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

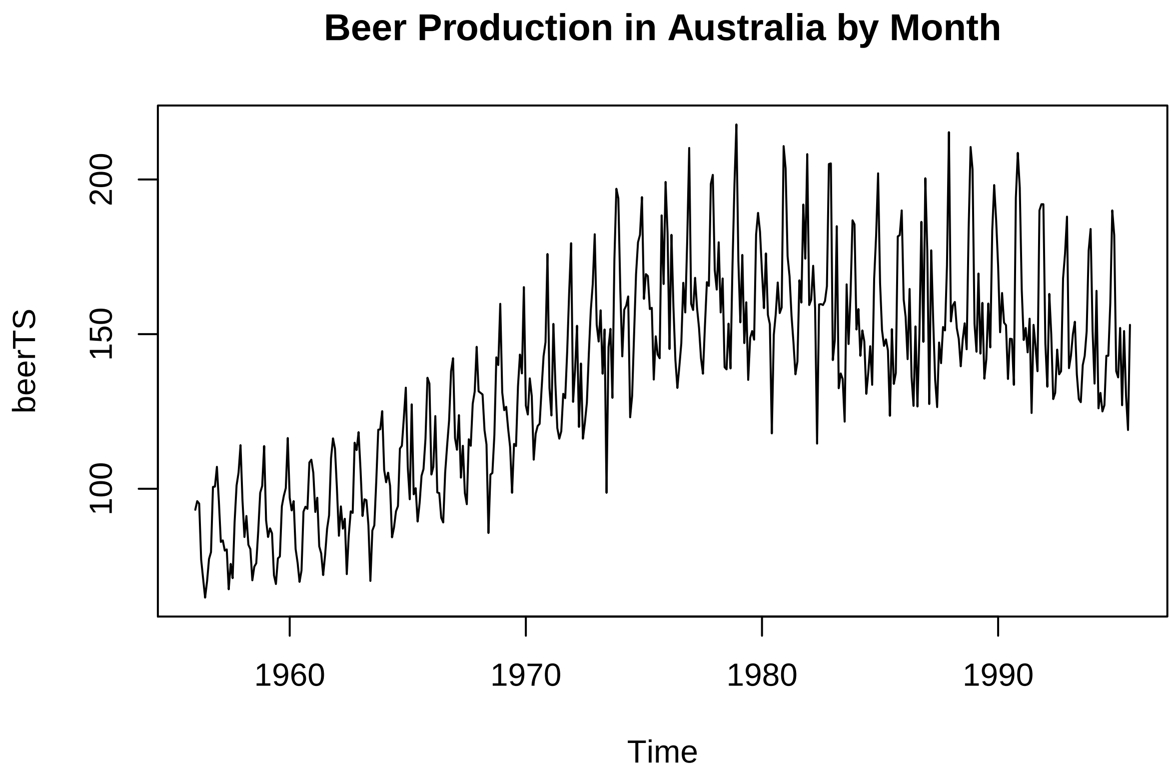
Load data (assumes file is in working directory)

```
#load the data
beerData<-read.csv("monthly-beer-production-in-austr.csv")
#
#turn into time series (cuts off last entry which is NA)
beerTS<-ts(beerData[1:(nrow(beerData)-1),2], frequency=12, start=c(1956,1))
```

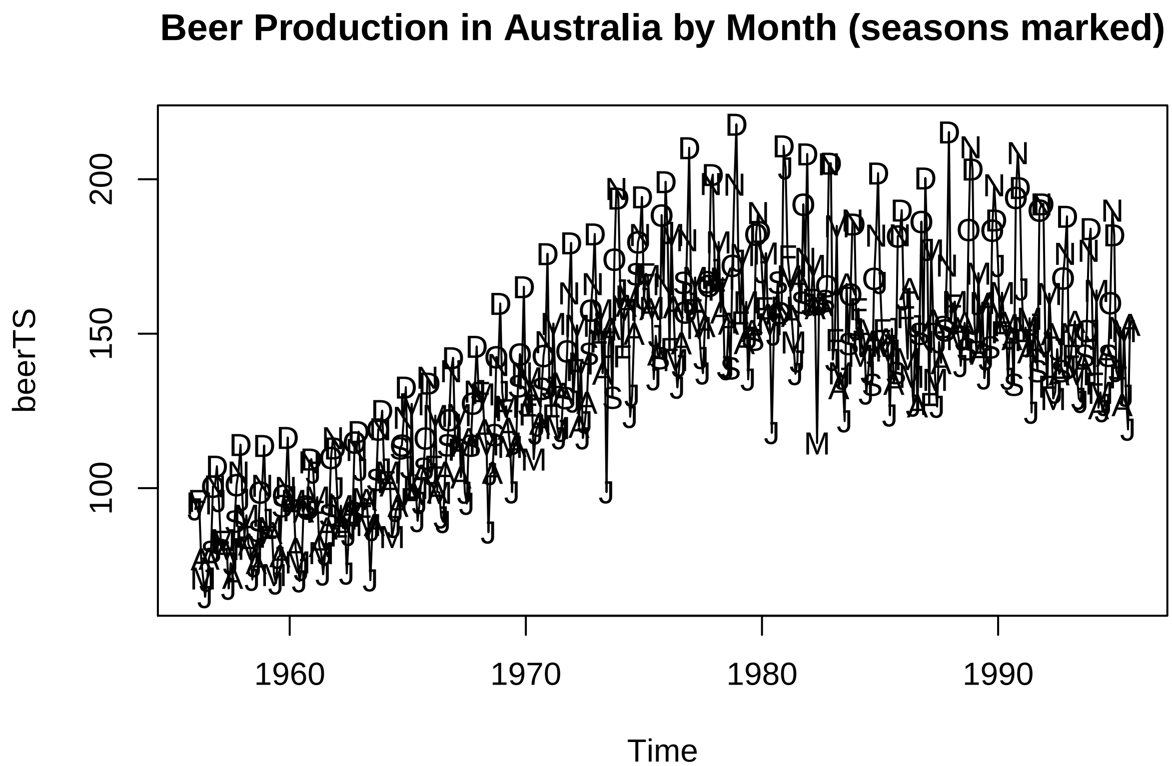
```
#load population data
#pop_totalData<-read.csv("Pop_total.csv", row.names=1)
#pop_total<-t(pop_totalData["Total",])
#pop_totalTS<-ts(pop_total[,1], frequency=1, start=c(1921))
```

Plot data

```
plot(beerTS, main="Beer Production in Australia by Month")
```




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



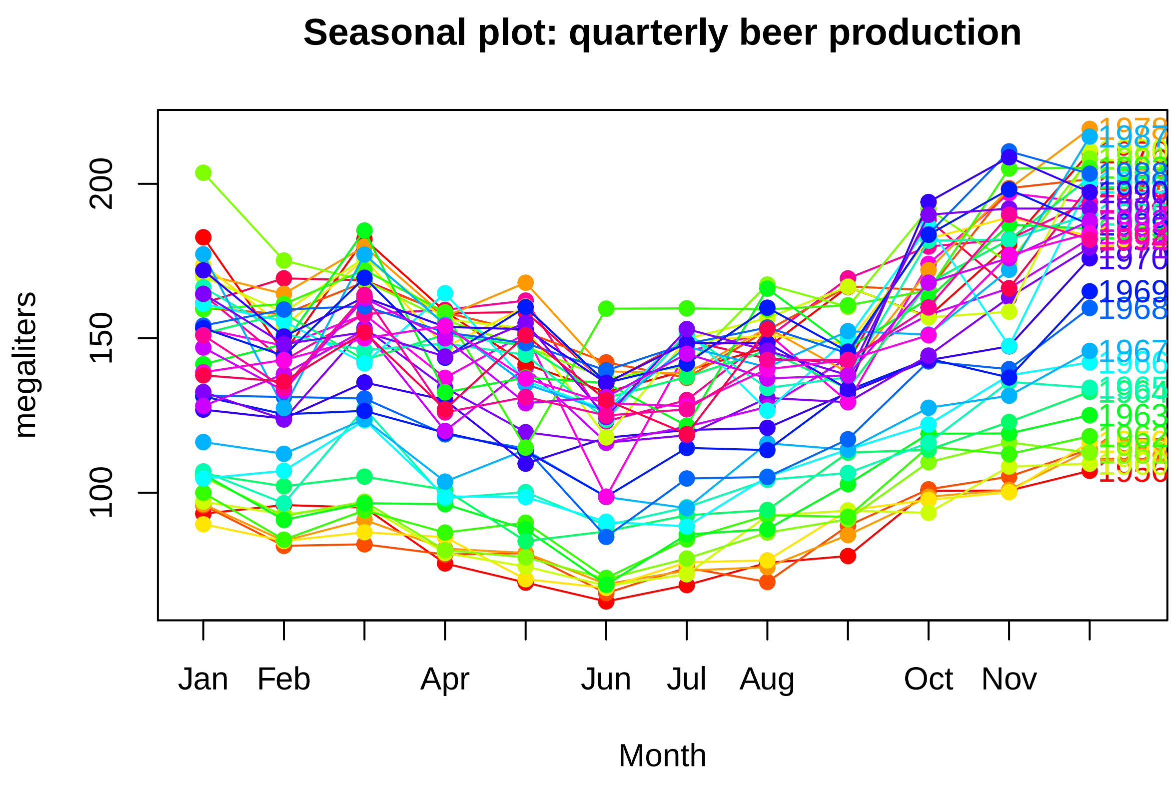
#another plot to show seasonality

```r
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
```

