

Time Series Lab and assignment

Neeraj Namani

TimeSeries Lab & Assignment

Dataset: “birth” from libraty astda, U.S. Monthly Live Births 1950-1980

```
library(astsa)
data(birth)
head(birth)
```

```
## [1] 295 286 300 278 272 268
```

```
#plot(birth)
library(ggplot2)
library(ggfortify)
library(dplyr)
```

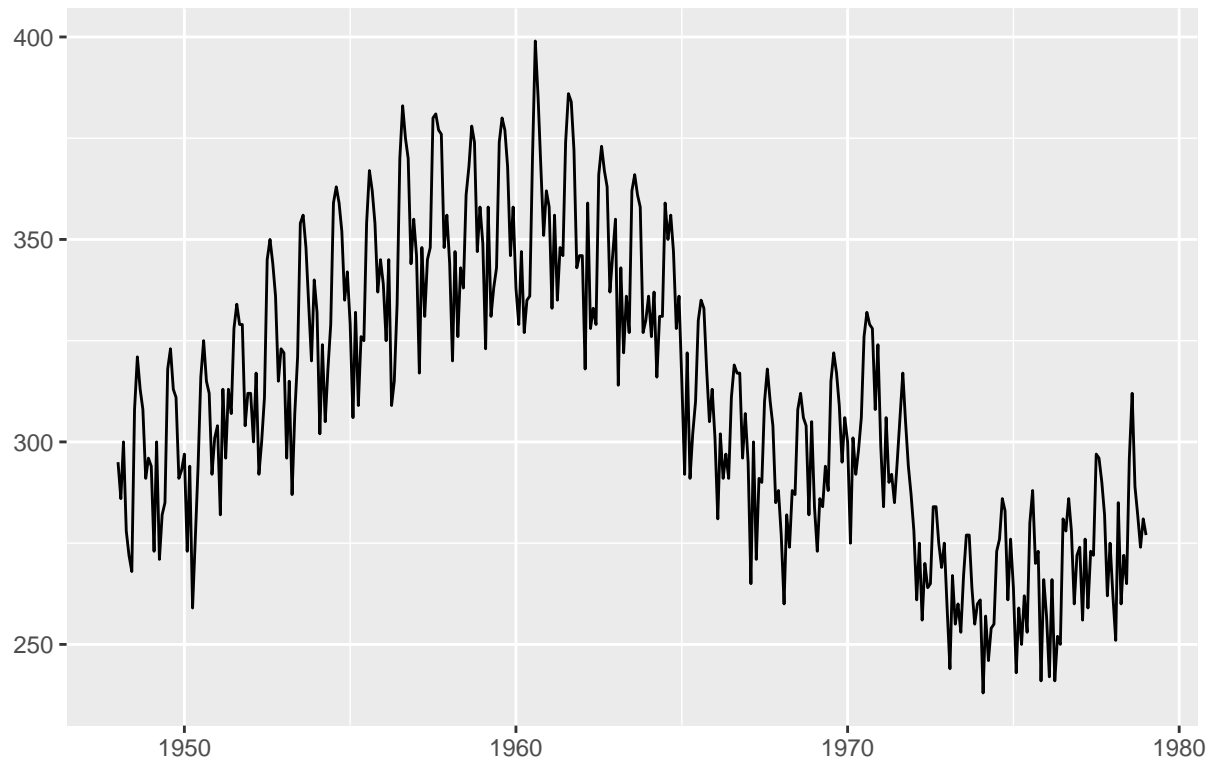
```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##   filter, lag

## The following objects are masked from 'package:base':
##
##   intersect, setdiff, setequal, union
```

```
birth %>%
  autoplot() + ggtitle("U.S. Monthly Live Births 1950-1980")
```

U.S. Monthly Live Births 1950–1980



A seasonal plot is similar to a time plot except that the data are plotted against the individual “seasons” in which the data were observed.

```
library(forecast)
```

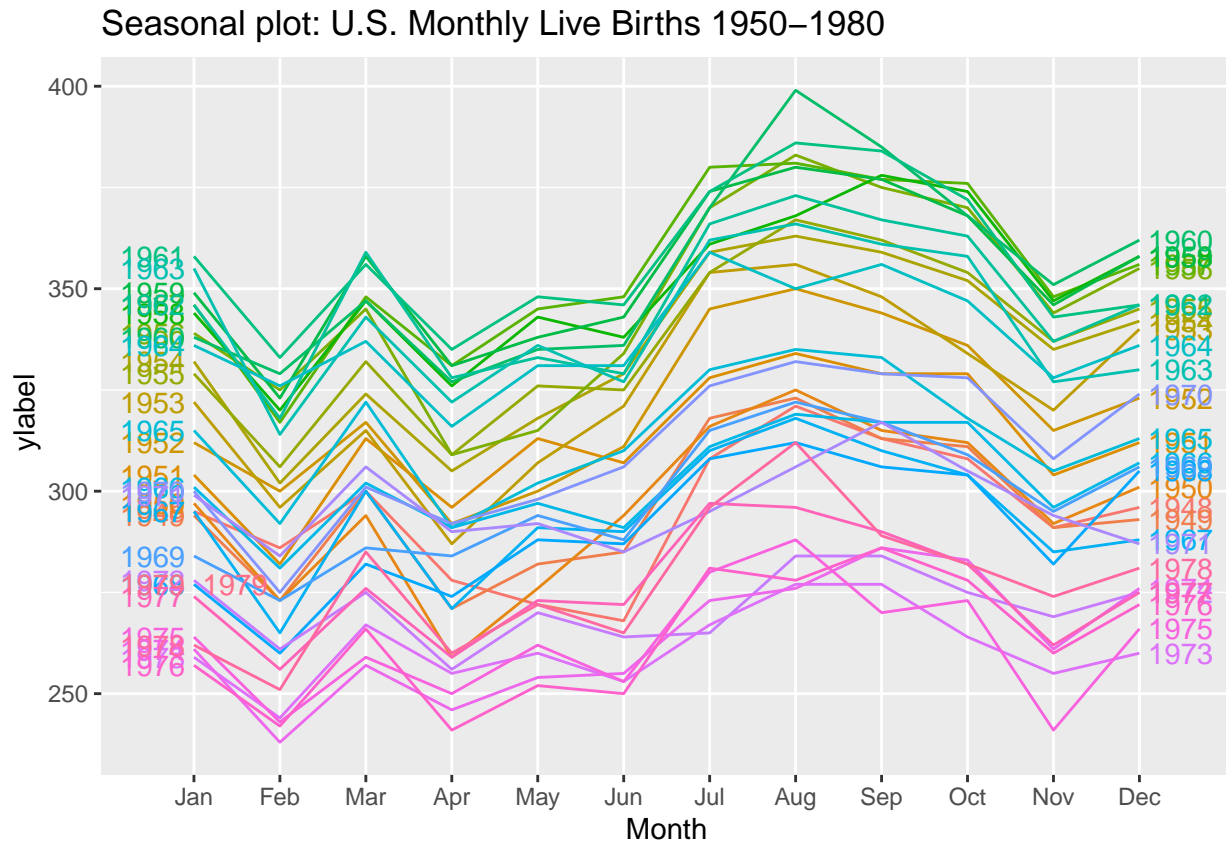
```
## Registered S3 method overwritten by 'quantmod':
##   method      from
##   as.zoo.data.frame zoo

## Registered S3 methods overwritten by 'forecast':
##   method      from
##   autoplot.Arima      ggfortify
##   autoplot.acf        ggfortify
##   autoplot.ar         ggfortify
##   autoplot.bats       ggfortify
##   autoplot.decomposed.ts ggfortify
##   autoplot.ets        ggfortify
##   autoplot.forecast   ggfortify
##   autoplot.stl        ggfortify
##   autoplot.ts         ggfortify
##   fitted.ar           ggfortify
##   fortify.ts          ggfortify
##   residuals.ar        ggfortify

##
## Attaching package: 'forecast'

## The following object is masked from 'package:astsa':
##
##   gas
```

```
ggseasonplot(birth, year.labels=TRUE, year.labels.left=TRUE) +
  ylab(" ylabel") +
  ggtitle("Seasonal plot: U.S. Monthly Live Births 1950-1980")
```



We are going to try a few things to get a feeling about the cyclical nature of the dataset.

There seems to be a yearly cycle. We can try adding monthly variables or use a sin and/or cos with the right frequency for a year repetition.

Note: I added numbers to the names of the month because otherwise r will order them alphabetically.

```
n<-length(birth)
#n=373, n/12 = 31.08
month<-rep(c("01Jan", "02Feb", "03Mar", "04Apr", "05May", "06Jun", "07Jul", "08Aug", "09Sep", "10Oct", "11Nov", "12Dec"),
times<-1:n

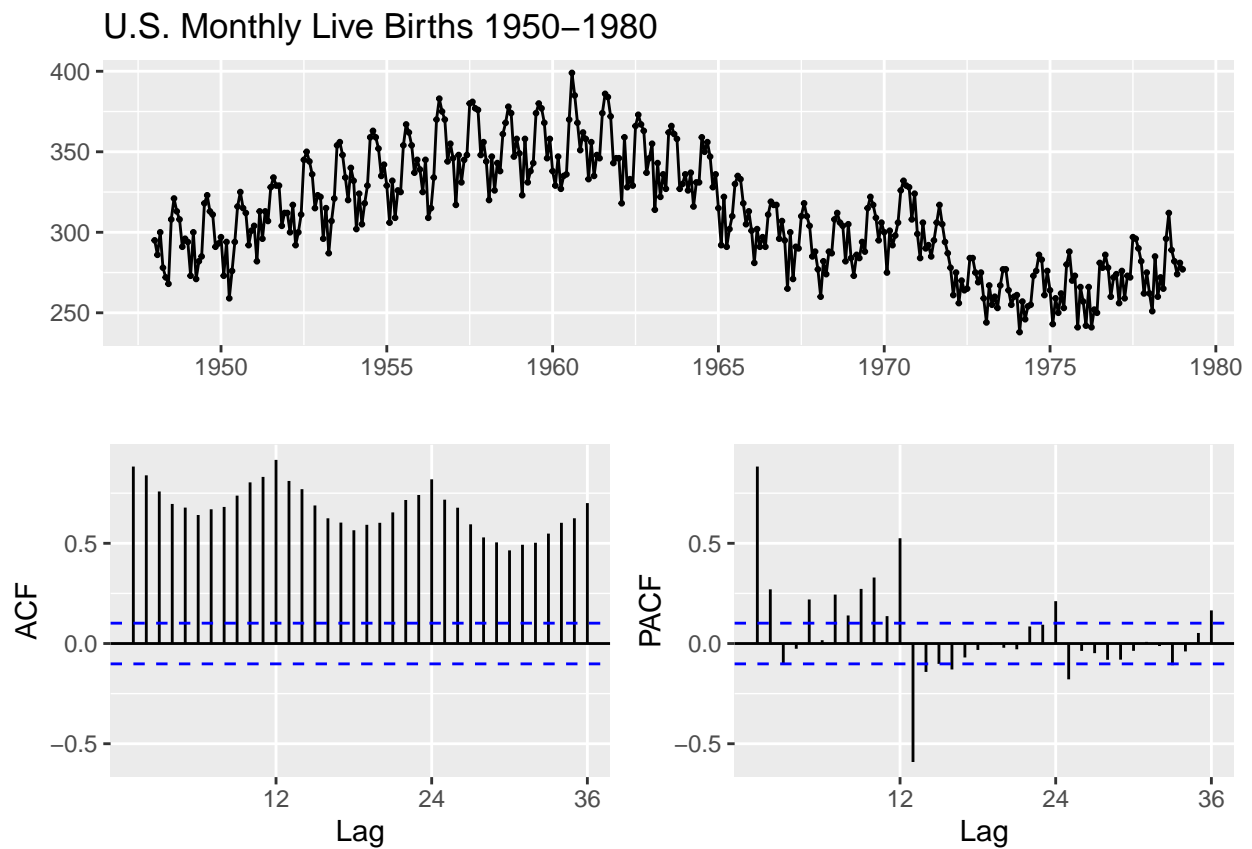
# we won't use all the monthly dummy variables because Jan = when all other are 0
#X<-as.data.frame(cbind(times,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec))
X<-data.frame(times=times,month=month)

# alternatively, seasons can be created with sin, cos
sint=sin(2*pi*times/12)
cost=cos(2*pi*times/12)
X_jan=data.frame(times=times,sint=sint,cost=cost,Jan=rep(c(1,0,0,0,0,0,0,0,0,0,0,0),32)[1:n])
```

Let's look at the auto correlation function and partial auto correlation functions

```
#acf(birth)
#pacf(birth)
```

