FITTING THE APPROPRIATE ARMA MODEL TO TIME SERIES

Pranav Gopalkrishna, 1940223

2022-03-08

# INTRODUCTION

In this assignment, we aim to to fit the appropriate ARIMA model i.e. autoregressive integrated moving average model (which generalizes autoregression, integration and moving average... these are discussed in the next-to-next section). We will limit our models to AR, MA and ARMA. Hence, the order of integration (discussed in the next-to-next section) will be limited to 0.

# INTRODUCTION TO ARMA MODELS

An ARMA(p, q) process is a combination of an autoregressive process of order p and a moving average process of order q, i.e. the value of an observation linearly depends on

* The previous p observations
* The previous q white noise terms

where

* is the white noise (random) component for time t i.e. a normally distributed random component that is used in a manner similar to an error term in regression models.
* is the *i*th constant coefficient i.e. the constant coefficient for the observation at lag *i*
* is the *i*th constant coefficient i.e. the constant coefficient for the observation at lag *i*

# INTRODUCTION TO ARIMA MODELS

**ARIMA** => **A**utoregressive **I**ntegrated **M**oving **A**verage

ARIMA is a generalization of AR models, MA (moving average) models and I (integrated) models. These components are explained below:

1. ***AR (Autoregression)***  
Current values depend on some number of past observations up to a certain lag.

2. ***I (Integrated)***  
A time series is said to be integrated of order d if taking repeated differences between zt values results in a stationary process. For example, if zt is integrated of order 2, then  
zt - zt-1 - zt-2  
is a stationary process. Here, ‘integrated’ refers the property of the time series being integrated at some order, i.e. it refers to time series that are non-stationary with respect to mean, but can become stationary after some number of repeated differencing.

3. ***MA (Moving Average)***  
The value at each time point is smoothened by averaging some number of past and future values around it.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ARIMA model has three integer components, namely:

1. ***Order of autoregression*** of the time series (p). This indicates the maximum lag at which observations are included i.e. extent to which past values are used to model current values.

2. ***Order of integration*** of the time series (d). This specifies the non-seasonal (i.e. trend) part of the model.

3. ***Order of moving average*** of the time series (q). This indicates the number of values around and including the current time point’s value that are averaged, in order to smoothen the current time point’s value. For example, MA(3) implies that each time point’s value is replaced by the average of the just preceding, current and just succeeding value.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CREATING THE TIME SERIES

# DATASET

## Importing necessary data

data = read.csv("~/Documents/Study/computerScience/programming/r/data/agriculturalRawMaterial.csv")[c(1, 22)]  
head(data)

## Month Soft.sawnwood.Price  
## 1 Apr-90 218.76  
## 2 May-90 213.00  
## 3 Jun-90 200.00  
## 4 Jul-90 210.05  
## 5 Aug-90 208.30  
## 6 Sep-90 199.59

summary(data)

## Month Soft.sawnwood.Price  
## Apr-00 : 1 Min. :183.6   
## Apr-01 : 1 1st Qu.:277.6   
## Apr-02 : 1 Median :295.0   
## Apr-03 : 1 Mean :291.1   
## Apr-04 : 1 3rd Qu.:310.9   
## Apr-05 : 1 Max. :372.6   
## (Other):355 NA's :34

We notice 34 NA values for soft sawnwood prices. Inspecting the dataset, we see that these NA values are grouped at the tail-end of the column. Hence, we can simply remove the last 34 rows of the columns.

prices = data$Soft.sawnwood.Price[c(1:(length(data$Month) - 34))]  
months = data$Month[c(1:(length(data$Month) - 34))]

## Defining and formatting the ‘Month’ variable

***(for better date summary)***

t = c()  
for(x in months)  
{  
 x = paste("01", x, sep = "-")  
 t = c(t, x)  
}  
t = as.Date(t, format = "%d-%b-%y")  
# %b => abbreviated month  
# %y => 2 digit year  
# It can recognize century on its own.  
# So for example, '93' will be interpreted as '1993'.

Summarising dates

summary(t)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## "1990-04-01" "1997-01-16" "2003-11-01" "2003-10-31" "2010-08-16" "2017-06-01"

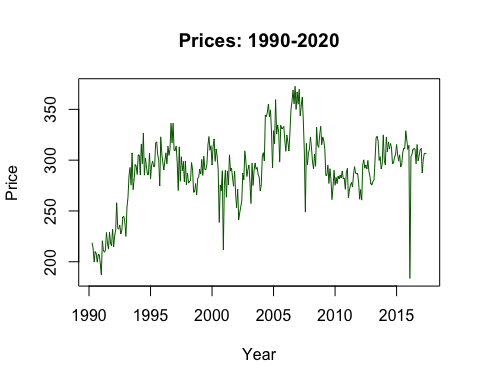
# CONVERTING TO TIME SERIES

Z = ts(prices, start = c(1990, 4, 1), end = c(2017, 6, 1), frequency = 12)  
# frequency = 12 -> monthly frequency  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