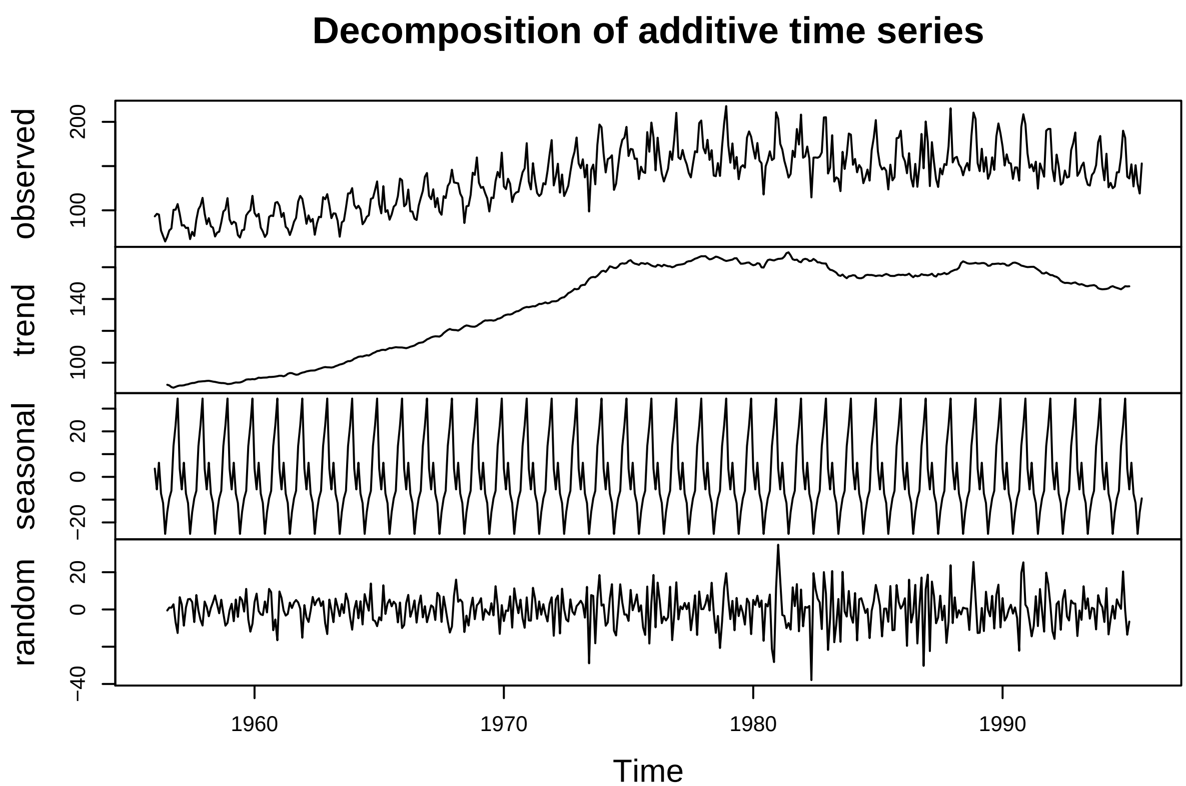
```r
seasonplot(beerTS,year.labels=TRUE,ylab="megaliters",main="Seasonal plot: quarterly beer production", c
```



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 varaince issue.

Decompsing the time series to see trends and patterns

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



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

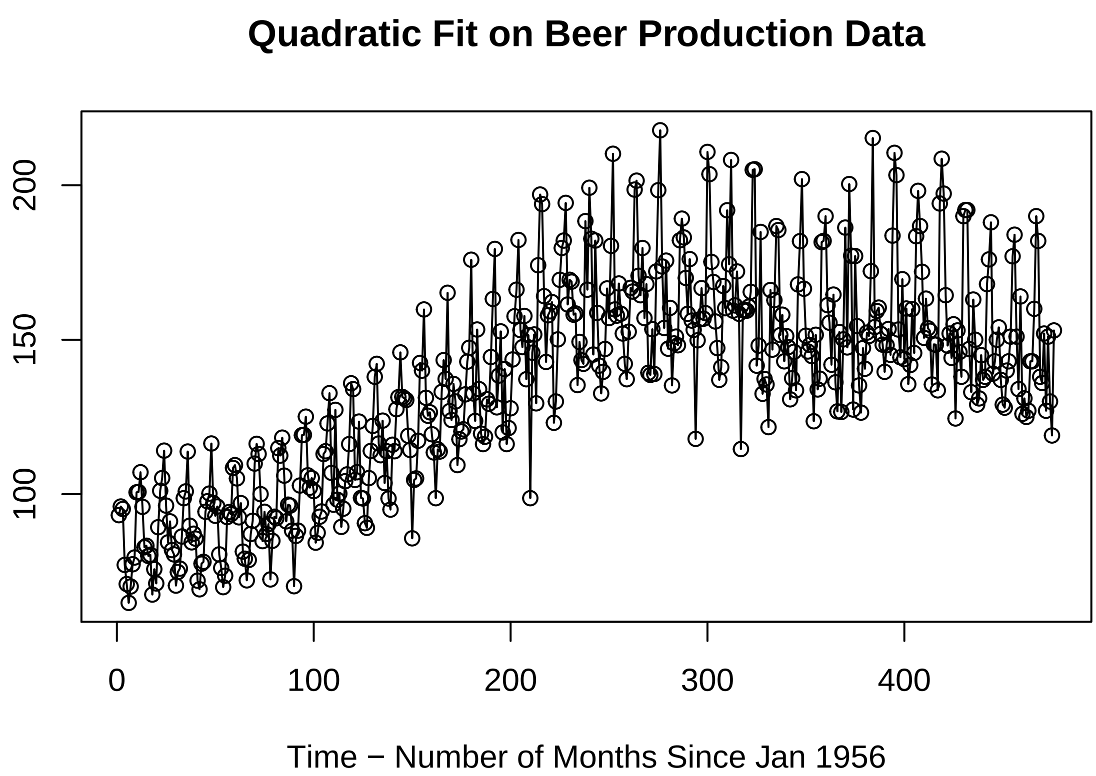
Try to figure out deterministic trend

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

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

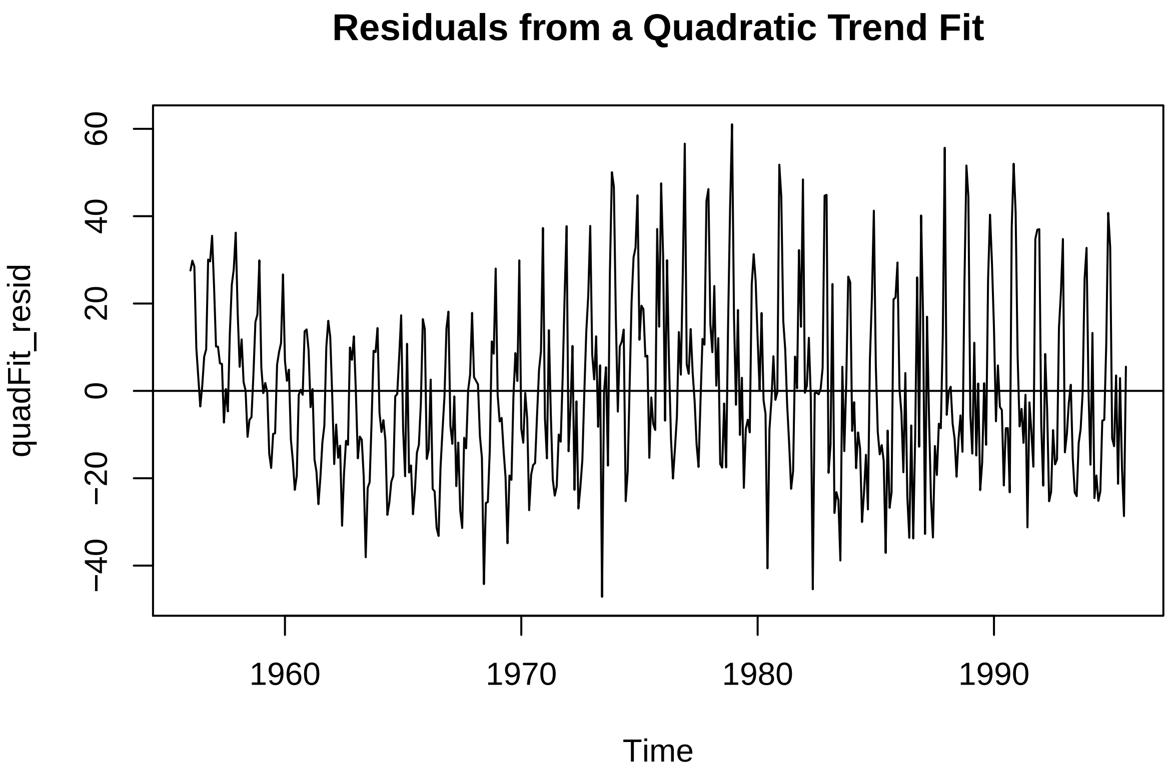
```
##
## Call:
```

```
## lm(formula = beerTS ~ t + t2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -47.157 -14.220  -2.119  11.721  61.046
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.509e+01  2.754e+00   23.64   <2e-16 ***
## t            5.517e-01  2.666e-02   20.69   <2e-16 ***
## t2          -7.954e-04  5.412e-05  -14.70   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.94 on 473 degrees of freedom
## Multiple R-squared:  0.6521, Adjusted R-squared:  0.6506
## F-statistic: 443.3 on 2 and 473 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="Qu
curve(expr = coef(quadFit)[1]+coef(quadFit)[2]*x+coef(quadFit)[3]*x^2+coef(quadFit)[4]*x^3,lty=1,add = '
```