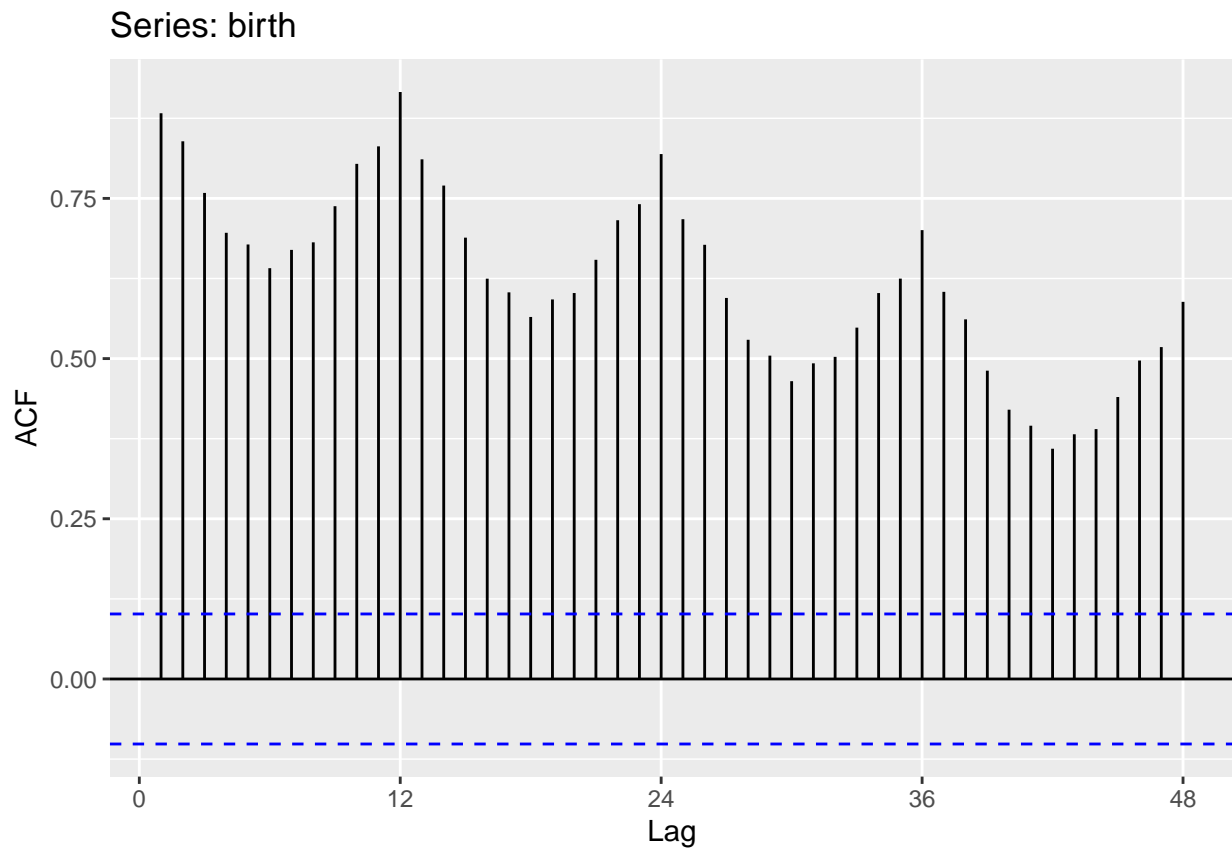
```
birth %>% ggtsdisplay(main="U.S. Monthly Live Births 1950-1980")
```



The acf also shows the cyclic pattern.

Using ggplot2

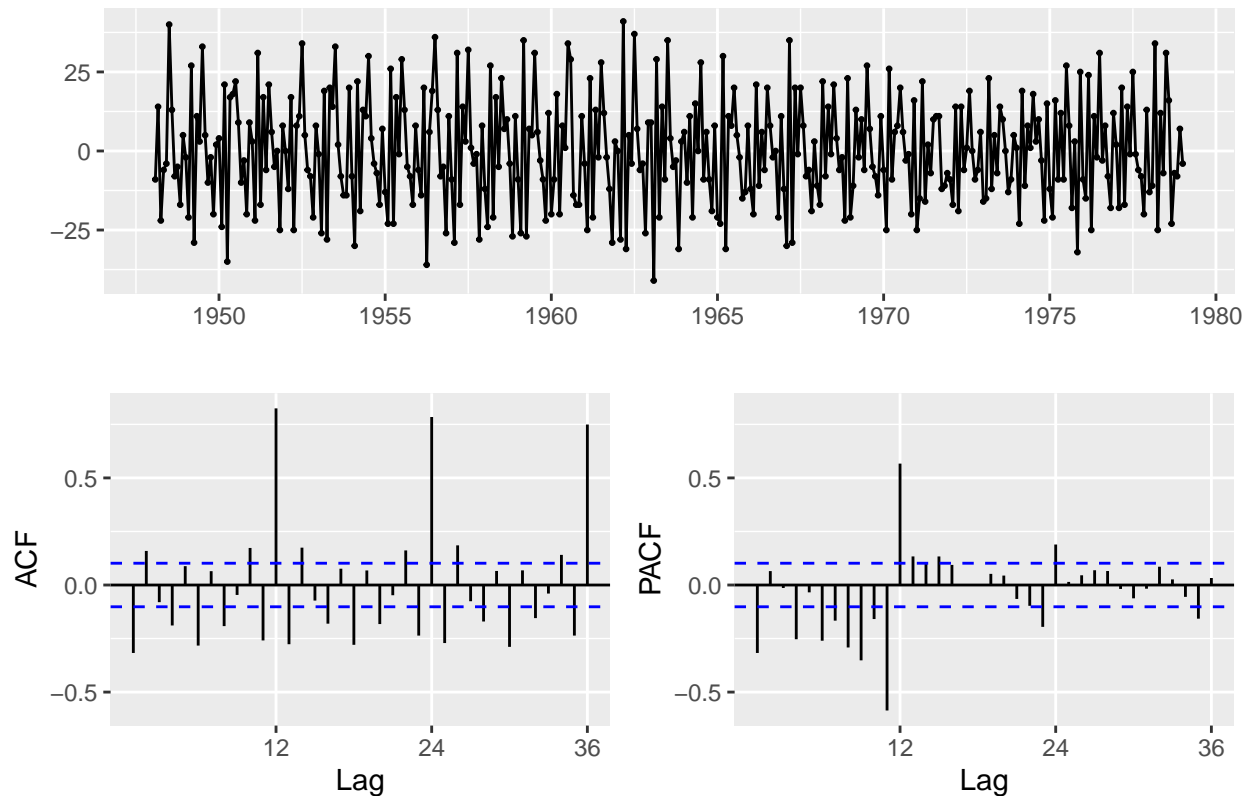
```
ggAcf(birth, lag=48) # default is lag=24
```



Looking at the differences:

```
#acf(diff(birth,1))  
birth %>% diff() %>% ggtsdisplay(main="Differences of Birth series")
```

Differences of Birth series



The spikes at 12, 24, 36 tell about the cycle of 12 months. The difference seems to make the series more stable but still have issues with autocorrelation.

Let's fit a model with monthly dummy variables. There is a curve trend that is beyond quadratic.

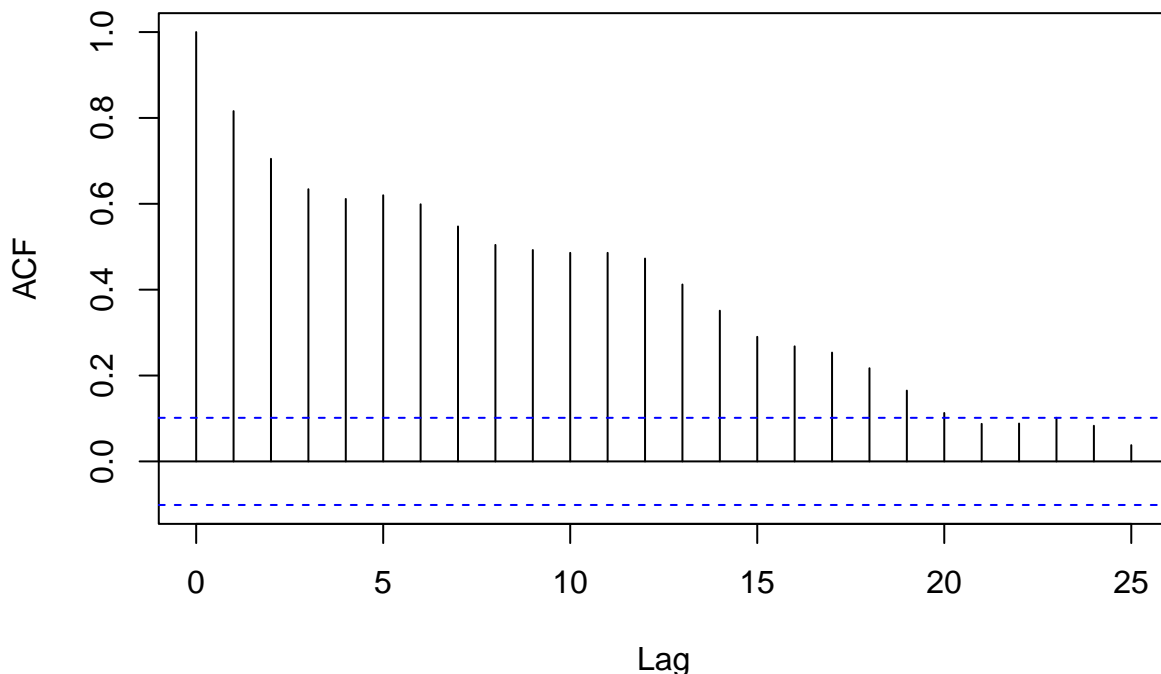
```
lsfit=lm(birth~poly(times,3)+month,
#       Feb+Mar+Apr+May+Jun+Jul+Aug+Sep+Oct+Nov+Dec,
#       data=X)
summary(lsfit)
```

```
##
## Call:
## lm(formula = birth ~ poly(times, 3) + month, data = X)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -30.806  -8.521  -1.008   9.051  41.496
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    307.478     2.164  142.119 < 2e-16 ***
## poly(times, 3)1 -356.462    12.241  -29.120 < 2e-16 ***
## poly(times, 3)2 -369.891    12.239  -30.222 < 2e-16 ***
## poly(times, 3)3  245.762    12.247   20.066 < 2e-16 ***
## month02Feb      -20.826     3.084   -6.752 5.90e-11 ***
## month03Mar        2.731     3.084    0.885  0.3766
## month04Apr      -17.837     3.084   -5.783 1.60e-08 ***
## month05May       -6.853     3.084   -2.222  0.0269 *
## month06Jun       -6.284     3.084   -2.038  0.0423 *
```

```
## month07Jul      19.869      3.084      6.442 3.79e-10 ***
## month08Aug      27.219      3.084      8.826 < 2e-16 ***
## month09Sep      23.154      3.084      7.507 4.84e-13 ***
## month10Oct      16.705      3.084      5.416 1.12e-07 ***
## month11Nov      -2.998      3.084     -0.972   0.3316
## month12Dec       6.398      3.084      2.074   0.0388 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.24 on 358 degrees of freedom
## Multiple R-squared:  0.884, Adjusted R-squared:  0.8795
## F-statistic: 194.9 on 14 and 358 DF,  p-value: < 2.2e-16
```

```
acf(lsfit$res)
```

Series lsfit\$res



Al-

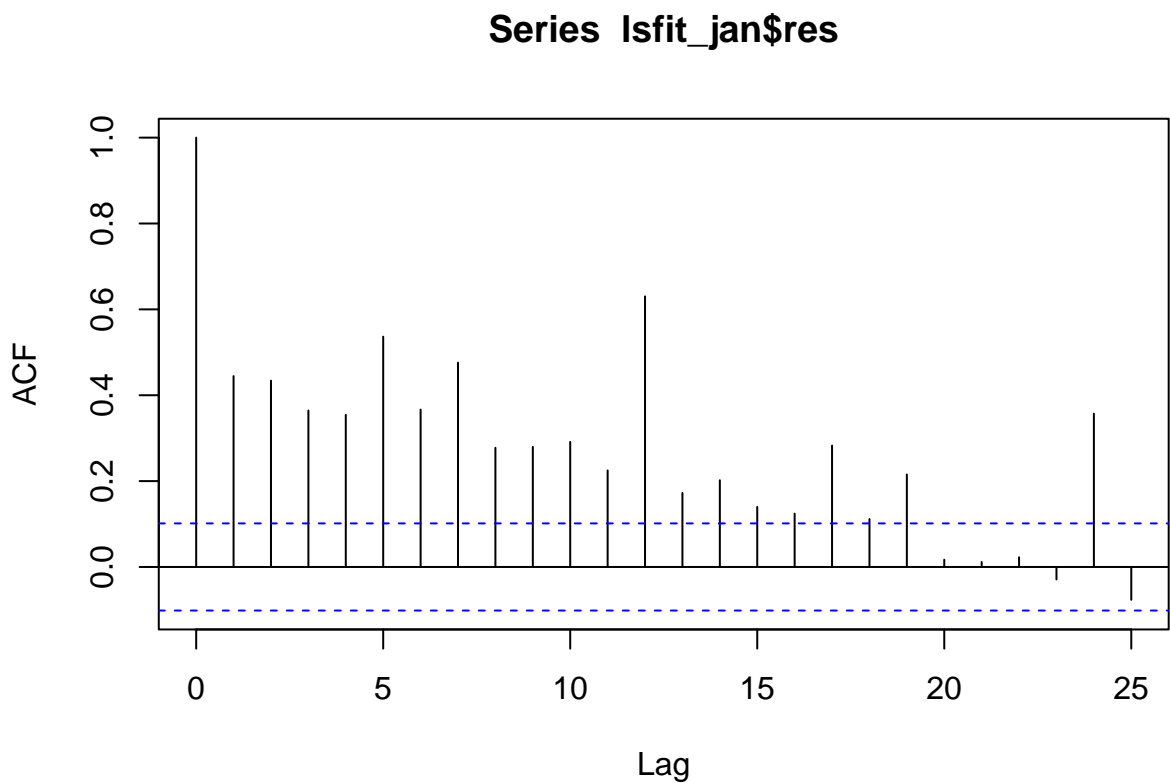
though this looks like a good fit, we see that the residuals have autocorrelation.

