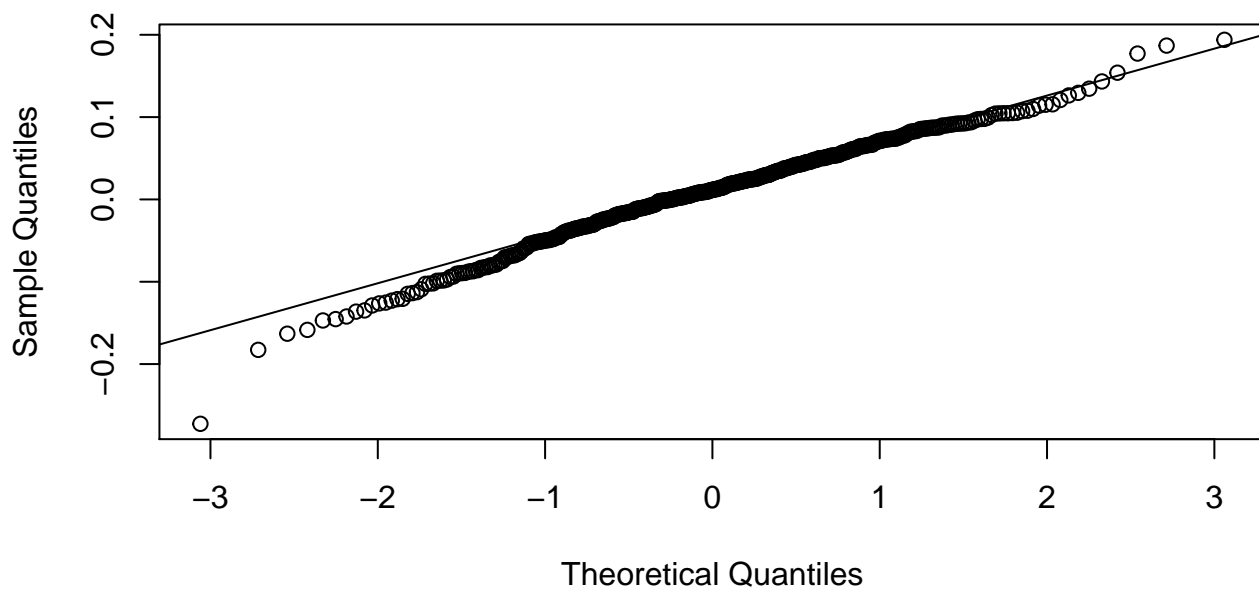
par(mfrow=c(1,1))
plot(residuals(chosenMod), main=paste("Residuals of Model", modelString))
abline(h=0)
```



Comment:

```
par(mfrow=c(1,1))
qqnorm(residuals(chosenMod), main=paste("Normal QQ Plot of Residuals from", modelString))
qqline(residuals(chosenMod))
```

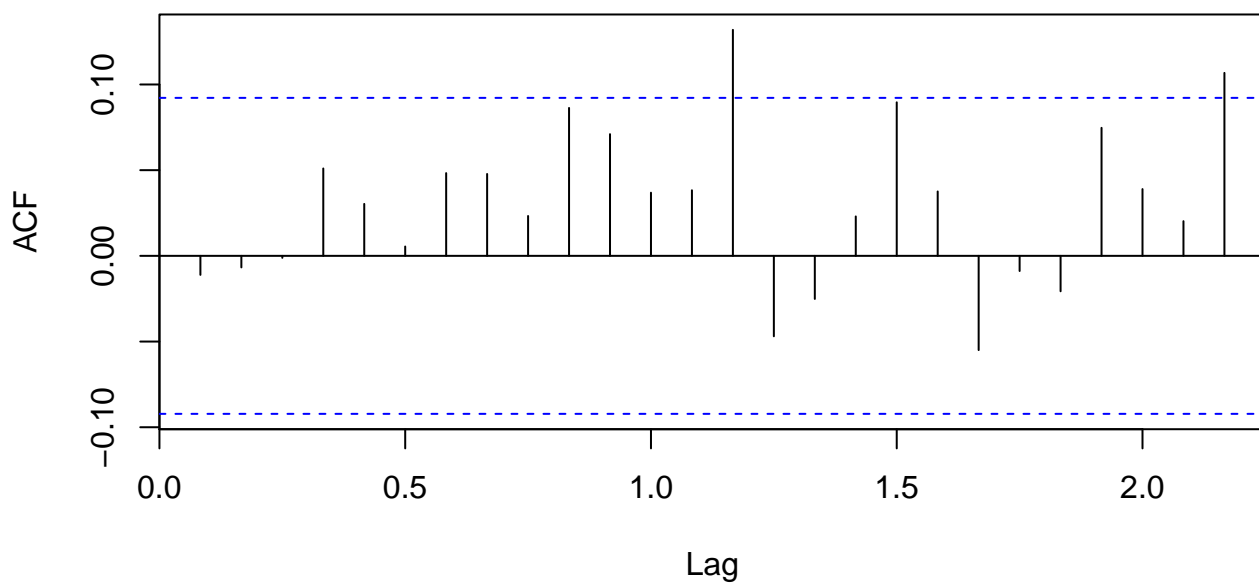
Normal QQ Plot of Residuals from SARIMA(12,0,0)(0,0,0)[12]



Comment:

