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

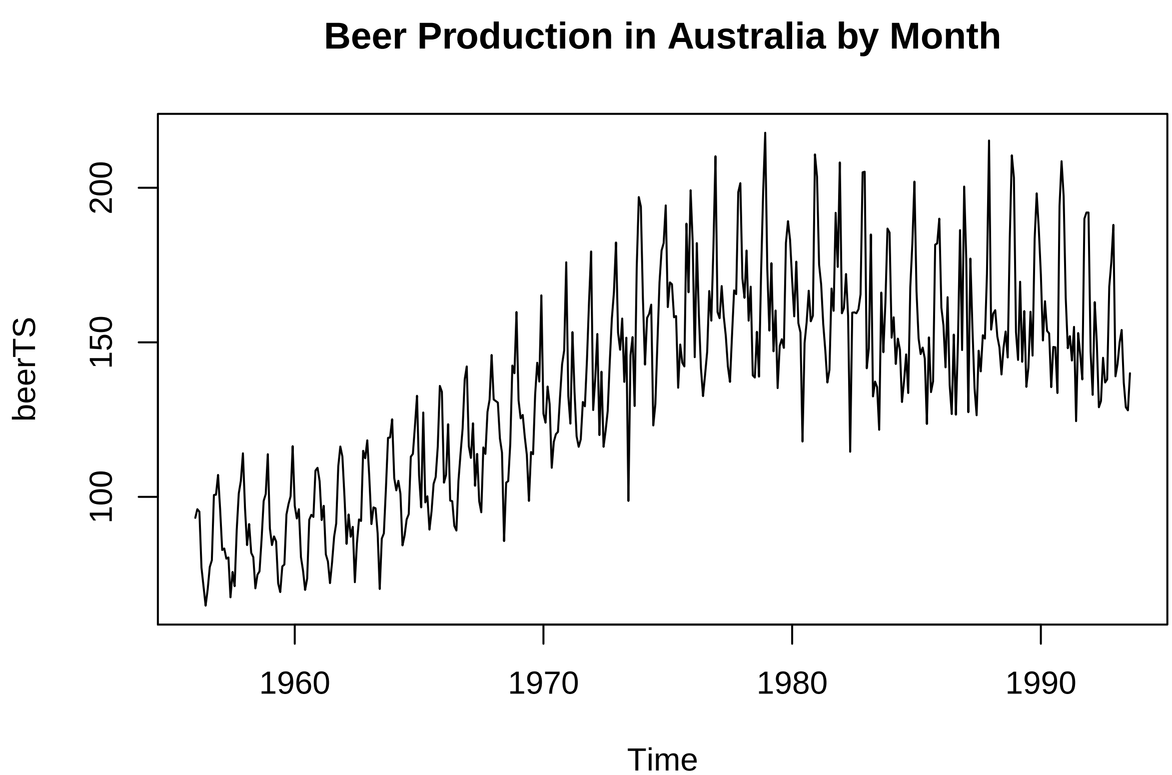
Change working directory here

Load data (assumes file is in working directory)

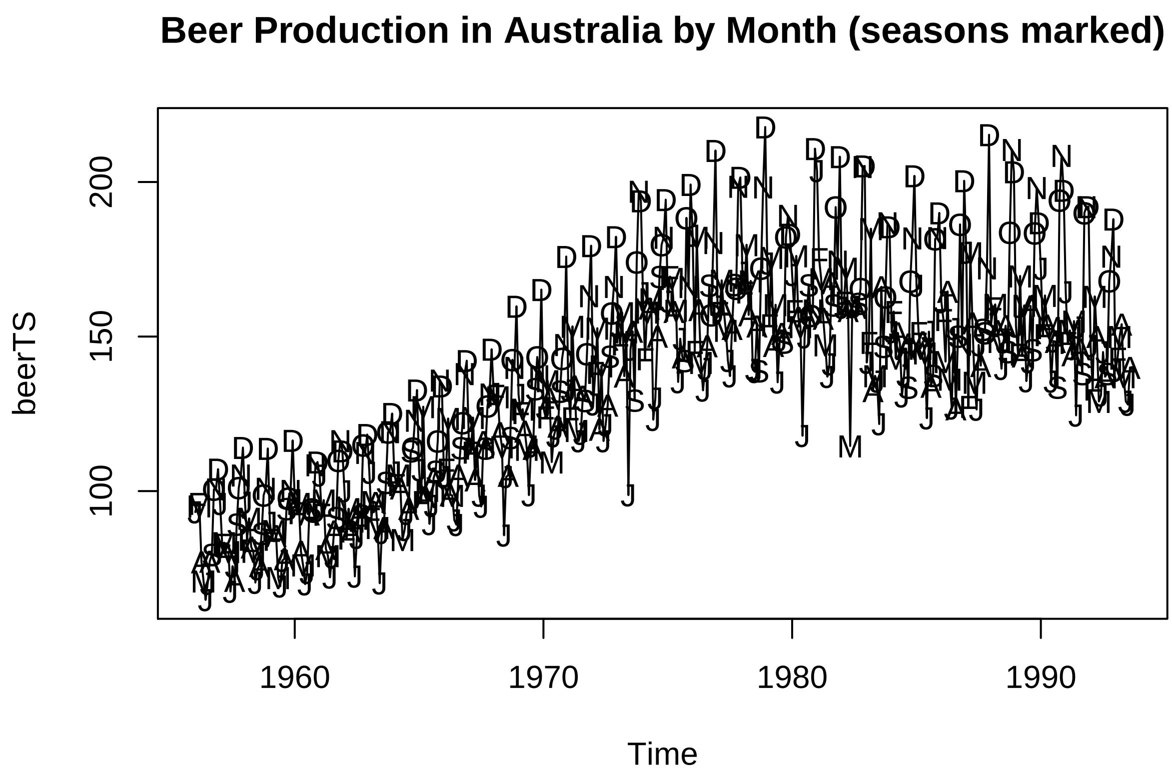
```
#load the data
beerData<-read.csv("monthly-beer-production-in-austr.csv")

#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)
```

Plot data

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

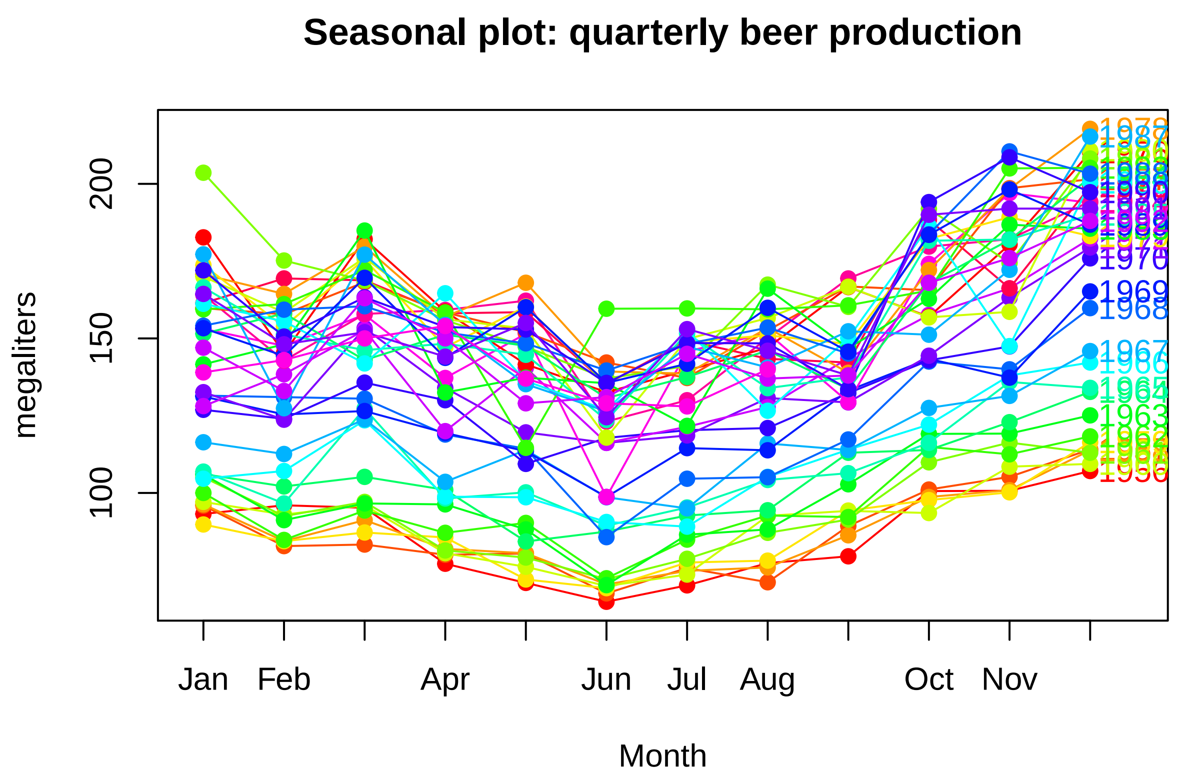


```r
plot(beerTS, main="Beer Production in Australia by Month (seasons marked)", type="l")
points(y=beerTS, x=time(beerTS), pch=as.vector(season(beerTS)))
```



Another plot to show seasonality

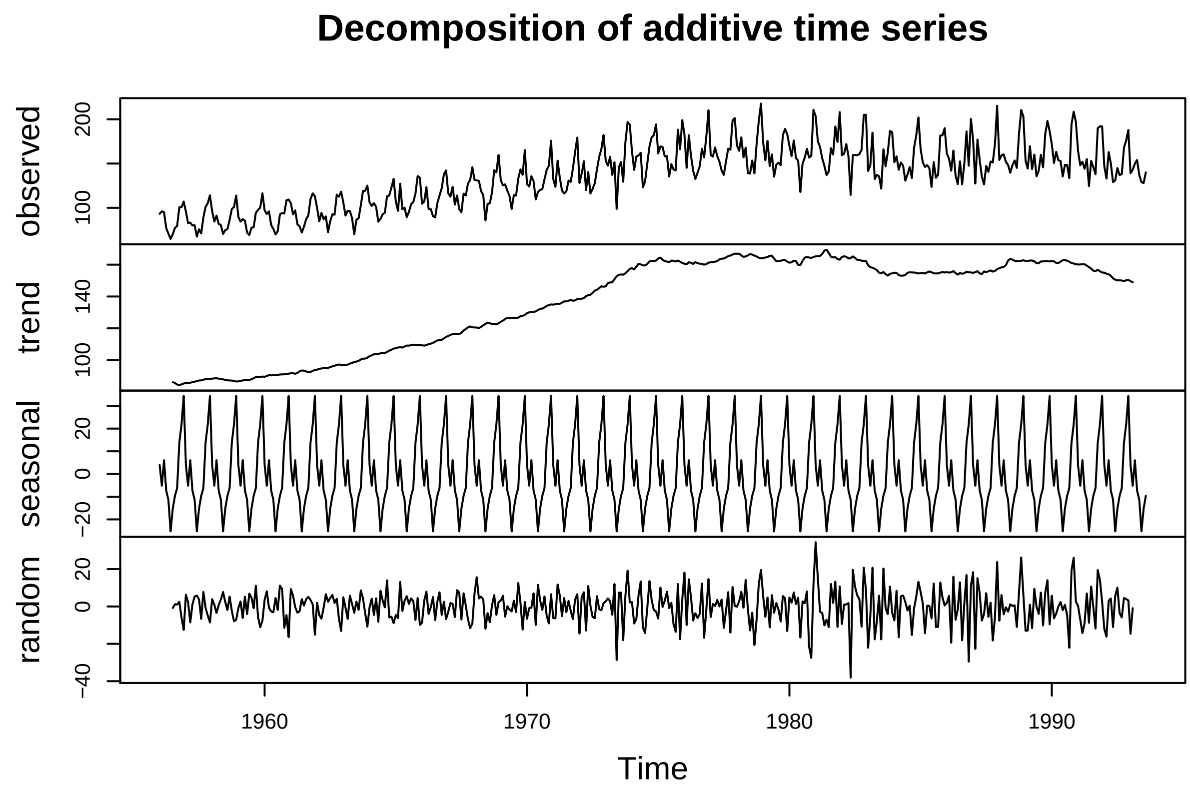
```r
require(fpp)
seasonplot(beerTS,year.labels=TRUE,ylab="megaliters",main="Seasonal plot: quarterly beer production", col=rain
```



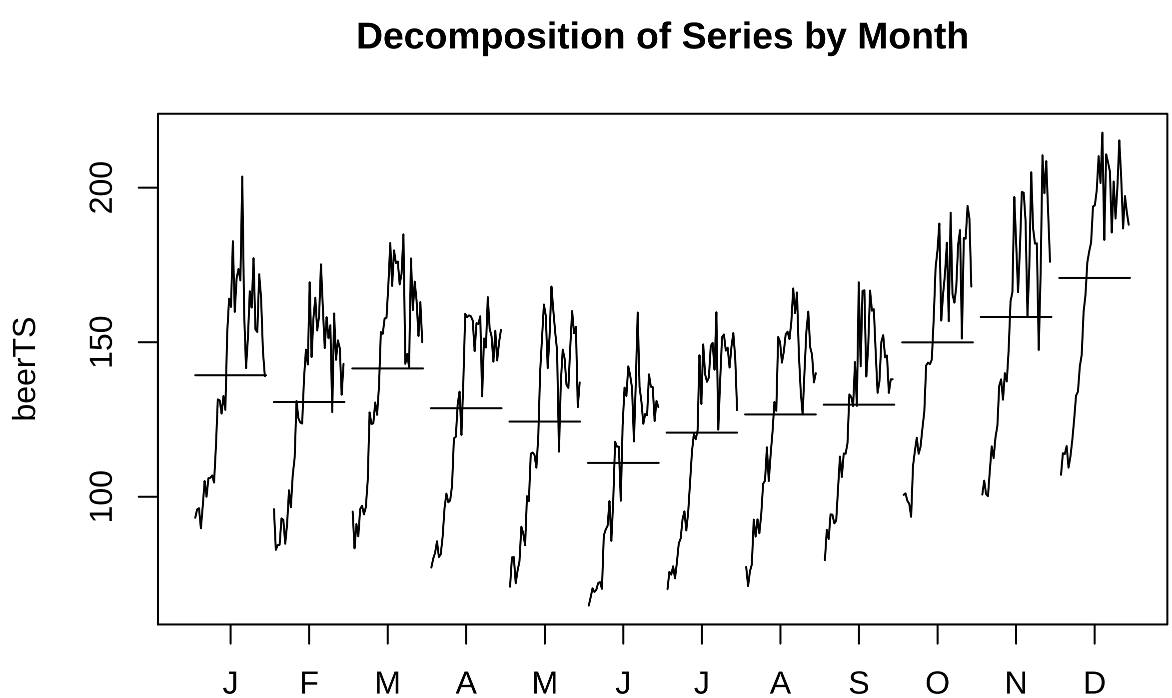
In the plot we see obvious seasonality with higher production in November and December and lower production in June and July. There is a trend which may be difficult to fit as it doesn't appear to be a "well known" function like a linear or quadratic function, so we'll have to experiment. It also looks 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)
```




```r
monthplot(beerTS, main="Decomposition of Series by Month")
```



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 ?

Investigate possible relationship with population data

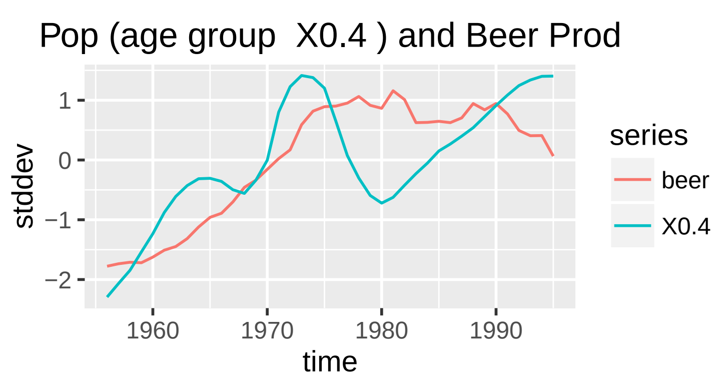
```
#load population data
library(reshape)
```

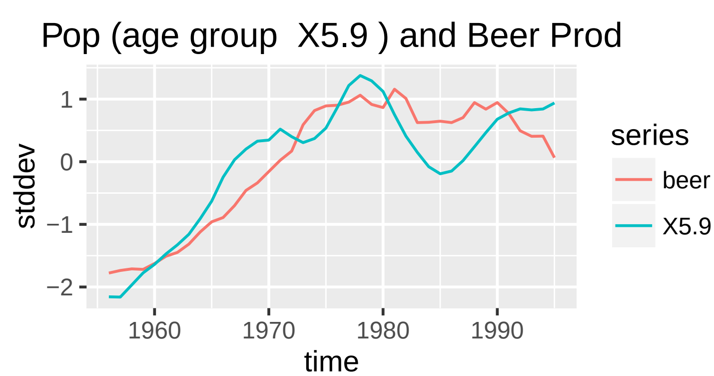
```
## Warning: package 'reshape' was built under R version 3.2.5
```

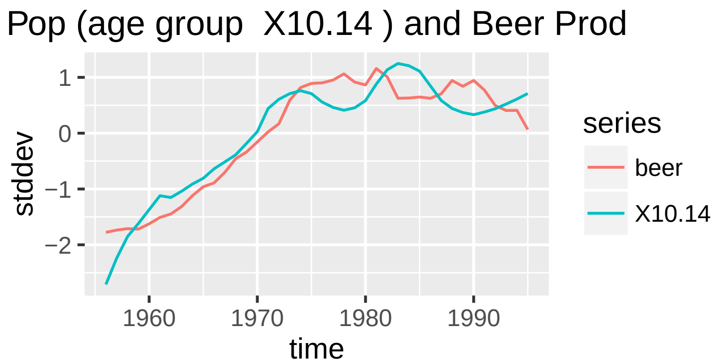
```
library(ggplot2)
```

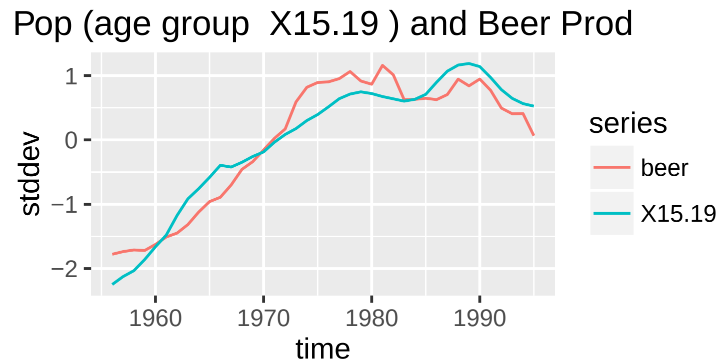
```
## Warning: package 'ggplot2' was built under R version 3.2.5
```

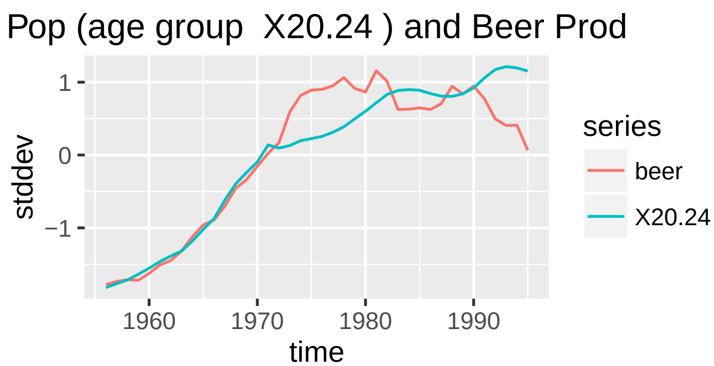
```
#Clean up population data
pop_totalData<-t(read.csv("Pop_total.csv", row.names=1))
dropCols<-colnames(pop_totalData) %in% c("Unspecified","Period not indicated")
rownames(pop_totalData)<-c(1921:2011)
pop_totalDataLong<-pop_totalData[,!dropCols]
pop_totalData<-pop_totalData[paste(1956:1995),!dropCols]

#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))]