Let's also fit a model with sin and cos to model cyclical nature.

```
lsfit_jan=lm(birth~poly(times,3)+sint+cost+Jan,data=X_jan) #you remove sin/cos and do all months
summary(lsfit_jan)
```

```
##
## Call:
## lm(formula = birth ~ poly(times, 3) + sint + cost + Jan, data = X_jan)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -33.58 -11.16  -1.32   10.30   48.34
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)      310.2118      0.8069 384.462 < 2e-16 ***
## poly(times, 3)1 -356.6168     14.7680 -24.148 < 2e-16 ***
## poly(times, 3)2 -369.8896     14.7665 -25.049 < 2e-16 ***
## poly(times, 3)3  245.5296     14.7736  16.620 < 2e-16 ***
## sint            -18.0085      1.1130 -16.181 < 2e-16 ***
## cost            -2.5458      1.1695  -2.177 0.03013 *
## Jan              8.4756      3.0262   2.801 0.00537 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.76 on 366 degrees of freedom
## Multiple R-squared:  0.8274, Adjusted R-squared:  0.8246
## F-statistic: 292.5 on 6 and 366 DF,  p-value: < 2.2e-16
acf(lsfit_jan$res)
```



Same

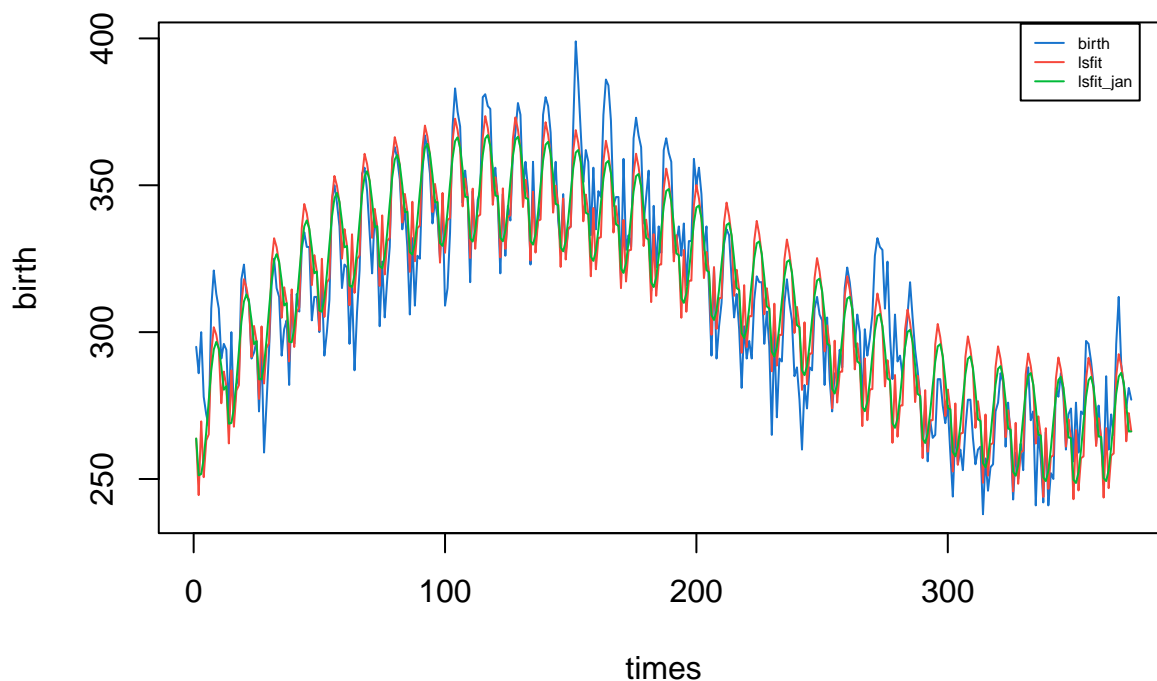
problem with this model, we also see that the residuals still have autocorrelation.

Let's plot both models:

```
plot(times,birth,type="l",main="U.S. Monthly Live Births 1950-1980",col=4)

lines(times,lsfit$fitted.values,col=2)
lines(times,lsfit_jan$fitted,col=3)
legend(329,405,c("birth","lsfit","lsfit_jan"),col=c(4,2,3),lty=1,cex=.5)
```


U.S. Monthly Live Births 1950–1980



```
#df<-data.frame(fit=lsfit$fitted.values, times=times)
#df2<-data.frame(fit=lsfit_jan$fitted.values, times=times)
#birth %>%
#  autoplot(col="darkgrey") +
#  ggtitle("U.S. Monthly Live Births 1950-1980") +
#  geom_line(data=df,aes(x=time(birth),y=fit),col=2)+
#  geom_line(data=df2,aes(x=time(birth),y=fit),col=3)
```

Which model performs better?

```
aic<-round(c(AIC(lsfit), AIC(lsfit_jan)),2)
bic<-round(c(BIC(lsfit), BIC(lsfit_jan)),2)
adjr2<-round(c(summary(lsfit)$ad,summary(lsfit_jan)$ad),2)
rbind(c("lsfit", "lsfit_jan"), aic,bic,adjr2)
```

```
##      [,1]      [,2]
##      "lsfit"  "lsfit_jan"
## aic  "2943.52" "3075.81"
## bic  "3006.26" "3107.19"
## adjr2 "0.88"   "0.82"
```

Now let's try the time series model with auto-regressive, integrated, moving averages and cyclic components:

```
library(forecast)
birthmod<-auto.arima(birth)
birthmod
```

```
## Series: birth
## ARIMA(0,1,2)(1,1,1)[12]
##
## Coefficients:
##          ma1          ma2          sar1          sma1
```

```
##          -0.3984  -0.1632  0.1018  -0.8434
## s.e.      0.0512   0.0486  0.0713   0.0476
##
## sigma^2 = 46.1:  log likelihood = -1204.93
## AIC=2419.86   AICc=2420.03   BIC=2439.29
```

The result is ARIMA(0,1,2)(1,1,1)[12] We also see the aic and the bic metrics and this model performed better than the ones we did earlier.

Equation corresponding to the time series model:

$$(I - sar1B^{12})(I - B^{12})(I - B)y_t = (I + sma1B^{12})(I + ma1B + ma2B^2)w_t$$

where $\{w_t\}$ are the random errors.

Plugging in the numbers:

$$(I - 0.1018B^{12})(I - B^{12})(I - B)y_t = (I - 0.8434B^{12})(I - 0.3984B - 0.1632B^2)w_t$$

Or

$$(I - B^{12})(I - B)y_t - 0.1018B^{12}(I - B^{12})(I - B)y_t = (I - 0.3984B - 0.1632B^2)w_t - 0.8434B^{12}(I - 0.3984B - 0.1632B^2)w_t$$

$$(I - B^{12})(y_t - y_{t-1}) - 0.1018B^{12}(I - B^{12})(y_t - y_{t-1}) = (w_t - 0.3984w_{t-1} - 0.1632w_{t-2}) - 0.8434(w_{t-12} - 0.3984w_{t-13} - 0.1632w_{t-14})$$

$$(y_t - y_{t-1}) - (y_{t-12} - y_{t-13}) - 0.1018B^{12}((y_t - y_{t-1}) - (y_{t-12} - y_{t-13})) = (w_t - 0.3984w_{t-1} - 0.1632w_{t-2}) - 0.8434(w_{t-12} - 0.3984w_{t-13} - 0.1632w_{t-14})$$

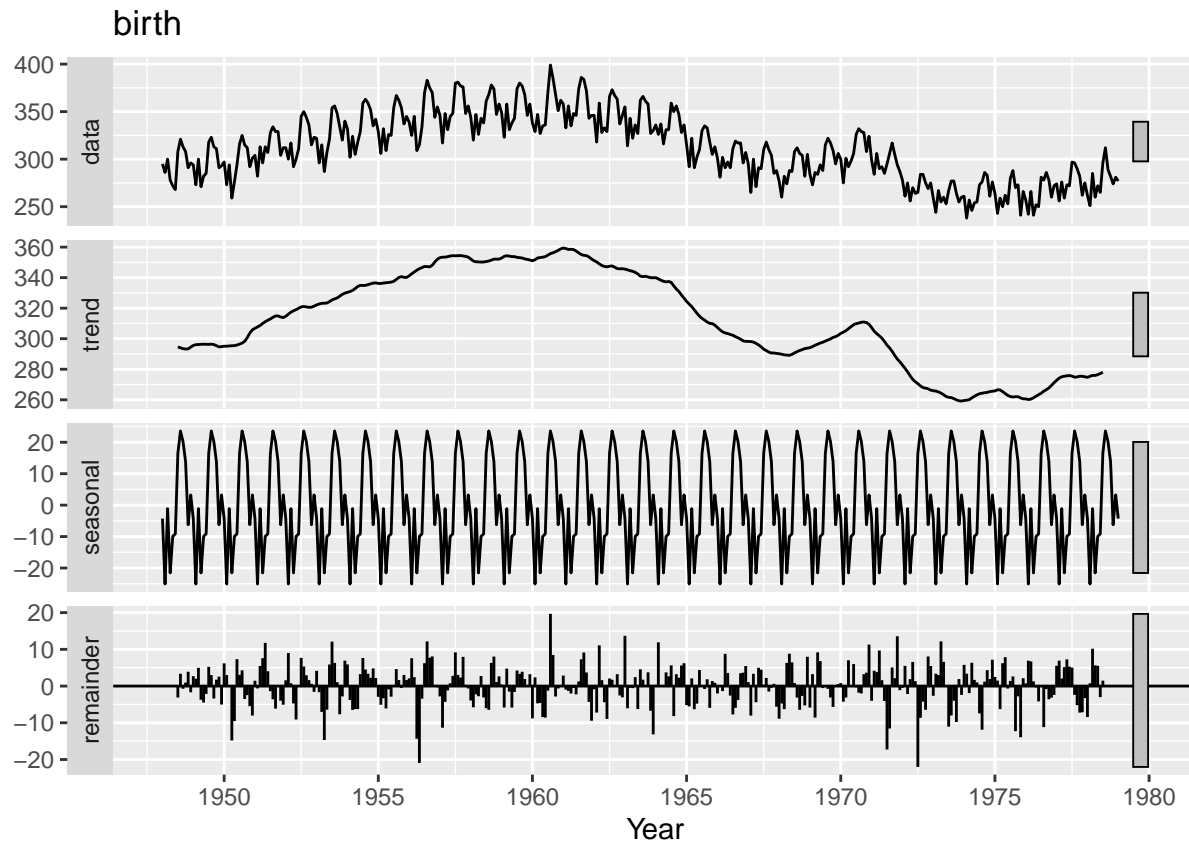
$$(y_t - y_{t-1}) - (y_{t-12} - y_{t-13}) - 0.1018((y_{t-12} - y_{t-13}) - (y_{t-24} - y_{t-25})) = w_t - 0.3984w_{t-1} - 0.1632w_{t-2} - 0.8434w_{t-12} + 0.8434*0.3984w_{t-13} - 0.8434*0.1632w_{t-14} + 0.8434*0.1632*0.3984w_{t-15} - 0.8434*0.1632*0.3984*0.1632w_{t-16}$$

$$(y_t = y_{t-1} + (y_{t-12} - y_{t-13}) + 0.1018((y_{t-12} - y_{t-13}) - (y_{t-24} - y_{t-25})) + w_t - 0.3984w_{t-1} - 0.1632w_{t-2} - 0.8434w_{t-12} + 0.8434*0.3984w_{t-13} - 0.8434*0.1632w_{t-14} + 0.8434*0.1632*0.3984w_{t-15} - 0.8434*0.1632*0.3984*0.1632w_{t-16})$$

We see that this is quite a complicated structure that captures a yearly cycle plus a 2 year cycle. That seems to account for the curved patterns we observed in the plot of the values.

Let's see the decomposition of the cycles:

```
birth %>% decompose() %>%
  autoplot() + xlab("Year") +
  ggtitle("birth")
```



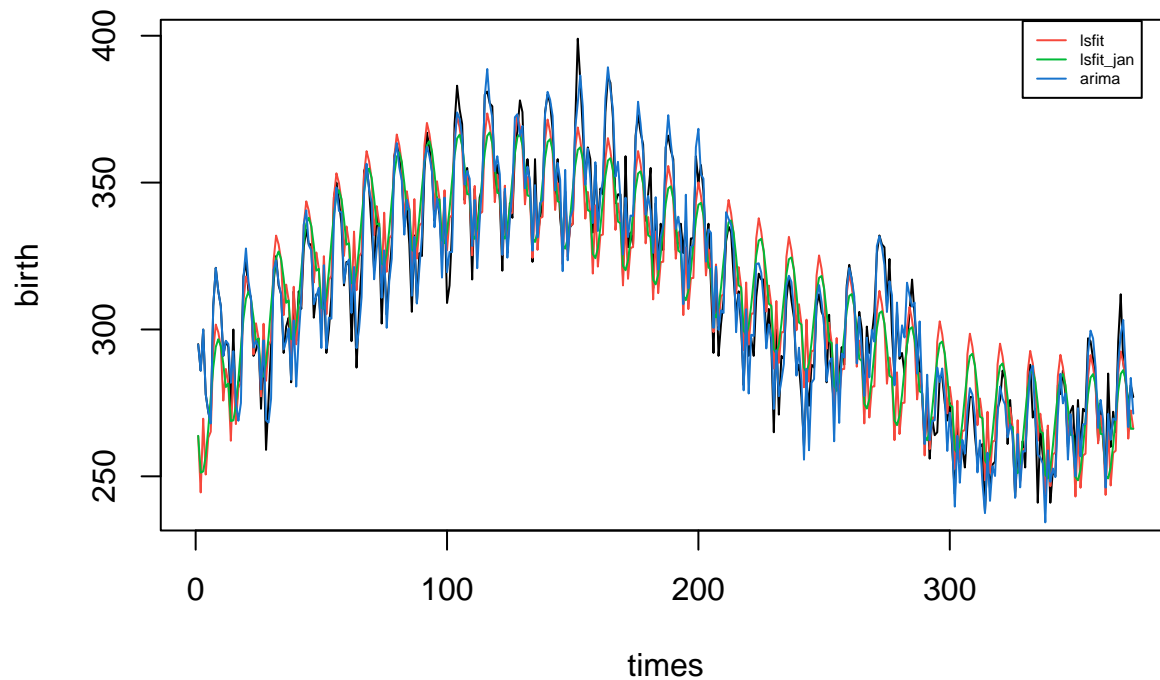
```
#dbirth<-decompose(birth)
#plot(dbirth)
```

We see the trend (2nd plot), the seasonal component (3rd plot) and the random part (4th plot). The 1st plot is the original series.