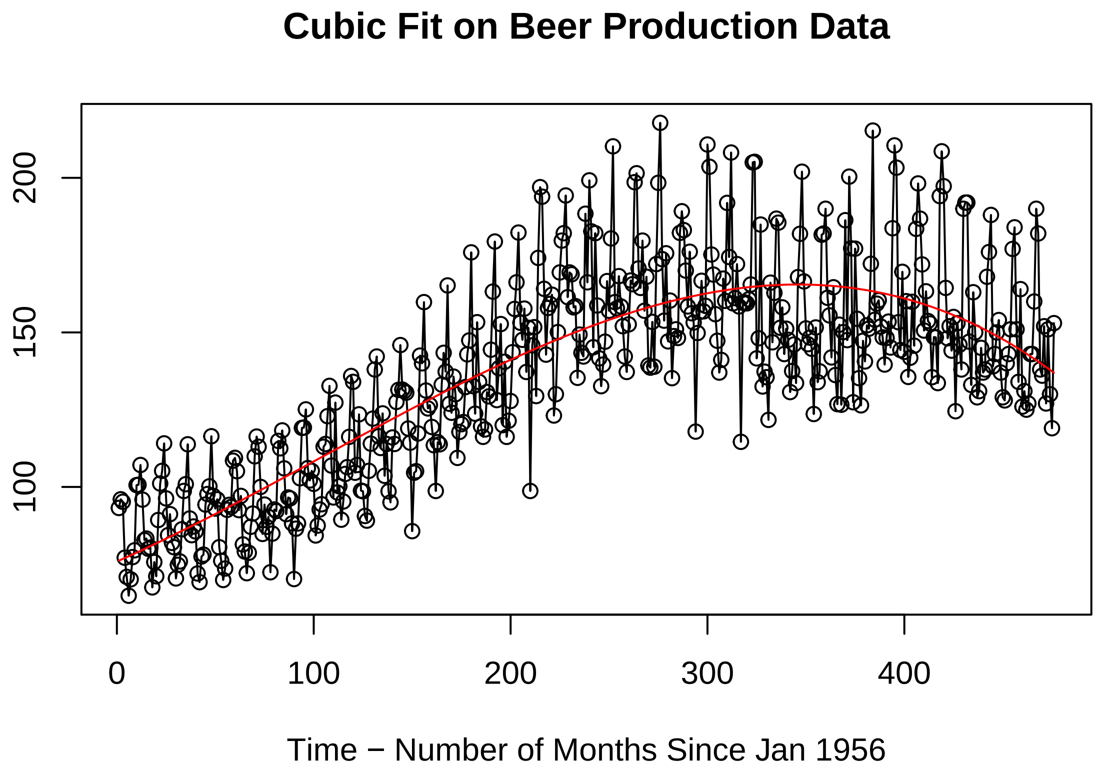

```
quadFit_resid<-ts(residuals(quadFit),frequency=12, start=c(1956,1))
plot(quadFit_resid, main="Residuals from a Quadratic Trend Fit")
abline(h=0)
```



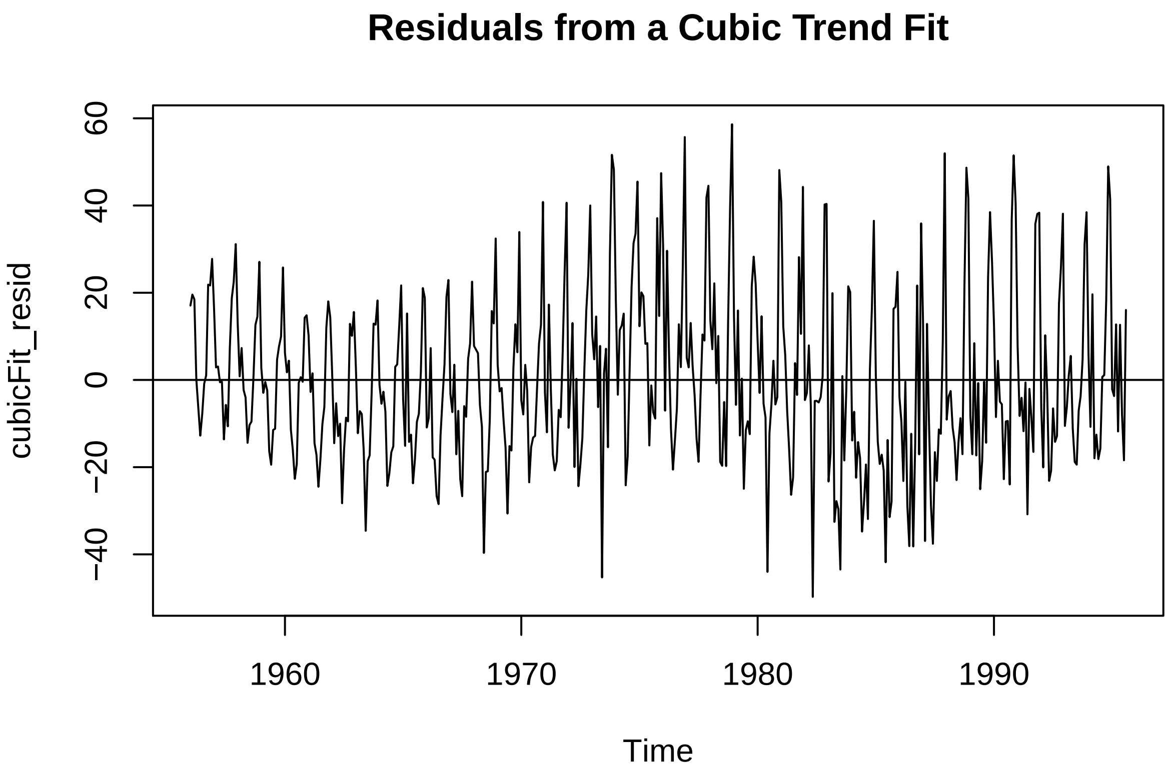

```r
cubicFit<-lm(beerTS~t+t2+t3)
summary(cubicFit)
```

```
##
## Call:
## lm(formula = beerTS ~ t + t2 + t3)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -49.744 -13.812  -2.959  12.679  58.635
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.586e+01  3.613e+00  20.998  < 2e-16 ***
## t            2.820e-01  6.552e-02   4.304 2.04e-05 ***
## t2           6.165e-04  3.190e-04   1.932   0.0539 .
## t3          -1.973e-06  4.397e-07  -4.488 9.05e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.55 on 472 degrees of freedom
## Multiple R-squared:  0.6663, Adjusted R-squared:  0.6642
## F-statistic: 314.2 on 3 and 472 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="Cu
curve(expr = coef(cubicFit)[1]+coef(cubicFit)[2]*x+coef(cubicFit)[3]*x^2+coef(cubicFit)[4]*x^3,lty=1,add
```



