Time Series Lab and assignment

Vinay Vaida

2023-11-02

# TimeSeries Lab & Assignment

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

library(astsa)  
data(birth)

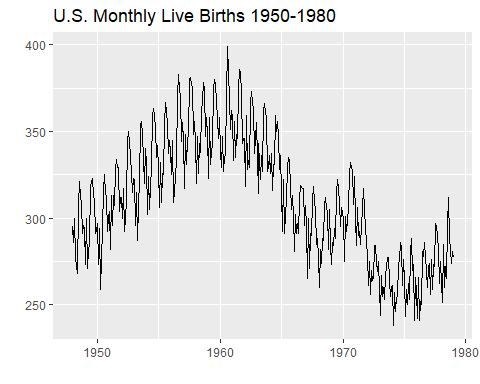
#plot(birth)  
library(ggplot2)  
library(ggfortify)  
library(dplyr)

##   
## 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")



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)

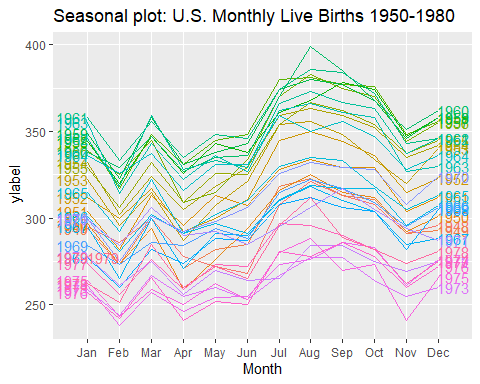
## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

## 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.

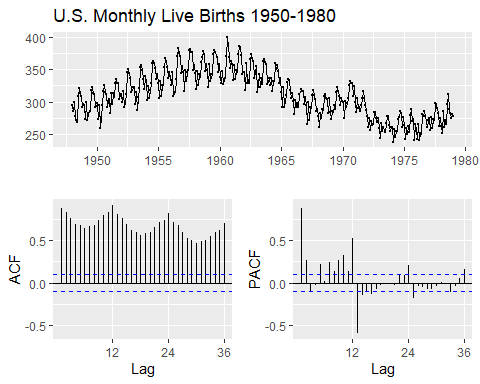
There seems to be a yearly cycle. We can try adding monthly variables or use a sin and/or cosing with the right frequency for a year repetition.

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"),32)[1:n]  
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])

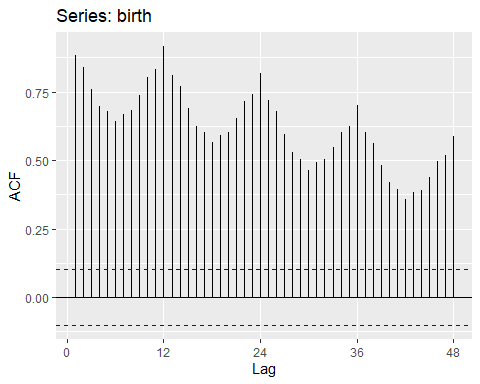
Let’s look at the auto correlation function and partial auto correlation functions

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

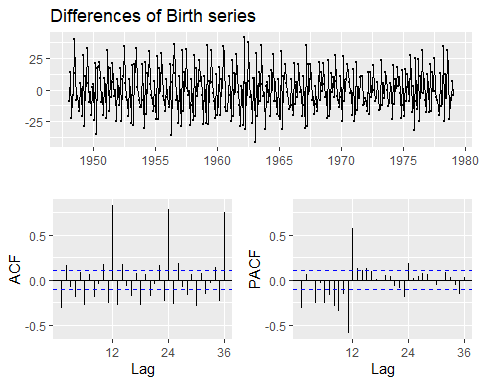


Ussing ggplot2

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



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

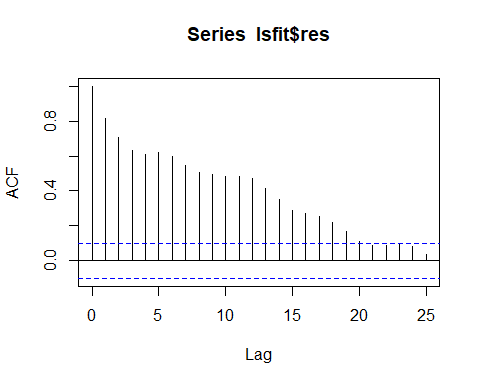


Let’s fit a model with monthly dummy variables. There is a curve trend that is beyond quadratic.