- Trend: the trend-cycle component T_t is a m -moving average, where m is the cycle. In our case of monthly data, $m = 12$. (moving average = average of previous m -observations)
- Detrended series: Calculate the detrended series as $y_t - T_t$
- Seasonal component: the seasonal component for each season is the average of the detrended values for that season. This gives a series called S_t .
- Error: The remainder component is calculated by subtracting the estimated seasonal and trend-cycle components: $R_t = Y_t - T_t - S_t$.

Let's plot the fitted values of the 3 models:

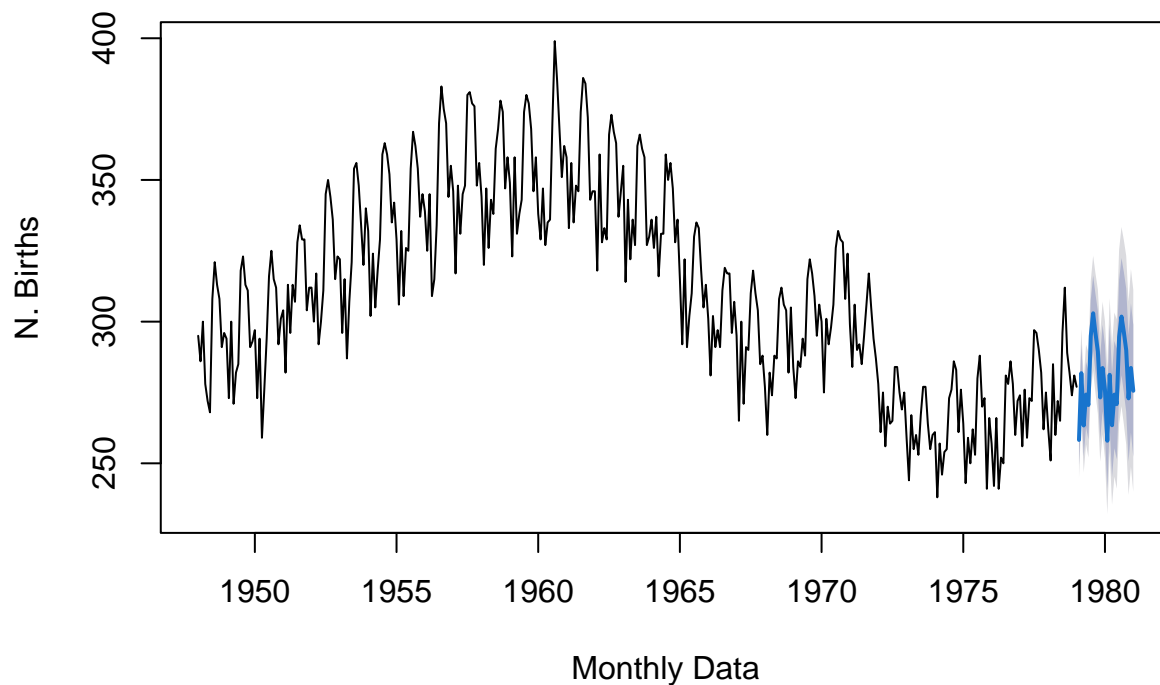
```
plot(times,birth,type="l") #plot on original scale
#lines(times,birth) #add lines to existing plot
lines(times,lsfit$fitted.values,col=2) #undo log for fitted model
lines(times,lsfit_jan$fitted,col=3) #undo log for fitted model
lines(times,birthmod$fitted,col=4)
legend(329,405,c("lsfit","lsfit_jan","arima"),col=c(2,3,4),lty=1,cex=.5)
```



Now let's use our arima model to do forecasts:

```
plot(forecast(birthmod, 24), xlab = "Monthly Data",
     ylab = "N. Births",
     main = "Number of Birth per month", col.main = "darkgreen")
```

Number of Birth per month



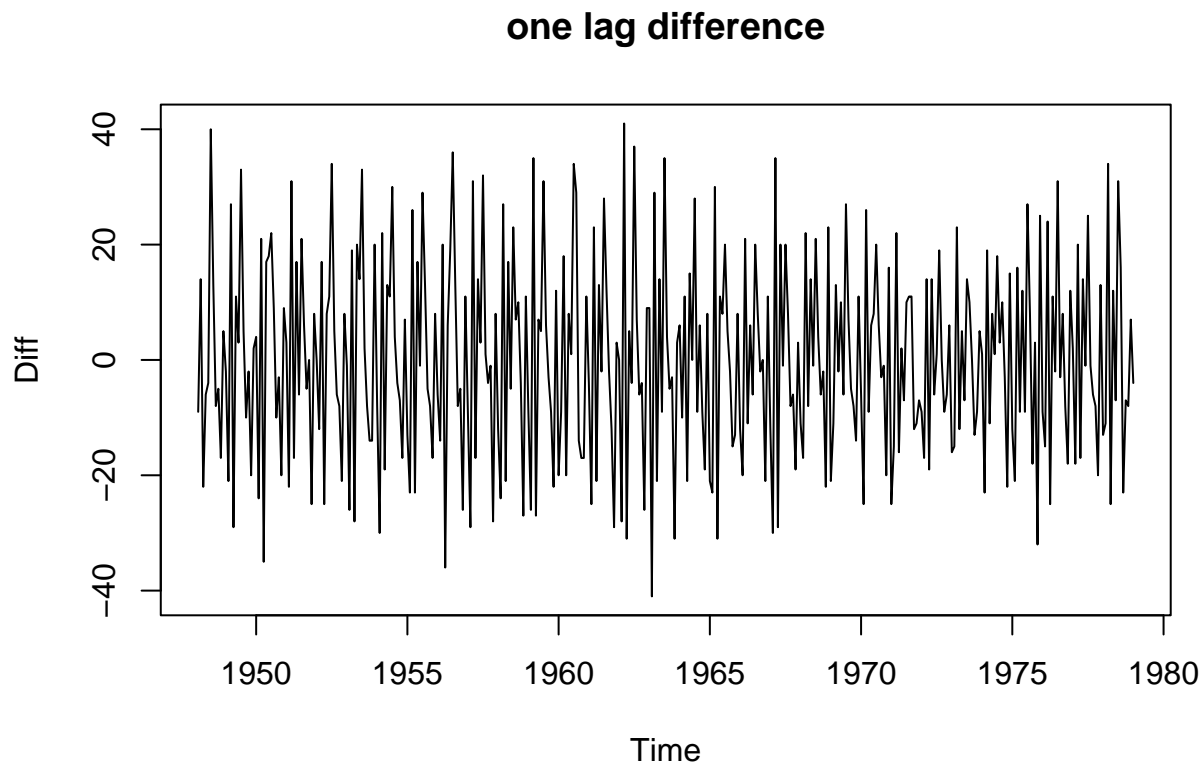
Let's check that the errors do not have any auto-correlation:

```
acf(birthmod$residuals)
```

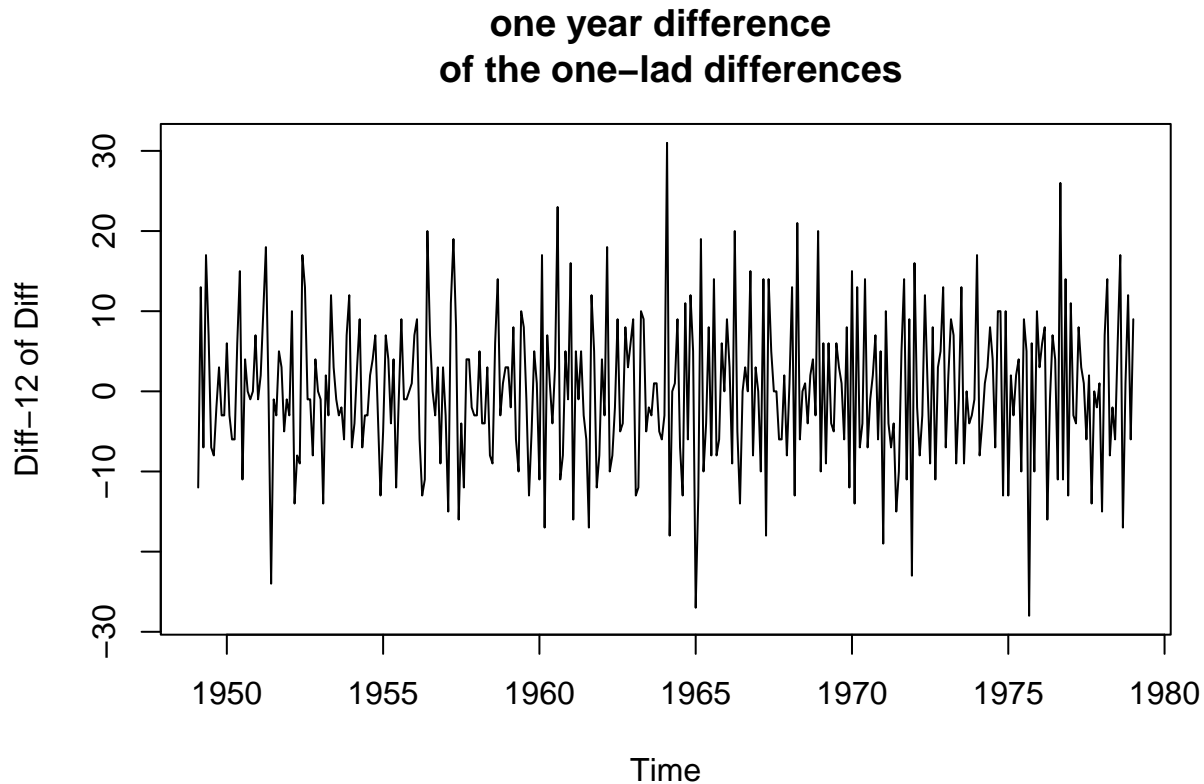


Just for the heck of it, let's look at the differences involved in the arima model:

```
plot(diff(birth,1),main="one lag difference",ylab="Diff")
```



```
plot(diff( diff(birth,1) ,12),main="one year difference \n of the one-lad differences",ylab="Diff-12 of
```



Time Series Assignment

We will fit a model to the log of the Australian wine sales.

- Plot wine and $\log(\text{wine})$.
- Plot the auto correlation and partial auto correlation functions for $\log(\text{wine})$.
- Just as we did for “birth”, fit a model allowing for a term for each month and time.
- Just as we did for “birth”, fit a model using sin and cos to model seasonality and time.
- compute the aic, bic and adjusted r^2 corresponding to both models.
- Use `auto.arima()` to obtain the arima model.
- compare the aic and bic of the arima model to the previous 2 models.
- Write down the equation corresponding to the arima model.
- Plot the decomposition of the series.
- Plot the fitted values of all 3 models over the values of wine. Remember that your models were for $\log(\text{wine})$ but you are plotting wine, so you need to adjust your fitted values.
- plot the predicted values for the next 12 months.
- `auto.arima` does not work with covariates. But we can use the structure it developed to add one or several covariates. Consider the models:
 - `Arima(y, order = c(1,1,1), xreg = X)` and

- Arima(y, order = c(1,0,1), xreg = X) where X is the data frame with times and the monthly dummy variables

```
wine=c(  
.46400E+03,  
.67500E+03,  
.70300E+03,  
.88700E+03,  
.11390E+04,  
.10770E+04,  
.13180E+04,  
.12600E+04,  
.11200E+04,  
.96300E+03,  
.99600E+03,  
.96000E+03,  
.53000E+03,  
.88300E+03,  
.89400E+03,  
.10450E+04,  
.11990E+04,  
.12870E+04,  
.15650E+04,  
.15770E+04,  
.10760E+04,  
.91800E+03,  
.10080E+04,  
.10630E+04,  
.54400E+03,  
.63500E+03,  
.80400E+03,  
.98000E+03,  
.10180E+04,  
.10640E+04,  
.14040E+04,  
.12860E+04,  
.11040E+04,  
.99900E+03,  
.99600E+03,  
.10150E+04,  
.61500E+03,  
.72200E+03,  
.83200E+03,  
.97700E+03,  
.12700E+04,  
.14370E+04,  
.15200E+04,  
.17080E+04,  
.11510E+04,  
.93400E+03,  
.11590E+04,  
.12090E+04,  
.69900E+03,  
.83000E+03,
```

