ARIMA modelling to beer production in India:

***Obtaining and comparing forecasts for testing data***

Pranav Gopalkrishna, 1940223

2022-04-26

# INTRODUCTION & AIMS

Model identification and fitting for a given time series data is key in helping predict future values for the time series i.e. forecasting. The aim is to obtain the best autoregressive integrated moving average model (ARIMA) for given dataset, using the **auto.arima** function (which compares many possible ARIMA models and selects the best one), and compare the model's accuracy using in-sample forecasts. For this purpose, we will be dividing the sample data into training and testing data.

Having identified and fitted the best ARIMA model, we will use residual analysis to comment on the suitability of our model, and compare forecasts for the testing data time periods to the actual testing data values.

# DATASET

This dataset contains the monthly beer production in India, from January, 1956 to August, 1987. The beer production's units were unspecified, hence we cannot comment on the magnitude of our findings.

setwd("~/Documents/Study/computerScience/programming/r/data/")  
data = read.csv("monthlyBeerProductionIndia.csv")  
head(data)

## Sneakpeek

**Month | Monthly.beer.production**  
1956-01 | 93.2  
1956-02 | 96.0  
1956-03 | 95.2  
1956-04 | 77.1  
1956-05 | 70.9  
1956-06 | 64.8

## **Unformatted summary**

summary(data)

**Month | Monthly.beer.production**  
1956-01: 1 | Min. : 64.8   
1956-02: 1 | 1st Qu.:112.9   
1956-03: 1 | Median :139.2   
1956-04: 1 | Mean :136.4   
1956-05: 1 | 3rd Qu.:158.8   
1956-06: 1 | Max. :217.8   
(Other):470

## Dividing dataset into training and testing data

**(To test for model accuracy later)**

fullDataLength = length(data$Month)

**80% of the data will be for training, the rest for testing**

trainingDataLength = as.integer(fullDataLength\*80/100)  
trainingData = data$Monthly.beer.production[c(1:trainingDataLength)]  
testingData = data$Monthly.beer.production[c((trainingDataLength+1):fullDataLength)]

**NOTE**: The above data values do not contain any columns, they are simply vectors of numeric values.

# TRAINING DATA HANDLING

## Checking first and last dates

data$Month[1]

[1] 1956-01  
**476 Levels**: 1956-01 1956-02 1956-03 1956-04 1956-05 1956-06 1956-07 ... 1995-08

data$Month[trainingDataLength]

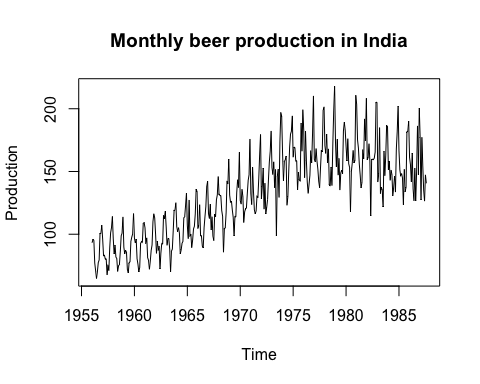
[1] 1987-08  
**476 Levels**: 1956-01 1956-02 1956-03 1956-04 1956-05 1956-06 1956-07 ... 1995-08

# Alternate code (since we have converted the month column to date objects)  
# min(data$Month[1])  
# max(data$Month[1])  
  
# Converting training data to time series  
z = ts(trainingData, start = c(1956, 1, 1), end = c(1987, 8, 1), frequency = 12)

# BASIC ANALYSIS

## Time plot

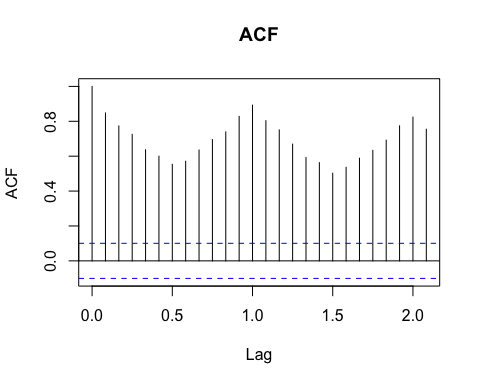
ts.plot(z,  
 main="Monthly beer production in India",  
 xlab="Time",  
 ylab="Production")



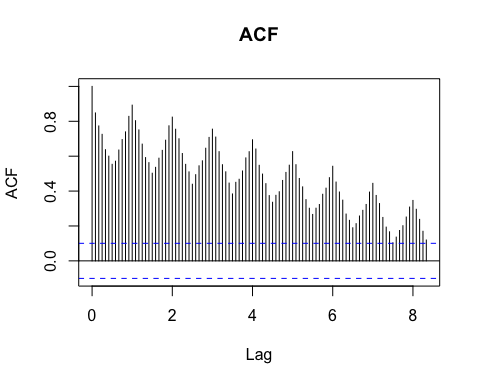
From the data, there can be said to be an overall upward trend, although this is not linear nor constantly increasing, as it stagnates and starts dipping after around 1975. There is no sign of cyclic fluctuation yet, but given a larger dataset, we may observe it, especially considering the wave-like motion of the data’s plot overall. Irregular fluctuations seem to be limited, and the main source of fluctuation seems to be periodic (in a smaller scale than seasonal periodicity i.e. for a time frame below a year). We also see that the range of fluctuation seems to be limited around the general trend of the data, implying constant variance.