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)

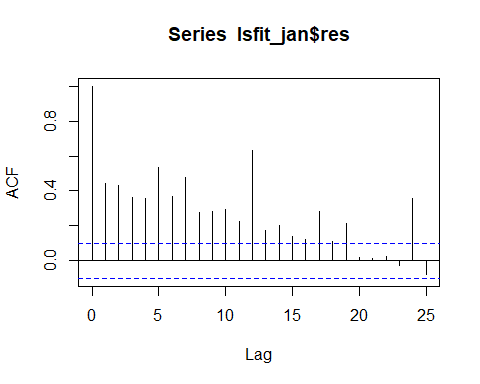
 Although this looks like a good fit, we see that the residuals have autocorrelation.

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)

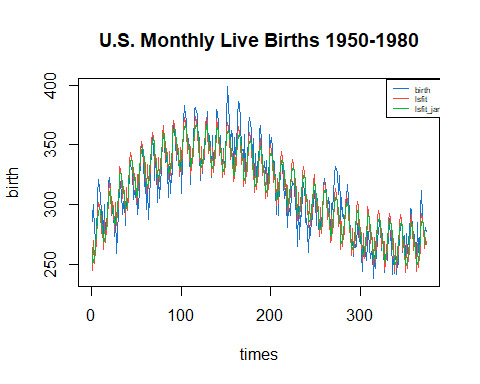
##   
## Call:  
## lm(formula = birth ~ poly(times, 3) + sint + cost + Jan, data = X\_jan)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -33.58 -11.16 -1.32 10.30 48.34   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 310.2118 0.8069 384.462 < 2e-16 \*\*\*  
## 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)



#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 that the ones we did earlier.

Equation corresponding to the time series model:

where are the random errors.

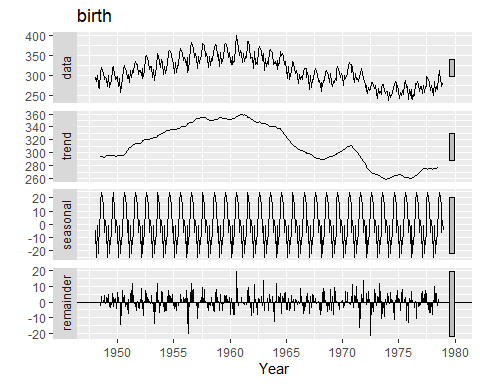
Plugging in the numbers:

Or

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")



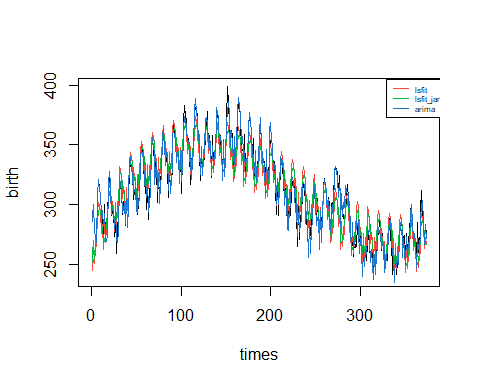
#dbirth<-decompose(birth)  
#plot(dbirth)

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

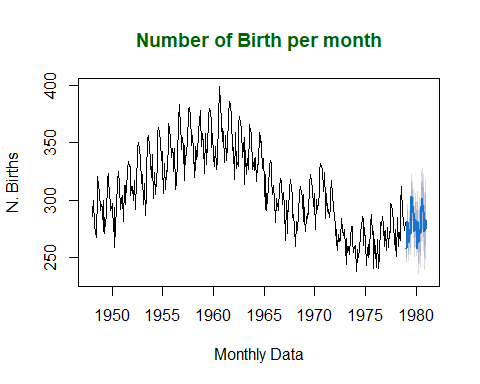
Let’s plot the fitted values of the 3 models:

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)



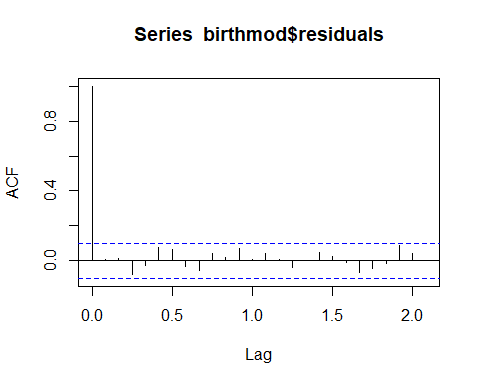
Now let’s use our arima model to do forecasts:

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



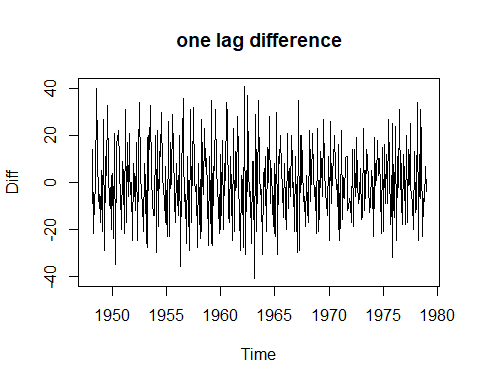
Let’s check that the errors do not have any auto-correlation:

acf(birthmod$residuals)

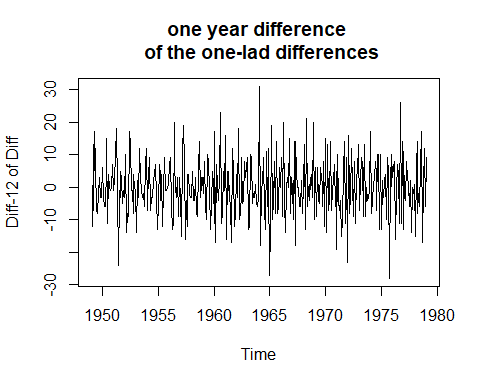


Just for the heck of it, let’s look at the differences involved in the arima model:

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 Diff")



# Time Series Assignment

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

* Plot wine and log(wine).
* Plot the auto correlation and partial auto correlation functions for log(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 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(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 covarites. 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

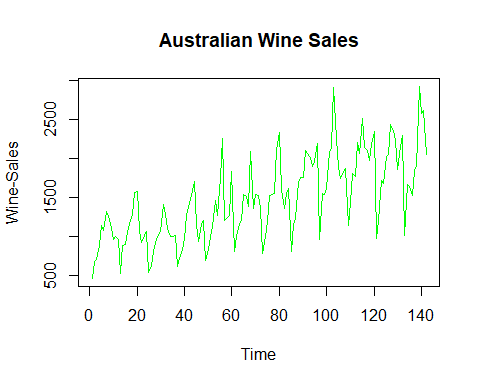
wine=c(  
.46400E+03,  
.67500E+03,  
.70300E+03,  
.88700E+03,  
.11390E+04,  
.10770E+04,  
.13180E+04,  
.12600E+04,  
.11200E+04,  
.96300E+03,  
.99600E+03,  
.96000E+03,  
.53000E+03,  
.88300E+03,  
.89400E+03,  
.10450E+04,  
.11990E+04,  
.12870E+04,  
.15650E+04,  
.15770E+04,  
.10760E+04,  
.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)

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=cbind(times,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec) #sin and cos and constant for Jan;   
X\_jan=cbind(times,sint,cost,Jan) #sin and cos and constant for Jan;

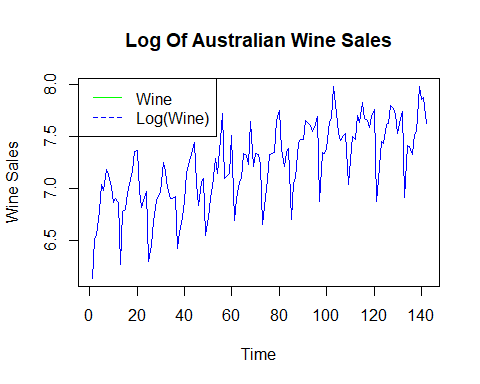
We will fit a model to the log of the Australian wine sales.