.99600E+03,
.11240E+04,
.14580E+04,
.12700E+04,
.17530E+04,
.22580E+04,
.12080E+04,
.12410E+04,
.12650E+04,
.18280E+04,
.80900E+03,
.99700E+03,
.11640E+04,
.12050E+04,
.15380E+04,
.15130E+04,
.13780E+04,
.20830E+04,
.13570E+04,
.15360E+04,
.15260E+04,
.13760E+04,
.77900E+03,
.10050E+04,
.11930E+04,
.15220E+04,
.15390E+04,
.15460E+04,
.21160E+04,
.23260E+04,
.15960E+04,
.13560E+04,
.15530E+04,
.16130E+04,
.81400E+03,
.11500E+04,
.12250E+04,
.16910E+04,
.17590E+04,
.17540E+04,
.21000E+04,
.20620E+04,
.20120E+04,
.18970E+04,
.19640E+04,
.21860E+04,
.96600E+03,
.15490E+04,
.15380E+04,
.16120E+04,
.20780E+04,
.21370E+04,
.29070E+04,


```
.22490E+04,
.18830E+04,
.17390E+04,
.18280E+04,
.18680E+04,
.11380E+04,
.14300E+04,
.18090E+04,
.17630E+04,
.22000E+04,
.20670E+04,
.25030E+04,
.21410E+04,
.21030E+04,
.19720E+04,
.21810E+04,
.23440E+04,
.97000E+03,
.11990E+04,
.17180E+04,
.16830E+04,
.20250E+04,
.20510E+04,
.24390E+04,
.23530E+04,
.22300E+04,
.18520E+04,
.21470E+04,
.22860E+04,
.10070E+04,
.16650E+04,
.16420E+04,
.15250E+04,
.18380E+04,
.18920E+04,
.29200E+04,
.25720E+04,
.26170E+04,
.20470E+04)
```

#####. solutions

Plot wine and log(wine). Warning

```
wine<-ts(wine,frequency = 12)
library(ggplot2)
library(ggfortify)
library(dplyr)
```

```
y=log(wine)
times=1:142
```

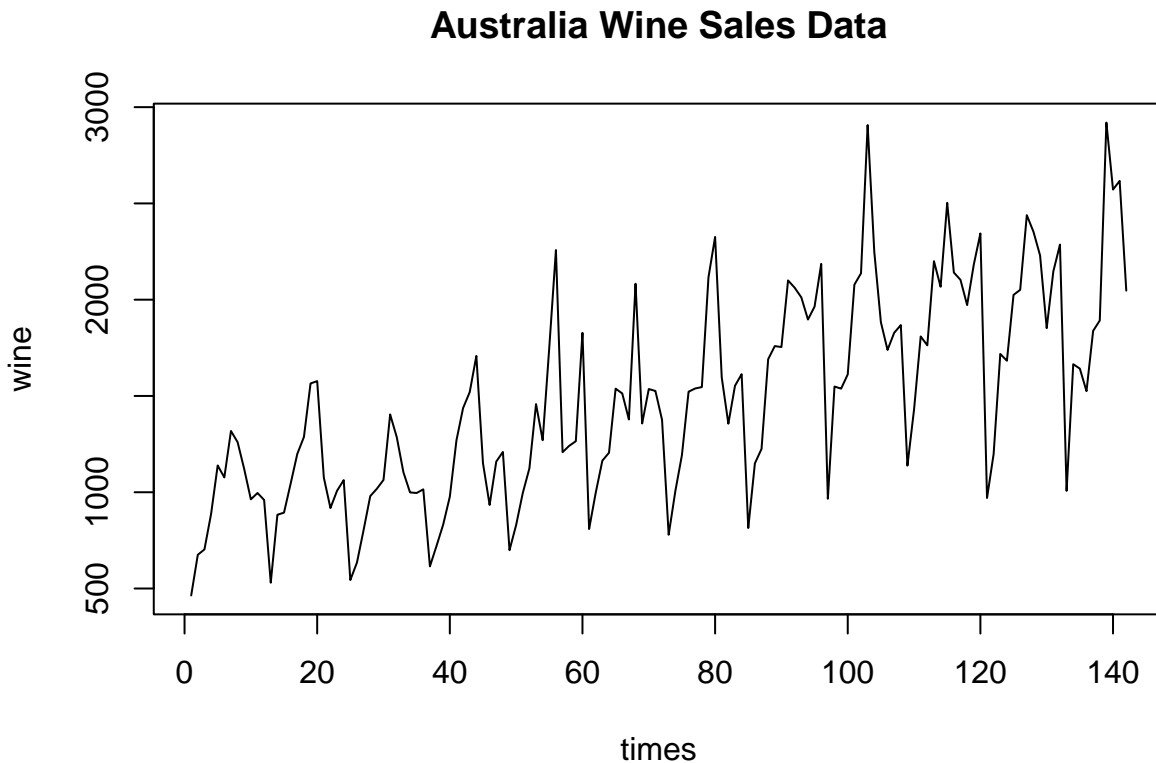
```
Jan=rep(c(1,0,0,0,0,0,0,0,0,0,0,0),12)[1:142]
Feb=rep(c(0,1,0,0,0,0,0,0,0,0,0,0),12)[1:142]
Mar=rep(c(0,0,1,0,0,0,0,0,0,0,0,0),12)[1:142]
```

```

Apr=rep(c(0,0,0,1,0,0,0,0,0,0,0,0),12)[1:142]
May=rep(c(0,0,0,0,1,0,0,0,0,0,0,0),12)[1:142]
Jun=rep(c(0,0,0,0,0,1,0,0,0,0,0,0),12)[1:142]
Jul=rep(c(0,0,0,0,0,0,1,0,0,0,0,0),12)[1:142]
Aug=rep(c(0,0,0,0,0,0,0,1,0,0,0,0),12)[1:142]
Sep=rep(c(0,0,0,0,0,0,0,0,1,0,0,0),12)[1:142]
Oct=rep(c(0,0,0,0,0,0,0,0,0,1,0,0),12)[1:142]
Nov=rep(c(0,0,0,0,0,0,0,0,0,0,1,0),12)[1:142]
Dec=rep(c(0,0,0,0,0,0,0,0,0,0,0,1),12)[1:142]
sint=sin(2*pi*times/12)
cost=cos(2*pi*times/12)
X=data.frame(times=times,Feb=Feb,Mar=Mar,Apr=Apr,May=May,Jun=Jun,Jul=Jul,Aug=Aug,
              Sep=Sep,Oct=Oct,Nov=Nov,Dec=Dec) #sin and cos and constant for Jan;
X_jan=data.frame(times=times,sint=sint,cost=cost,Jan=Jan) #sin and cos and constant for Jan;

plot(times, wine, type = "l", main = "Australia Wine Sales Data")

```

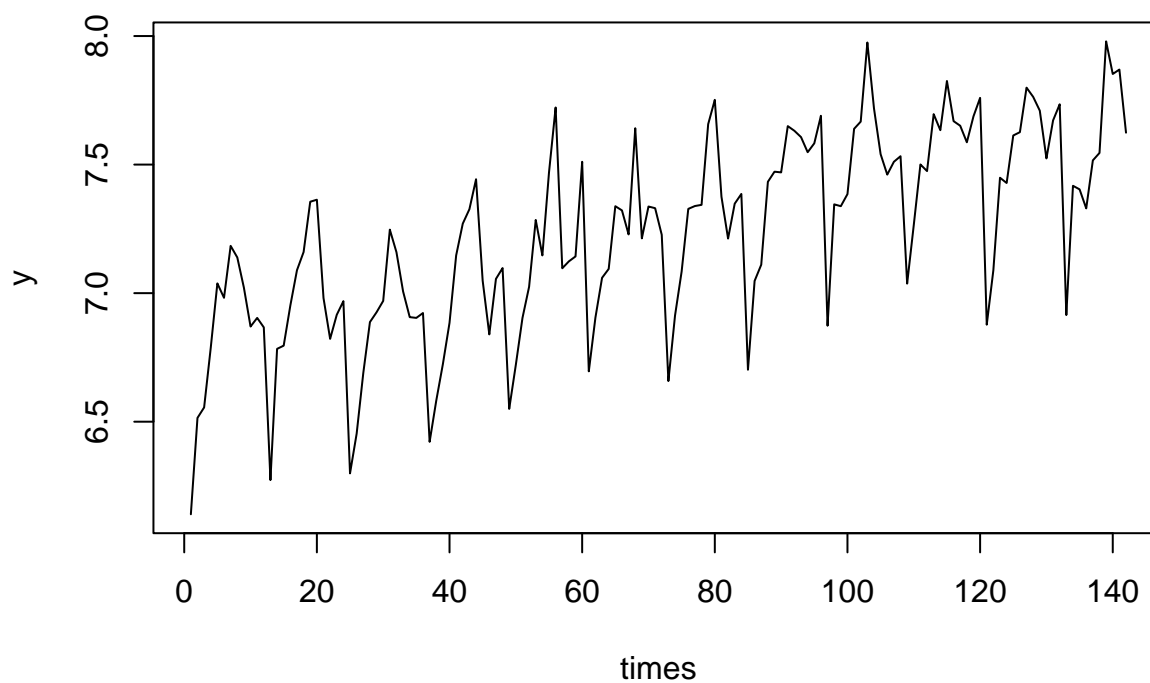


```

plot(times, y, type = "l", main = "Australia Wine Sales Data")

```

Australia Wine Sales Data



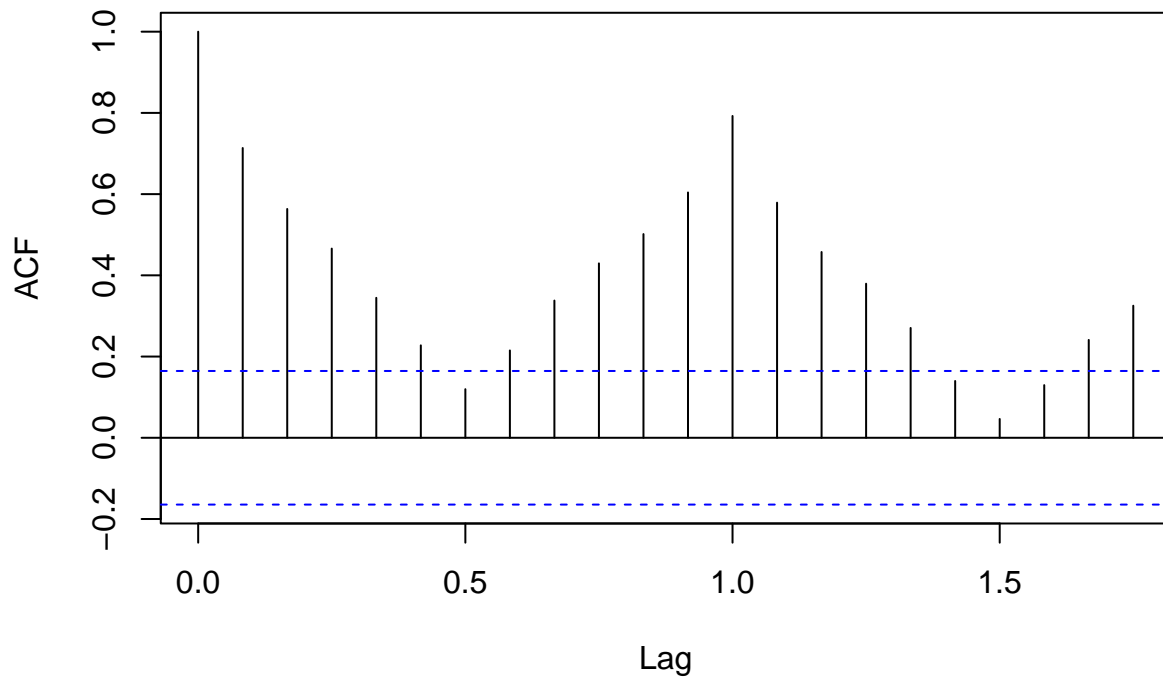
- Plot the auto correlation and partial auto correlation functions for $\log(\text{wine})$ correlation functions

```
library(forecast)
```

```
# ACF and PACF plots for log transformed data
```

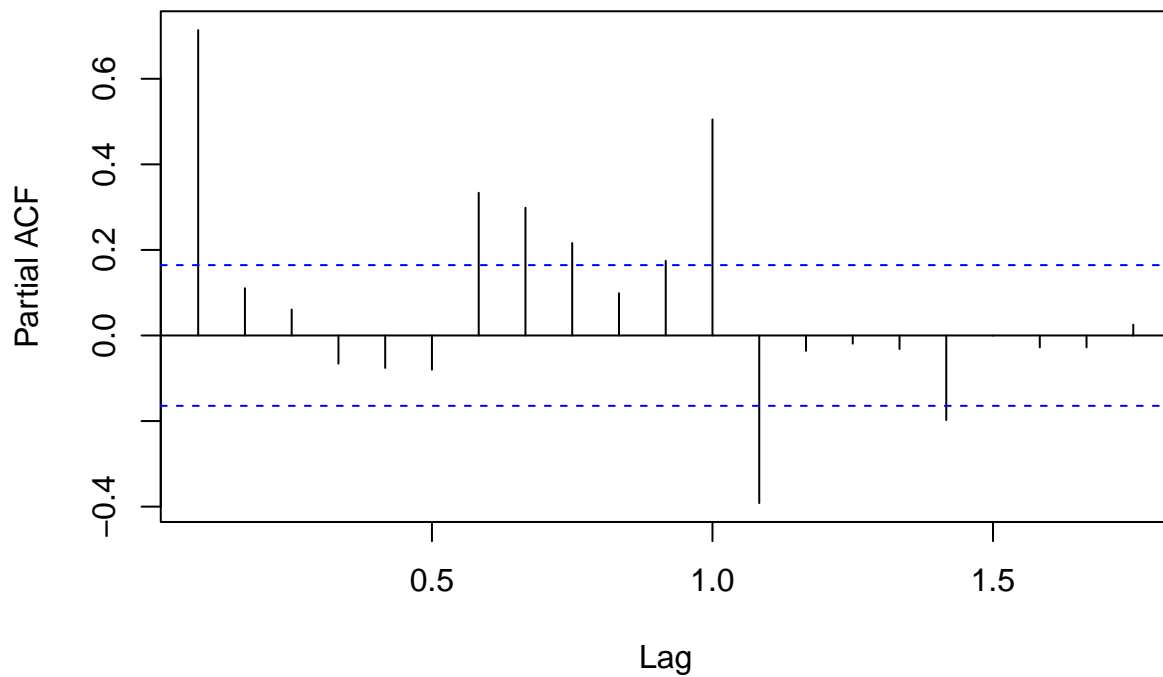
```
acf(y, main = "Auto Correlation Function plot of Log(Wine)")
```