#Plot data for each age group and beer data on same plot
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
  print(newPlot)
}
```

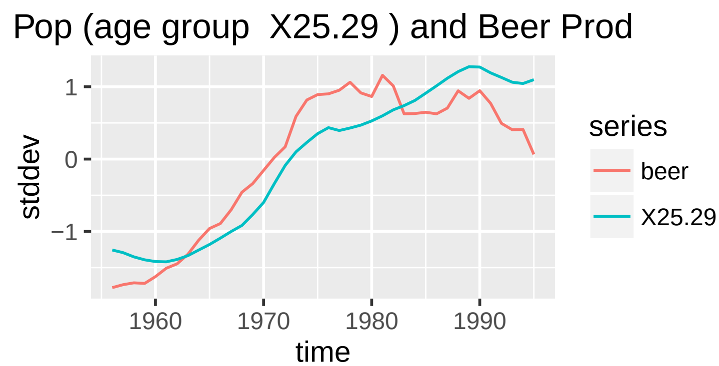


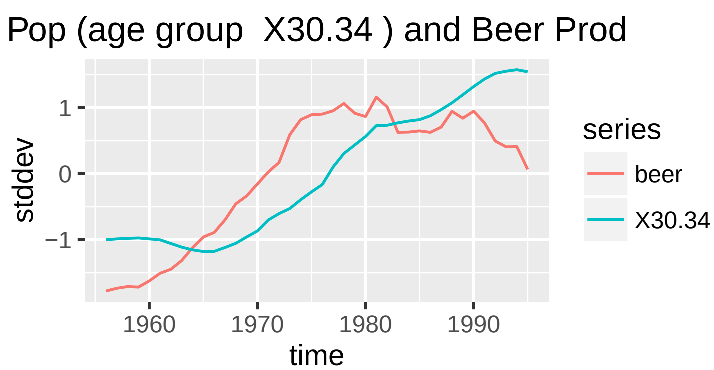


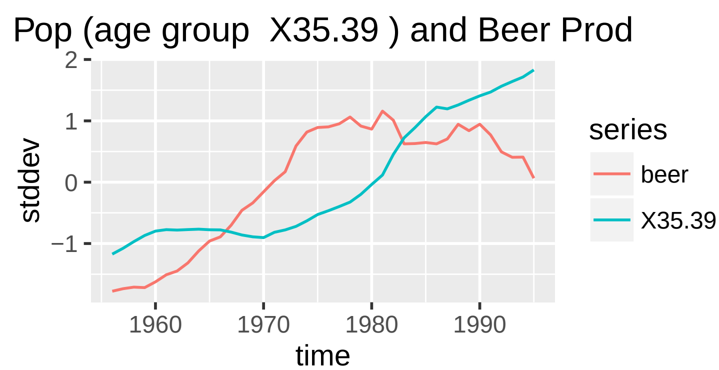

Pop (age group X10.14) and Beer Prod
stddev
time
series
beer
X10.14

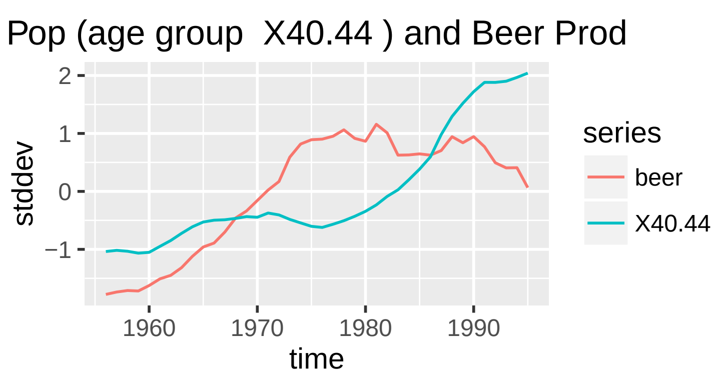

Pop (age group X15.19) and Beer Prod
stddev
time
series
beer
X15.19

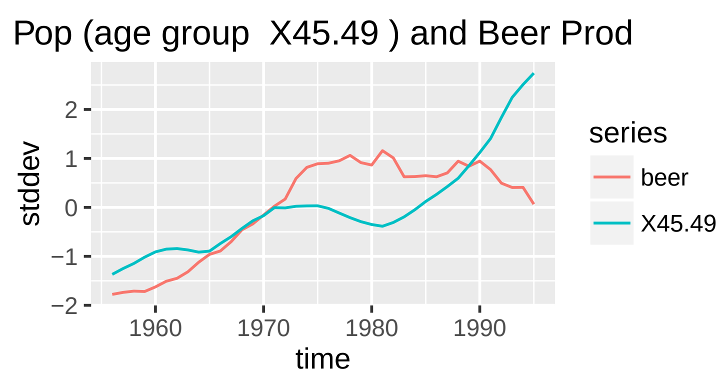

Pop (age group X20.24) and Beer Prod
stddev
time
series
beer
X20.24

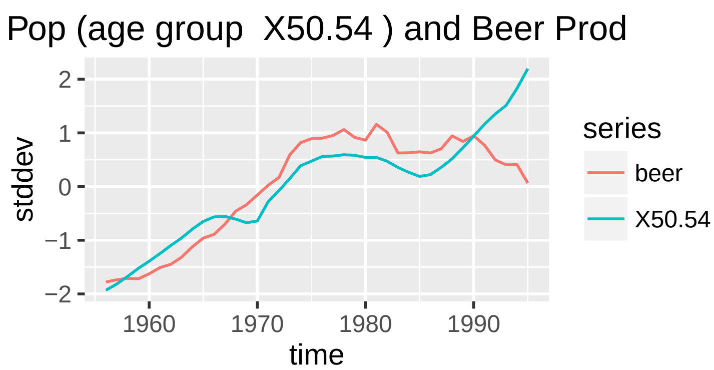

Pop (age group X25.29) and Beer Prod
stddev
time
series
beer
X25.29

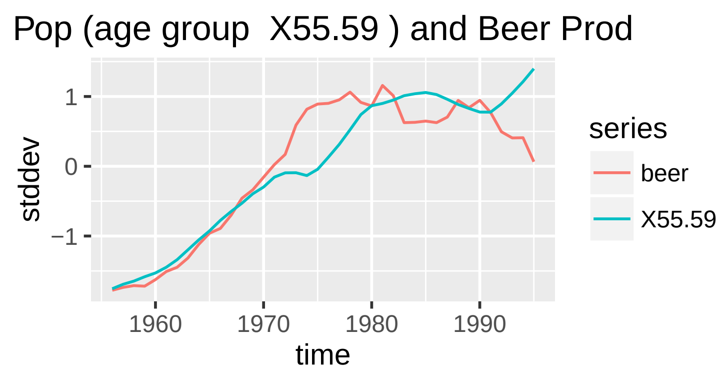

Pop (age group X30.34) and Beer Prod
stddev
time
series
beer
X30.34

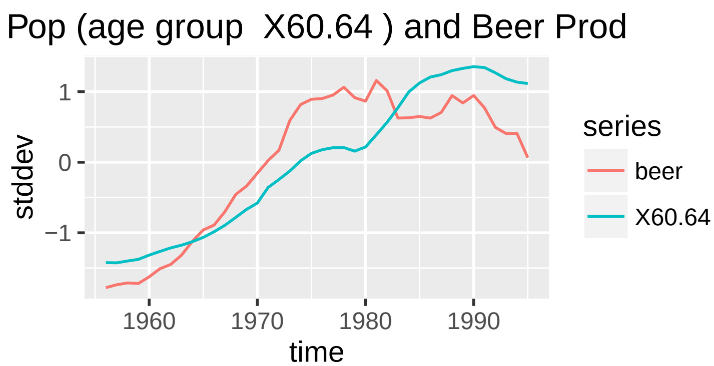

Pop (age group X35.39) and Beer Prod
stddev
time
series
beer
X35.39

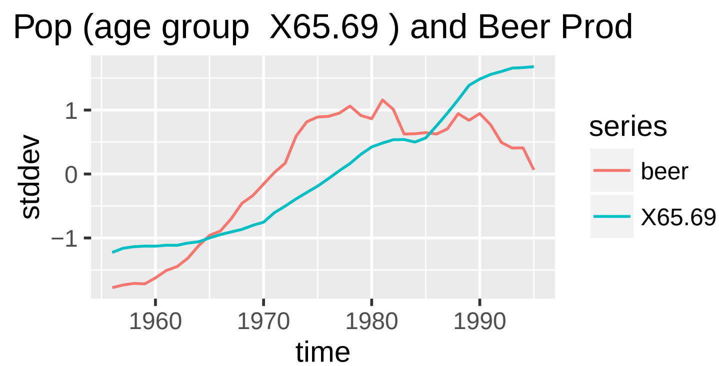

Pop (age group X40.44) and Beer Prod
stddev
time
series
beer
X40.44

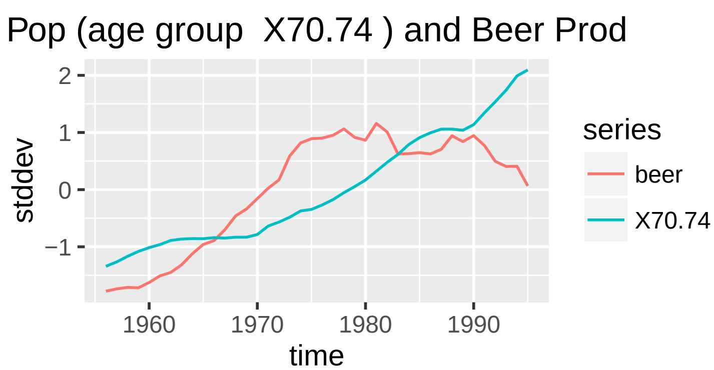

Pop (age group X45.49) and Beer Prod
stddev
time
series
beer
X45.49

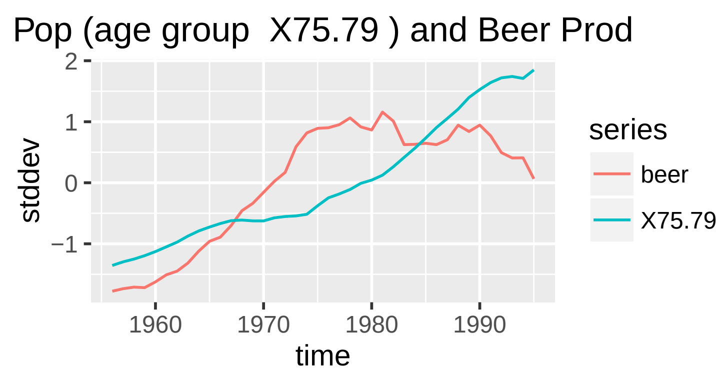

Pop (age group X50.54) and Beer Prod
stddev
time
series
beer
X50.54

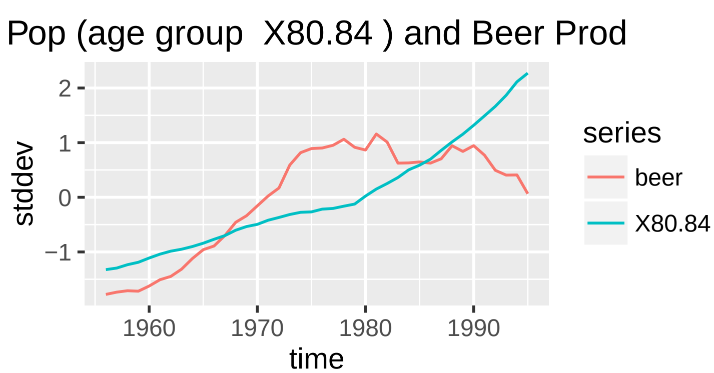

Pop (age group X55.59) and Beer Prod
stddev
time
series
beer
X55.59

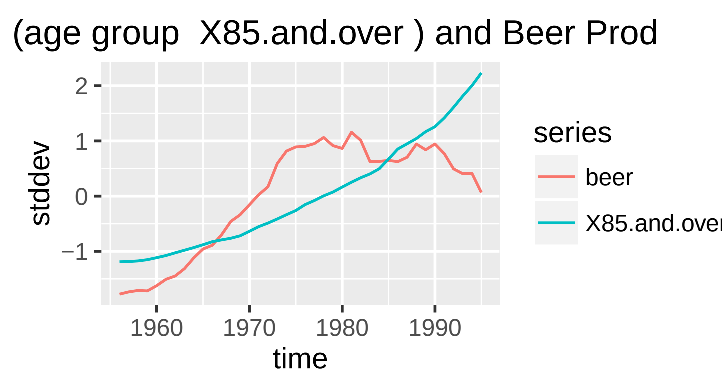

Pop (age group X60.64) and Beer Prod
stddev
time
series
beer
X60.64

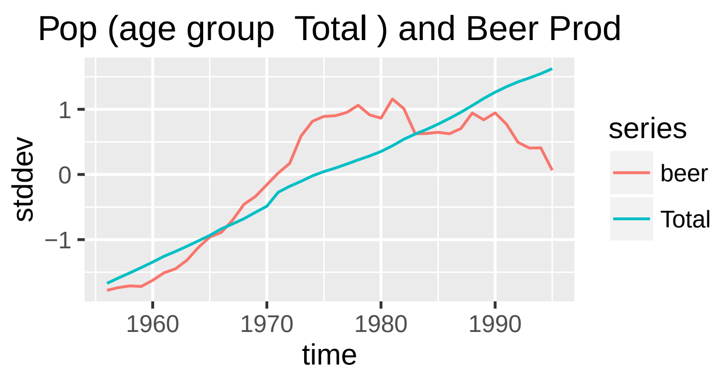

Pop (age group X65.69) and Beer Prod
stddev
time
series
beer
X65.69


Pop (age group X70.74) and Beer Prod
stddev
time
series
beer
X70.74


Pop (age group X75.79) and Beer Prod
stddev
time
series
beer
X75.79


Pop (age group X80.84) and Beer Prod
stddev
time
series
beer
X80.84

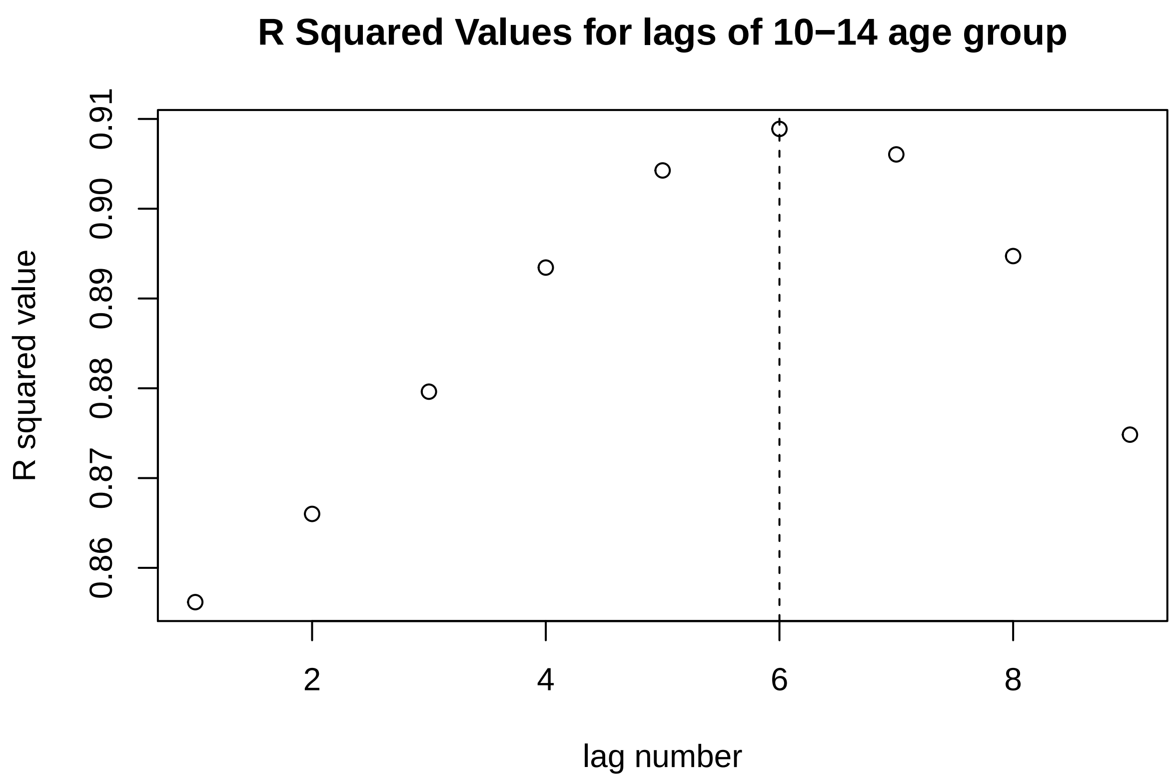

(age group X85.and.over) and Beer Prod
stddev
time
series
beer
X85.and.over



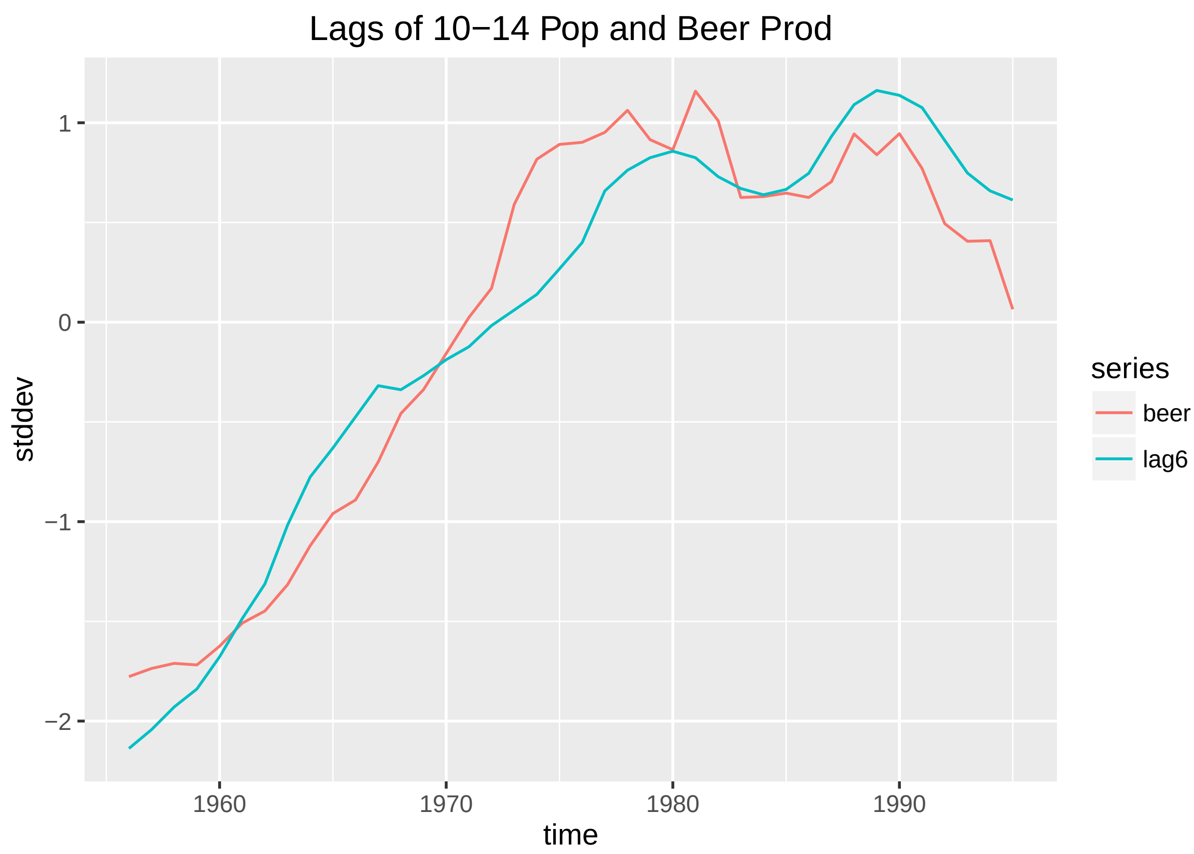

```r
par(mfrow=c(1,1))