* Plot wine and log(wine).

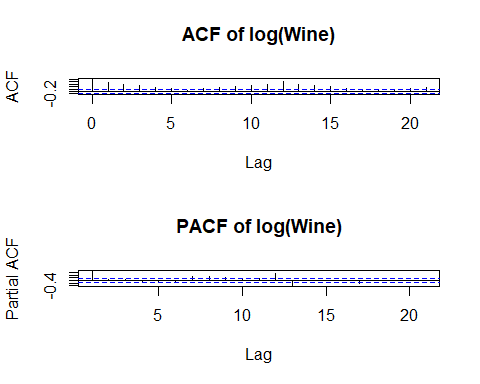
library(ggplot2)  
  
if (!requireNamespace("astsa", quietly = TRUE)) {  
 install.packages("astsa")  
}  
  
# Loading required libraries  
library(astsa)  
  
# Plotting wine and log(wine)  
plot(wine, type = "l", col = "green", ylab = "Wine-Sales", xlab = "Time", main = "Australian Wine Sales")



plot(log(wine), type = "l", col = "blue", ylab = "Wine Sales", xlab = "Time", main = "Log Of Australian Wine Sales")  
legend("topleft", legend = c("Wine", "Log(Wine)"), col = c("green", "blue"), lty = c(1, 2))

 \* Plot the auto correlation and partial auto correlation functions for log(wine).

# Calculating and plotting ACF and PACF for log(wine)  
par(mfrow = c(2, 1))  
acf(log(wine), main = "ACF of log(Wine)")  
pacf(log(wine), main = "PACF of log(Wine)")



* Just as we did for “birth”, fit a model allowing for a term for each month and time.

# Fitting a model with a term for each month and time  
lm\_model <- lm(log(wine) ~ poly(times,3) + Jan + Feb + Mar + Apr + May + Jun + Jul + Aug + Sep + Oct + Nov + Dec, data = data.frame(times,Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec, log(wine)))  
  
# Displaying the summary of the model  
summary(lm\_model)

##   
## Call:  
## lm(formula = log(wine) ~ poly(times, 3) + Jan + Feb + Mar + Apr +   
## May + Jun + Jul + Aug + Sep + Oct + Nov + Dec, data = data.frame(times,   
## Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec,   
## log(wine)))  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.29932 -0.06594 -0.01251 0.06661 0.28057   
##   
## Coefficients: (1 not defined because of singularities)  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 7.331166 0.031477 232.905 < 2e-16 \*\*\*  
## poly(times, 3)1 3.087022 0.104514 29.537 < 2e-16 \*\*\*  
## poly(times, 3)2 -0.165071 0.104539 -1.579 0.116813   
## poly(times, 3)3 -0.502162 0.104869 -4.788 4.60e-06 \*\*\*  
## Jan -0.689645 0.043614 -15.812 < 2e-16 \*\*\*  
## Feb -0.394616 0.043595 -9.052 2.08e-15 \*\*\*  
## Mar -0.267949 0.043581 -6.148 9.39e-09 \*\*\*  
## Apr -0.156363 0.043571 -3.589 0.000473 \*\*\*  
## May 0.013090 0.043566 0.300 0.764313   
## Jun 0.011525 0.043564 0.265 0.791791   
## Jul 0.220215 0.043567 5.055 1.47e-06 \*\*\*  
## Aug 0.228804 0.043574 5.251 6.18e-07 \*\*\*  
## Sep -0.003657 0.043586 -0.084 0.933274   
## Oct -0.113620 0.043602 -2.606 0.010260 \*   
## Nov -0.053197 0.044472 -1.196 0.233849   
## Dec NA NA NA NA   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.1043 on 127 degrees of freedom  
## Multiple R-squared: 0.9327, Adjusted R-squared: 0.9253   
## F-statistic: 125.7 on 14 and 127 DF, p-value: < 2.2e-16

* Just as we did for “birth”, fit a model using sin and cos to model seasonality and time.