```
par(mfrow=c(1,1))  
acf(residuals(chosenMod), main=paste("ACF of Residuals from", modelString))
```

ACF of Residuals from SARIMA(12,0,0)(0,0,0)[12]



Comment:

```
shapiro.test(residuals(chosenMod))
```

##


```
## Shapiro-Wilk normality test
##
## data: residuals(chosenMod)
## W = 0.98657, p-value = 0.000348
```

Comment:

```
LB.test(chosenMod, lag=24)
```

```
##
## Box-Ljung test
##
## data: residuals from chosenMod
## X-squared = 30.485, df = 12, p-value = 0.002359
```

Comment:

Make the forecasts

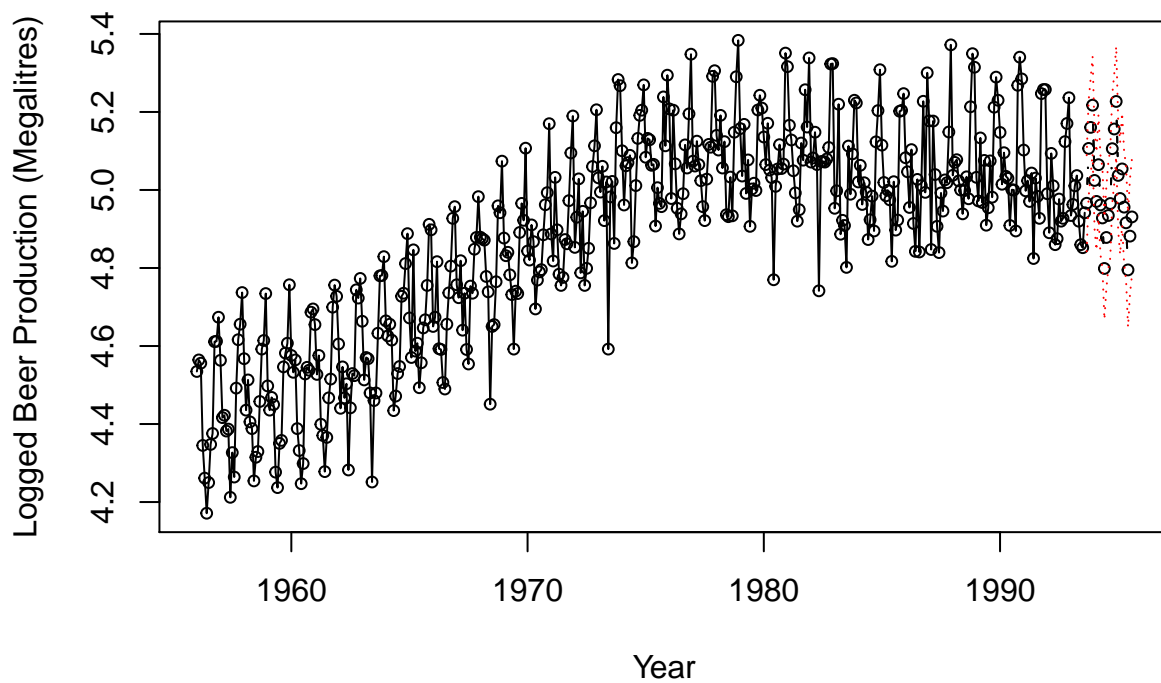
Set up external regressor data frame