Auto Correlation Function plot of Log(Wine)

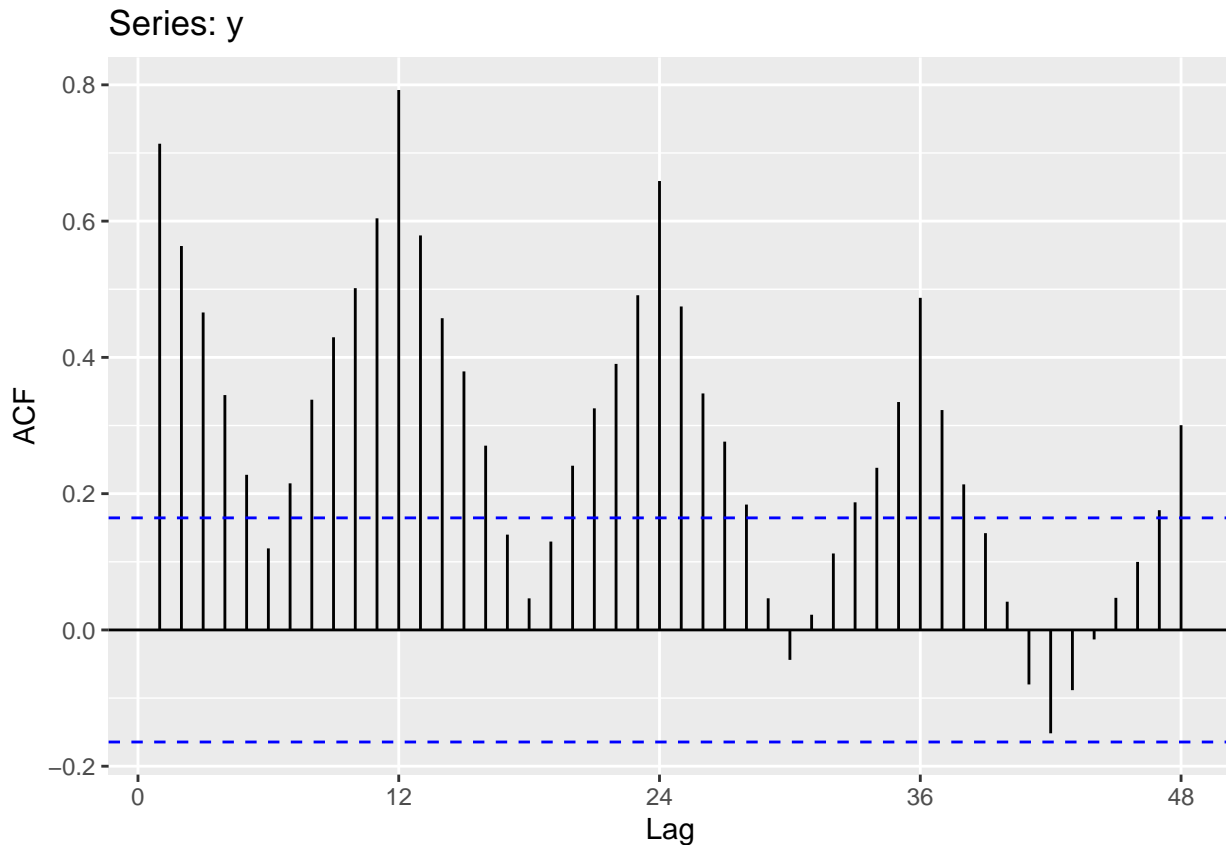


```
pacf(y, main = "Partial Auto Correlation Function plot of Log(Wine)")
```

Partial Auto Correlation Function plot of Log(Wine)



```
ggAcf(y, lag = 48)
```



- Just as we did for “birth”, fit a model allowing for a term for each month and time by using the data frame X.

```
# Assuming y is the name of the dataset column containing wine sales
n <- length(y)

# Creating the times variable
times <- 1:n

# Generating a sequence of months to match the length of the data
months <- rep(c("01Jan", "02Feb", "03Mar", "04Apr", "05May", "06Jun",
               "07Jul", "08Aug", "09Sep", "10Oct", "11Nov", "12Dec"),
             each = ceiling(n / 12))[1:n]

# Assuming X is another variable you want to include in the model
X_df <- data.frame(months, X)

# Creating a data frame with month and times variables
X_wine <- data.frame(times = times, month = months)

# Fitting a model with terms for each month and time
# Assuming y and X_wine are part of the same observation set
lsfit_wine <- lm(y ~ poly(times, 3) + month, data = X_wine)
summary(lsfit_wine)

##
## Call:
## lm(formula = y ~ poly(times, 3) + month, data = X_wine)
```

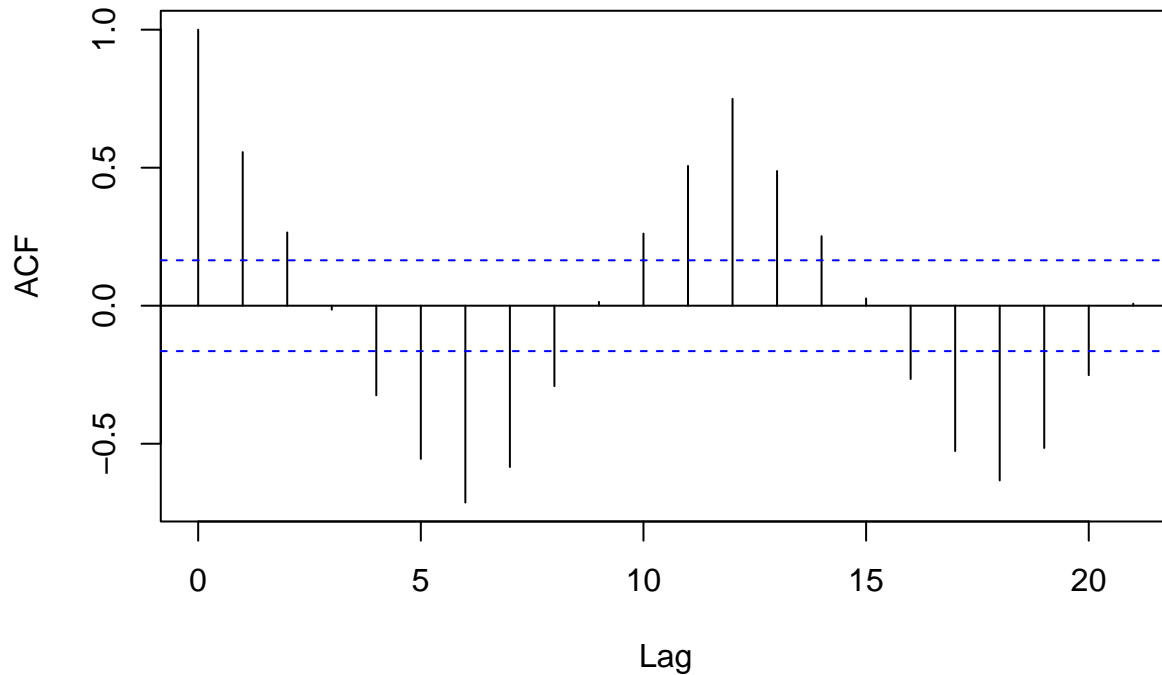
```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.42543 -0.16557  0.00536  0.14626  0.50724
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    10.1533     0.3899  26.037 < 2e-16 ***
## poly(times, 3)1  25.2902     2.8255   8.951 3.65e-15 ***
## poly(times, 3)2   0.6795     1.1926   0.570 0.56985
## poly(times, 3)3   0.4122     0.7954   0.518 0.60522
## month02Feb      -0.5036     0.1624  -3.101 0.00238 **
## month03Mar      -1.2010     0.2474  -4.854 3.49e-06 ***
## month04Apr      -1.6736     0.3139  -5.332 4.30e-07 ***
## month05May      -2.1087     0.3713  -5.680 8.73e-08 ***
## month06Jun      -2.6422     0.4277  -6.178 8.15e-09 ***
## month07Jul      -3.1581     0.4857  -6.502 1.64e-09 ***
## month08Aug      -3.6435     0.5426  -6.715 5.64e-10 ***
## month09Sep      -4.1924     0.5936  -7.063 9.52e-11 ***
## month10Oct      -4.7895     0.6354  -7.538 7.94e-12 ***
## month11Nov      -5.5298     0.6707  -8.244 1.78e-13 ***
## month12Dec      -6.1842     0.7101  -8.709 1.40e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2169 on 127 degrees of freedom
## Multiple R-squared:  0.7088, Adjusted R-squared:  0.6767
## F-statistic: 22.08 on 14 and 127 DF, p-value: < 2.2e-16
```

The to see the auto correlation of the residuals of the model (acf)

```
# Calculate residuals from the model
residuals_lsfit_wine <- residuals(lsfit_wine)

# Plot the autocorrelation function of the residuals
acf(residuals_lsfit_wine, main = "ACF of Residuals from lsfit_wine")
```

ACF of Residuals from lsfit_wine



- Just as we did for “birth”, fit a model using sin and cos to model seasonality and time using the data frame X_jan

```
n <- length(y)

# Creating sine and cosine terms for seasonality
sint <- sin(2 * pi * times / 12) # Frequency for yearly seasonality
cost <- cos(2 * pi * times / 12)

# Creating the Jan indicator (assuming January as the starting point)
Jan <- rep(c(1,0,0,0,0,0,0,0,0,0,0,0),ceiling(n / 12))[1:n]

# Fitting a model with sine, cosine and Jan indicator
lsfit_wine_jan <- lm(y ~ poly(times, 3) + sint + cost + Jan, data = X_jan)
summary(lsfit_wine_jan)
```

```
##
## Call:
## lm(formula = y ~ poly(times, 3) + sint + cost + Jan, data = X_jan)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.31228 -0.09852 -0.00531  0.09645  0.46227
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    7.26388    0.01226  592.512 < 2e-16 ***
## poly(times, 3)1  3.08221    0.13837   22.275 < 2e-16 ***
## poly(times, 3)2 -0.20972    0.13825   -1.517  0.131622
## poly(times, 3)3 -0.51123    0.13879   -3.683  0.000332 ***
```

```
## sint          -0.17542      0.01682 -10.428 < 2e-16 ***
## cost          -0.13436      0.01789  -7.508 7.31e-12 ***
## Jan           -0.41833      0.04629  -9.037 1.49e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1381 on 135 degrees of freedom
## Multiple R-squared:  0.8745, Adjusted R-squared:  0.8689
## F-statistic: 156.8 on 6 and 135 DF,  p-value: < 2.2e-16
```

- compute the aic, bic and adjusted r2 corresponding to both models.

```
aic <- round(c(AIC(lsfit_wine), AIC(lsfit_wine_jan)),2)
bic <- round(c(BIC(lsfit_wine), BIC(lsfit_wine_jan)),2)
adjr2 <- round(c(summary(lsfit_wine)$ad,summary(lsfit_wine_jan)$ad))
rbind(c("lsfit_wine", "lsfit_wine_jan"), aic,bic,adjr2)
```

```
##      [,1]      [,2]
##      "lsfit_wine" "lsfit_wine_jan"
## aic  "-14.93"    "-150.44"
## bic  "32.37"     "-126.79"
## adjr2 "1"        "1"
```

- Use auto.arima() to obtain the arima model.

```
winemod <- auto.arima(y)
summary(winemod)
```

```
## Series: y
## ARIMA(1,0,1)(0,1,1)[12] with drift
##
## Coefficients:
##          ar1          ma1          sma1      drift
##          0.8930   -0.6841   -0.7372   0.0062
## s.e.    0.0785    0.1249    0.0951   0.0008
##
## sigma^2 = 0.0125:  log likelihood = 97.74
## AIC=-185.48   AICc=-185   BIC=-171.14
##
## Training set error measures:
##              ME          RMSE          MAE          MPE          MAPE          MASE
## Training set -7.957843e-05 0.1053301 0.07939004 -0.01674237 1.090572 0.6036391
##              ACF1
## Training set 0.02659543
```

- compare the aic and bic of the arima model to the previous 2 models.

```
aic <- round(c(AIC(lsfit_wine), AIC(lsfit_wine_jan), AIC(winemod)),2)
bic <- round(c(BIC(lsfit_wine), BIC(lsfit_wine_jan), BIC(winemod)),2)
rbind(c("lsfit_wine", "lsfit_wine_jan","winemod"), aic,bic)
```

```
##      [,1]      [,2]      [,3]
##      "lsfit_wine" "lsfit_wine_jan" "winemod"
## aic  "-14.93"    "-150.44"    "-185.48"
## bic  "32.37"     "-126.79"    "-171.14"
```

- Write down the equation corresponding to the arima model.

Ans) The ARIMA (1,1,1) model can be written in the following form:

$(I - ar1B)(I - B)yt = (I + ma1B)wt$
 Equation corresponding to the above ARIMA model is
 $(I - 0.8930B)(I - B)yt = (I - 0.6841B)wt$

- Plot the decomposition of the series.