#Make a model based on the 15-19 age group
yearModel1<-lm(beer ~ X15.19, data=beerPop)
```

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 to remove some of our trend. However, the 10-14 age group numbers look like they might have potential if shifted forward a few years. This makes sense since these children will grow up and start drinking beer.

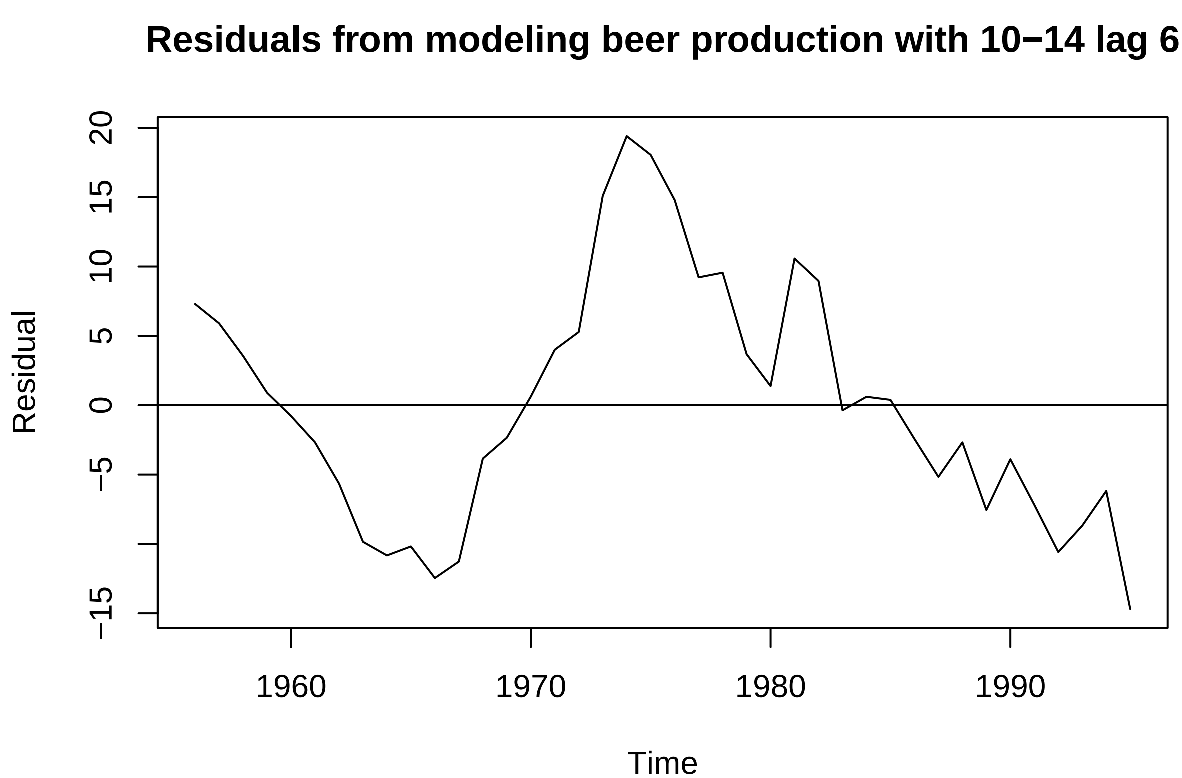
```r
#Explore lagged x10.14 data
laggedData<-data.frame(beer=beerAg[,2])
models<-list()
modelRsq<-c()
for (lag in 0:8) {
  newColNames<-c(colnames(laggedData), paste("lag", lag, sep=""))
  newLag<-pop_totalDataLong[paste(1956:1995-lag), "10-14"]
  laggedData<-data.frame(laggedData, newLag)
  newModel<-lm(beer ~ newLag, data=laggedData)
  models[[paste("lag", lag, sep="")]]<-newModel
  modelRsq<-c(modelRsq, summary(newModel)$r.squared)
  colnames(laggedData)<-newColNames

}

plot(modelRsq, main="R Squared Values for lags of 10-14 age group", xlab="lag number", ylab="R squared value")
abline(v=6, lty=2)
```




```
lagDataScale<-scale(laggedData)[,c(1,8)]
lagDataMelt<-melt(lagDataScale, variable.name="series")
colnames(lagDataMelt)<-c("time", "series", "stddev")

newPlot<-ggplot(lagDataMelt, aes(time,stddev)) + geom_line(aes(colour = series)) +ggtitle(paste("Lags of 10-14
print(newPlot)
```



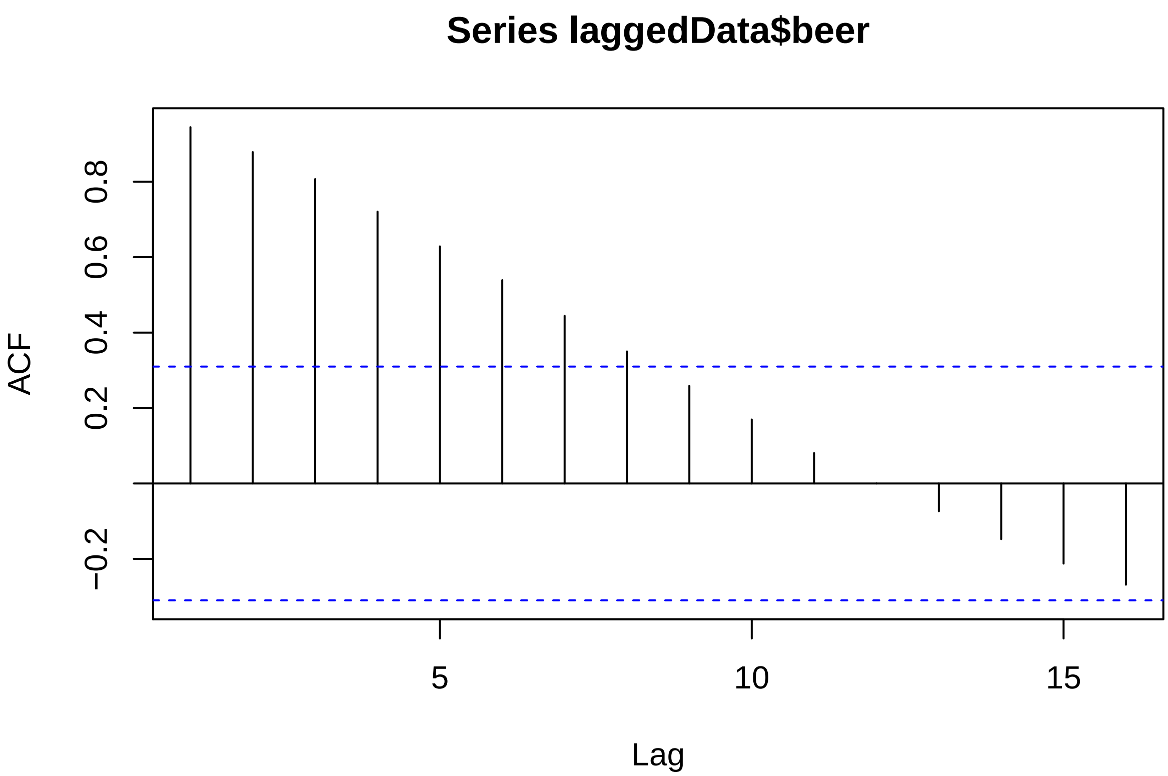

```r
plot(ts(residuals(models[["lag6"]]), frequency=1, start=c(1956)), main="Residuals from modeling beer productio
abline(h=0)
```



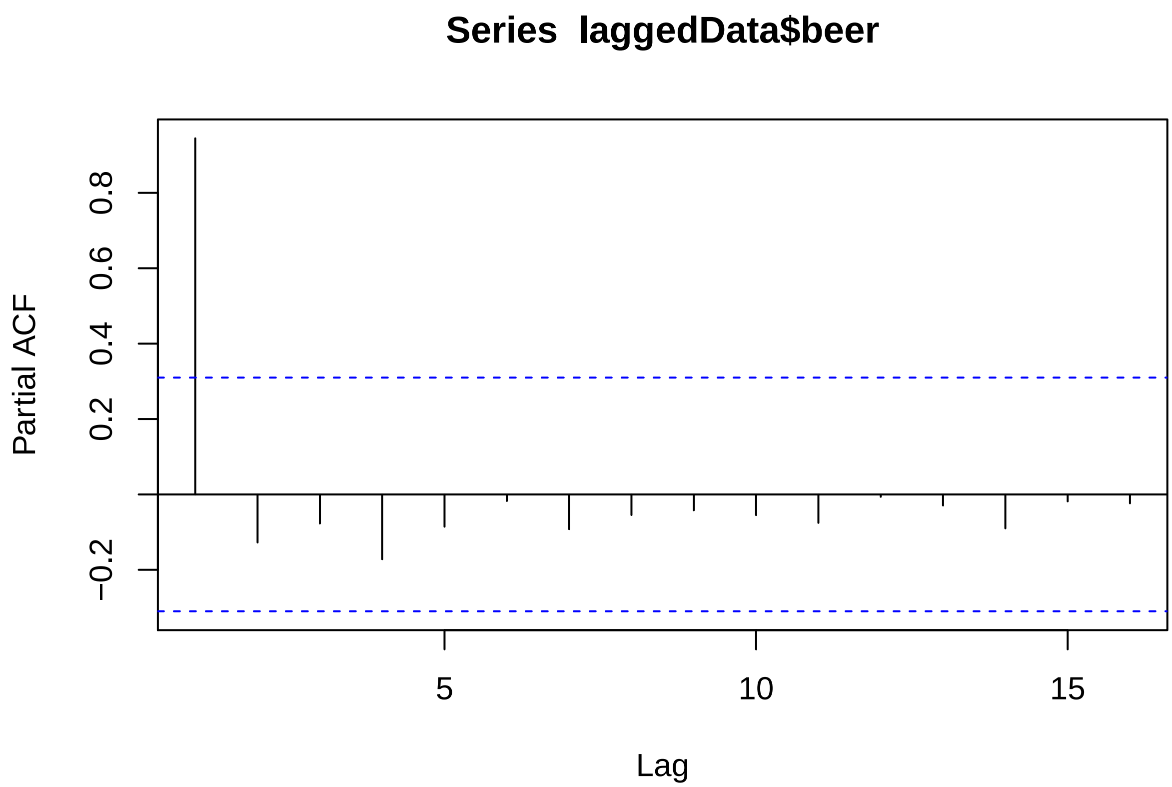
```r
lag6_resid<-ts(residuals(models[["lag6"]]), frequency=1, start=c(1956))
adf.test(laggedData$beer)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  laggedData$beer
## Dickey-Fuller = -0.26924, Lag order = 3, p-value = 0.987
## alternative hypothesis: stationary
```

```r
acf(laggedData$beer)
```




```r
pacf(laggedData$beer)
```




```
ar1YearlyModel<-arima(laggedData$beer, order=c(1,1,0), xreg=laggedData$lag6)
```

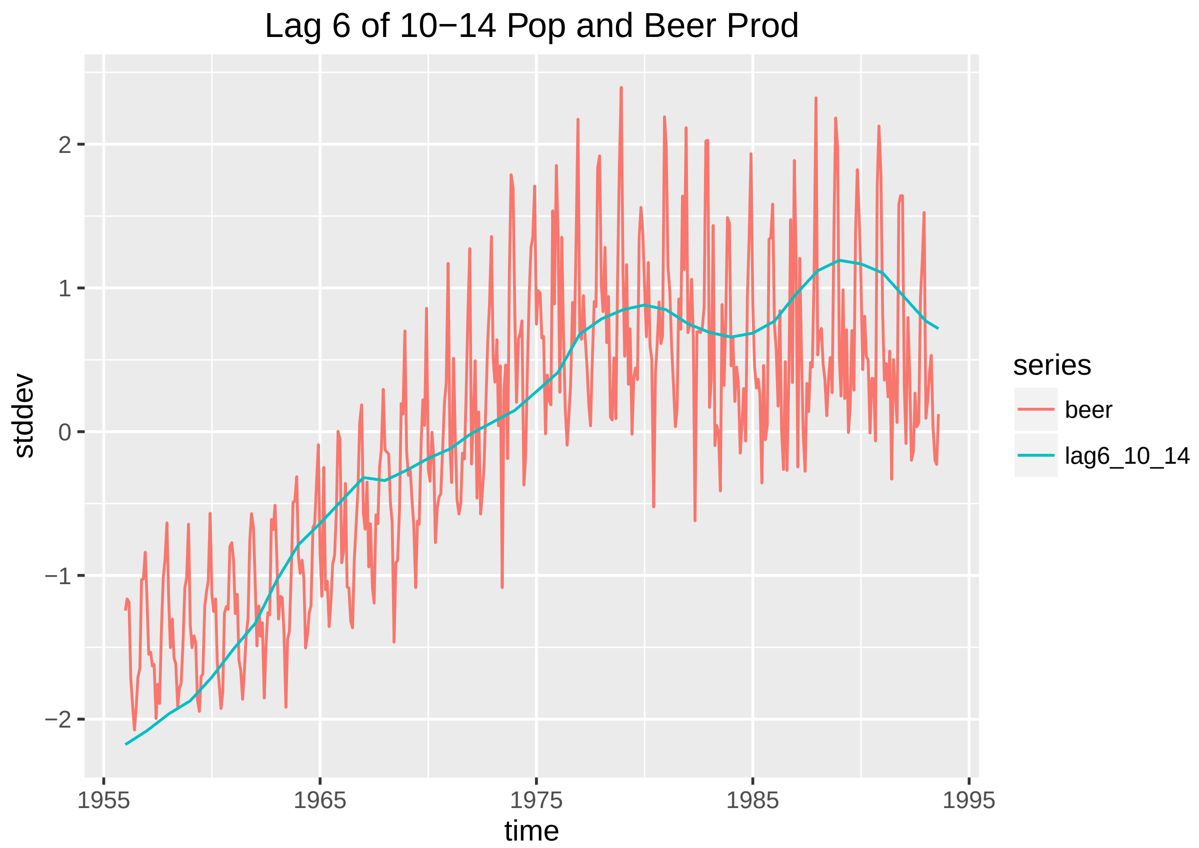
We see that lag 6 is the optimal lag in terms of R-squared values. This makes sense because in 6 years, this age group will be 16-20, or right around drinking age. We can see in the residuals that it isn't perfect, but this pattern may be easier to model than what we had before, it looks much more like a regular polynomial.

We can model just the yearly trend, but this isn't a stationary model, so we want to try to model the monthly data with the seasonality.

If we are interested in modeling the seasonal trend in our data, we would need to extend the population data to a monthly series. To do this, we need to assume that the population will change at about the same rate over the year or that it will remain approximately constant over a year. In this case we will choose to assume that the population will change at about the same rate over the year. This may not be true if there is a particular month in which there is a high influx of immigrants, however it seems reasonable to assume that the majority of the population from year to year is composed of people who were in Australia the previous year (source for net migration rate preferrably broken up by age group needed).
#Interpolate Monthly Numbers

```
library(zoo)

#Create a vector with missing values for zoo to interpolate
withNA<-c()
for (year in 1:nrow(laggedData)) {
  withNA<-c(withNA, laggedData$lag6[year], rep(NA, 11))
}

#Interpolate values using zoo library
zooSeries<-zoo(withNA, frequency=12)
wAppx<-na.approx(zooSeries, na.rm=FALSE)
monthlyLag6<-wAppx

#Reattach to beer numbers for plotting
beerPopMonth<-data.frame(beer=beerTS, lag6_10_14=monthlyLag6[1:length(beerTS)])
```

```r
rownames(beerPopMonth)<-round(seq(from=1956, length.out=length(beerTS), by=1/12),2)
scaleMonth<-scale(beerPopMonth)
scaleMonthMelt<-melt(scaleMonth, variable.name="series")
colnames(scaleMonthMelt)<-c("time", "series", "stddev")

#Make a pretty plot
newPlot<-ggplot(scaleMonthMelt, aes(time,stddev)) + geom_line(aes(colour = series)) +ggtitle(paste("Lag 6 of 1
print(newPlot)
```

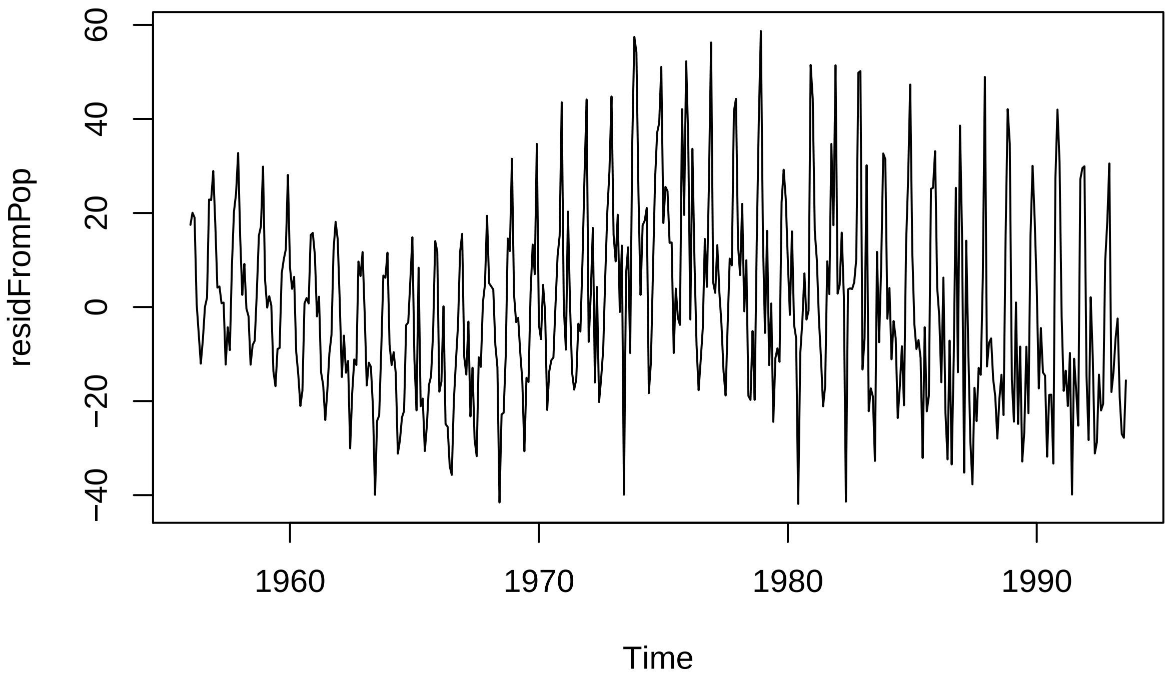


Now we are ready to investigate models using the population numbers

```r
monthlyPopModel<-lm(beer ~ lag6_10_14, data=beerPopMonth)
summary(monthlyPopModel)
```

```
##
## Call:
## lm(formula = beer ~ lag6_10_14, data = beerPopMonth)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -41.862 -14.399  -2.561  13.099  58.717
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.919e+00  4.554e+00   1.519    0.129
## lag6_10_14  1.180e-04  4.076e-06  28.946   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 20.25 on 450 degrees of freedom
## Multiple R-squared:  0.6506, Adjusted R-squared:  0.6498
## F-statistic: 837.9 on 1 and 450 DF,  p-value: < 2.2e-16
```

```r
residFromPop<-ts(residuals(monthlyPopModel), frequency=12, start=c(1956,1))
plot(residFromPop)
```



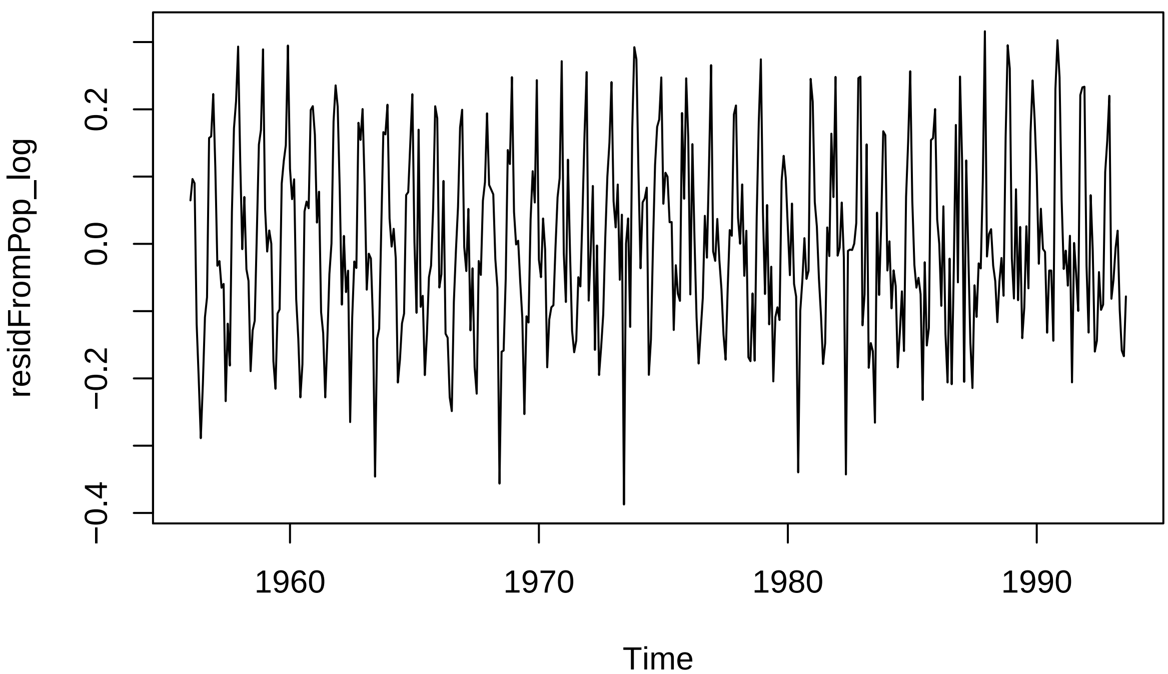
We notice that there still appears to be pattern in the residuals and that the variance appears to be larger at later lags, so the first thing we should do is log the beer data to fix the variance problem. We can also try to fit the remainder of the pattern with a polynomial

```r
logBeer<-log(beerPopMonth$beer)
lag6_10_14<-beerPopMonth$lag6_10_14
t<-1:length(beerTS)
t2<-t^2
t3<-t^3
t4<-t^4
monthlyPopModel_log<-lm(logBeer ~ lag6_10_14+t+t2+t3+t4)
summary(monthlyPopModel_log)
```

```
##
## Call:
## lm(formula = logBeer ~ lag6_10_14 + t + t2 + t3 + t4)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.38742 -0.09499 -0.01169  0.08863  0.31604
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.233e+00  1.167e-01  36.286  < 2e-16 ***
## lag6_10_14   4.122e-07  2.091e-07   1.971   0.0493 *
## t           -3.114e-03  1.322e-03  -2.356   0.0189 *
## t2           4.855e-05  8.989e-06   5.401 1.08e-07 ***
## t3          -1.647e-07  2.938e-08  -5.604 3.67e-08 ***
```

```
## t4           1.668e-10  3.217e-11   5.185 3.28e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1357 on 446 degrees of freedom
## Multiple R-squared:  0.748,  Adjusted R-squared:  0.7452
## F-statistic: 264.8 on 5 and 446 DF,  p-value: < 2.2e-16
```

```
residFromPop_log<-ts(residuals(monthlyPopModel_log), frequency=12, start=c(1956,1))
plot(residFromPop_log)
```



Obviously there is a pattern in the residuals here, so we can either try to fit the remainder of the trend or continue on and count that as error in favor of a more simple model. For now we will continue on, but remember to check the residuals of any model we come up with.