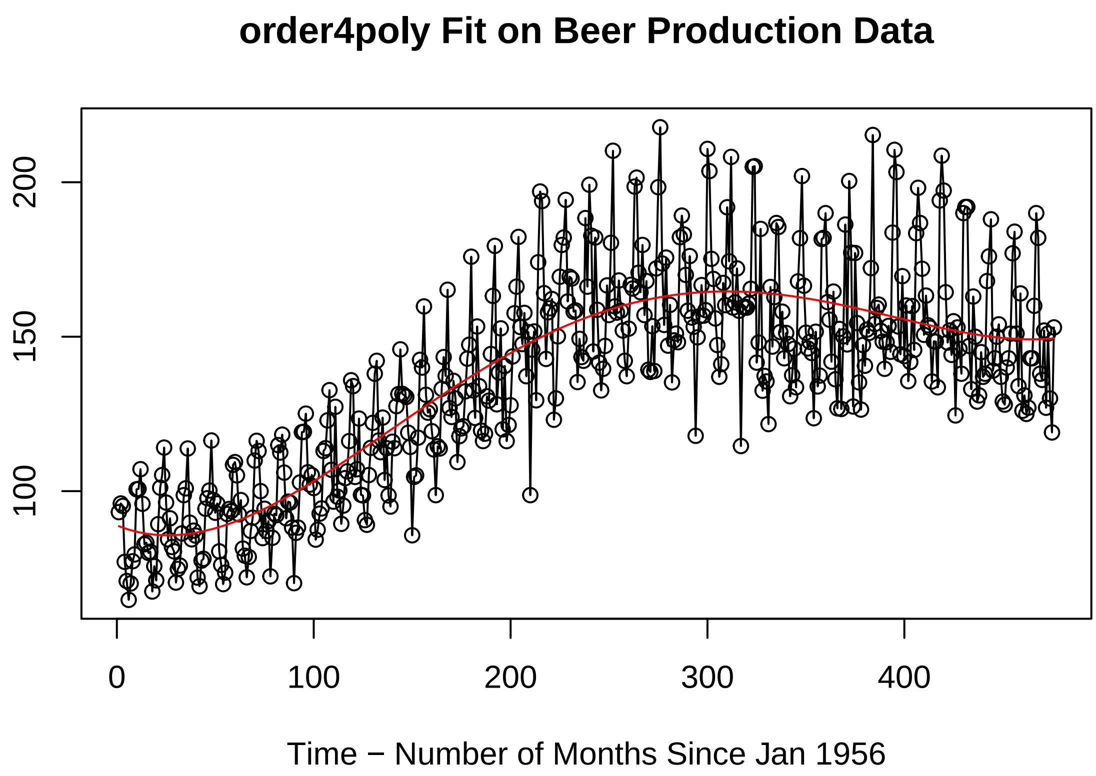

```
cubicFit_resid<-ts(residuals(cubicFit),frequency=12, start=c(1956,1))
plot(cubicFit_resid, main="Residuals from a Cubic Trend Fit")
abline(h=0)
```



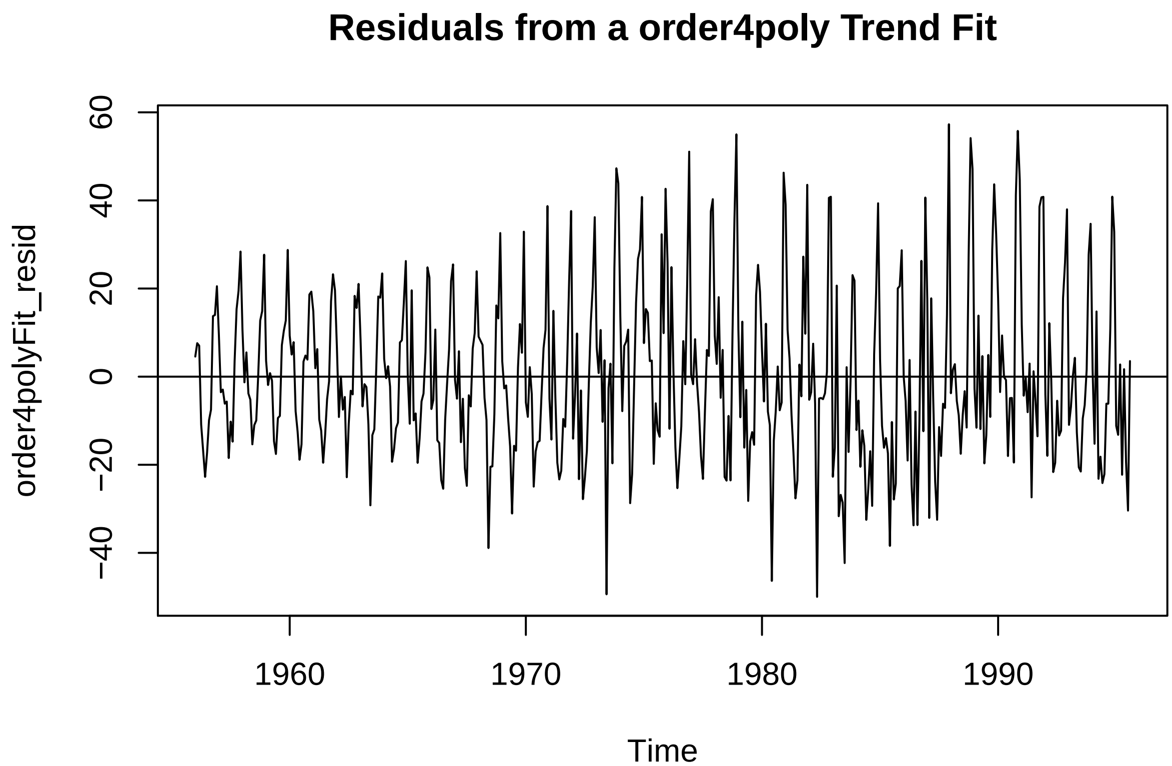

```
order4polyFit<-lm(beerTS~t+t2+t3+t4)
summary(order4polyFit)
```

```
##
## Call:
## lm(formula = beerTS ~ t + t2 + t3 + t4)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -49.98 -12.80  -3.25  10.61  57.29
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.890e+01  4.432e+00  20.058  < 2e-16 ***
## t           -2.598e-01  1.285e-01  -2.022   0.0437 *
## t2           5.715e-03  1.094e-03   5.226 2.61e-07 ***
## t3          -1.859e-05  3.443e-06  -5.399 1.06e-07 ***
## t4           1.742e-08  3.581e-09   4.864 1.57e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.1 on 471 degrees of freedom
## Multiple R-squared:  0.6823, Adjusted R-squared:  0.6796
## F-statistic: 252.9 on 4 and 471 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="o
curve(expr = coef(order4polyFit)[1]+coef(order4polyFit)[2]*x+coef(order4polyFit)[3]*x^2+coef(order4polyF
```



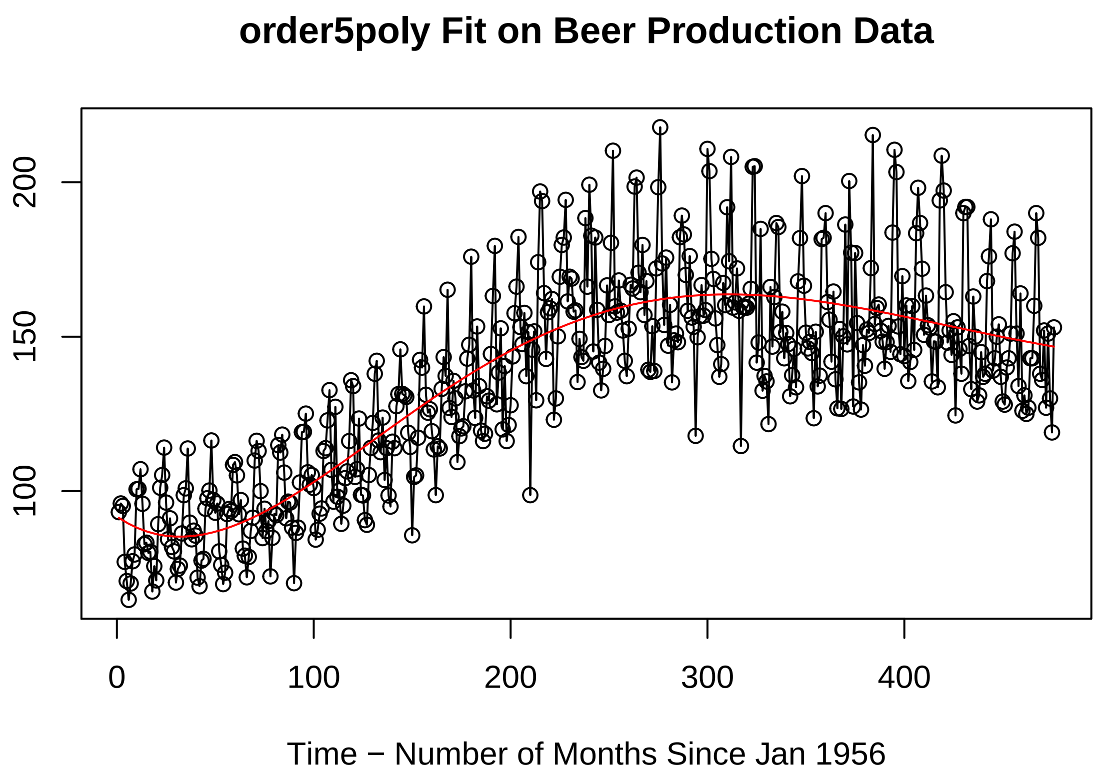

```
order4polyFit_resid<-ts(residuals(order4polyFit),frequency=12, start=c(1956,1))
plot(order4polyFit_resid, main="Residuals from a order4poly Trend Fit")
abline(h=0)
```