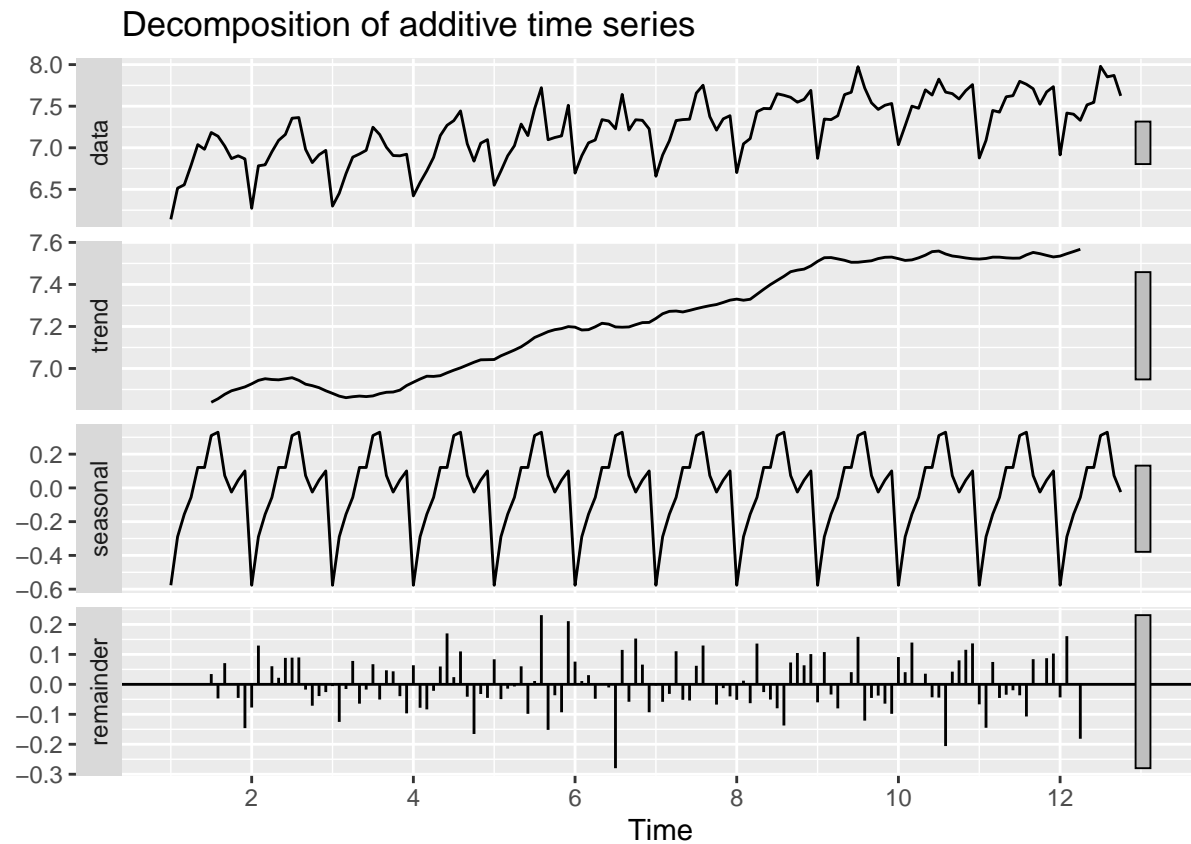
```

wine_ts <- ts(y, frequency = 12)

# Try decomposing the time series
decomposed <- decompose(wine_ts)

# Plot the decomposition if successful
autoplot(decomposed)

```



```

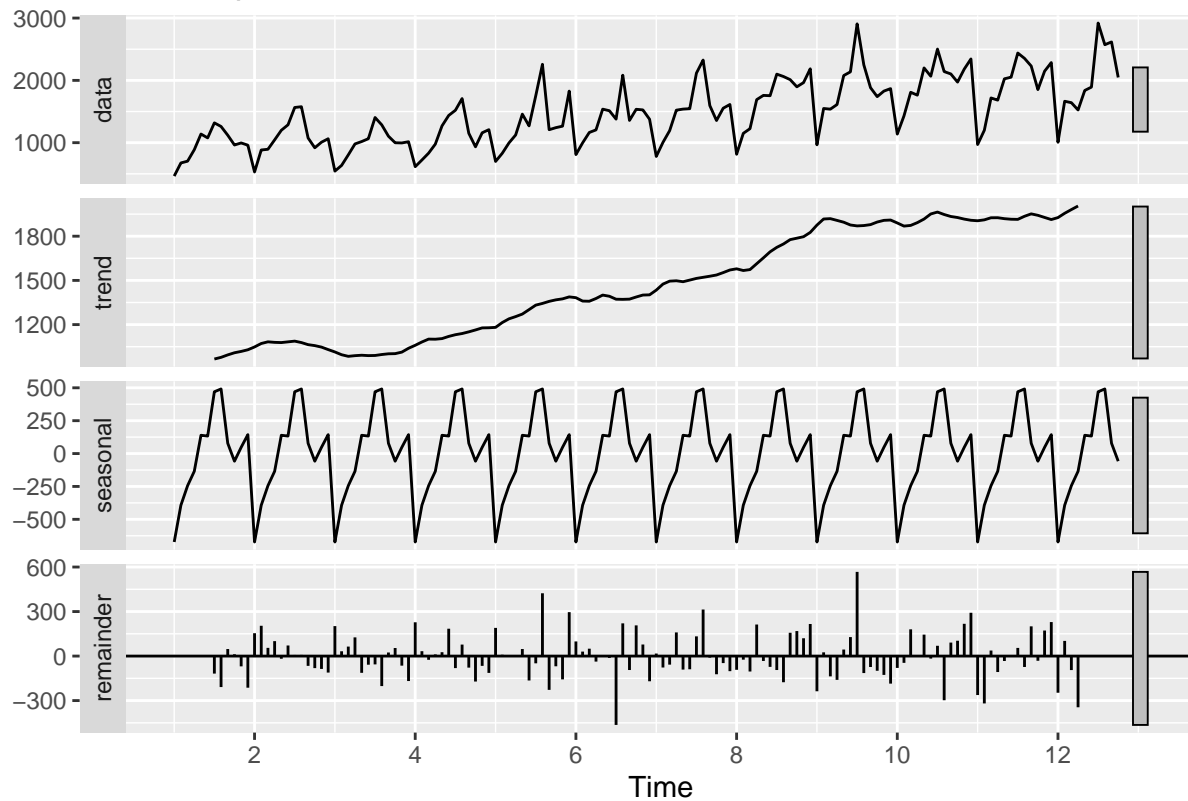
wine_ts <- ts(wine, frequency = 12)

# Try decomposing the time series
decomp_wine <- decompose(wine_ts)

# Plot the decomposition
autoplot(decomp_wine)

```

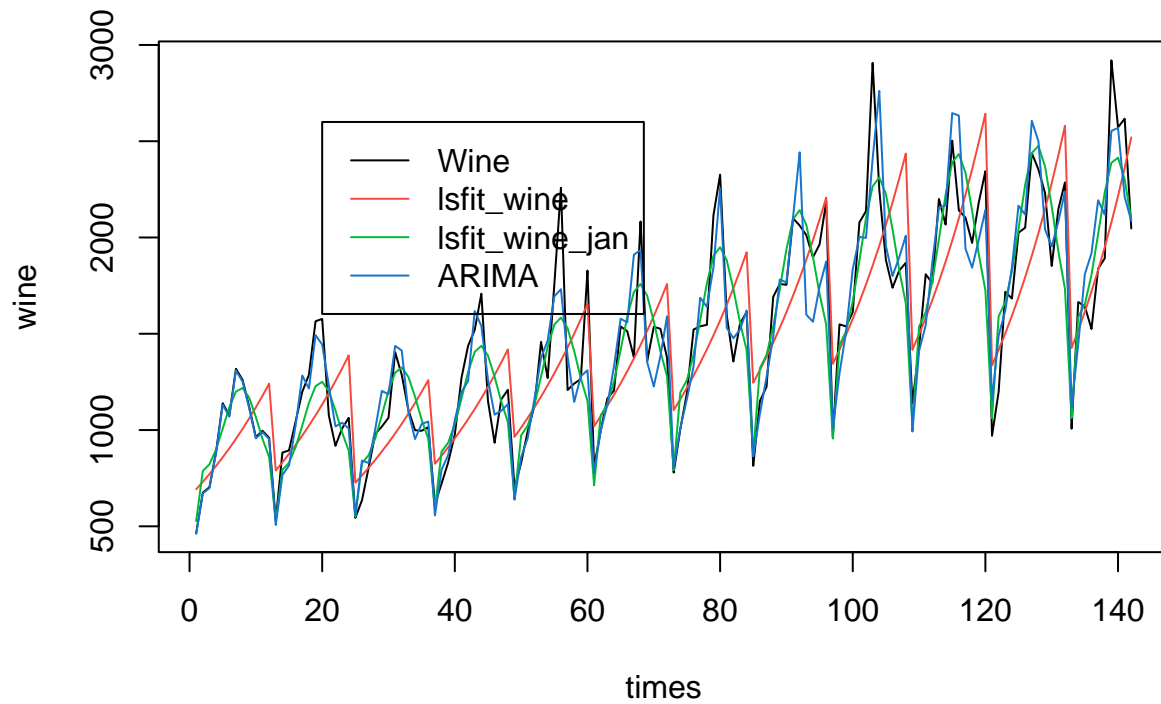
Decomposition of additive time series



- Plot the fitted values of all 3 models over the values of wine. Remember that your models were for $\log(\text{wine})$ but you are plotting wine, so you need to adjust your fitted values.

```
plot(times, wine, type = "l", main = "Australian Wine Sales")
lines(times, exp(lsfit_wine$fitted.values), col = 2)
lines(times, exp(lsfit_wine_jan$fitted), col = 3)
lines(times, exp(winemod$fitted), col = 4)
legend(20, 2600, c("Wine", "lsfit_wine", "lsfit_wine_jan", "ARIMA"), col = c(1, 2, 3, 4), lty = 1)
```

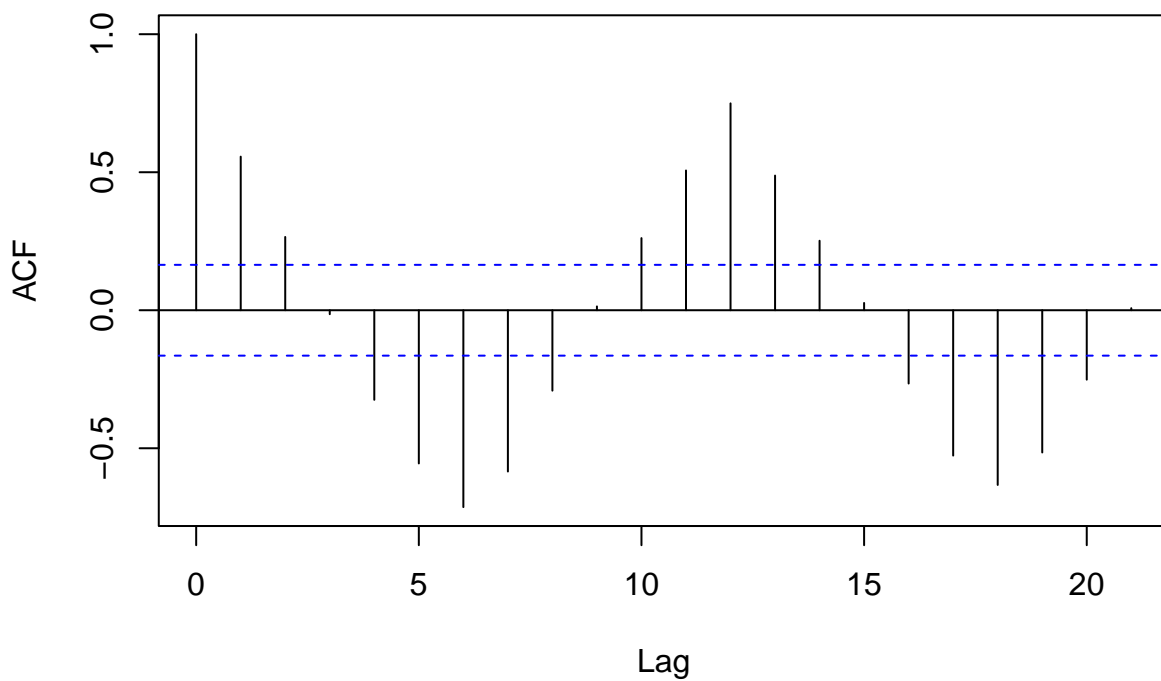
Australian Wine Sales



Check that the errors do not have any auto-correlation (acf)