```
adf.test(residFromPop_log)
```

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

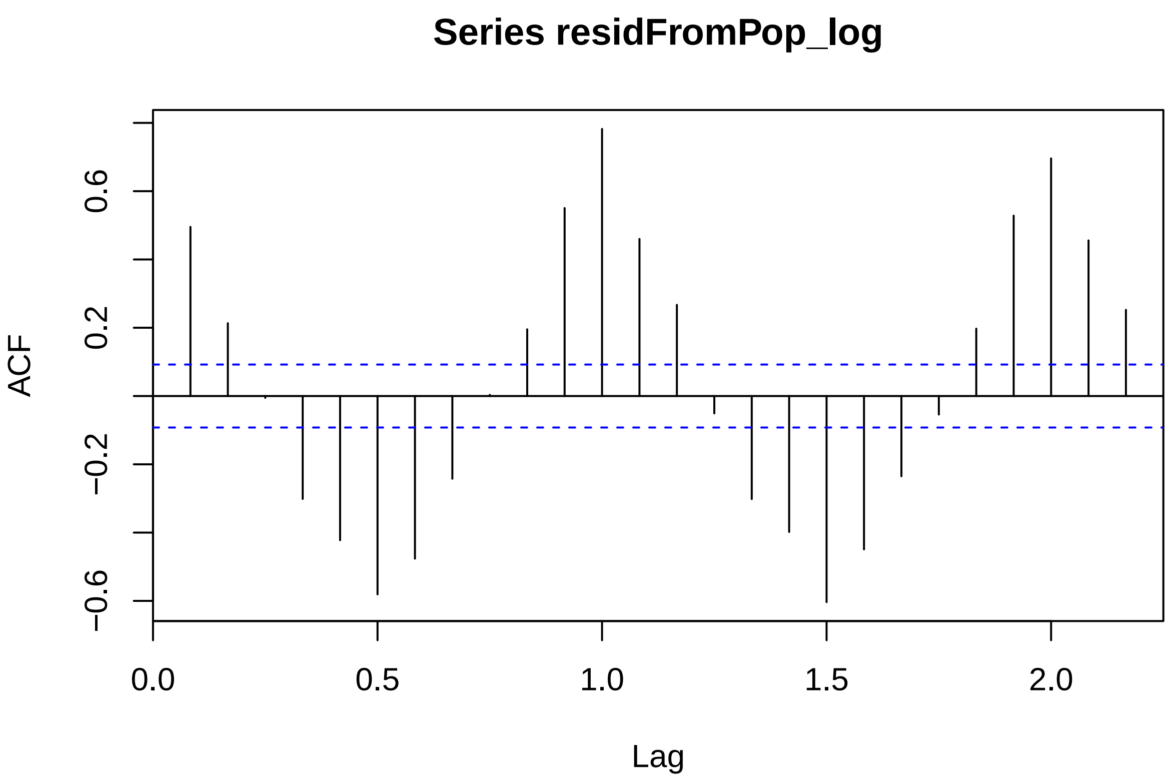
```
##
##  Augmented Dickey-Fuller Test
##
## data:  residFromPop_log
## Dickey-Fuller = -14.166, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

```
pp.test(residFromPop_log)
```

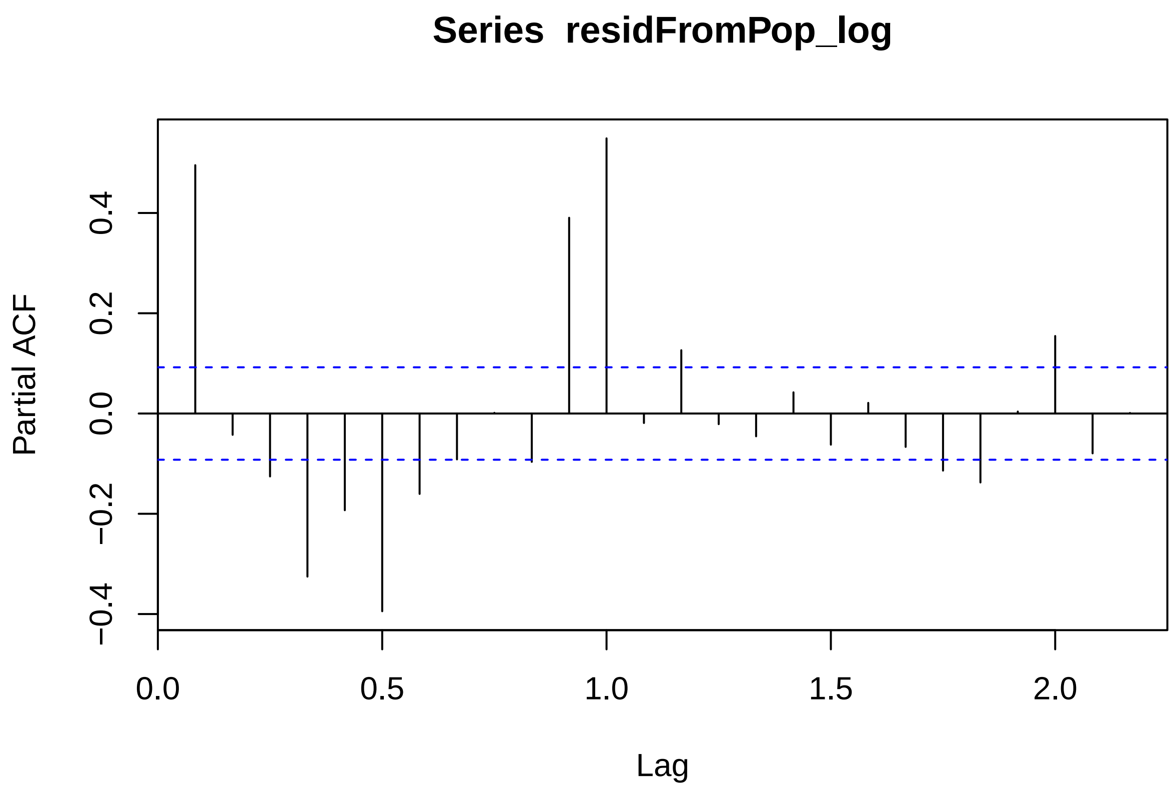
```
## Warning in pp.test(residFromPop_log): p-value smaller than printed p-value
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  residFromPop_log
## Dickey-Fuller Z(alpha) = -217.82, Truncation lag parameter = 5,
## p-value = 0.01
## alternative hypothesis: stationary
```

```
acf(residFromPop_log)
```




```
pacf(residFromPop_log)
```




```
eacf(residFromPop_log)
```

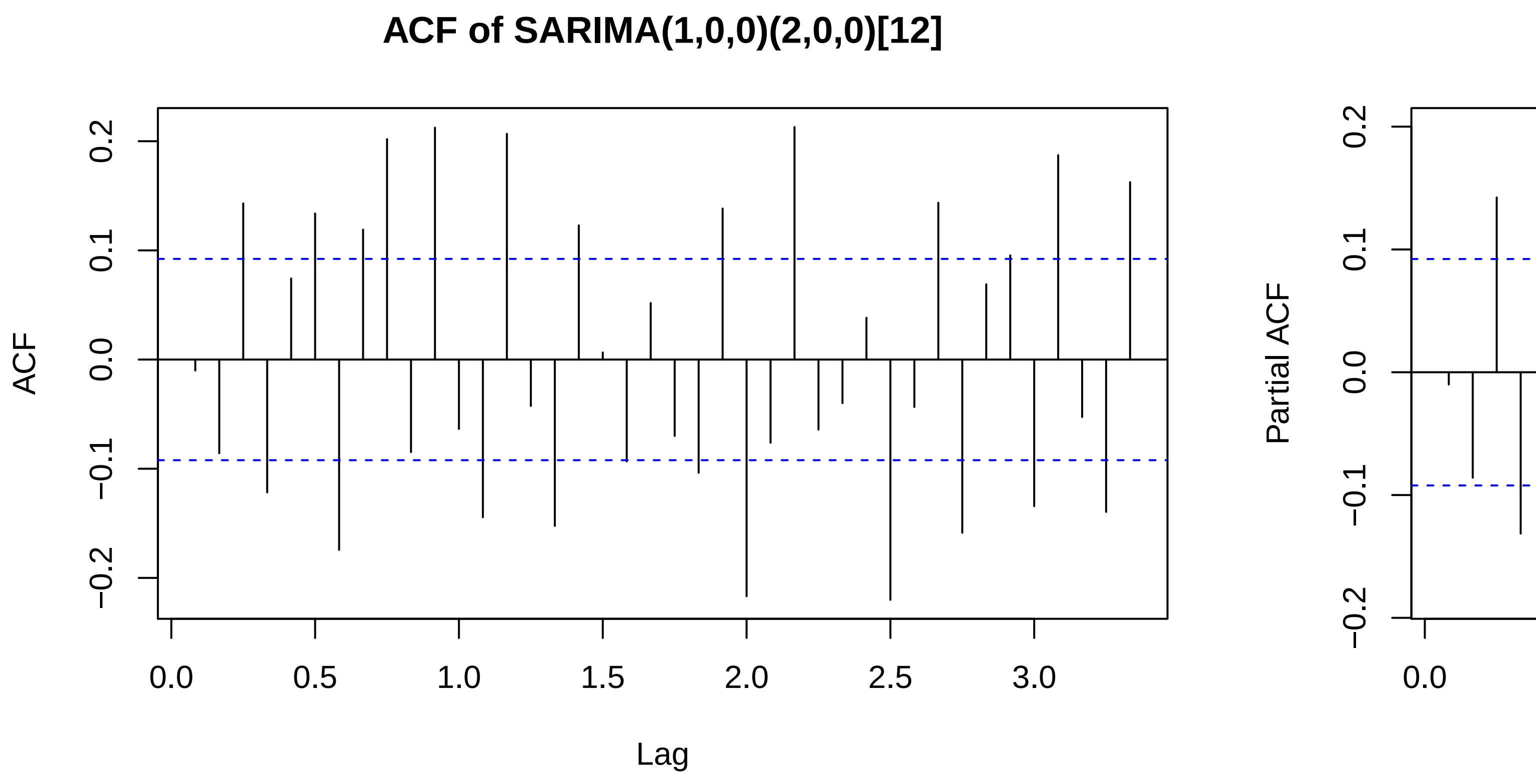
```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x x o x x x x x o x x  x  x  x
## 1 o x o x o x x x o o x  x  o  x
## 2 x o x x o x o o o o o  x  o  x
## 3 x o x o x x o o o x o  x  x  x
## 4 x x x o x x o o o o o  x  x  x
## 5 x x x x x o o o o o o  x  x  x
## 6 x o x x x o o o o x o  x  x  o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

Based on the pacf plot, perhaps p=1, P=1 in an SARIMA(1,0,0)(1,0,0)[12]

Write up a quick function for plotting acf and pacf of residuals

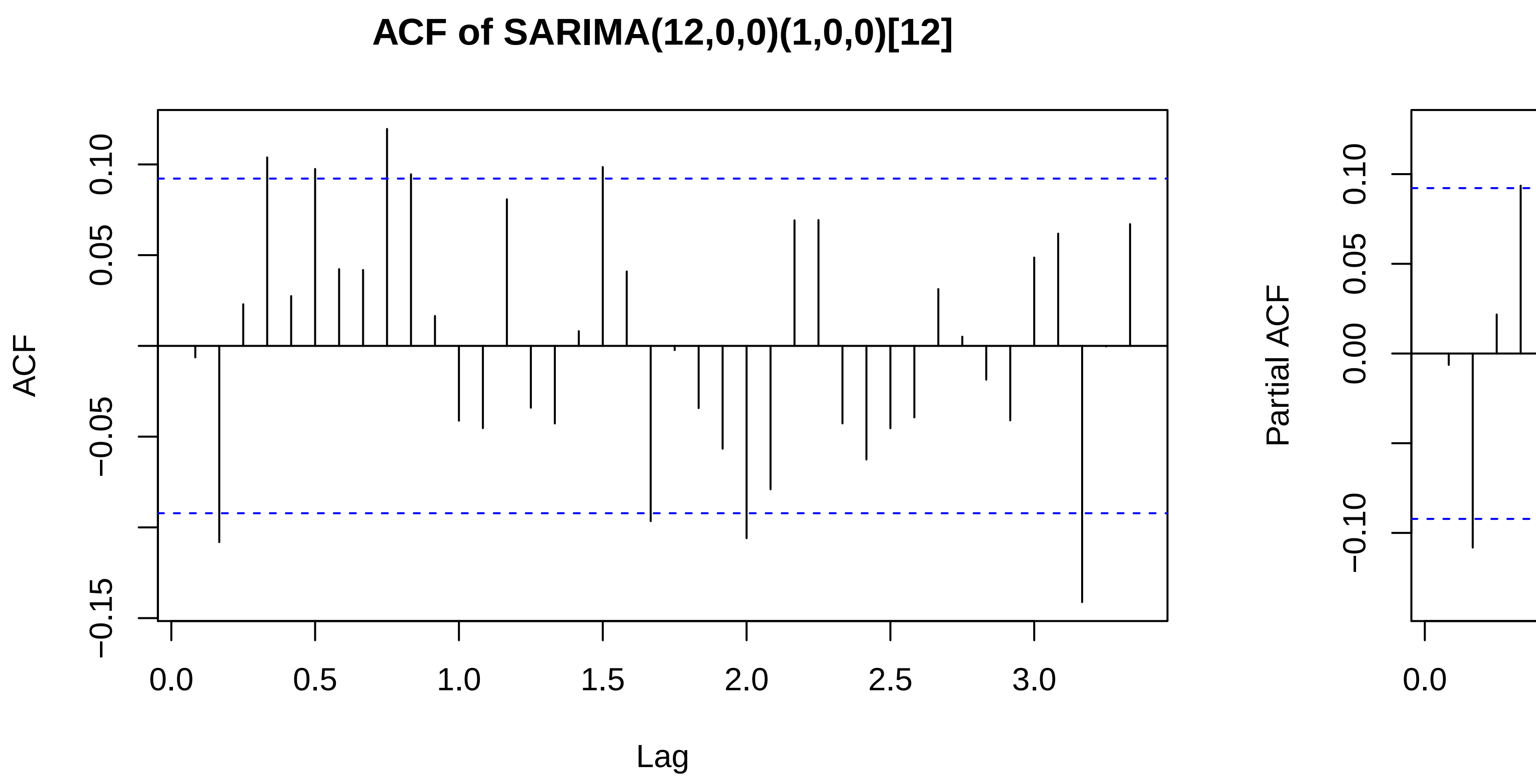
```
getModelString<-function(model) {
  modSpec<-model$arma
  modelString<-paste("SARIMA(", modSpec[1],",", modSpec[6], ",", modSpec[2],")(", modSpec[3], ",", modSpec[7],

  return(modelString)
}

plotResid<-function(model) {
  residuals<-ts(residuals(model), frequency=12, start=c(1956,1))
  modelString<-getModelString(model)
  par(mfrow=c(1,1))
  acf(residuals, main=paste("ACF of", modelString), lag.max=40, cex=.5)
  pacf(residuals, main=paste("PACF of", modelString), lag.max=40, cex=.5)
  par(mfrow=c(1,1))
}
```