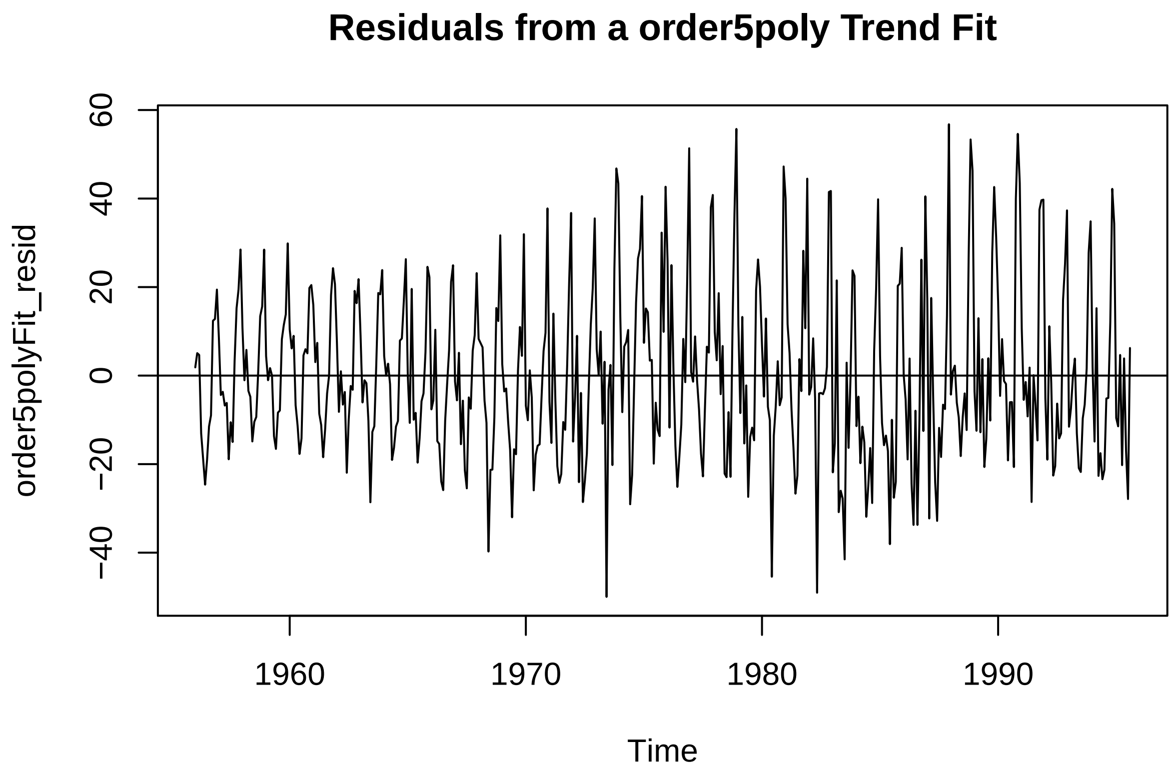

```r
order5polyFit<-lm(beerTS~t+t2+t3+t4+t5)
summary(order5polyFit)
```

```
##
## Call:
## lm(formula = beerTS ~ t + t2 + t3 + t4 + t5)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -49.99 -12.86  -3.47  10.48  56.79
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.177e+01  5.351e+00  17.152  < 2e-16 ***
## t           -4.376e-01  2.258e-01  -1.938  0.05328 .
## t2           8.313e-03  2.926e-03   2.841  0.00469 **
## t3          -3.309e-05  1.554e-05  -2.130  0.03369 *
## t4           5.160e-08  3.589e-08   1.438  0.15118
## t5          -2.866e-11  2.995e-11  -0.957  0.33896
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.1 on 470 degrees of freedom
## Multiple R-squared:  0.6829, Adjusted R-squared:  0.6795
## F-statistic: 202.4 on 5 and 470 DF,  p-value: < 2.2e-16
```

```r
#### 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="o
curve(expr = coef(order5polyFit)[1]+coef(order5polyFit)[2]*x+coef(order5polyFit)[3]*x^2+coef(order5polyF
```




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



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?)

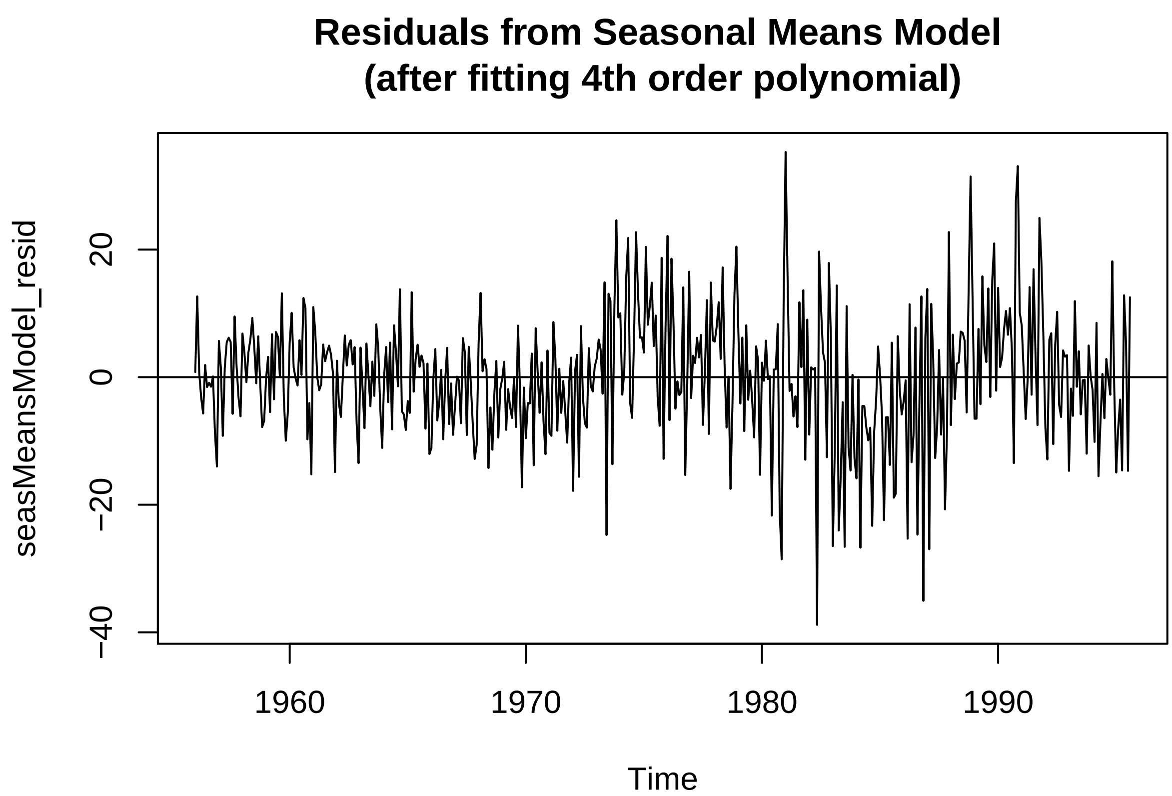
Assume we go with the 4th order polynomial for now. Let's see what we can do about the seasonality with a seasonal means model

```r
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
## -38.833  -6.262   0.163   5.865  35.303
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)       3.762      1.619   2.324 0.020577 *
## monthFebruary    -8.838      2.290  -3.860 0.000129 ***
## monthMarch        2.520      2.290   1.100 0.271686
```

```
## monthApril      -11.413      2.290  -4.985 8.79e-07 ***
## monthMay        -14.914      2.290  -6.513 1.91e-10 ***
## monthJune       -28.417      2.290 -12.411  < 2e-16 ***
## monthJuly       -19.470      2.290  -8.503 2.55e-16 ***
## monthAugust     -12.758      2.290  -5.572 4.28e-08 ***
## monthSeptember   -9.781      2.304  -4.245 2.64e-05 ***
## monthOctober      9.852      2.304   4.276 2.31e-05 ***
## monthNovember    18.926      2.304   8.214 2.15e-15 ***
## monthDecember    30.769      2.304  13.353  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.24 on 464 degrees of freedom
## Multiple R-squared:  0.7168, Adjusted R-squared:  0.7101
## F-statistic: 106.8 on 11 and 464 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 polynomi
abline(h=0)
```

