Exponential Smoothing Using Holt's & Winter's methods

Pranav Gopalkrishna, 1940223

2022-02-02

# REQUIRED LIBRARIES

# ASTSA => Applied Statistical Time Series Analysis  
# install.packages("astsa")  
# install.packages("forecast")  
# install.packages("tseries")

library(astsa)  
library(forecast)  
library(tseries)

# INTRODUCTION & AIM

Forecasting future observations is the main aim of time series analysis. Modelling the deterministic components such as trend and periodicity are key in this endeavor, although future forecasts will almost always be subject to some random error. Smoothing involves taking averages of past values to predict future values.

Exponential smoothing involves taking a weighted average of past values to predict future values. Here, more recent observations are given more weightage, and weights decrease as we go to more previous observations. In this assignment, our aim is to use Holt’s and Winter’s exponential smoothing methods to create models for our time series, in order to predict five future observations.

# DATA

## Importing necessary data

setwd("~/Documents/Study/computerScience/programming/r/data/")  
data = read.csv("agriculturalRawMaterial.csv")[c(1, 4)]  
head(data)

Month | Copra.Price  
Apr-90 | 236  
May-90 | 234  
Jun-90 | 216  
Jul-90 | 205  
Aug-90 | 198  
Sep-90 | 196

## Defining and formatting the ‘Month’ variable

t = c()  
for(x in data$Month)  
{  
 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'.

# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

HOLT-WINTER EXPONENTIAL SMOOTHING

Holt's exponential smoothing is used when the time series has only trend and no seasonality, whereas Winter's exponential smoothing considers both trend and seasonality. These two methods involve the estimation of trend component 'beta' and the smoothing constant 'alpha', which is in fact 1 - 'omega', where omega represents the weight whose different powers are applied to the past observations when attempting to estimate future observations.

# DATA FORMATTING & TIME SERIES CREATION

## Defining and reformatting the ‘Copra.Price’ variable

z = c()  
for(p in data$Copra.Price)  
{  
 p = gsub(',', '', p)  
 # Removing commas in the numbers  
 # 1st argument => what to replace  
 # 2nd argument => what to put instead  
 # 3rd argument => full string  
 z = c(z, p)  
}  
z = as.numeric(z)

## Summarizing the data

df = data.frame(t, z)  
summary(df)

t | z   
Min. :1990-04-01 | Min. : 182   
1st Qu.:1997-10-01 | 1st Qu.: 372   
Median :2005-04-01 | Median : 458   
Mean :2005-04-01 | Mean : 542   
3rd Qu.:2012-10-01 | 3rd Qu.: 714   
Max. :2020-04-01 | Max. :1503   
 | NA's : 22

We can see there are 22 missing values in the price column. Checking in the dataset itself, values for copra prices are available until a certain point, after which we have these missing values. Hence, we can simply remove the tail end of the dataset where these missing values are concentrated.

t\_new = t[c(1:(length(t) - 22))]  
z\_new = z[c(1:(length(z) - 22))]  
df = data.frame(t\_new, z\_new)  
summary(df)

t\_new | z\_new   
Min. :1990-04-01 | Min. : 182   
1st Qu.:1997-10-01 | 1st Qu.: 372   
Median :2005-04-01 | Median : 458   
Mean :2005-04-01 | Mean : 542   
3rd Qu.:2012-10-01 | 3rd Qu.: 714   
Max. :2020-04-01 | Max. :1503

## Creating a time series for softlog prices

Z = ts(z\_new, start = c(1990, 4, 1), end = c(2018, 6, 1), frequency = 12)  
# frequency = 12 -> monthly frequency