```
newMonthDummy<-seasonaldummy(beer_forecast)
```

Plot the model forecasts

```
library(TSA)
```

```
TSA::plot.Arima(chosenMod,n.ahead=24,n1=c(1956,1), newxreg=newMonthDummy,
type='b',ylab='Logged Beer Production (Megalitres)',xlab='Year', col="red", lty=2, cex=.75)
```



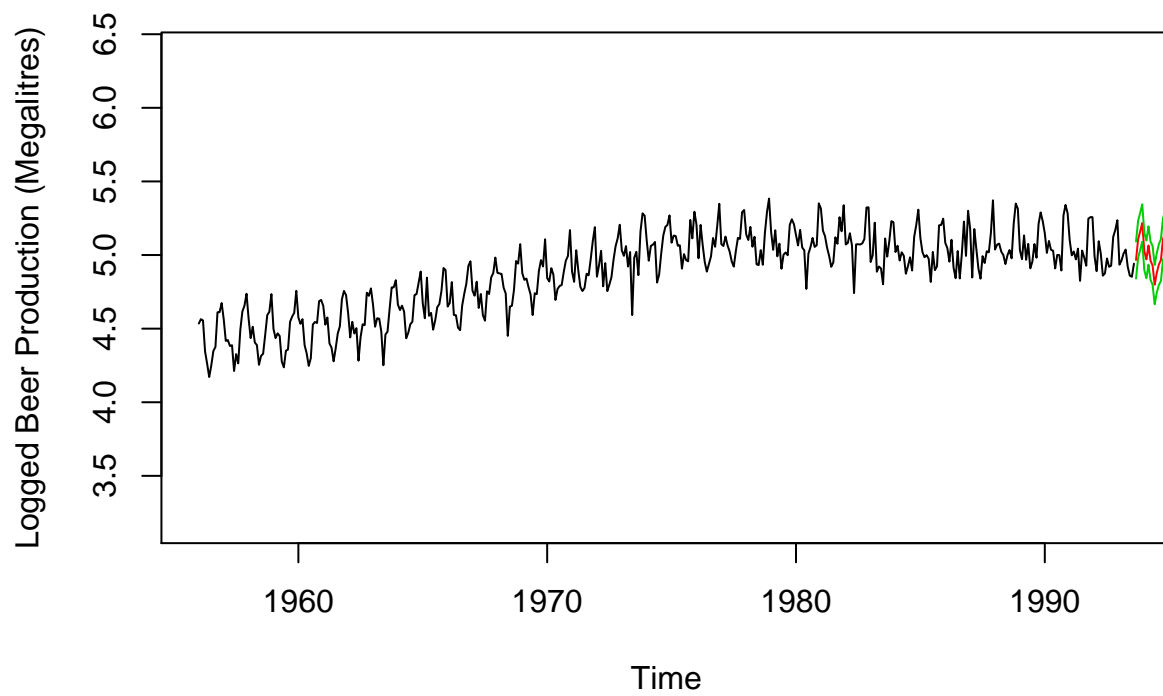
```

predictions<-predict(chosenMod, newxreg=newMonthDummy, n.ahead=24)
pred<-predictions$pred
uci<-pred+2*predictions$se
lci<-pred-2*predictions$se

ymin=min(c(as.vector(lci),logBeer))-1
ymax=max(c(as.vector(uci),logBeer))+1
plot(logBeer,ylim=c(ymin,ymax),main=modelString, ylab='Logged Beer Production (Megalitres)')
lines(pred,col=2)
lines(uci,col=3)
lines(lci,col=3)

```

SARIMA(12,0,0)(0,0,0)[12]



```
ymin=min(c(as.vector(lci),logBeer))-1
ymax=max(c(as.vector(uci),logBeer))+1
plot(logBeer,xlim=c(1993, 1996), ylim=c(4.5,5.5),main=modelString, ylab='Logged Beer Production (Megalitres)')
lines(pred,col=2)
lines(uci,col=3)
lines(lci,col=3)
lines(log(beer_forecast), col="black")
```

SARIMA(12,0,0)(0,0,0)[12]