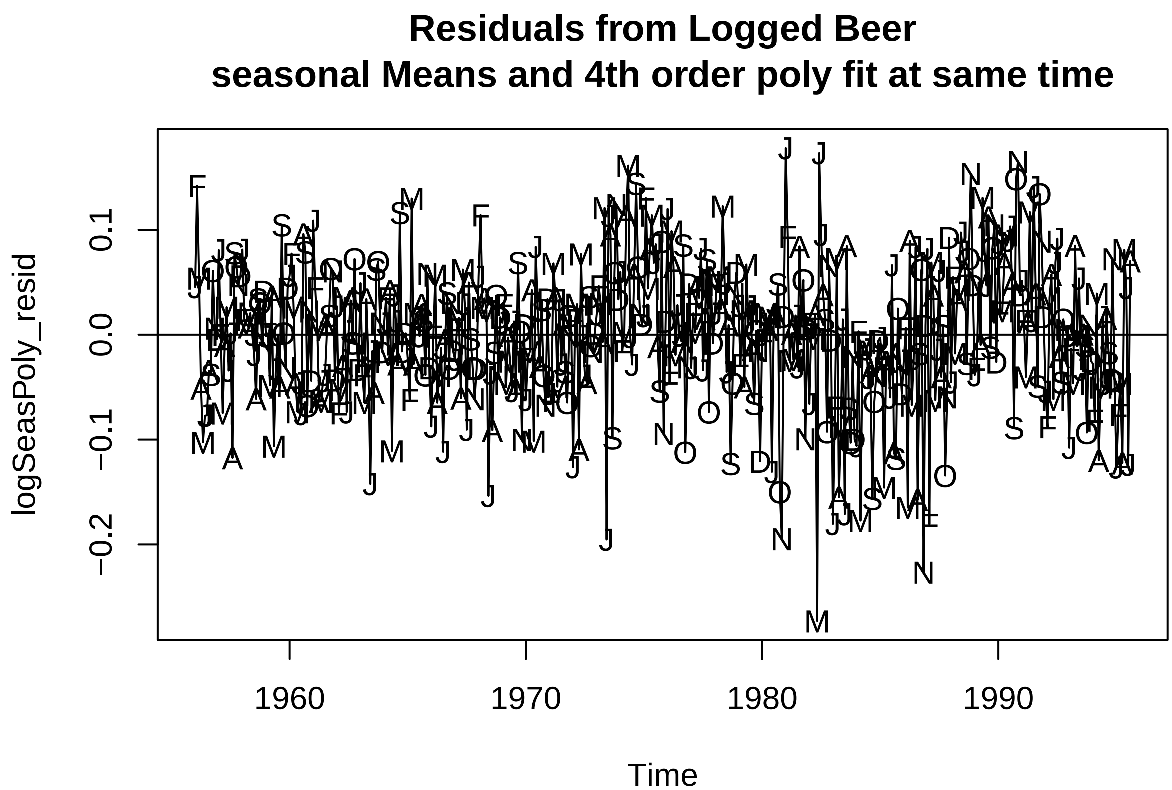


With an adjusted R-squared value of 71%, this is looking pretty good, but in the residual plot you can still the the variance increasing over time. In addition, there is a noticeable "wave" in the residuals that starts around 1970, but I'm not sure what to do about that yet. For now, let's go back, log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

```r
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
## -0.273007 -0.039226  0.002222  0.045637  0.177923
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     4.490e+00  1.918e-02 234.056  < 2e-16 ***
## t              -1.112e-03  4.694e-04  -2.368   0.0183 *
## t2              4.149e-05  3.996e-06  10.384  < 2e-16 ***
## t3             -1.446e-07  1.258e-08 -11.492  < 2e-16 ***
## t4              1.420e-10  1.308e-11  10.857  < 2e-16 ***
## monthFebruary  -6.604e-02  1.559e-02  -4.235 2.76e-05 ***
## monthMarch      1.470e-02  1.559e-02   0.943   0.3464
## monthApril     -9.043e-02  1.559e-02  -5.799 1.24e-08 ***
## monthMay       -1.219e-01  1.559e-02  -7.818 3.68e-14 ***
## monthJune      -2.366e-01  1.559e-02 -15.172  < 2e-16 ***
## monthJuly      -1.577e-01  1.560e-02 -10.114  < 2e-16 ***
## monthAugust    -1.020e-01  1.560e-02  -6.539 1.65e-10 ***
## monthSeptember -6.994e-02  1.570e-02  -4.456 1.05e-05 ***
## monthOctober    6.689e-02  1.570e-02   4.261 2.47e-05 ***
## monthNovember   1.230e-01  1.570e-02   7.834 3.30e-14 ***
## monthDecember   1.954e-01  1.570e-02  12.449  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06973 on 460 degrees of freedom
## Multiple R-squared:  0.9328, Adjusted R-squared:  0.9306
## F-statistic: 425.6 on 15 and 460 DF,  p-value: < 2.2e-16
```

```r
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
points(y=logSeasPoly_resid, x=time(logSeasPoly_resid), pch=as.vector(season(logSeasPoly_resid)))
abline(h=0)
```



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

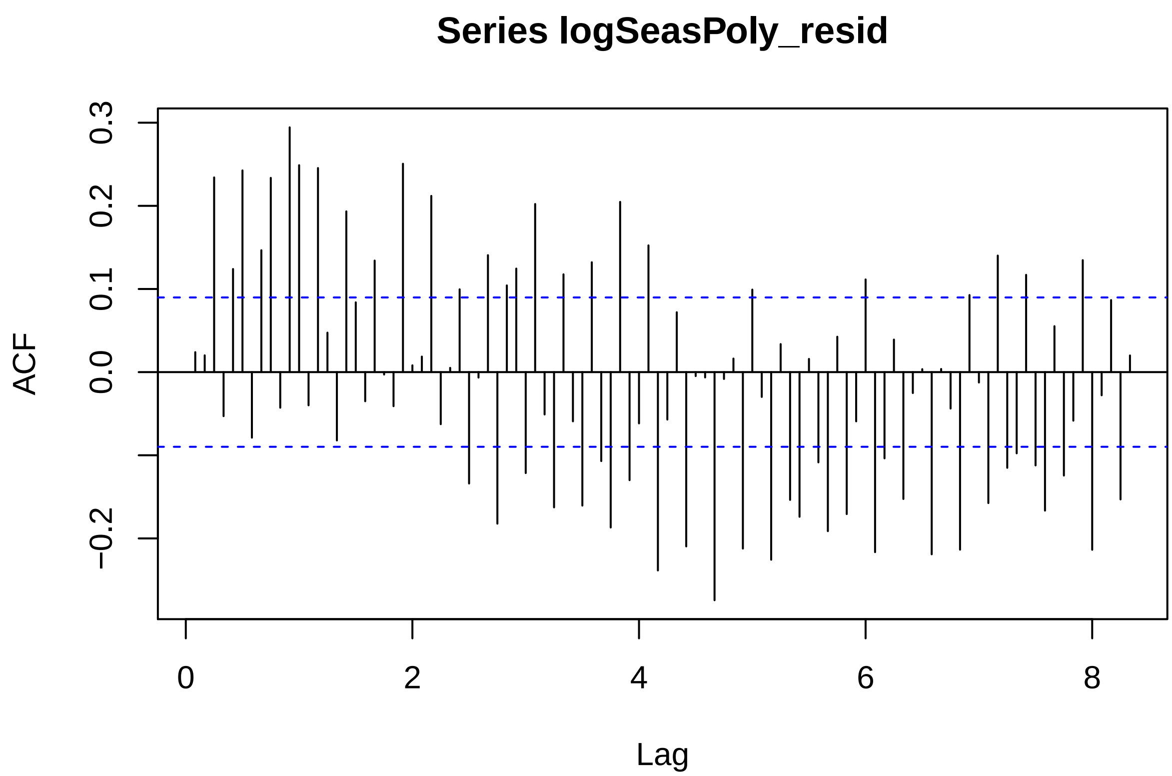
```
# d
adf.test(logSeasPoly_resid)
```

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

##
##  Augmented Dickey-Fuller Test
##
## data:  logSeasPoly_resid
## Dickey-Fuller = -5.1622, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