MAKING TIME SERIES STATIONARY

# TIME PLOT AND POTENTIAL TIME SERIES COMPONENTS

ts.plot(Z,  
 main = "Prices: 1990-2020",  
 ylab = "Price",  
 xlab = "Year",  
 col = "darkgreen")



Based on the time plot, we may conclude that there is no clear seasonality (i.e. no periodic fluctuations annually). We shall confirm this more rigorously using the ADF test. There seems to be an overall upward trend over time, with a seemingly high level of irregular fluctuations. We also see no discernible long term fluctuation pattern, hence we may conclude that there is no cyclical fluctuation. Hence, the major components of this time series seem to be trend and irregular fluctuations.

# TESTING THE STATIONARITY OF THE ORIGINAL TIME SERIES

## ADF test

ADF test is used to check the stationarity of a time series. Its null hypothesis is that there is the data is not stationary, and its alternative hypothesis is that the data is stationary. As with all statistical tests, the p-value indicates whether we may or may not reject the null hypothesis. Only if it is lower than the level of significance (in our case, this is 5% or 0.05) may we reject the null hypothesis with the given level of confidence.

## Performing the test using the ‘tseries’ library

library(tseries)

## Warning: package 'tseries' was built under R version 3.6.2

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

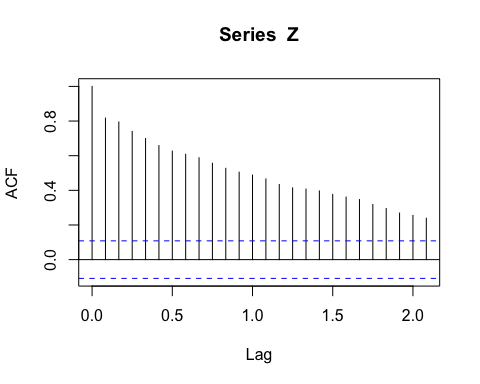
### Testing for stationarity

adf.test(Z)

##   
## Augmented Dickey-Fuller Test  
##   
## data: Z  
## Dickey-Fuller = -3.1445, Lag order = 6, p-value = 0.09756  
## alternative hypothesis: stationary

### Observing ACF plot

acf(Z)



Based on the ACF plot, we may conclude that there is no significant seasonality in the time series. Given that the p-value for the ADF test is 0.09756 > 0.05, (where 0.05 is the level of significance) we may accept the null hypothesis that the time series is not stationary, with a 95% confidence. Hence, we may conclude that the time series is not stationary. Given the seasonal component is not significant, and given that there is no string indication of long-term fluctuations, we may conclude that the trend component is significant.

# HOW WE WOULD ELIMINATE TREND

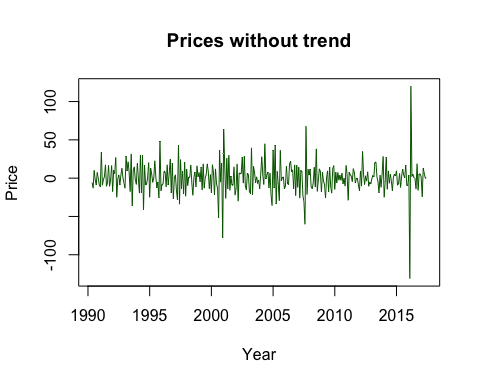
***(and why we need not do so in this case)***

Based on the above conclusions, we will use the model zt = st + et, where z\_t is the softlog price at time t, st is the seasonal component at time t, et is the error at time t (irregular fluctuations). We assume there to be a constant downward trend component.

To use the method of differencing to remove trend and seasonality, we use the ‘diff’ function. To remove trend, we use lag = 1 (the default lag). Since we concluded that the seasonality us not significant, we will not perform differencing for it.

**In our case**, however, since we are using the ‘auto.arima’ function from the ‘forecast’ library of R, we do not need to eliminate trend and differencing on our own, since the function does the appropriate differencing operations based on statistical tests that determine the significance of the trend and seasonality. This fact is mentioned in the documentation of this function. However, we will still eliminate trend…

Z1 = diff(Z)  
ts.plot(Z1,  
 main = "Prices without trend",  
 ylab = "Price",  
 xlab = "Year",  
 col = "darkgreen")



# TESTING STATIONARITY OF NEW TIME SERIES

## Performing the ADF test using the ‘tseries’ library

### Testing for stationarity

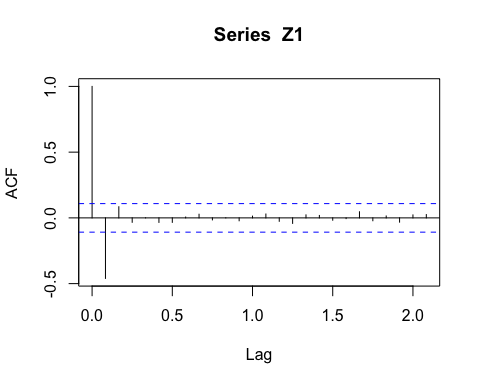
adf.test(Z1)