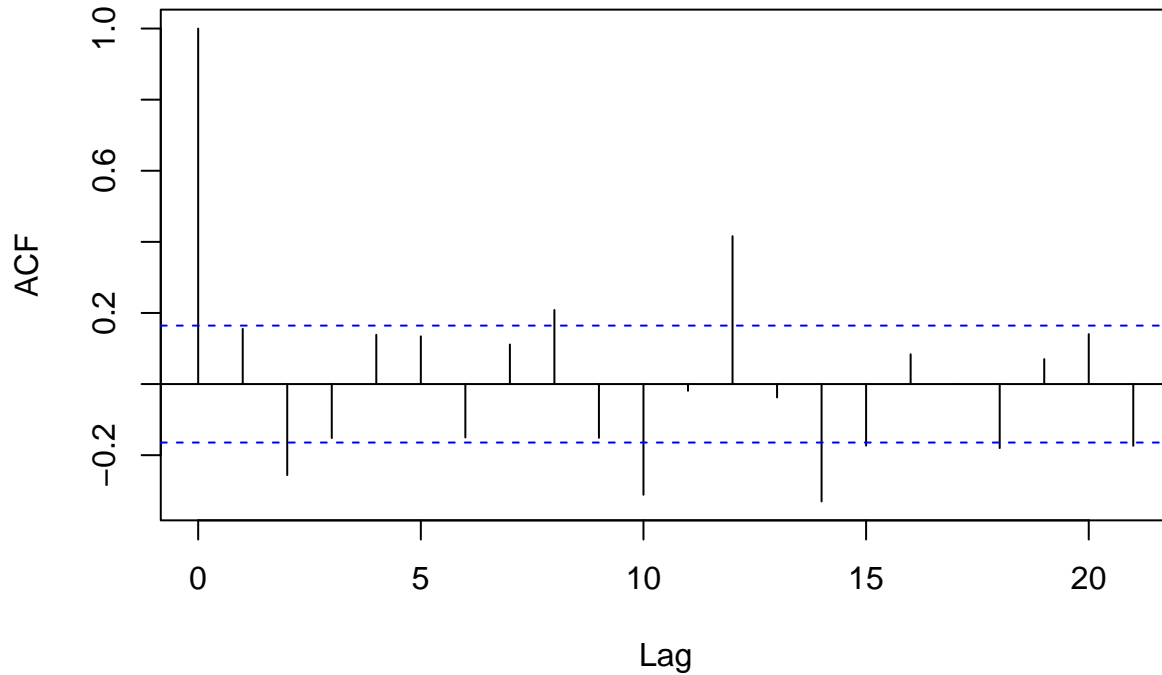
```
# Check autocorrelation in residuals of lsfit_wine
acf(residuals(lsfit_wine), main = "ACF of lsfit_wine Residuals")
```

ACF of lsfit_wine Residuals



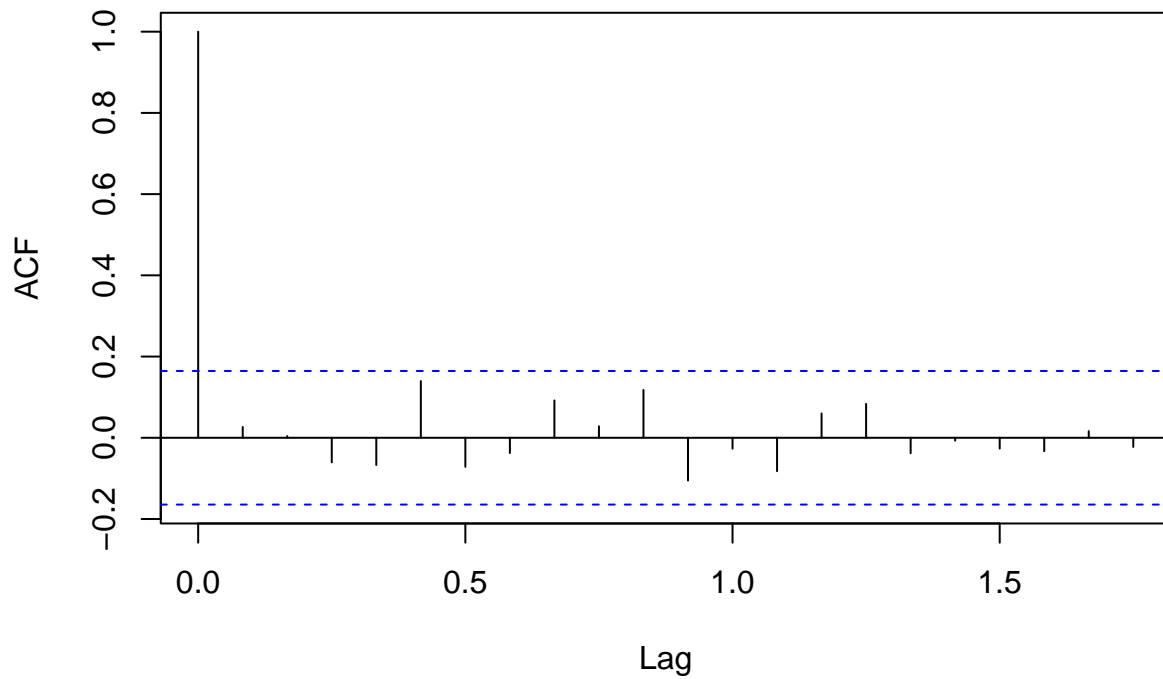
```
# Check autocorrelation in residuals of lsfit_wine_jan  
acf(residuals(lsfit_wine_jan), main = "ACF of lsfit_wine_jan Residuals")
```

ACF of lsfit_wine_jan Residuals



```
# Check autocorrelation in residuals of winemod (ARIMA model)  
acf(residuals(winemod), main = "ACF of winemod (ARIMA) Residuals")
```

ACF of winemod (ARIMA) Residuals

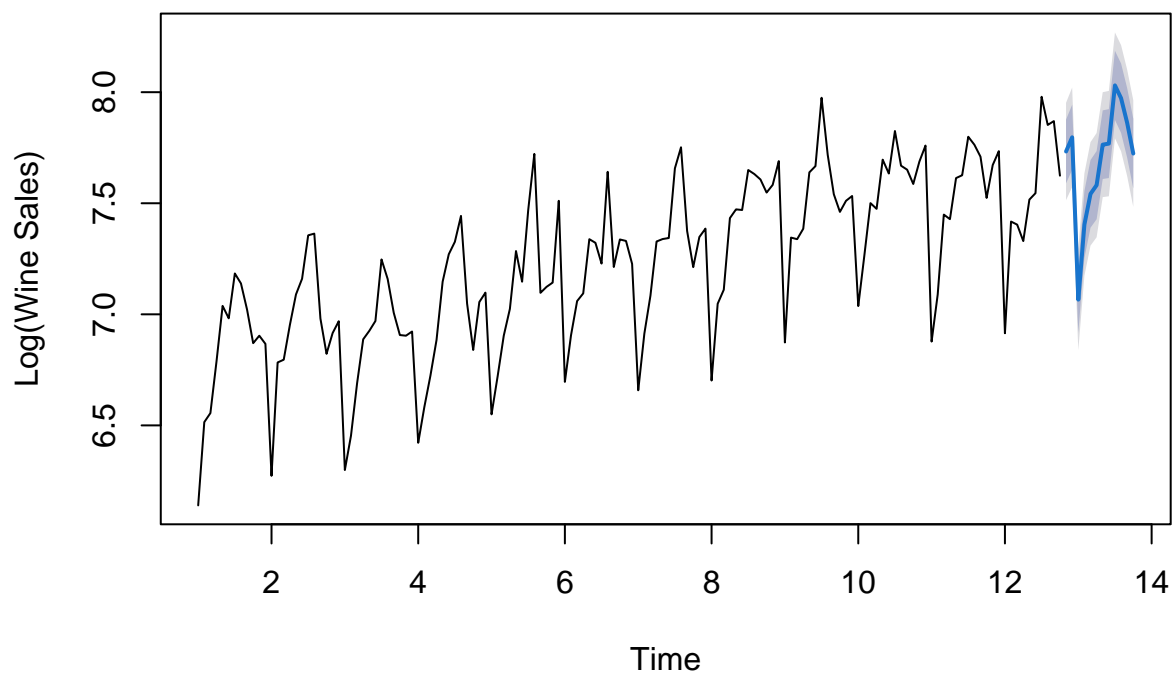


- plot the predicted values for the next 12 months.

Now let's use our arima model to do forecasts:

```
plot(forecast(winemod, h = 12), xlab = "Time", ylab = "Log(Wine Sales)", main = "Forecasted Values")
```

Forecasted Values



- auto.arima does not work with covariates. But we can use the structure it developed to add one or several covariates. Consider the models: – Arima(y, order = c(1,1,1), xreg = X) and – Arima(y, order = c(1,0,1), xreg = X) where X is the data frame with times and the monthly dummy variables

```
# Convert all columns of X to numeric if they are not already
X <- data.frame(lapply(X, function(col) as.numeric(as.character(col))))
X_matrix <- data.matrix(X)
tsmod2<-Arima(y, order = c(1,1,1), xreg = data.matrix(X))
tsmod3<-Arima(y, order = c(1,0,1), xreg = data.matrix(X))
summary(tsmod2)
```

```
## Series: y
## Regression with ARIMA(1,1,1) errors
##
## Coefficients:
##          ar1          ma1      times      Feb      Mar      Apr      May      Jun      Jul
##      0.0985   -0.8350   0.0058   0.2940   0.4195   0.5300   0.6984   0.6957   0.9033
## s.e.  0.1093   0.0668   0.0016   0.0362   0.0382   0.0386   0.0387   0.0389   0.0389
##          Aug      Sep      Oct      Nov      Dec
##      0.9107   0.6771   0.5659   0.6322   0.6841
## s.e.  0.0389   0.0389   0.0388   0.0391   0.0372
##
## sigma^2 = 0.01128: log likelihood = 122.98
## AIC=-215.97  AICc=-212.13  BIC=-171.74
##
## Training set error measures:
##              ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 0.001359156 0.100425 0.0790184 0.005170204 1.087867 0.6008134
##              ACF1
## Training set -0.00813603
```

```
summary(tsmod3)
```

```
## Series: y
## Regression with ARIMA(1,0,1) errors
##
## Coefficients:
##          ar1          ma1 intercept      times      Feb      Mar      Apr      May      Jun
##      0.8694   -0.6558      6.1964   0.0063   0.2936   0.4189   0.529   0.6969   0.6938
## s.e.  0.0891   0.1412      0.0474   0.0005   0.0363   0.0373   0.038   0.0385   0.0388
##          Jul      Aug      Sep      Oct      Nov      Dec
##      0.9009   0.9078   0.6736   0.5619   0.6308   0.6832
## s.e.  0.0390   0.0389   0.0387   0.0383   0.0382   0.0373
##
## sigma^2 = 0.01092: log likelihood = 127.02
## AIC=-222.05  AICc=-217.7  BIC=-174.75
##
## Training set error measures:
##              ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 0.0003300068 0.09881831 0.07768568 -0.01376188 1.070394 0.5906801
##              ACF1
## Training set 0.01067203
```

```
newdata<-cbind(times=143:154,Feb=c(0,0,0,1,rep(0,8)),
               Mar=c(0,0,0,0,1,rep(0,7)),
               Apr=c(0,0,0,0,0,1,rep(0,6)),
```

```

May=c(rep(0,6),1,rep(0,5)),
Jun=c(rep(0,7),1,rep(0,4)),
Jul=c(rep(0,8),1,rep(0,3)),
Aug=c(rep(0,9),1,rep(0,2)),
Sep=c(rep(0,10),1,0),Oct=c(rep(0,11),1),
Nov=c(1,rep(0,11)),Dec=c(0,1,rep(0,10)))

```

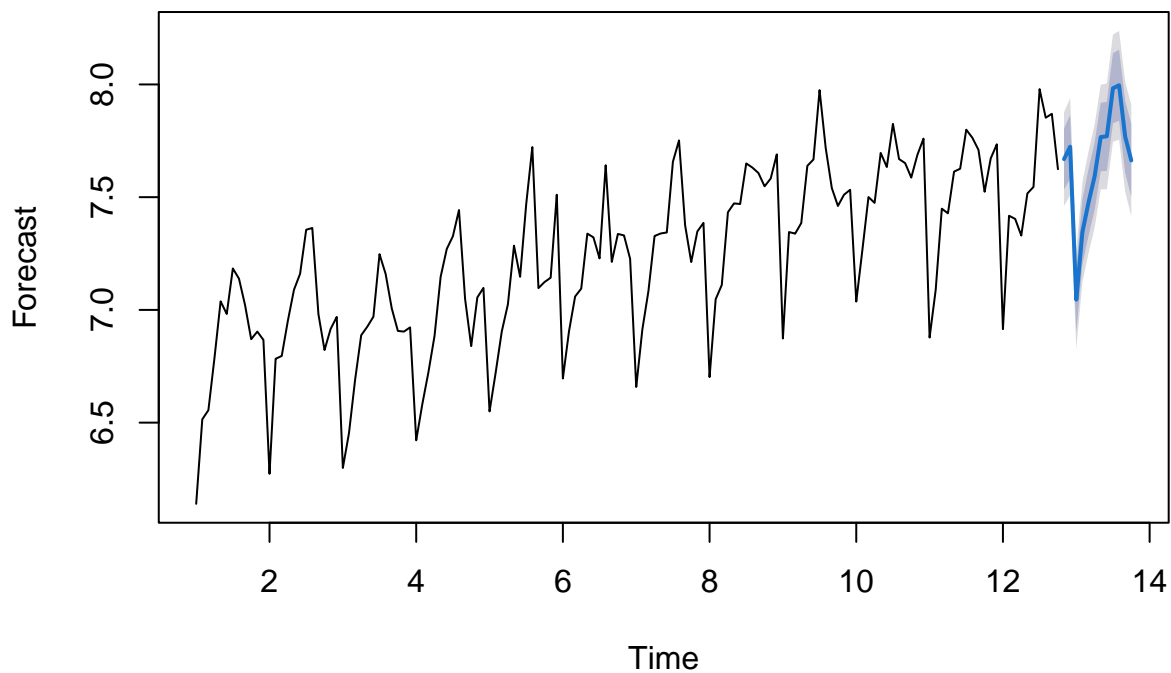
PLot forecast for the two models using newdata. Put titles in your plots.

```

forecast_tsmod2 <- forecast(tsmod2, xreg = newdata)
plot(forecast_tsmod2, main = "FORECAST from ARIMA (1,1,1) Model",
     xlab = "Time", ylab = "Forecast")

```

FORECAST from ARIMA (1,1,1) Model



```

forecast_tsmod3 <- forecast(tsmod3, xreg = newdata)
plot(forecast_tsmod3, main = "FORECAST from ARIMA (1,0,1) Model",
     xlab = "Time", ylab = "Forecast")

```

FORECAST from ARIMA (1,0,1) Model