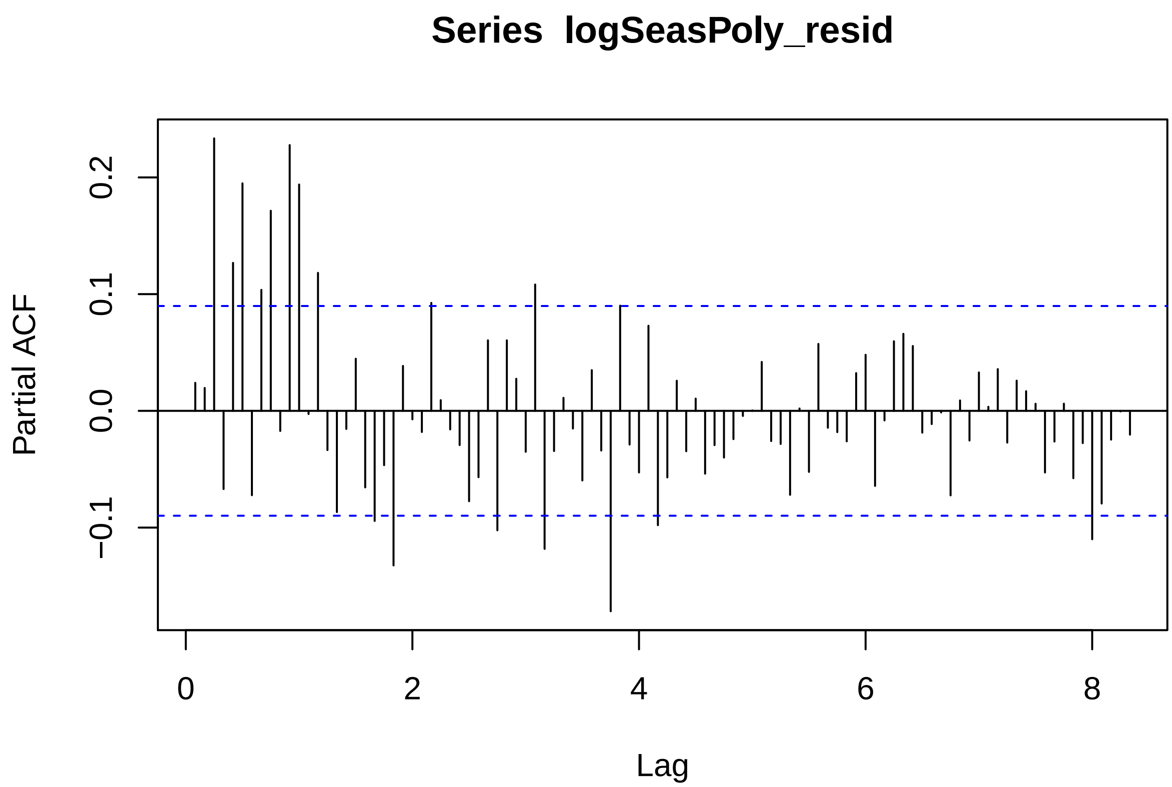
```
pp.test(logSeasPoly_resid)
```

```
## 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) = -522.68, Truncation lag parameter = 5,
## p-value = 0.01
## alternative hypothesis: stationary
```

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




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

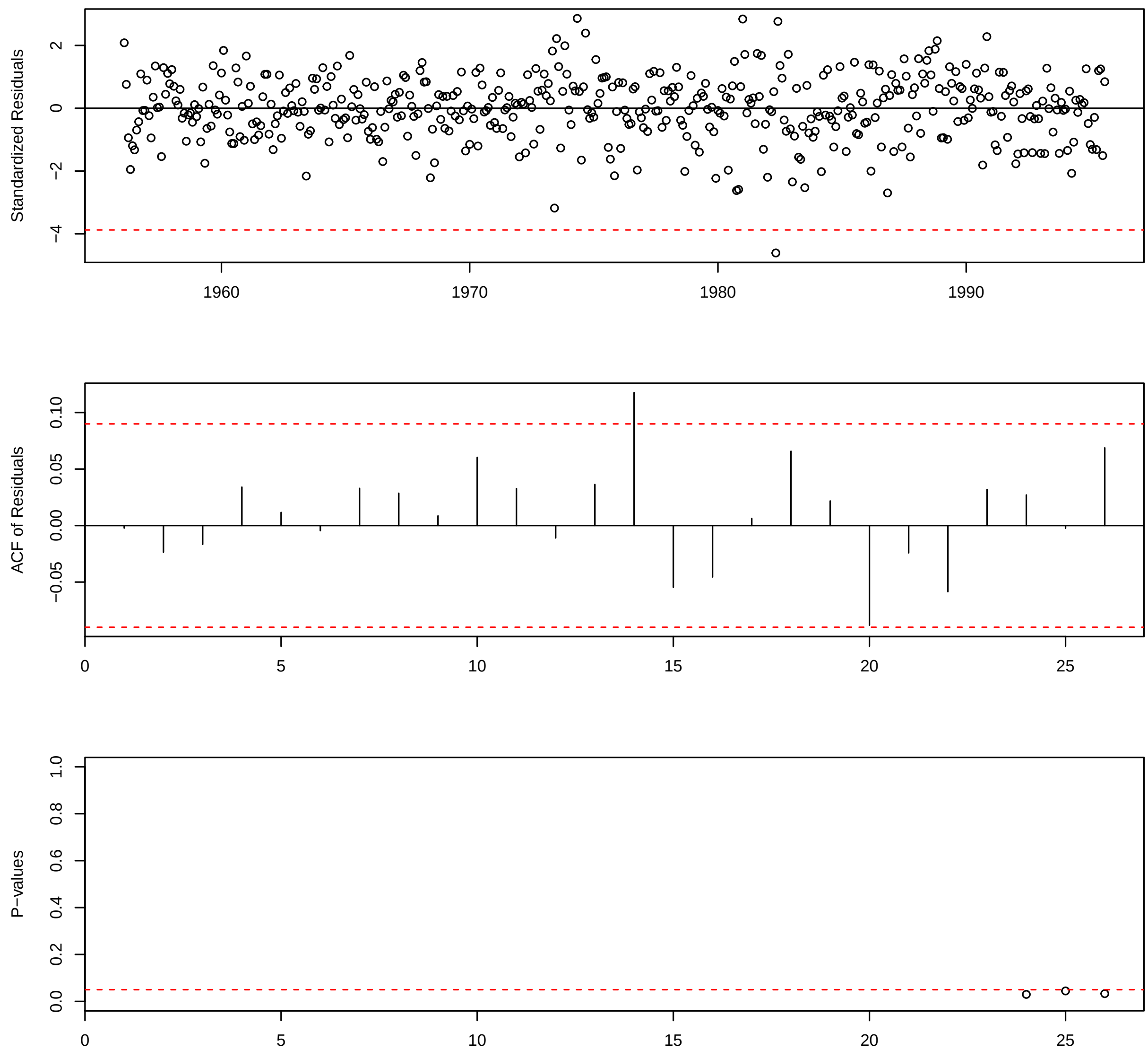



```r
par(mfrow=c(1,1))
eacf(logSeasPoly_resid)
```