# Fitting a model with sin and cos for seasonality and time  
lm\_model\_sin\_cos <- lm(log(wine) ~ poly(times,3)+sint + cost + Jan, data = data.frame(times, sint, cost, Jan, log(wine)))  
  
# Displaying the summary of the model  
summary(lm\_model\_sin\_cos)

##   
## Call:  
## lm(formula = log(wine) ~ poly(times, 3) + sint + cost + Jan,   
## data = data.frame(times, sint, cost, Jan, log(wine)))  
##   
## 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 corresponding to both models.

# Computing AIC, BIC, and adjusted R^2 for the model with a term for each month and time  
lm\_aic\_bic\_adjr2 <- c(AIC(lm\_model), BIC(lm\_model), summary(lm\_model)$adj.r.squared)  
lm\_aic\_bic\_adjr2

## [1] -222.8709896 -175.5777567 0.9252505

# Computing AIC, BIC, and adjusted R^2 for the model with sin and cos for seasonality and time  
lm\_sin\_cos\_aic\_bic\_adjr2 <- c(AIC(lm\_model\_sin\_cos), BIC(lm\_model\_sin\_cos), summary(lm\_model\_sin\_cos)$adj.r.squared)  
lm\_sin\_cos\_aic\_bic\_adjr2

## [1] -150.4375221 -126.7909057 0.8689176

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

library(forecast)  
  
# Using auto.arima to obtain the ARIMA model  
arima\_model <- auto.arima(log(wine))  
  
# Displaying the obtained ARIMA model  
arima\_model

## Series: log(wine)   
## ARIMA(1,1,1)   
##   
## Coefficients:  
## ar1 ma1  
## 0.5214 -0.9277  
## s.e. 0.0821 0.0268  
##   
## sigma^2 = 0.06157: log likelihood = -3.02  
## AIC=12.04 AICc=12.21 BIC=20.88

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

# AIC and BIC of the model with a term for each month and time  
lm\_aic\_bic <- c(AIC(lm\_model), BIC(lm\_model))  
  
# AIC and BIC of the model with sin and cos for seasonality and time  
lm\_sin\_cos\_aic\_bic <- c(AIC(lm\_model\_sin\_cos), BIC(lm\_model\_sin\_cos))  
  
# Displaying all AIC and BIC values for comparison  
comparison <- data.frame(  
 Models = c("lm\_model", "lm\_model\_sin\_cos", "arima\_model"),  
 AIC = c(lm\_aic\_bic[1], lm\_sin\_cos\_aic\_bic[1], arima\_model$aic),  
 BIC = c(lm\_aic\_bic[2], lm\_sin\_cos\_aic\_bic[2], arima\_model$bic)  
)  
comparison

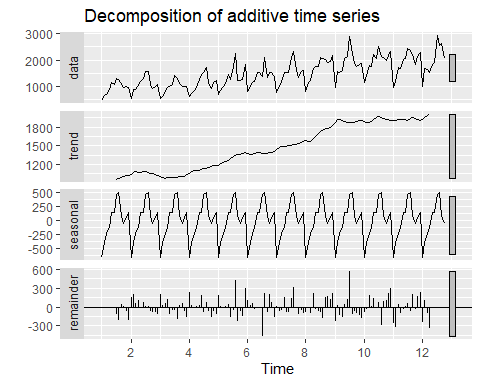
## Models AIC BIC  
## 1 lm\_model -222.87099 -175.5778  
## 2 lm\_model\_sin\_cos -150.43752 -126.7909  
## 3 arima\_model 12.03603 20.8823

* Write down the equation corresponding to the arima model.