```r
popModel1<-arima(residFromPop_log, order=c(1,0,0), seasonal=list(order=c(2,0,0), period=12))
plotResid(popModel1)
```

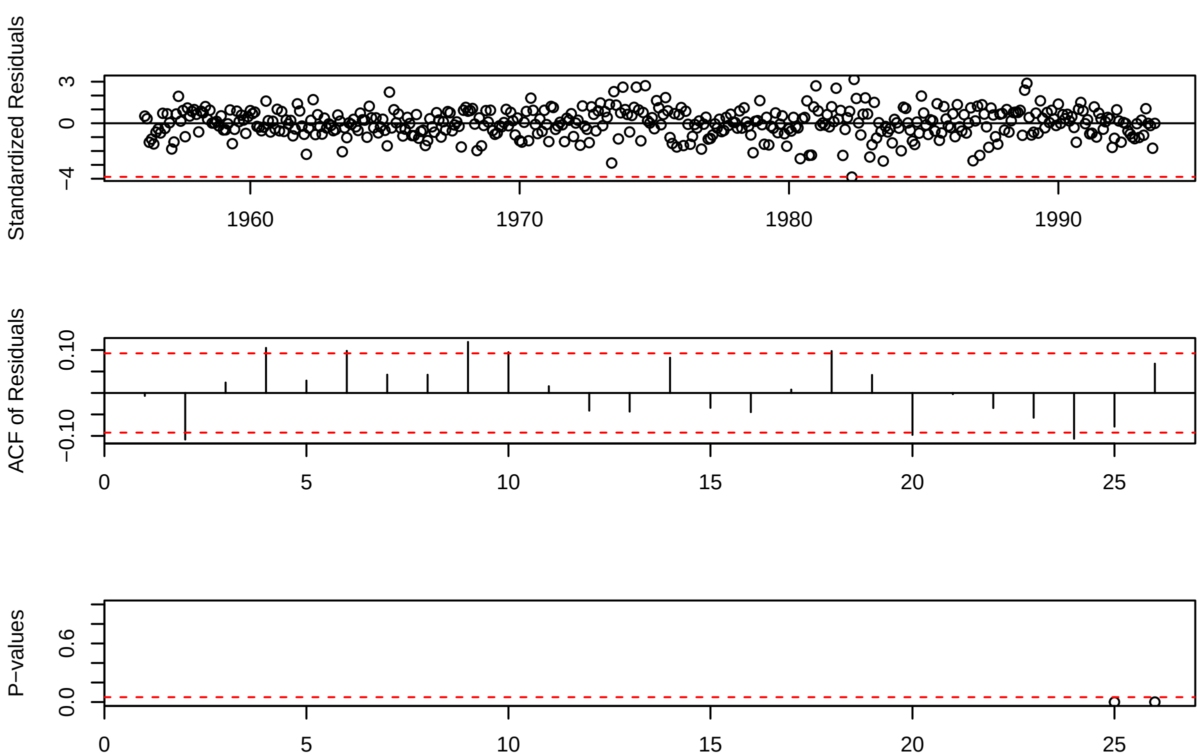


There is still a significant autocorrelation at lag 12 in both plots. Perhaps try either p=12 or q=12?

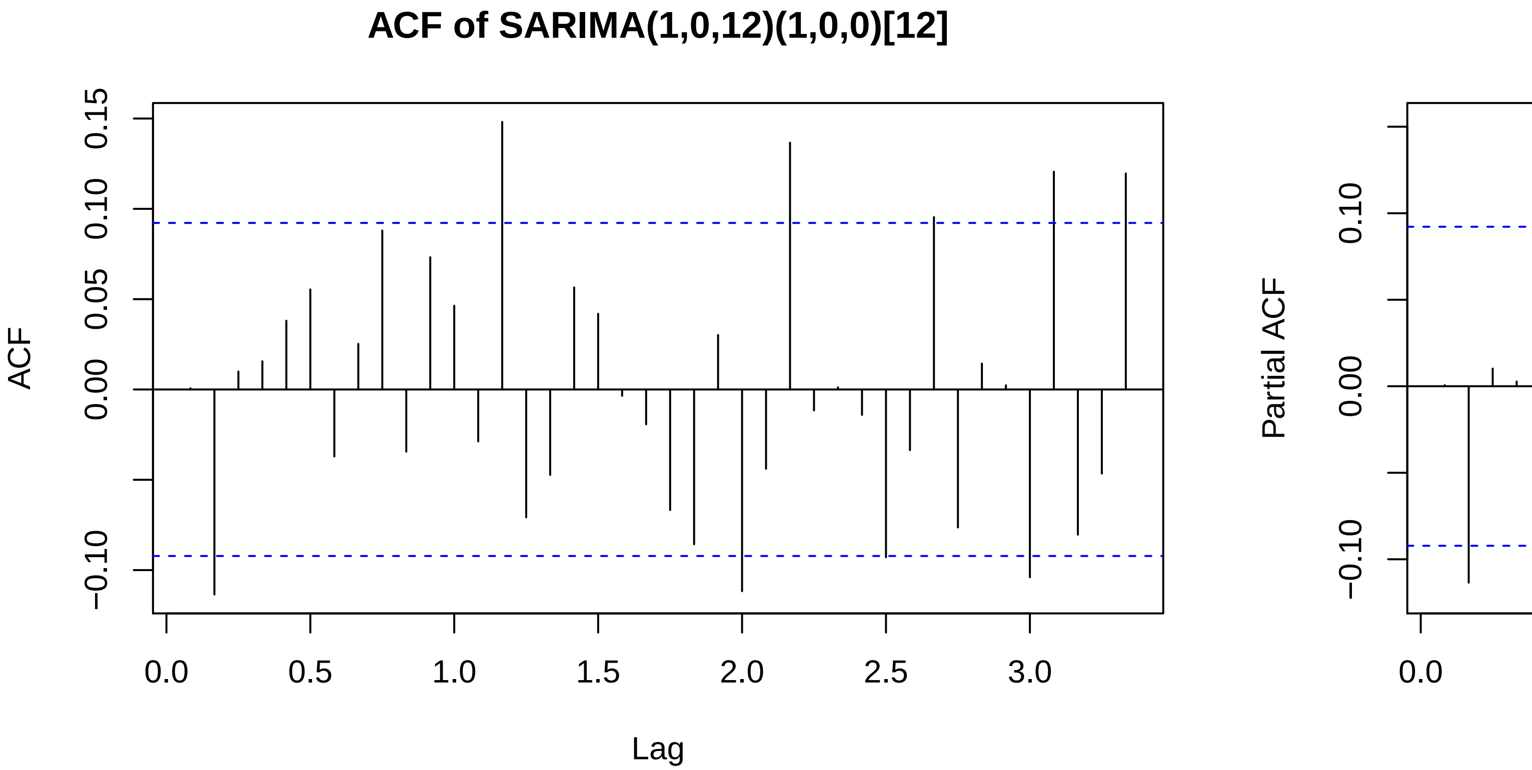
```r
popModel2<-arima(residFromPop_log, order=c(12,0,0), seasonal=list(order=c(1,0,0), period=12))
plotResid(popModel2)
```



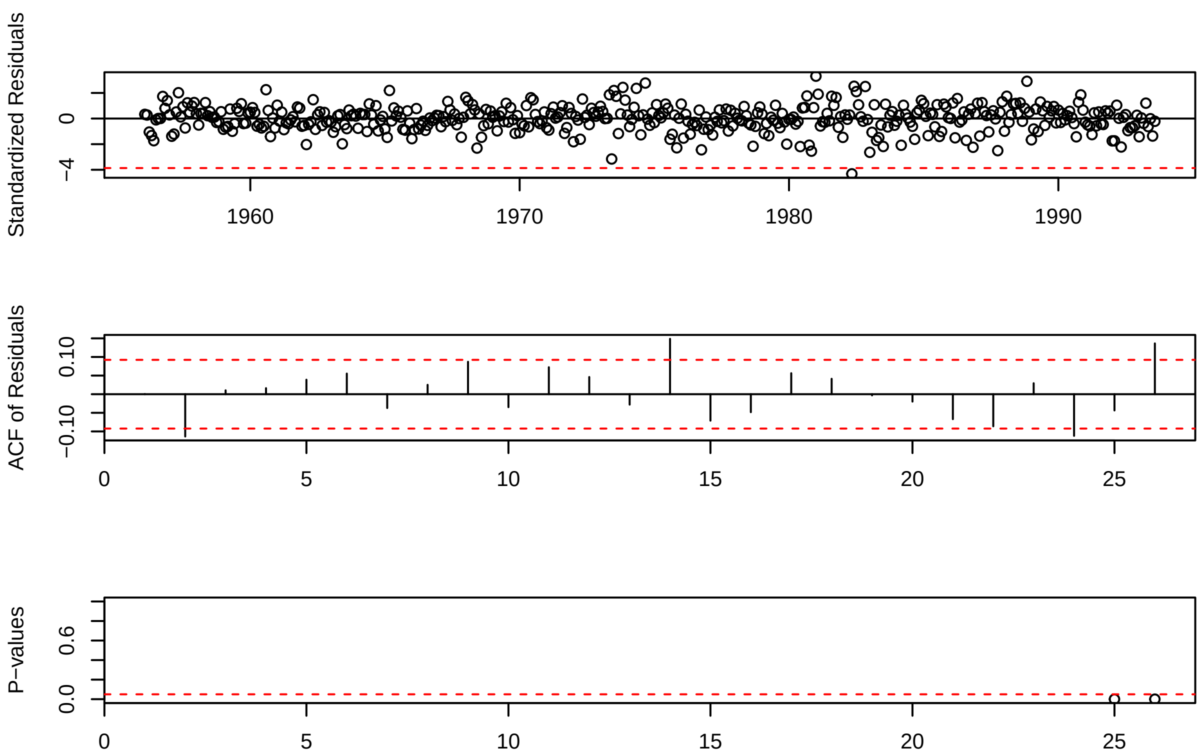

```r
tsdiag(popModel2)
```



```r
popModel3<-arima(residFromPop_log, order=c(1,0,12), seasonal=list(order=c(1,0,0), period=12))
plotResid(popModel3)
```




```r
tsdiag(popModel3)
```



That definitely looks better, we still have significant lags, but they now lie very close to the boundary. This is true more so for the SARIMA(12,0,0)(1,0,0)[12] model than the SARIMA(0,0,12)(1,0,0)[12] model.

Try overfitting the SARIMA(12,0,0)(1,0,0)[12]

```
#popModel4<-arima(residFromPop_log, order=c(13,0,0), seasonal=list(order=c(1,0,0), period=12))
#Produces error

popModel5<-arima(residFromPop_log, order=c(12,0,1), seasonal=list(order=c(1,0,0), period=12))
popModel5
```

```
##
## Call:
## arima(x = residFromPop_log, order = c(12, 0, 1), seasonal = list(order = c(1,
##     0, 0), period = 12))
##
## Coefficients:
##          ar1     ar2     ar3      ar4     ar5      ar6      ar7     ar8
##       0.0083  0.0269  0.0119  -0.1320  0.1293  -0.1074  -0.0709  0.0316
## s.e.  0.0493  0.0327  0.0295   0.0291  0.0311   0.0307   0.0336  0.0299
##          ar9     ar10    ar11    ar12      ma1     sar1  intercept
##       0.0174  -0.1302  0.2063  0.7417  -0.0301  -0.3180    -0.0008
## s.e.  0.0296   0.0292  0.0318  0.0405   0.0825   0.0528     0.0082
##
## sigma^2 estimated as 0.00456:  log likelihood = 571,  aic = -1112
```

```
popModel6<-arima(residFromPop_log, order=c(12,0,0), seasonal=list(order=c(2,0,0), period=12))
popModel6
```

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

```
## arima(x = residFromPop_log, order = c(12, 0, 0), seasonal = list(order = c(2,
##     0, 0), period = 12))
##
## Coefficients:
##           ar1      ar2     ar3      ar4     ar5     ar6      ar7     ar8
##       -0.0548  -0.1008  0.0923  -0.0492  0.0658  0.0729  -0.0987  0.0549
## s.e.   0.0500   0.0483  0.0465   0.0460  0.0473  0.0515   0.0506  0.0481
##          ar9     ar10    ar11     ar12    sar1    sar2  intercept
##       0.2046  -0.0134  0.2072  -0.0683  0.5945  0.3155    -0.0092
## s.e.  0.0470   0.0490  0.0491   0.0707  0.0723  0.0650     0.0401
##
## sigma^2 estimated as 0.004827:  log likelihood = 554.21,  aic = -1078.41
```

```r
popModel7<-arima(residFromPop_log, order=c(12,0,0), seasonal=list(order=c(1,0,1), period=12))
popModel7
```

```
##
## Call:
## arima(x = residFromPop_log, order = c(12, 0, 0), seasonal = list(order = c(1,
##     0, 1), period = 12))
##
## Coefficients:
##           ar1     ar2      ar3      ar4     ar5      ar6      ar7     ar8
##       -0.0127  0.0180  -0.0031  -0.0785  0.0743  -0.0670  -0.0219  0.0234
## s.e.   0.0197  0.0219   0.0183   0.0284  0.0339   0.0244   0.0232  0.0226
##           ar9     ar10    ar11    ar12    sar1     sma1  intercept
##       -0.0127  -0.0726  0.0907  0.9079  0.1563  -0.7465     0.0008
## s.e.   0.0183   0.0273  0.0418  0.0390  0.0779   0.1105     0.0056
##
## sigma^2 estimated as 0.004344:  log likelihood = 580.28,  aic = -1130.57
```

We see that the extra coefficients added in models 6 (SARIMA(12,0,0)(2,0,0)[12]) and 7 (SARIMA(12,0,0)(1,0,1)[12]) are both significant, so we try adding them to the model together.

```r
popModel8<-arima(residFromPop_log, order=c(12,0,0), seasonal=list(order=c(2,0,1), period=12))
popModel8
```

```
##
## Call:
## arima(x = residFromPop_log, order = c(12, 0, 0), seasonal = list(order = c(2,
##     0, 1), period = 12))
##
## Coefficients:
##           ar1      ar2     ar3      ar4     ar5     ar6      ar7     ar8
##       -0.0246  -0.0221  0.0663  -0.0715  0.0796  0.0831  -0.1084  0.0583
## s.e.   0.0482   0.0505  0.0453   0.0447  0.0451  0.0482   0.0472  0.0457
##          ar9     ar10    ar11    ar12    sar1    sar2     sma1  intercept
##       0.1370  -0.0217  0.2249  0.2416  0.9178  0.0803  -0.9069    -0.0014
## s.e.  0.0492   0.0478  0.0462  0.1065  0.1056  0.1054   0.0335     0.0711
##
## sigma^2 estimated as 0.003779:  log likelihood = 602.36,  aic = -1172.72
```

With both together, the sar2 coefficent is no longer significant, overfit model 7 (SARIMA(12,0,0)(1,0,1)[12])

```r
popModel9<-arima(residFromPop_log, order=c(13,0,0), seasonal=list(order=c(1,0,1), period=12))
popModel9
```

```
##
## Call:
## arima(x = residFromPop_log, order = c(13, 0, 0), seasonal = list(order = c(1,
##     0, 1), period = 12))
##
## Coefficients:
```

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

```
##           ar1    ar2     ar3      ar4     ar5      ar6      ar7     ar8
##       -0.0600  0.016  0.0068  -0.0947  0.0988  -0.0754  -0.0276  0.0322
## s.e.   0.0056    NaN  0.0037      NaN  0.0047      NaN   0.0033     NaN
##           ar9     ar10    ar11    ar12    ar13    sar1     sma1  intercept
##       -0.0027  -0.0964  0.1196  0.8842  0.0480  0.0909  -0.6295    -0.0015
## s.e.   0.0040   0.0013  0.0071     NaN  0.0038     NaN   0.0345     0.0076
##
## sigma^2 estimated as 0.004419:  log likelihood = 577.49,  aic = -1122.98
```

```
popModel10<-arima(residFromPop_log, order=c(12,0,1), seasonal=list(order=c(1,0,1), period=12))
popModel10
```

```
##
## Call:
## arima(x = residFromPop_log, order = c(12, 0, 1), seasonal = list(order = c(1,
##     0, 1), period = 12))
##
## Coefficients:
##           ar1     ar2     ar3      ar4     ar5      ar6      ar7     ar8
##       -0.0005  0.0087  0.0001  -0.0702  0.0667  -0.0649  -0.0088  0.0146
## s.e.   0.0208  0.0219  0.0179   0.0292  0.0380   0.0268   0.0226  0.0234
##           ar9     ar10    ar11    ar12      ma1    sar1     sma1
##       -0.0089  -0.0649  0.0790  0.9158  -0.0730  0.1851  -0.7909
## s.e.   0.0180   0.0279  0.0451  0.0412   0.0561  0.0748   0.1079
##       intercept
##          0.0009
## s.e.     0.0051
##
## sigma^2 estimated as 0.004323:  log likelihood = 581.08,  aic = -1130.15
```

```
popModel11<-arima(residFromPop_log, order=c(12,0,0), seasonal=list(order=c(1,0,2), period=12))
popModel11
```

```
##
## Call:
## arima(x = residFromPop_log, order = c(12, 0, 0), seasonal = list(order = c(1,
##     0, 2), period = 12))
##
## Coefficients:
##           ar1     ar2      ar3      ar4     ar5      ar6      ar7     ar8
##       -0.0124  0.0179  -0.0020  -0.0773  0.0729  -0.0658  -0.0217  0.0232
## s.e.   0.0196  0.0227   0.0183   0.0296  0.0363   0.0253   0.0236  0.0236
##           ar9     ar10    ar11    ar12     sar1     sma1     sma2
##       -0.0118  -0.0710  0.0892  0.9091  -0.0100  -0.5788  -0.1180
## s.e.   0.0183   0.0283  0.0448  0.0413   0.3515   0.3665   0.2292
##       intercept
##          0.0010
## s.e.     0.0057
##
## sigma^2 estimated as 0.004342:  log likelihood = 580.41,  aic = -1128.83
```

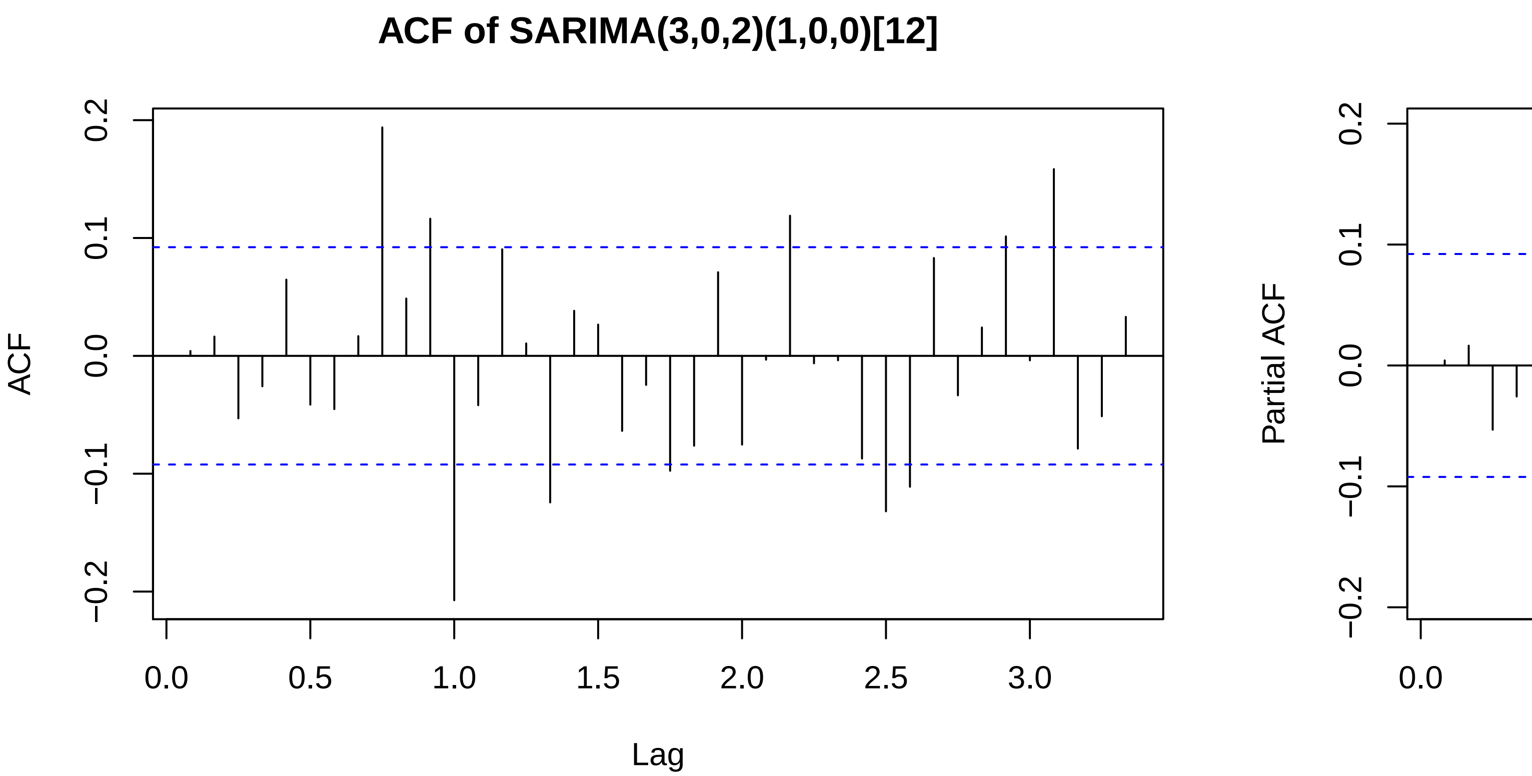
The new coefficients are not significant (with the exception of model 9, which appears to have some issues with fitting), suggesting that we should stick with model 7 (or possibly 8).

Let auto.arima choose a model

```
autoRes<-auto.arima(residFromPop_log, max.p=13, max.order=14, test="adf")
autoRes
```

```
## Series: residFromPop_log
## ARIMA(3,0,2)(1,0,0)[12] with zero mean
##
## Coefficients:
##           ar1      ar2      ar3     ma1     ma2    sar1
##       -1.5646  -1.1175  -0.1432  1.5324  0.9317  0.8543
## s.e.   0.0552   0.0757   0.0536  0.0251  0.0252  0.0253
##
## sigma^2 estimated as 0.005738:  log likelihood=519.88
## AIC=-1025.75   AICc=-1025.5   BIC=-996.96
```

```
plotResid(autoRes)
```



auto.arima chose SARIMA(3,0,2)(1,0,0)[12] but this model still has the same problems as the others we have tried so far (error terms not independent) and the AIC is larger than models 7 and 8, so the only reason we would want to consider this model is if we were really interested in a more parsimonious model.

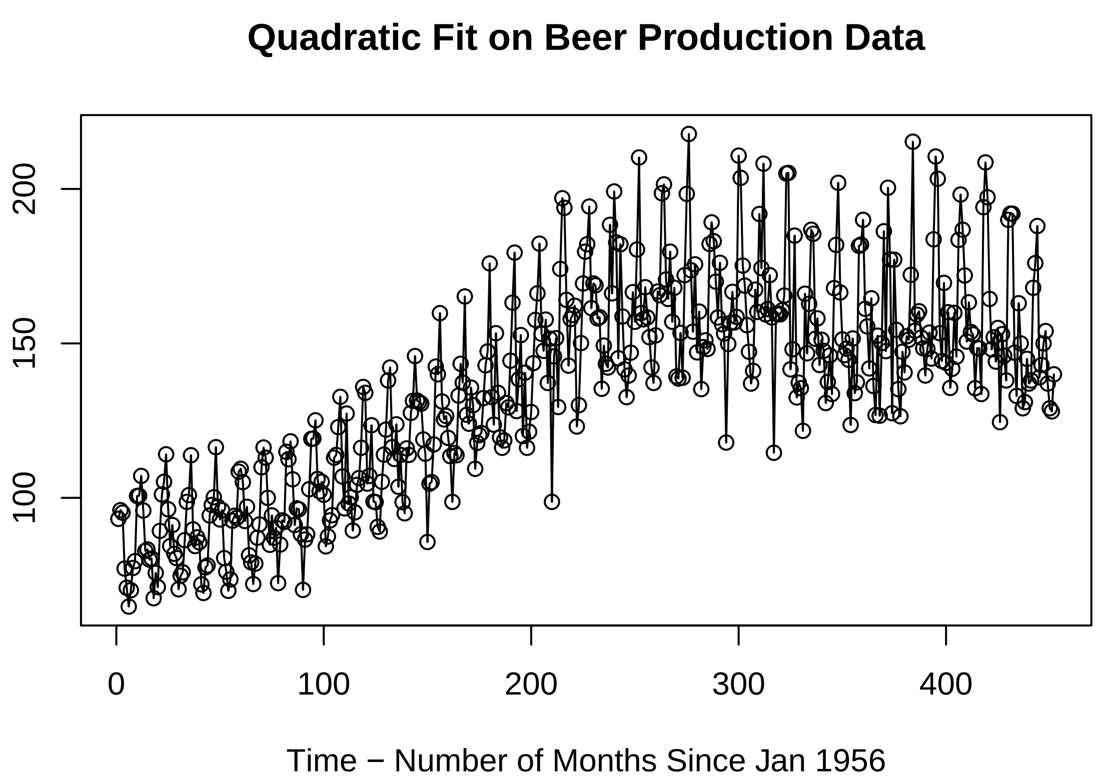
Try to figure out deterministic trend