# TIME PLOT & POTENTIAL TIME SERIES COMPONENTS

## General function to create time plots with specified intervals

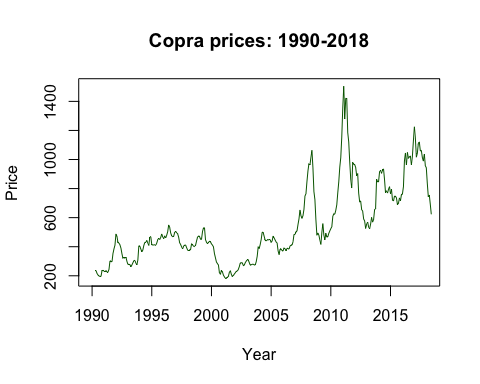
timeplot = function(min, max)  
{  
 ts.plot(Z,  
 main = paste("Copra prices: ", min, "-", max, sep = ''),  
 ylab = "Price",  
 xlab = "Year",  
 xlim = c(min, max),  
 col = "darkgreen")  
}

## Time plots for different intervals of time

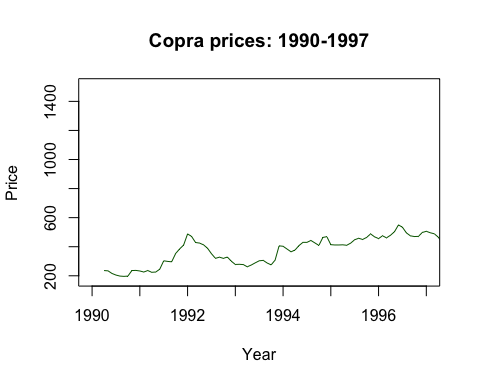
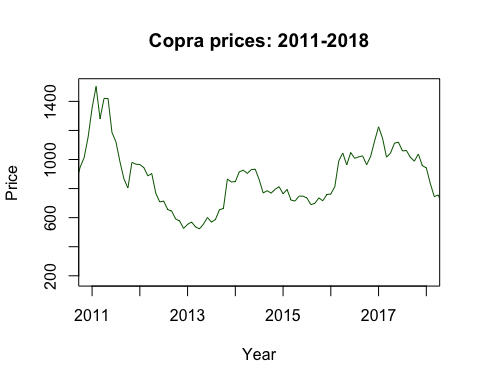
To study the graphs more closely for recognizing time series components on an annual level...

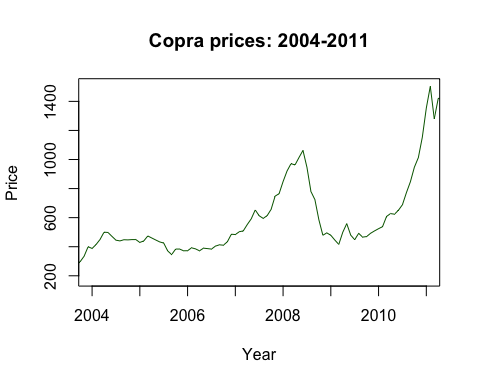
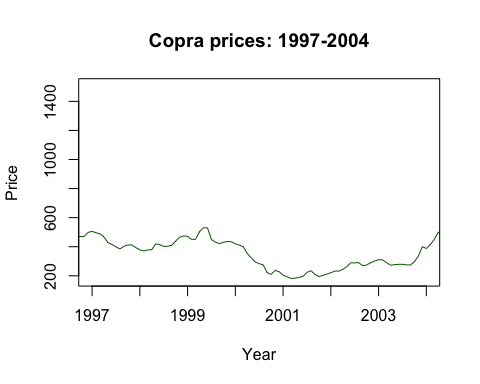
timeplot(1990, 2018)  
timeplot(1990, 1997)  
timeplot(1997, 2004)  
timeplot(2004, 2011)  
timeplot(2011, 2018)

## Overall time plot



## For the specific time intervals





Based on the time plots, we may conclude that there is no clear seasonality (i.e. no periodic fluctuations annually). There seems to be an overall upward trend over time, with a seemingly high level of irregular fluctuations. We may also observe a discernible long-term fluctuation pattern over a around a year period, hence we may conclude that there is some cyclical fluctuation. However, the major components of this time series seem to be trend and irregular fluctuations.