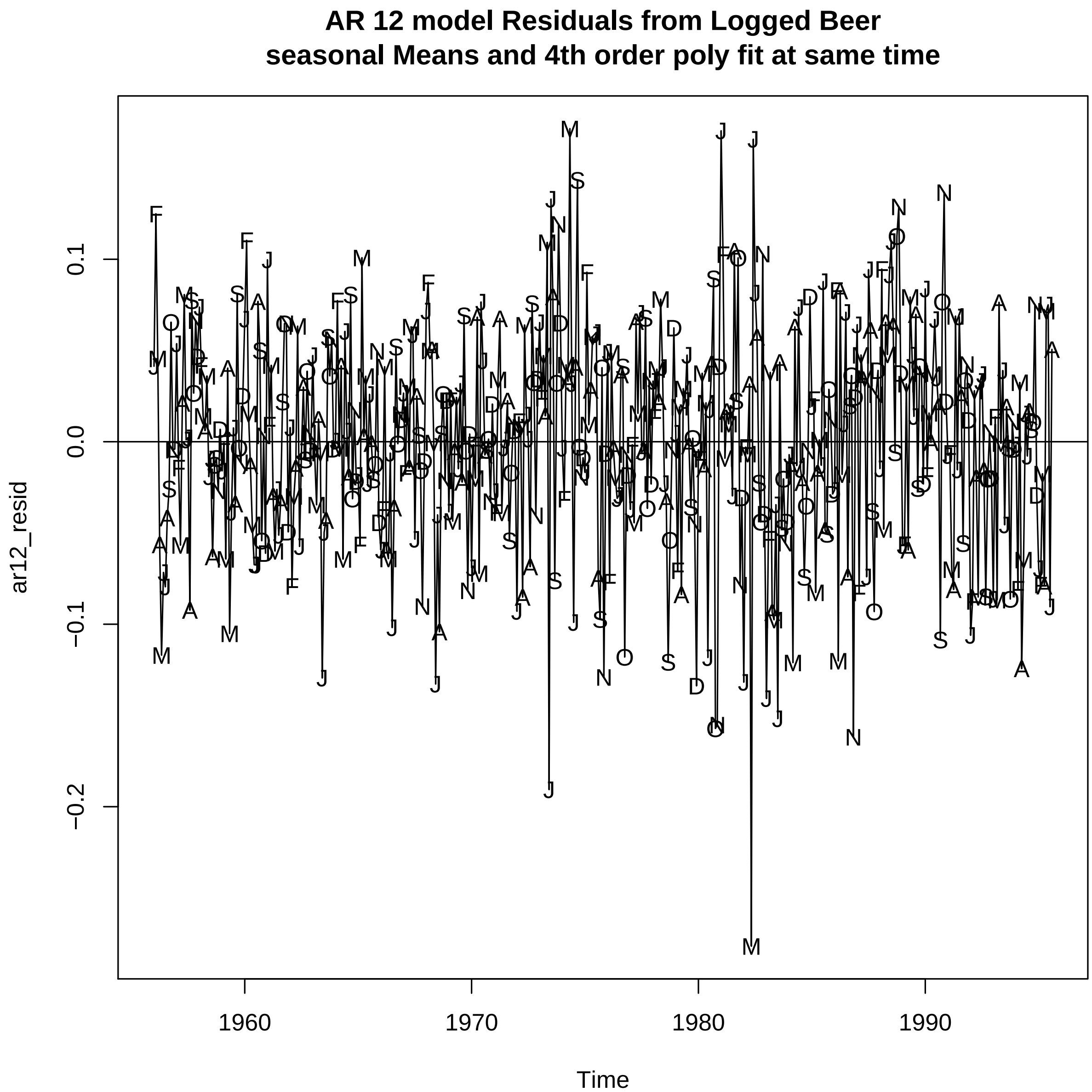
```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 o o x o x x o x x o x  x  o  x
## 1 x o x o o x o o x o x  x  o  x
## 2 o o x o x x x x x x x  x  x  x
## 3 x x x o o o o o o o x  x  x  o
## 4 x x x x o o o o o o o  x  o  o
## 5 x x x o x o o o o o o  o  o  x
## 6 x x x o o x o o o o o  o  o  x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

Try an AR(12) model and examine residuals

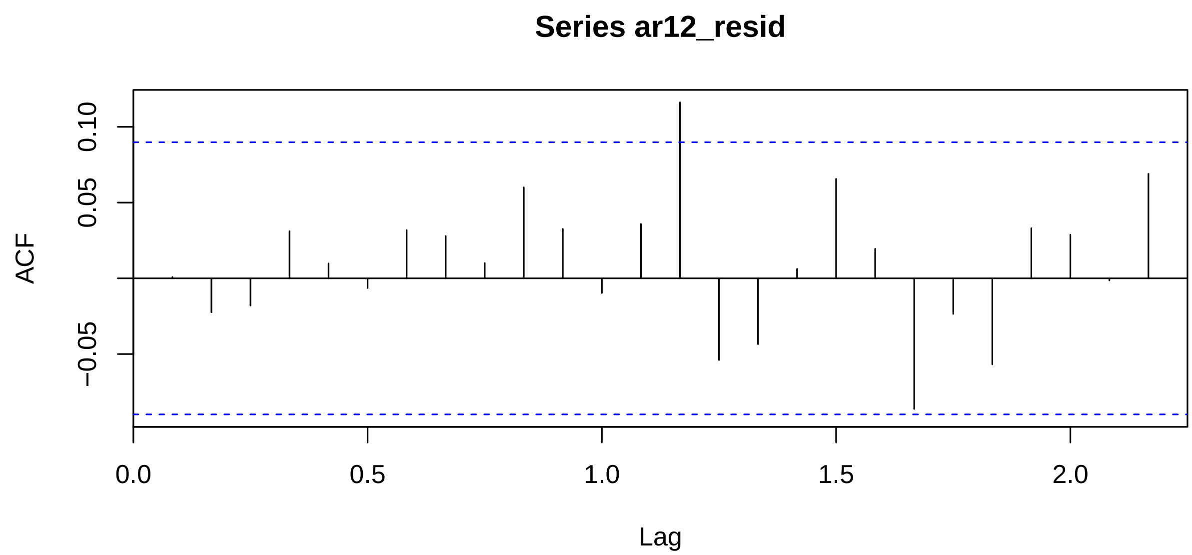
```r
ar12<-arima(logSeasPoly_resid, order=c(12,0,0))
tsdiag(ar12)
```

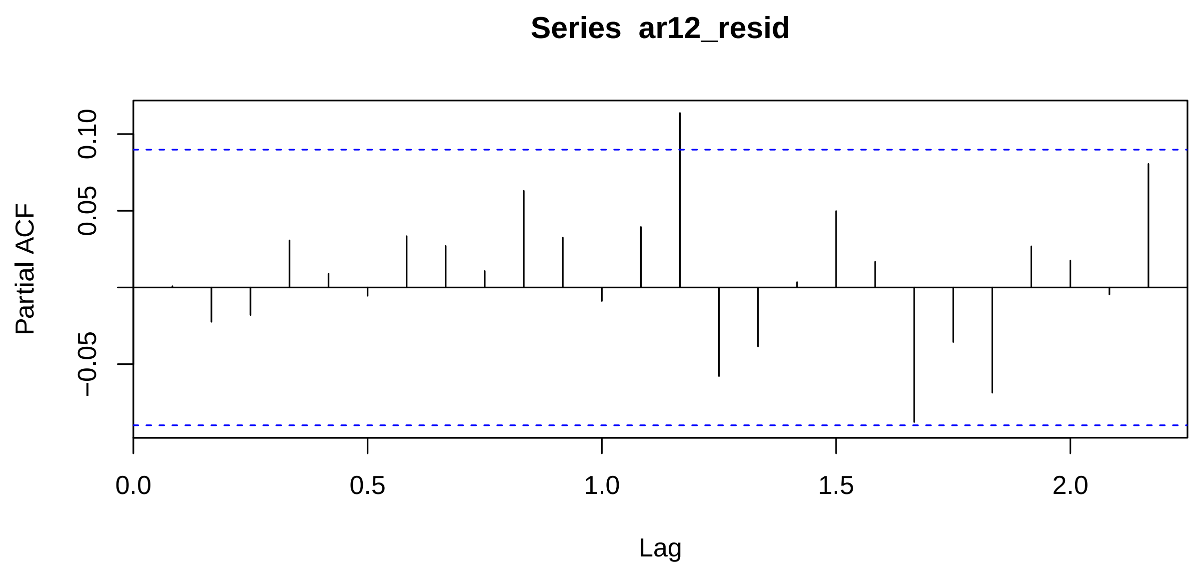



```r
#residuals
ar12_resid<-ts(residuals(ar12), frequency=12, start=c(1956,1))
plot(ar12_resid, main="AR 12 model Residuals from Logged Beer\nseasonal Means and 4th order poly fit at
points(y=ar12_resid, x=time(ar12_resid), pch=as.vector(season(ar12_resid)))
abline(h=0)
```




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






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

```
##          acfVal       pacfVal
## 1  0.0008457085  0.0008457085
## 2 -0.0223783342 -0.0223790654
## 3 -0.0179778658 -0.0179485925
## 4  0.0311454580  0.0306937786
## 5  0.0098755884  0.0090423253
## 6 -0.0064192587 -0.0054068145
## 7  0.0318540813  0.0334455994
## 8  0.0279137800  0.0270664426
## 9  0.0101096168  0.0107284139
```

```
## 10  0.0600942051  0.0629987233
## 11  0.0326258877  0.0325299802
## 12 -0.0096628825 -0.0088024988
## 13  0.0359362601  0.0394344435
## 14  0.1161355971  0.1137447927
## 15 -0.0539597875 -0.0577666411
## 16 -0.0434919566 -0.0384278098
## 17  0.0062476940  0.0034661972
## 18  0.0657053836  0.0497873953
## 19  0.0194632971  0.0168106924
## 20 -0.0862278209 -0.0876967355
## 21 -0.0234896381 -0.0355530425
## 22 -0.0568409892 -0.0686028626
## 23  0.0330801172  0.0268137564
## 24  0.0287814455  0.0175935565
## 25 -0.0014144415 -0.0045787971
## 26  0.0689969972  0.0805470391
```