## Warning in adf.test(Z1): p-value smaller than printed p-value

##   
## Augmented Dickey-Fuller Test  
##   
## data: Z1  
## Dickey-Fuller = -9.4481, Lag order = 6, p-value = 0.01  
## alternative hypothesis: stationary

### Observing ACF plot

acf(Z1)



Based on the ACF plot, we can see that for all lags other than 1, the autocorrelation coefficient is not significant, indicating stationarity. This is supported by the ACF, whose p-value is below 0.05 (where 0.05 is the level of significance).

# FITTING THE APPROPRIATE STATIONARY MODEL

library(forecast)

## Warning: package 'forecast' was built under R version 3.6.2

We will not only be fitting an appropriate ARIMA model for our dataset(s), but also trace the models that the function has considered. This is to demonstrate that the function is intended to search for the best possible ARIMA model (with the most suitable orders of autoregression (p), integration (d) and moving average(q))

Also, since our focus is on ARMA models in particular, we will be limiting the integration component of the ARIMA model to 0, by using the argument max.d = 0.

Fitting the stationary dataset, we have obtained…

auto.arima(Z1, max.d = 0, seasonal = "FALSE", trace = TRUE)

##   
## Fitting models using approximations to speed things up...  
##   
## ARIMA(2,0,2) with non-zero mean : 2803.831  
## ARIMA(0,0,0) with non-zero mean : 2889.745  
## ARIMA(1,0,0) with non-zero mean : 2815.061  
## ARIMA(0,0,1) with non-zero mean : 2801.68  
## ARIMA(0,0,0) with zero mean : 2887.778  
## ARIMA(1,0,1) with non-zero mean : 2804.081  
## ARIMA(0,0,2) with non-zero mean : 2803.601  
## ARIMA(1,0,2) with non-zero mean : 2805.707  
## ARIMA(0,0,1) with zero mean : 2800.05  
## ARIMA(1,0,1) with zero mean : 2802.442  
## ARIMA(0,0,2) with zero mean : 2801.929  
## ARIMA(1,0,0) with zero mean : 2813.195  
## ARIMA(1,0,2) with zero mean : 2804.035  
##   
## Now re-fitting the best model(s) without approximations...  
##   
## ARIMA(0,0,1) with zero mean : 2800.252  
##   
## Best model: ARIMA(0,0,1) with zero mean

## Series: Z1   
## ARIMA(0,0,1) with zero mean   
##   
## Coefficients:  
## ma1  
## -0.5512  
## s.e. 0.0500  
##   
## sigma^2 = 311.5: log likelihood = -1398.11  
## AIC=2800.21 AICc=2800.25 BIC=2807.79

To show that the data is made stationary if it is not, we shall also fit the original time series data…

auto.arima(Z, max.d = 0, seasonal = "FALSE", trace = TRUE)

##   
## Fitting models using approximations to speed things up...  
##   
## ARIMA(2,0,2) with non-zero mean : 2806.68  
## ARIMA(0,0,0) with non-zero mean : 3239.449  
## ARIMA(1,0,0) with non-zero mean : 2868.115  
## ARIMA(0,0,1) with non-zero mean : 3075.697  
## ARIMA(0,0,0) with zero mean : 4644.938  
## ARIMA(1,0,2) with non-zero mean : 2804.981  
## ARIMA(0,0,2) with non-zero mean : 2983.027  
## ARIMA(1,0,1) with non-zero mean : 2803.507  
## ARIMA(2,0,1) with non-zero mean : 2804.688  
## ARIMA(2,0,0) with non-zero mean : 2810.82  
## ARIMA(1,0,1) with zero mean : Inf  
##   
## Now re-fitting the best model(s) without approximations...  
##   
## ARIMA(1,0,1) with non-zero mean : 2810.341  
##   
## Best model: ARIMA(1,0,1) with non-zero mean

## Series: Z   
## ARIMA(1,0,1) with non-zero mean   
##   
## Coefficients:  
## ar1 ma1 mean  
## 0.9693 -0.5200 286.2942  
## s.e. 0.0162 0.0546 14.0700  
##   
## sigma^2 = 309.7: log likelihood = -1401.11  
## AIC=2810.22 AICc=2810.34 BIC=2825.38

At least in this case, there is very little difference between the estimated values of the models, whether you use the original dataset, or a stationary dataset derived from the original non-stationary dataset.

Through the function's process trace, we can see how it searches through various ARIMA models with different orders (i.e. different p, d and q values), to ultimately find the one that produces the least error i.e. the least overall difference between model estimations and actual values for any time t.