# TESTING STATIONARITY OF TIME SERIES

## Performing the test using the ‘tseries’ library

### Original time series stationarity

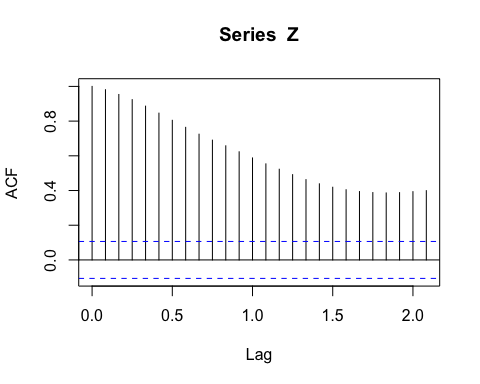
adf.test(Z)

**Augmented Dickey-Fuller Test**  
  
data: Z  
***Dickey-Fuller*** = -3.9003, ***Lag order*** = 6, ***p-value*** = 0.01431  
**alternative hypothesis:** stationary

As we can see, the p value is lower than 0.05, meaning that the series is stationary, given a 0.05 significance level. But it exceeds 0.01, meaning that the series is non-stationary, given a 0.01 significance level. In any case, we will apply exponential smoothing to further remove the relatively insignificant trend component.

### Observing ACF plot

acf(Z)



The above graph of the autocorrelation coefficient of the time series at lags from 0 to 2 indicate that the time series is not stationary, since autocorrelation is significant for all lags from 1 to 2, implying some deterministic movement of the time series observation based on past values.

# EXPONENTIAL SMOOTHING MODEL CREATION

In our time plots, we have observed no seasonal component. Hence, the exponential smoothing model appropriate for our time series is Holt’s exponential smoothing, which applies for time series without seasonality. To obtain Holt’s method using the **HoltWinters** function, you must give the argument ‘gamma = FALSE’. Otherwise, it will apply Winter’s method of exponential smoothing.

model = HoltWinters(Z, gamma = FALSE)  
print(model)

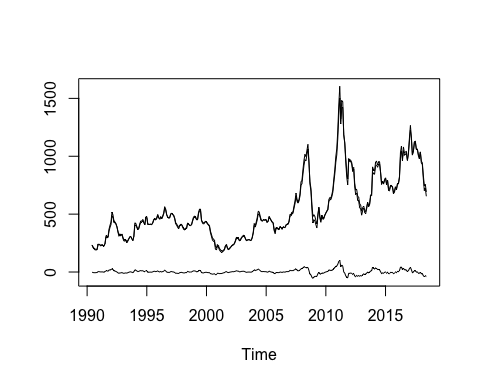
*Holt-Winters exponential smoothing with trend and without seasonal component.*

**Call:**  
HoltWinters(x = Z, gamma = FALSE)

**Smoothing parameters:**  
alpha: 1  
beta : 0.1733467  
gamma: FALSE

**Coefficients:**  
a: 623.00000  
b: -43.12296

'beta' is the estimate of the slope of the trend component.

ts.plot(model$fitted)

The above graph shows the original time plot against the smoothened time plot below, which has a much weaker trend component and more resembles white noise data, indicating stationarity.

# FORECASTING USING ABOVE MODEL

## Creating forcast data

forecastedData = forecast(model, h = 5)

The above command uses the exponential smoothing model we created for our data to forecast the following five observations i.e. for the next five months.

forecastedData

| Point Forecast | Lo 80 | Hi 80 | Lo 95 | Hi 95  
Jul 2018 | 579.8770 | 515.3358 | 644.4183 | 481.1698 | 678.5843  
Aug 2018 | 536.7541 | 437.2529 | 636.2553 | 384.5801 | 688.9280  
Sep 2018 | 493.6311 | 361.5134 | 625.7488 | 291.5745 | 695.6877  
Oct 2018 | 450.5082 | 285.9489 | 615.0674 | 198.8366 | 702.1798  
Nov 2018 | 407.3852 | 209.8384 | 604.9320 | 105.2634 | 709.5070

## Plotting the forecasted data

plot(forecastedData,  
 main = "Past data + forecasts for the next 5 monthts",  
 xlab = "Year",  
 ylab = "Price")