```
t<-1:length(beerTS)
t2<-t^2
```

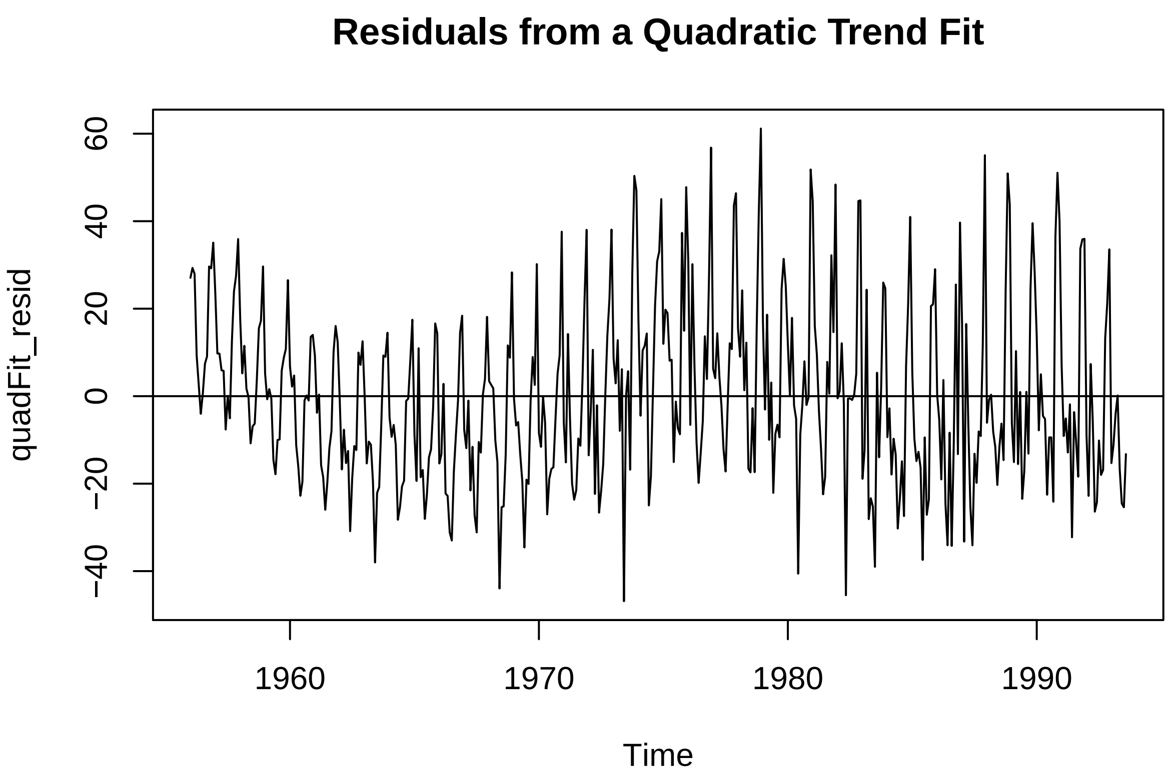
```r
t3<-t^3
t4<-t^4
t5<-t^5

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
```

```r
#### 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="Quadrati
curve(expr = coef(quadFit)[1]+coef(quadFit)[2]*x+coef(quadFit)[3]*x^2+coef(quadFit)[4]*x^3,lty=1,add = TRUE, c
```



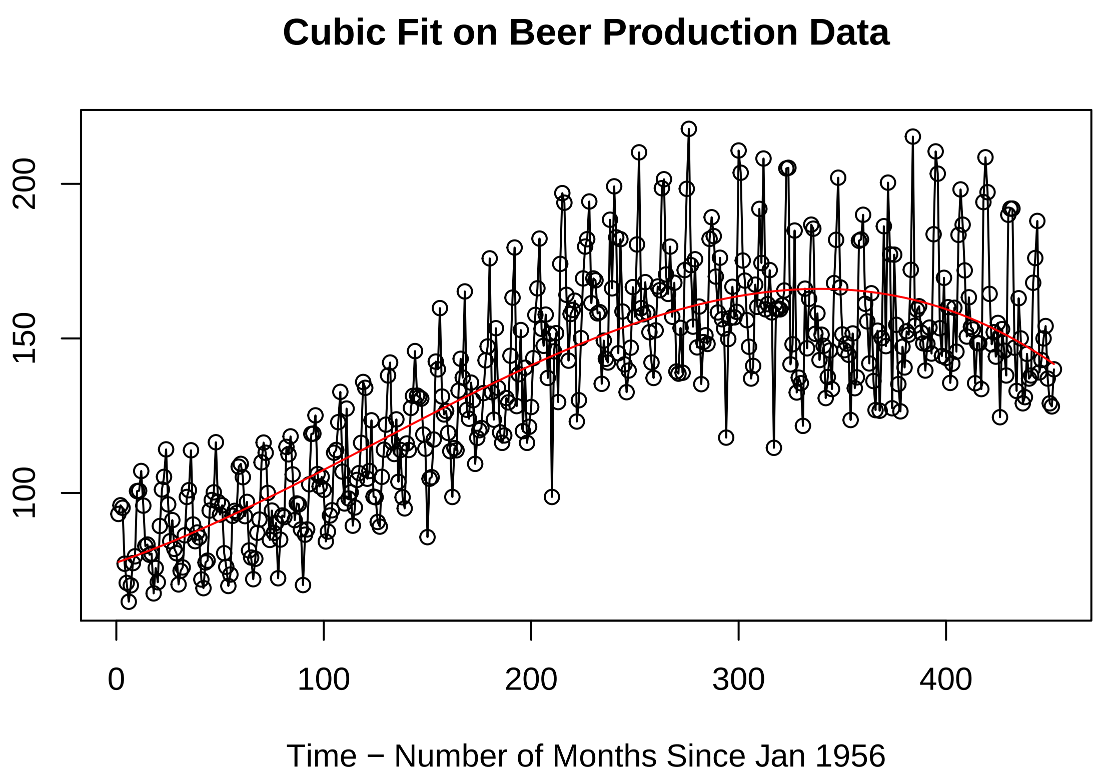
```r
quadFit_resid<-ts(residuals(quadFit),frequency=12, start=c(1956,1))
plot(quadFit_resid, main="Residuals from a Quadratic Trend Fit")
abline(h=0)
```



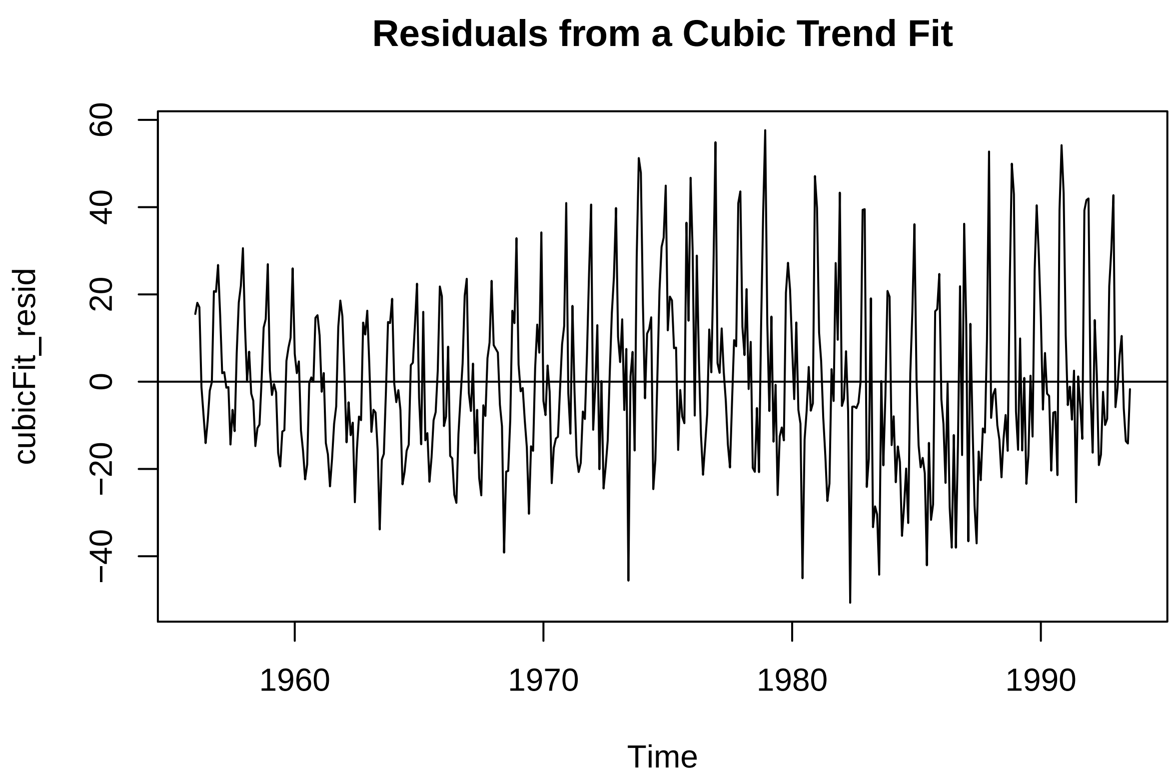

```r
cubicFit<-lm(beerTS~t+t2+t3)
summary(cubicFit)
```

```
##
## Call:
## lm(formula = beerTS ~ t + t2 + t3)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -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
```

```r
#### 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 Fi
curve(expr = coef(cubicFit)[1]+coef(cubicFit)[2]*x+coef(cubicFit)[3]*x^2+coef(cubicFit)[4]*x^3,lty=1,add = TRU
```



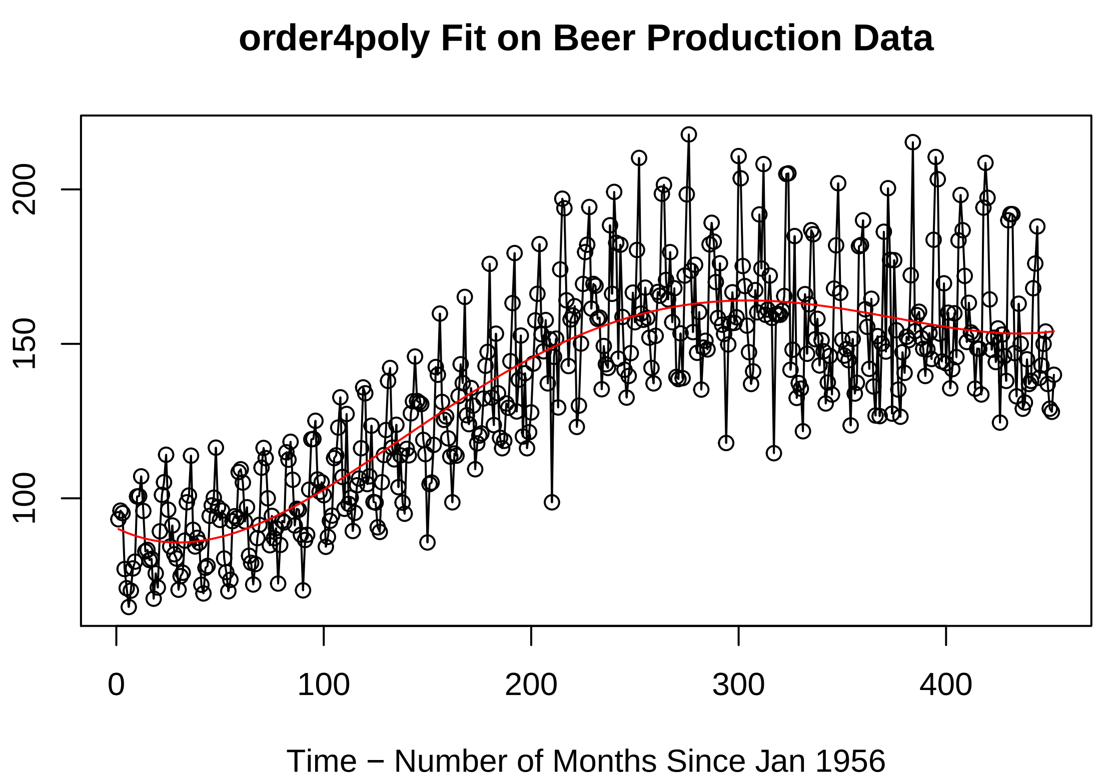

```
cubicFit_resid<-ts(residuals(cubicFit),frequency=12, start=c(1956,1))
plot(cubicFit_resid, main="Residuals from a Cubic Trend Fit")
abline(h=0)
```



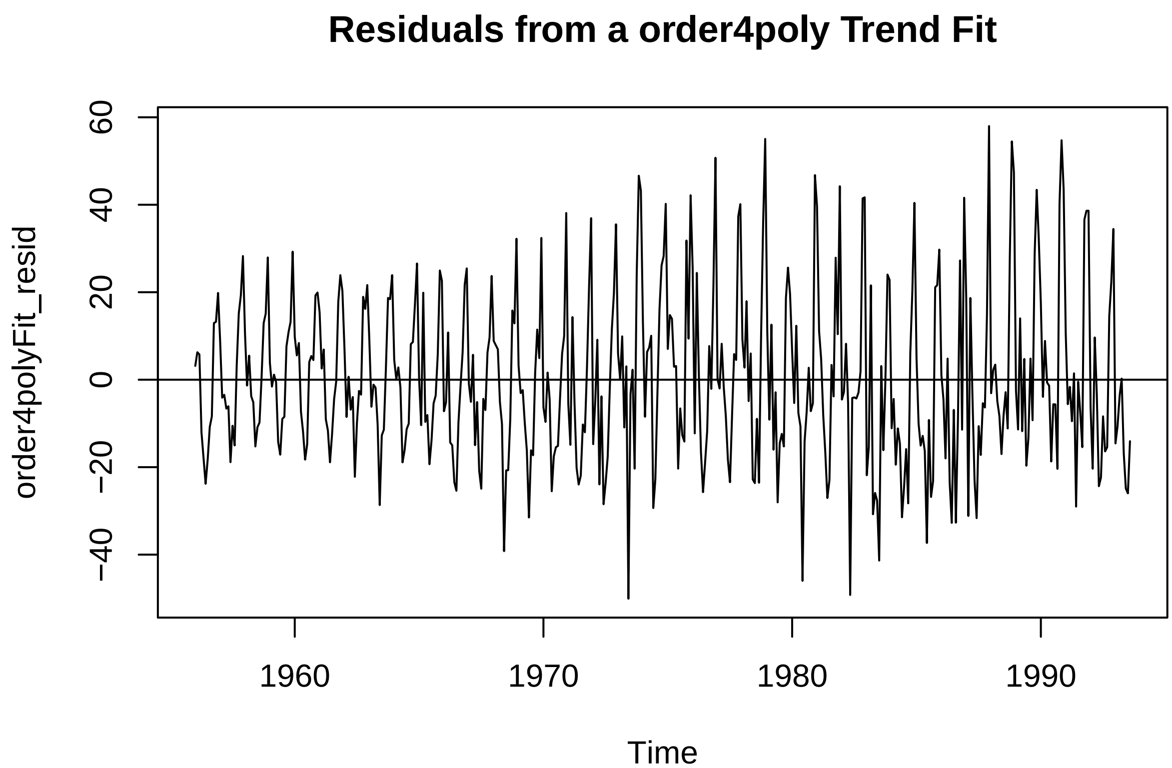
```r
order4polyFit<-lm(beerTS~t+t2+t3+t4)
summary(order4polyFit)
```

```
##
## 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
```

```r
#### 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="order4po
curve(expr = coef(order4polyFit)[1]+coef(order4polyFit)[2]*x+coef(order4polyFit)[3]*x^2+coef(order4polyFit)[4]
```



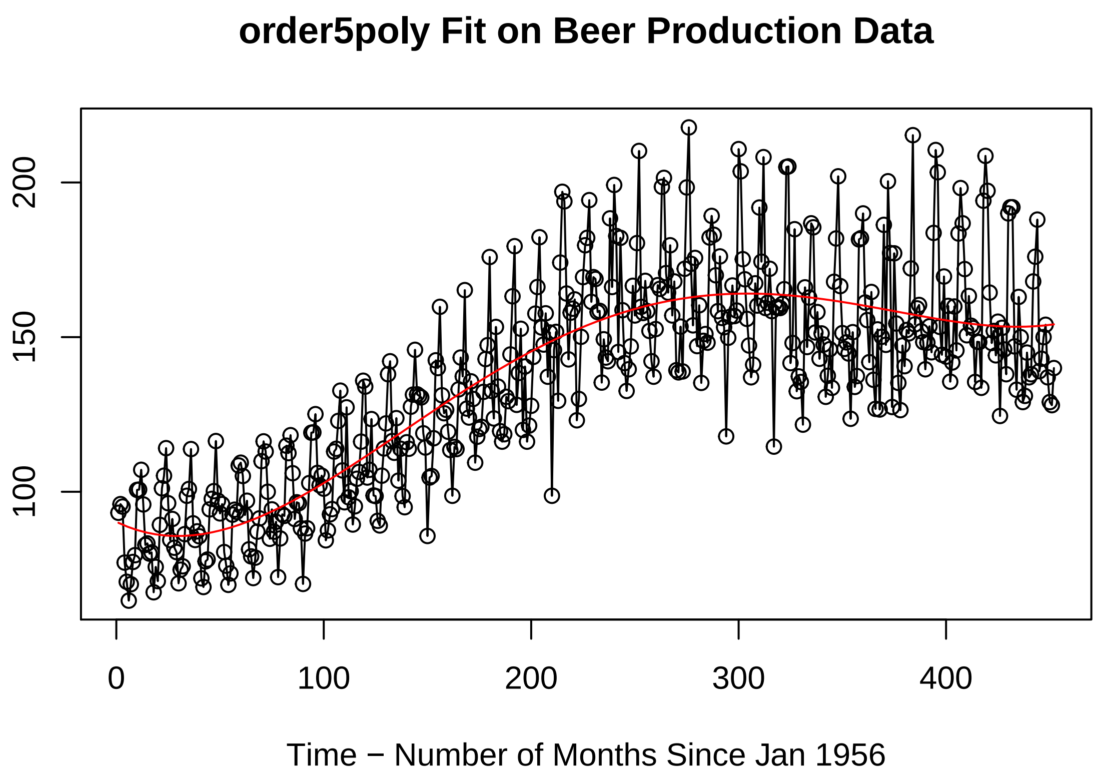
```r
order4polyFit_resid<-ts(residuals(order4polyFit),frequency=12, start=c(1956,1))
plot(order4polyFit_resid, main="Residuals from a order4poly Trend Fit")
abline(h=0)
```



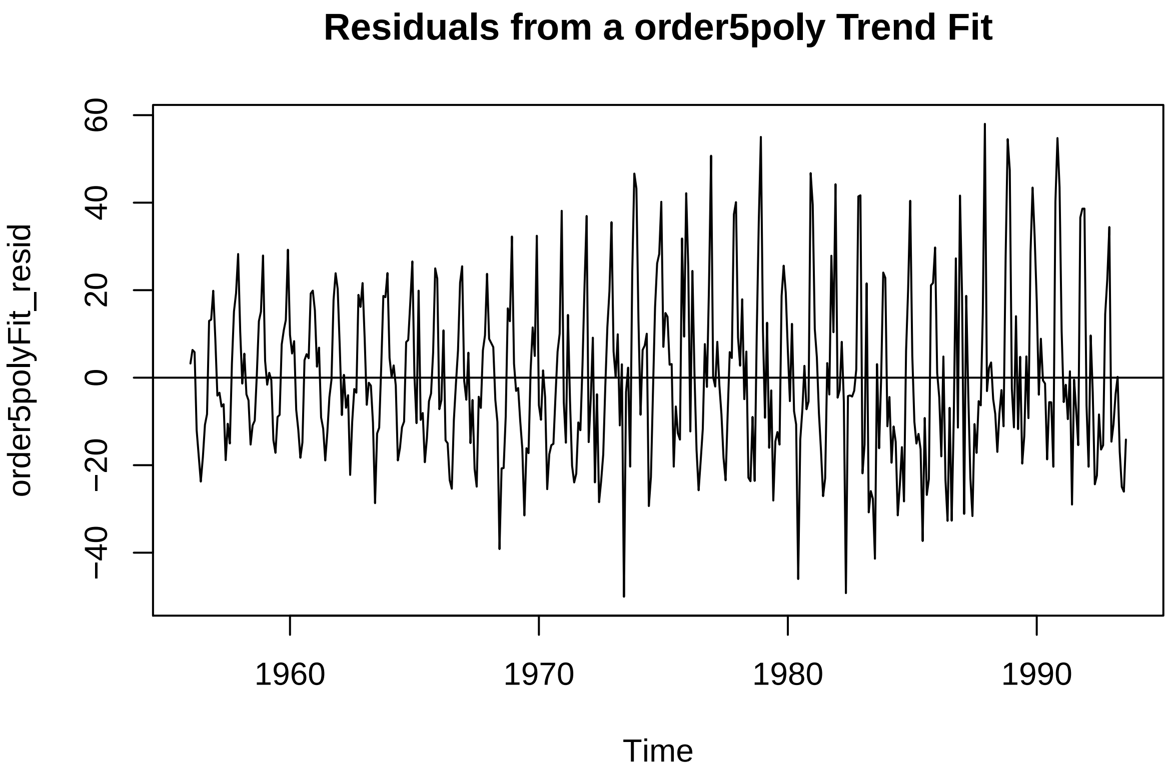

```r
order5polyFit<-lm(beerTS~t+t2+t3+t4+t5)
summary(order5polyFit)
```

```
##
## Call:
## lm(formula = beerTS ~ t + t2 + t3 + t4 + t5)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -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="order5po
curve(expr = coef(order5polyFit)[1]+coef(order5polyFit)[2]*x+coef(order5polyFit)[3]*x^2+coef(order5polyFit)[4]
```




```
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?)