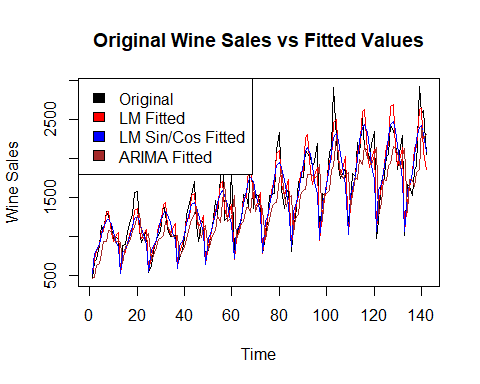
:(1-Ø1B)(1-B)(yt – yt-1)=(1+Ɵ1B)wt - Ø1 is the autoregressive parameter, - B is the backshift operator (used for differencing), - yt is the observed time series, - yt-1 is the lagged value of the time series, - Ɵ1 is the moving average parameter, - wt is the white noise series. The estimated values for Ø1 and Ɵ1 are 0.5214 and -0.9277, respectively, based on your model summary. You can substitute these values into the equation to get the specific form for your ARIMA(1,1,1) model.

* Plot the decomposition of the series.

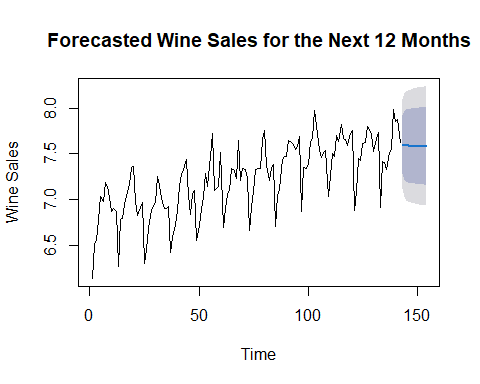
library(forecast)  
# Convert the y to a time series (replace 'frequency = 12' with your actual frequency if different)  
wine\_ts <- ts(wine, frequency = 12)  
# Try decomposing the time series  
decomposed <- try(decompose(wine\_ts))  
# Plot the decomposition if successful  
if (class(decomposed) != "try-error") {  
autoplot(decomposed)  
} else {  
cat("Unable to decompose the time series.")  
}

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

# Creating a dataframe with original and fitted values in the original scale  
# Creating a data frame with original and fitted values  
df\_fitted <- data.frame(  
 times = 1:length(wine),  
 wine = wine,  
 lm\_fitted = exp(predict(lm\_model)),  
 lm\_sin\_cos\_fitted = exp(predict(lm\_model\_sin\_cos)),  
 arima\_fitted = exp(fitted(arima\_model))  
)  
  
# Plotting the original wine values and the fitted values from the models  
plot(df\_fitted$times, df\_fitted$wine, type = "l", xlab = "Time", ylab = "Wine Sales",  
 main = "Original Wine Sales vs Fitted Values")  
lines(df\_fitted$times, df\_fitted$lm\_fitted, col = "red")  
lines(df\_fitted$times, df\_fitted$lm\_sin\_cos\_fitted, col = "blue")  
lines(df\_fitted$times, df\_fitted$arima\_fitted, col = "brown")  
legend("topleft", legend = c("Original", "LM Fitted", "LM Sin/Cos Fitted", "ARIMA Fitted"),  
 fill = c("black", "red", "blue", "brown"))

 \* plot the predicted values for the next 12 months.

# Generating forecasts for the next 12 months  
forecast\_values <- forecast(arima\_model, h = 12)  
  
# Plotting the forecasted values  
plot(forecast\_values, xlab = "Time", ylab = "Wine Sales",  
 main = "Forecasted Wine Sales 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 covarites. 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

# Fitting the ARIMA model with covariates and differencing  
arima\_model\_xreg\_diff <- Arima(log(wine), order = c(1, 1, 1), xreg = X)  
  
# Displaying the model summary  
summary(arima\_model\_xreg\_diff)

## Series: log(wine)   
## Regression with ARIMA(1,1,1) errors   
##   
## Coefficients:  
## ar1 ma1 times Feb Mar Apr May Jun Jul  
## 0.0986 -0.8350 0.0058 0.2939 0.4195 0.5300 0.6984 0.6958 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 0.001359051 0.100425 0.07901908 0.00517242 1.087874 0.4068724  
## ACF1  
## Training set -0.008184978

# Fitting the ARIMA model with covariates without differencing  
arima\_model\_xreg <- Arima(log(wine), order = c(1, 0, 1), xreg = X)  
  
# Displaying the model summary  
summary(arima\_model\_xreg)

## Series: log(wine)   
## 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 0.4000067  
## ACF1  
## Training set 0.01067203