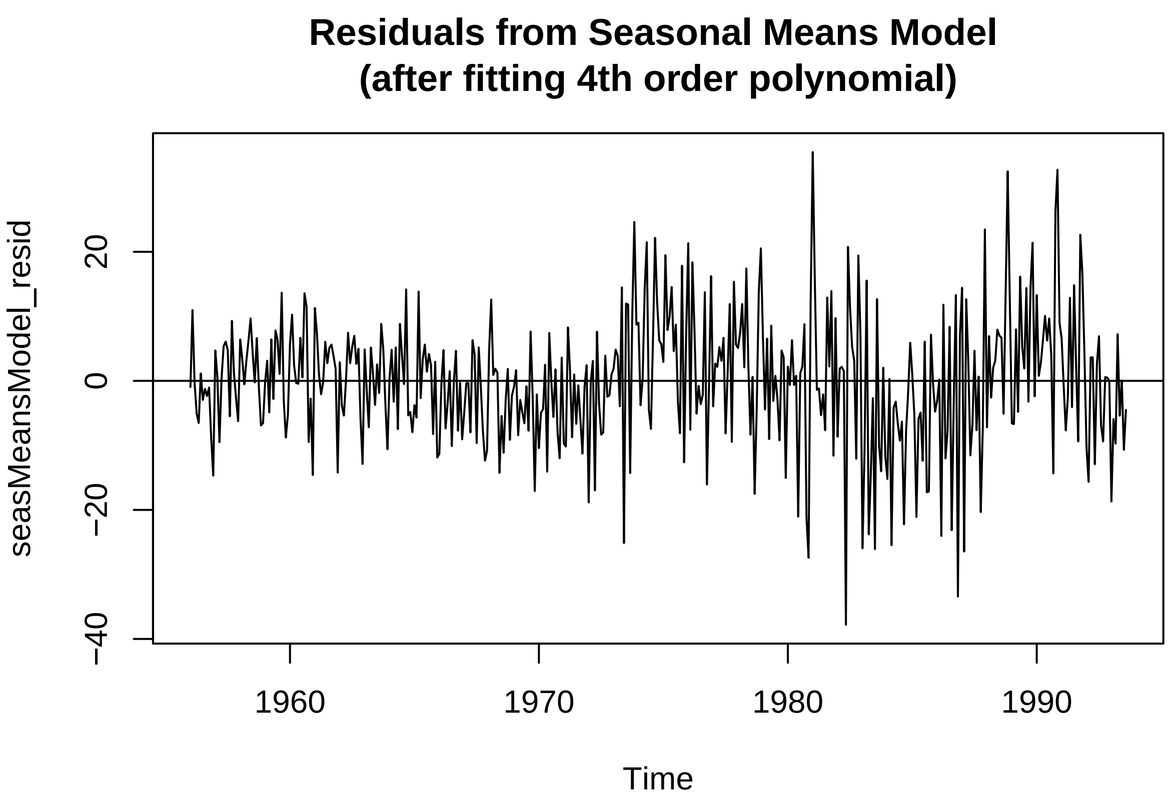
Assume we go with the 4th order polynomial for now. Let's see what we can do about the seasonality with a seasonal means model

```
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)
```



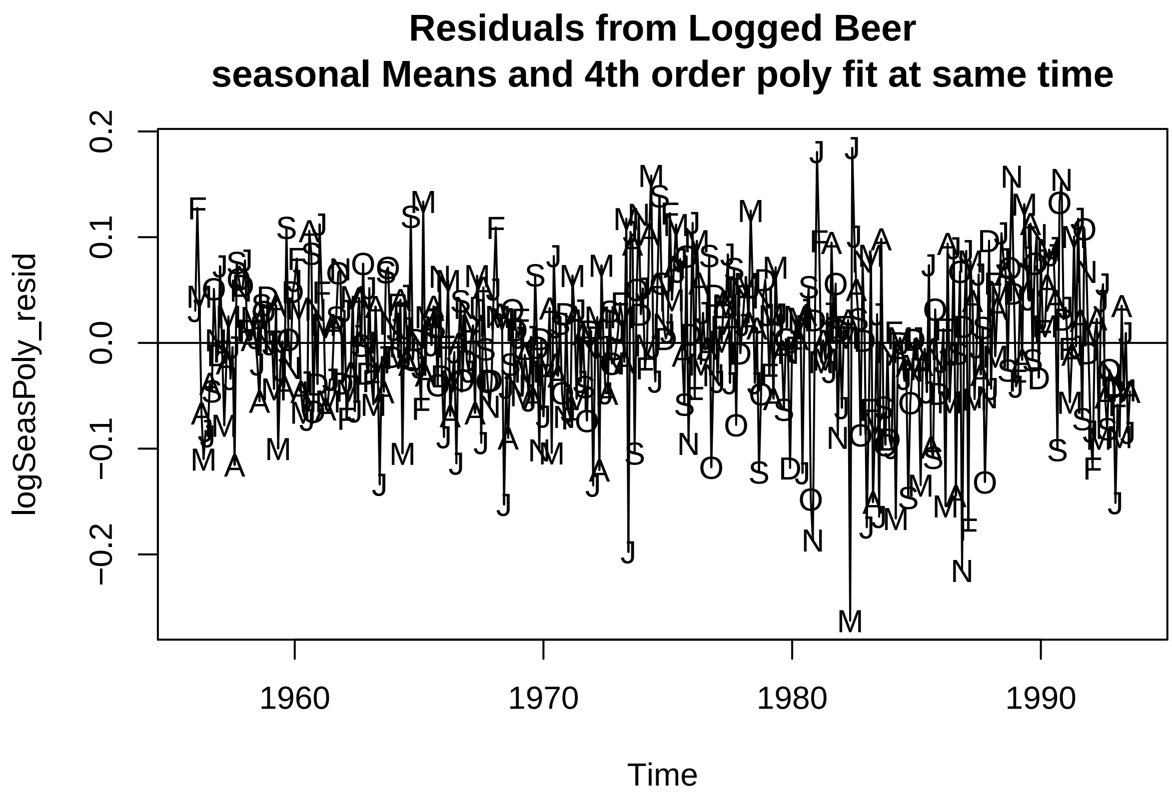
With an adjusted R-squared value of 71%, this is looking pretty good, but in the residual plot you can still the the variance increasing over time. In addition, there is a noticeable "wave" in the residuals that starts around 1970, but I'm not sure what to do about that yet. For now, let's go back, log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

```
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)
```



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

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

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

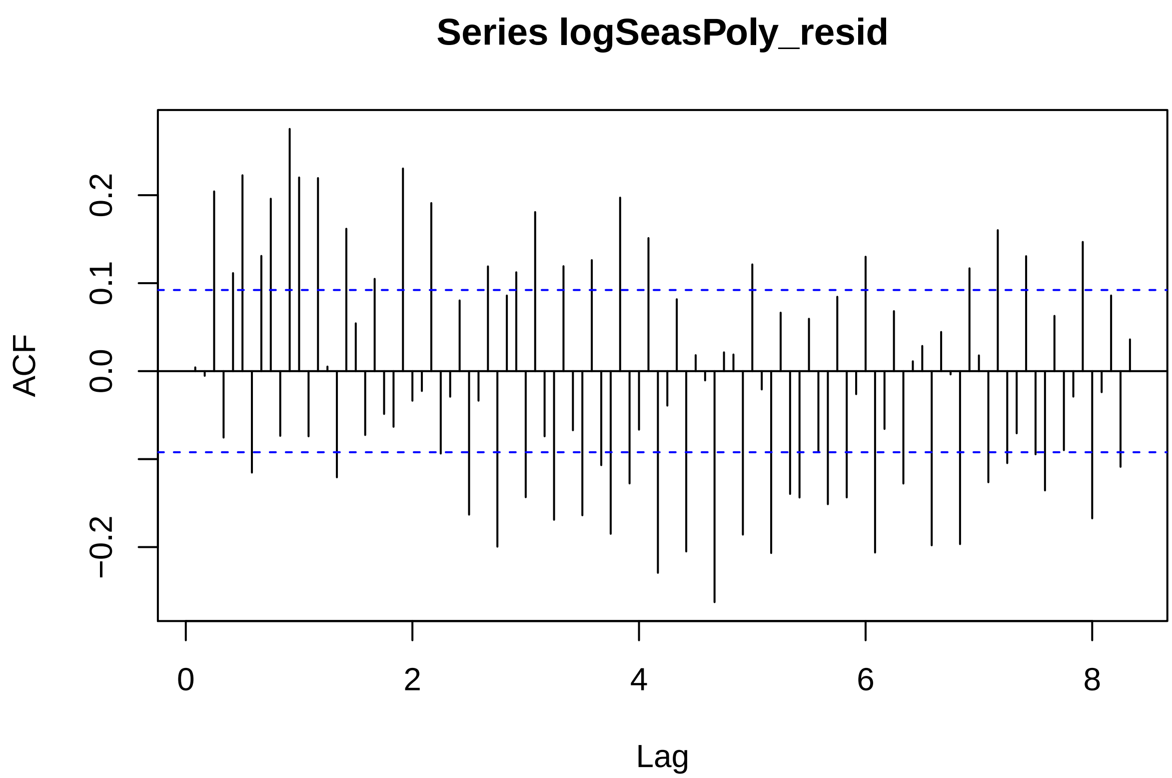
```
##
##  Augmented Dickey-Fuller Test
##
## data:  logSeasPoly_resid
## Dickey-Fuller = -5.3999, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

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

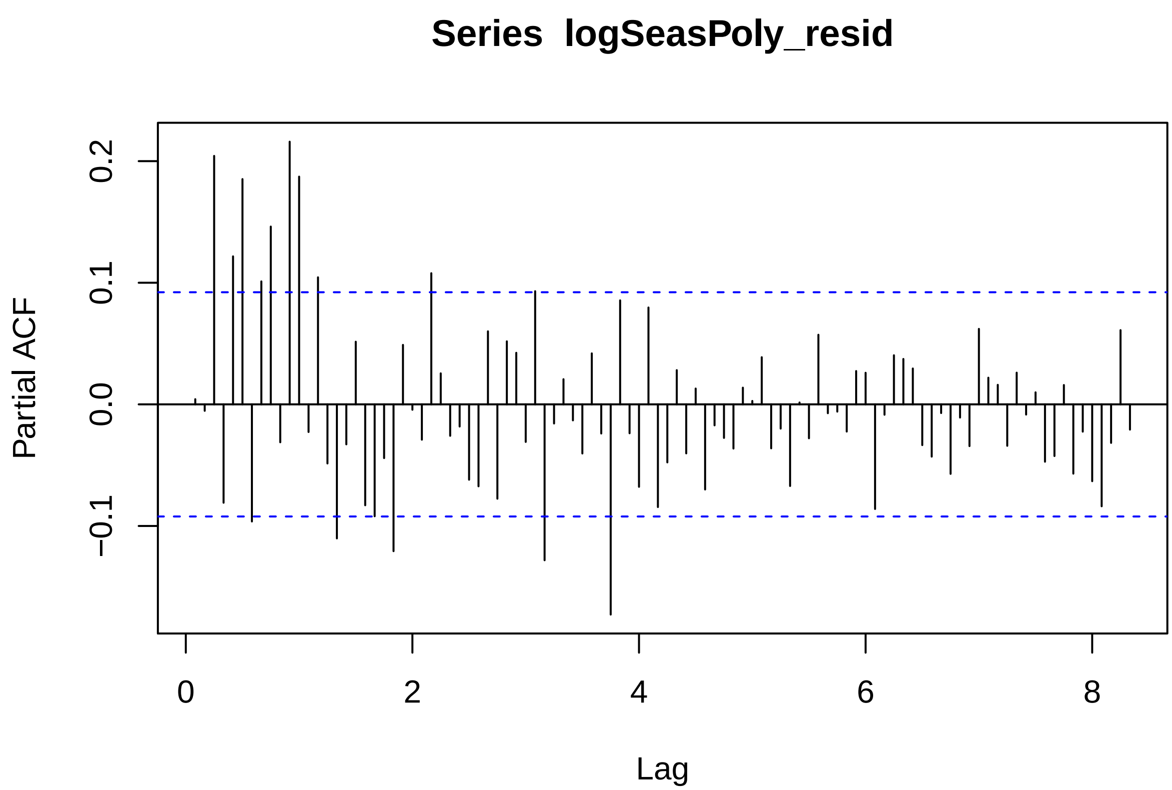
```
## Warning in pp.test(logSeasPoly_resid): p-value smaller than printed p-value
```

```
##
##  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
```

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




```r
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 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 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 x o o o o o  o  o  x
```

Try an AR(12) model and examine residuals

```r
#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
```

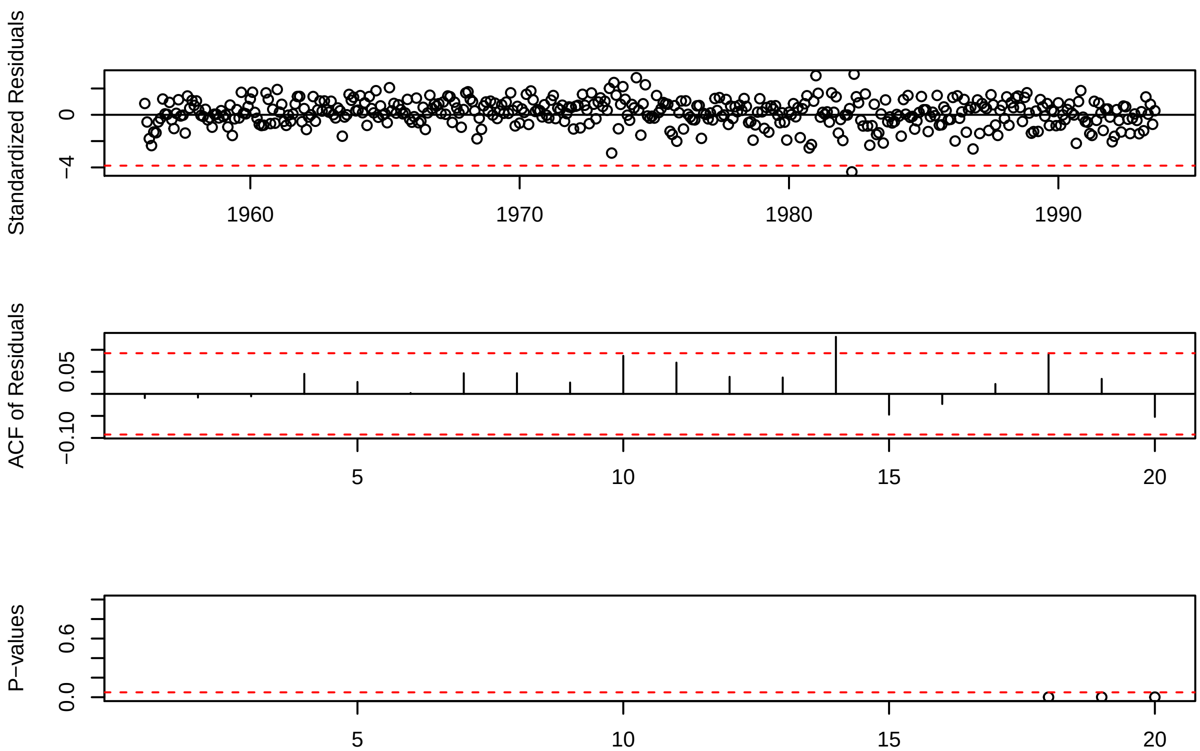
We seem to be having trouble getting fits for the trend line, ask about this Monday, try just using the month dummies.

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

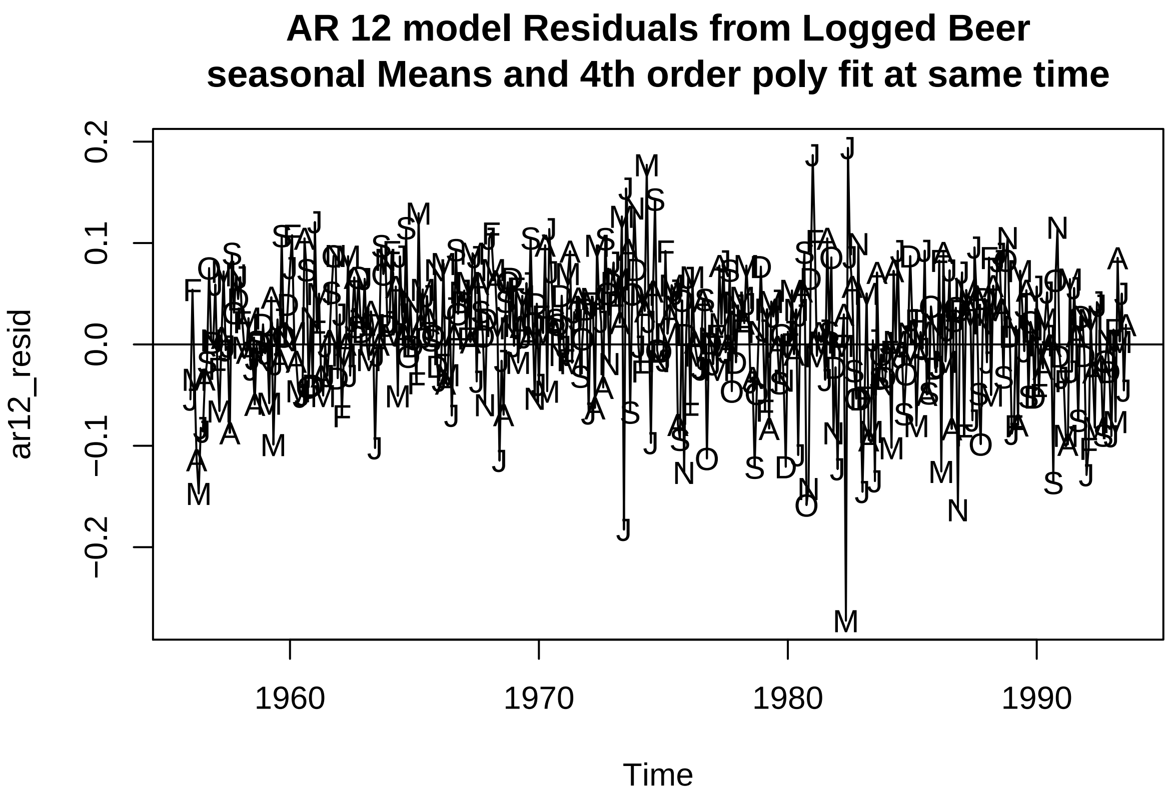
```
##
## Call:
## arima(x = logBeer, order = c(12, 0, 0), xreg = monthDummies)
##
## Coefficients:
##          ar1     ar2     ar3      ar4     ar5     ar6      ar7     ar8
##       0.0631  0.0283  0.1255  -0.0127  0.1286  0.1447  -0.0733  0.0983
## s.e.  0.0464  0.0458  0.0458   0.0457  0.0458  0.0460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

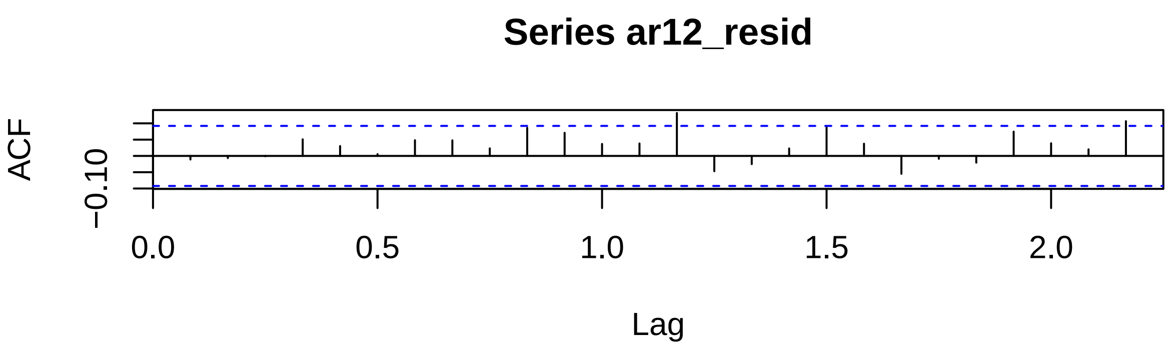
```r
tsdiag(ar12, gof.lag=20)
```

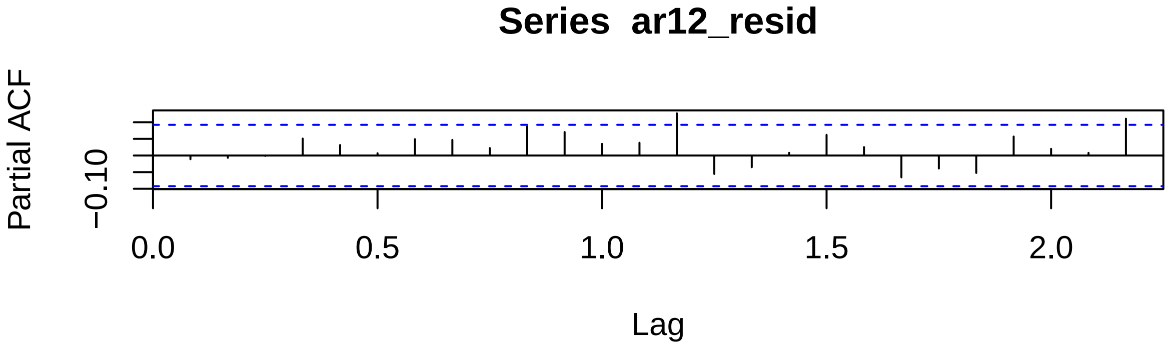



```r
#residuals
ar12_resid<-ts(residuals(ar12), frequency=12, start=c(1956,1))
plot(ar12_resid, main="AR 12 model Residuals from Logged Beer\nseasonal Means and 4th order poly fit at same t
points(y=ar12_resid, x=time(ar12_resid), pch=as.vector(season(ar12_resid)))
abline(h=0)
```




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

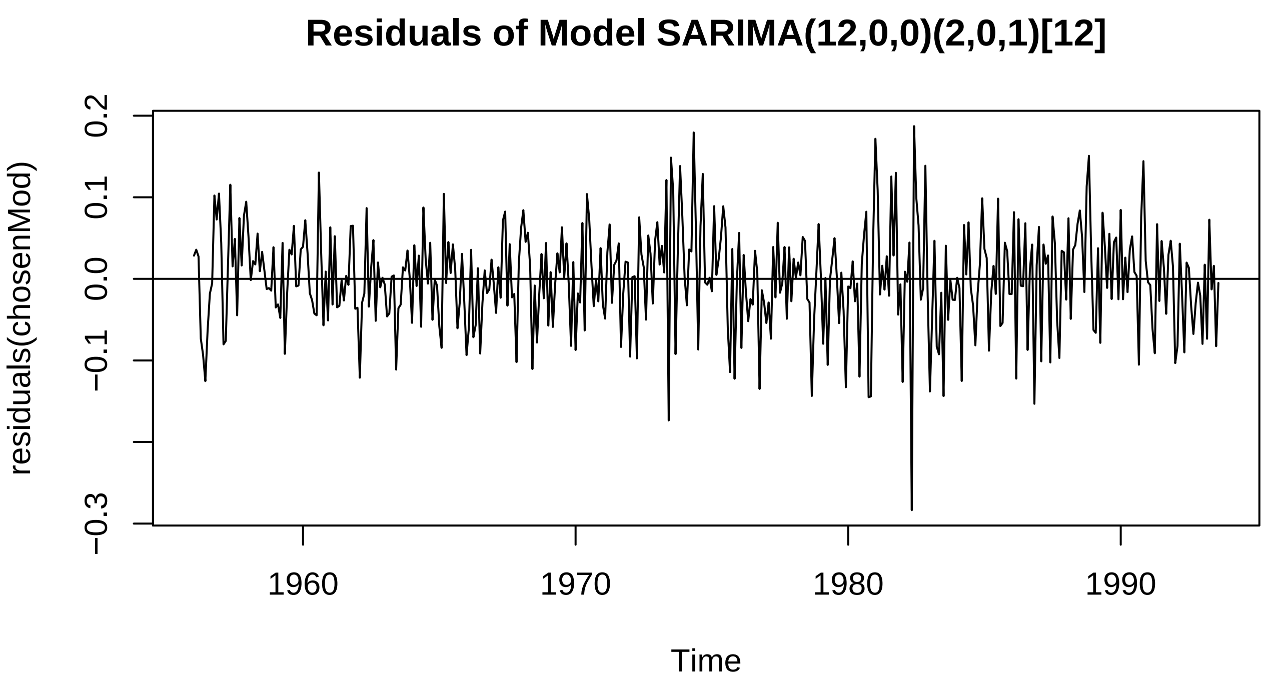




```
pacf_acf<-data.frame(acfVal=acf(ar12_resid, plot=FALSE)$acf, pacfVal=pacf(ar12_resid, plot=FALSE)$acf)
#print(pacf_acf)
```

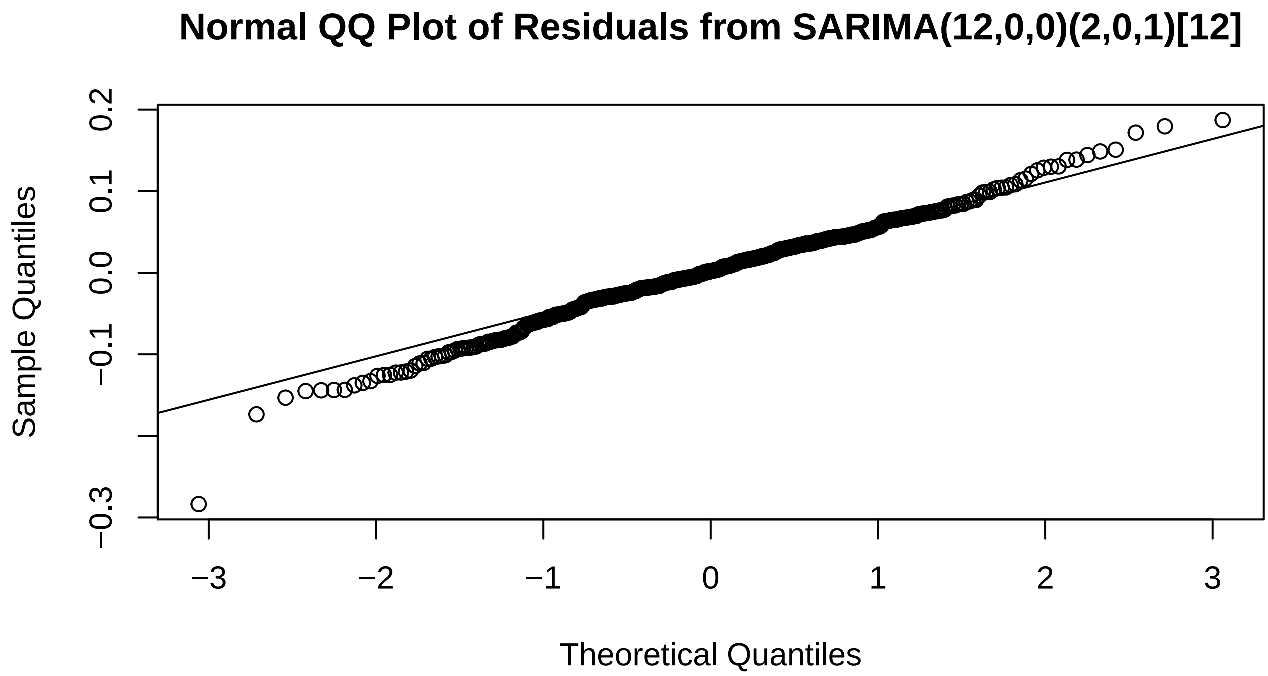
After we choose a model, run all of the diagnostic tests

```
chosenMod<-popModel8
modelString<-getModelString(chosenMod)

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



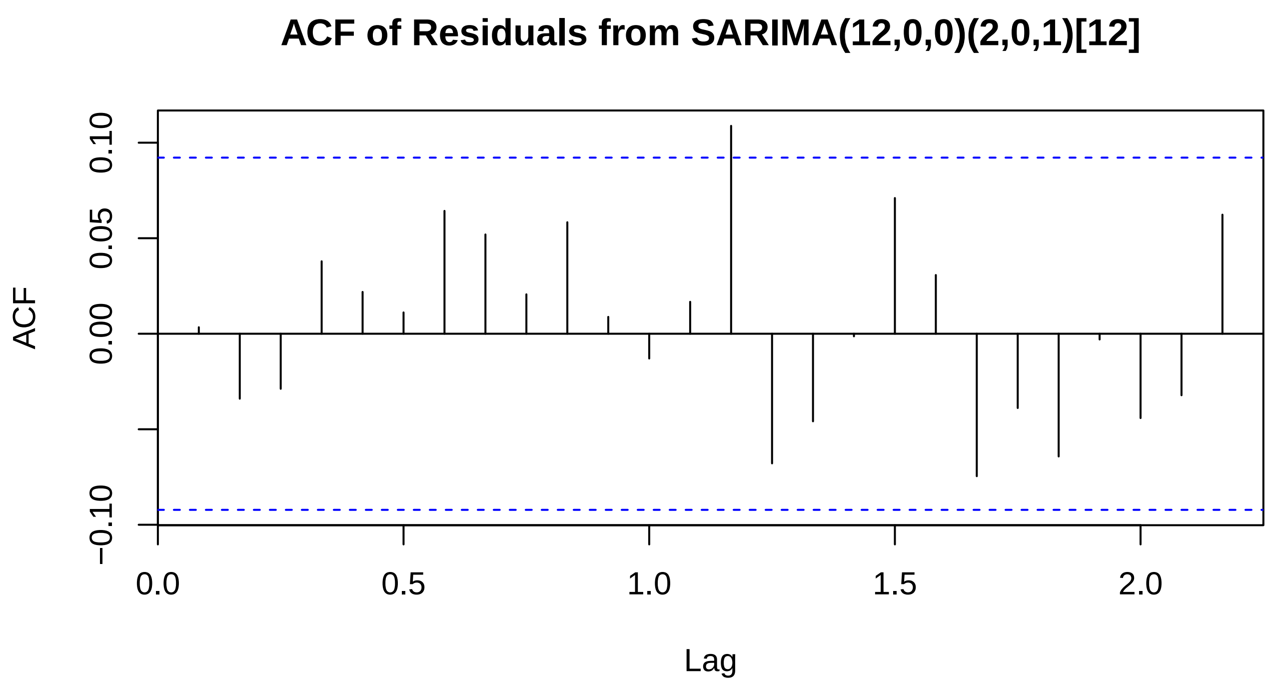
Comment:

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



Comment:

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



Comment:

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

```
##
```

```
##  Shapiro-Wilk normality test
##
## data:  residuals(chosenMod)
## W = 0.99006, p-value = 0.003776
```

Comment:

```r
LB.test(chosenMod, lag=35)
```

```
##
##  Box-Ljung test
##
## data:  residuals from  chosenMod
## X-squared = 37.5, df = 20, p-value = 0.01019
```

Comment:

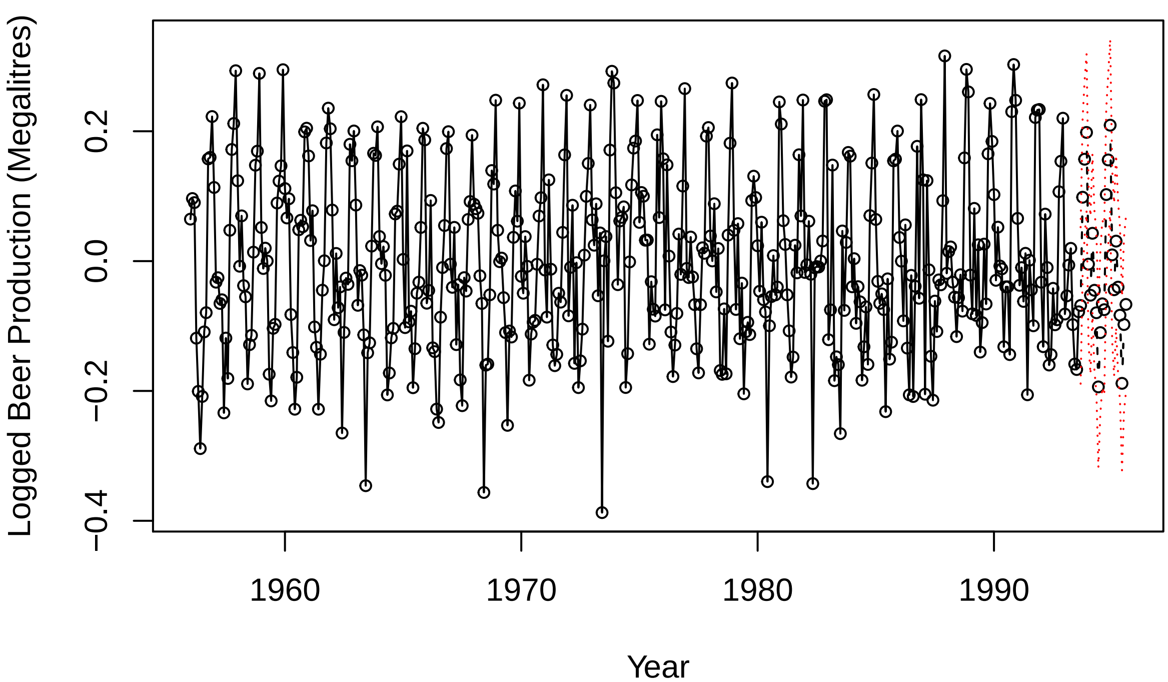
Make the forecasts

Set up external regressor data frame

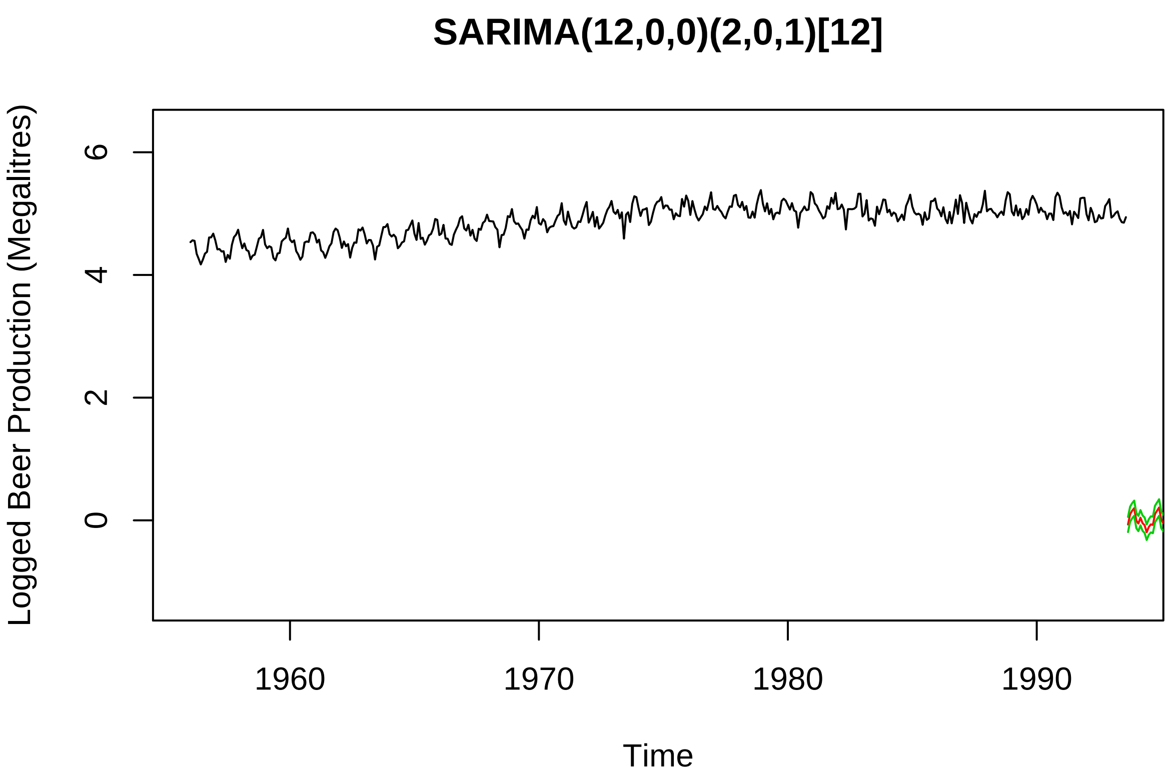
```r
newMonthDummy<-seasonaldummy(beer_forecast)
```

Plot the model forecasts

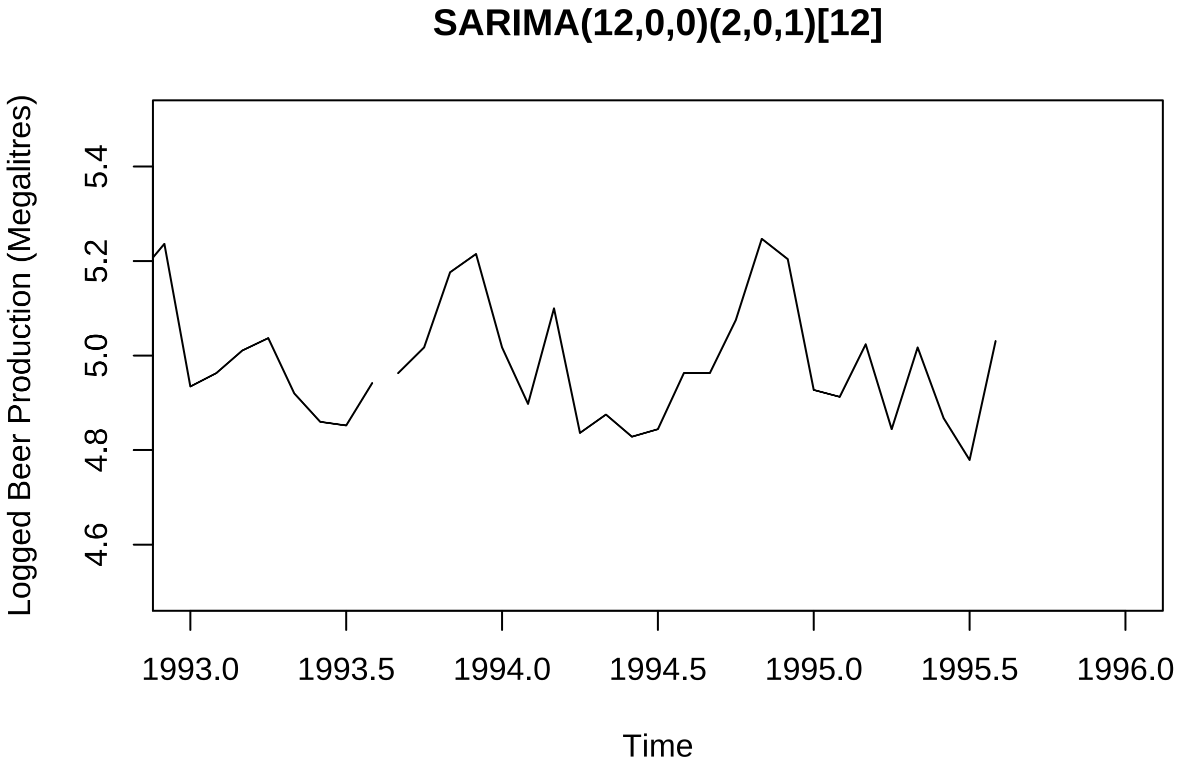
```r
library(TSA)

TSA::plot.Arima(chosenMod,n.ahead=24,n1=c(1956,1), type='b',ylab='Logged Beer Production (Megalitres)',xlab='Y
```



```
predictions<-predict(chosenMod, 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)
```




```
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)(2,0,1)[12]
Logged Beer Production (Megalitres)
5.4
5.2
5.0
4.8
4.6
1993.0
1993.5
1994.0
1994.5
1995.0
1995.5
1996.0
Time