## ACF plot

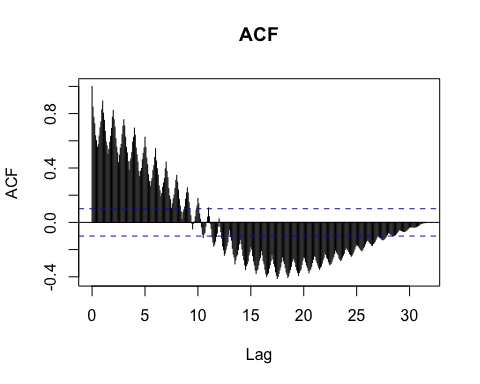
acf(z, main="ACF")



acf(z, main="ACF", 100)



acf(z, main="ACF", 1000)



Hence, from the ACF, due to the ACF being a periodic function of the lag 1, we observe autoregression of order 1. Furthermore, we saw in the time plot that the variance may be constant. However, we observed some sort of trend, which is not consistent, and may infact present an overall lack of trend in the long run (which we cannot confirm right now).

## Stationarity testing

We will now test the stationarity of the time series using the ADF (augmented Dickey-Fuller test), whose hypotheses are as follows...

**H0** (null hypothesis): Data is not stationary  
**H1** (alternate hypothesis: Data is stationary

library(tseries)adf.test(z)

**Augmented Dickey-Fuller Test**

**data**: z  
**Dickey-Fuller** = -4.4575, **Lag order** = 7, **p-value** = 0.01  
**alternative hypothesis**: stationary

p-value is below 0.05. Hence, for a 0.05 significance level, we may reject the null hypothesis and conclude that the data is stationary.

# FINDING THE BEST ARMA MODEL

Model shows no clear trend or periodicity, hence we give ‘seasonal=FALSE’. We expect there to be at least an AR(1) (higher orders may be possible) component. Data was tested to be stationary using the ADF test, so we expect no integration component.

library(forecast)

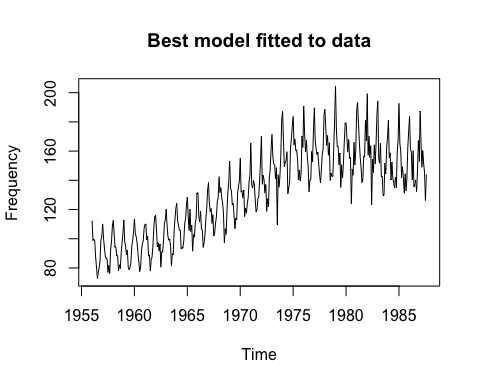
model = auto.arima(z, seasonal = FALSE, max.d = 0)  
summary(model)

Series: z   
ARIMA(3,0,1) with non-zero mean

Coefficients:  
 **| ar1 | ar2 | ar3 | ma1 | mean**  
 **|**-0.0397 |0.5710 |0.2790 |0.723 |130.9045  
**s.e.** **|** 0.0946 | 0.0798|0.0492 |0.088 | 7.9343  
   
sigma^2 = 309.9: log likelihood = -1627.35  
AIC=3266.7 AICc=3266.92 BIC=3290.34

Training set error measures:  
**ME |RMSE |MAE |MPE |MAPE |MASE |ACF1**  
0.1836168|17.48887|13.7737|-1.637533|10.54398|1.502645|0.007761517

plot(model$fitted,  
 main="Best model fitted to data",  
 ylab="Frequency")



We see that model’s plot closely resembles the actual data, indicating a goodness of fit. We can perform residual analysis to further confirm whether this model is suitable…

# QUICK RESIDUAL ANALYSIS

Residual analysis checks whether the residuals of a model (i.e. the differences between the model’s estimates and the actual values) are uncorrelated and normally distributed. The reasoning behind this is discussed at the end of this section.

residuals = model$residuals

## Checking mean of residuals

mean(residuals)

[1] 0.1836168

The mean is close to 0. However, to see if the mean is close enough to zero that we may consider the mean of all possible residuals to be 0, we will perform a one-sample t-test to compare the residual mean to the theoretical value 0. For the sample of residuals and theoretical mean 0, we have the following hypotheses...  
***H0****: True mean of residuals is 0*  
***H1****: True mean of residuals not 0*

t.test(residuals, mu=0)

**One Sample t-test**

**data**: residuals  
**t** = 0.20441, **df** = 379, **p-value** = 0.8381  
**alternative hypothesis**: true mean is not equal to 0

**95 percent confidence interval**:  
-1.582646 1.949880  
**sample estimates**:  
mean of x   
0.1836168

p-value exceeds 0.05. Hence, given a 0.05 significance level, we may accept the null hypothesis and conclude that the true mean of the residuals of this model is 0.

## Using Portmanteau test (Box-Pierce test) to test for uncorrelatedness

***H0****: No significant autocorrelation*  
***H1****: Significant autocorrelation for at least some lags*

Box.test(residuals)

**Box-Pierce test**

**data**: residuals  
**X-squared** = 0.022892, **df** = 1, **p-value** = 0.8797

p-value is greater than 0.05. Hence, given a 0.05 significance level, we may accept the null hypothesis and conclude that there is no significant autocorrelation in the residuals i.e. they are uncorrelated.

## Using Shapiro-Wilk test to test for normality

***H0****: No difference between normal distribution and sample distribution*  
***H1****: There is difference between normal distribution and sample distribution*

shapiro.test(residuals)

**Shapiro-Wilk normality test**

**data**: residuals  
**W** = 0.99353, **p-value** = 0.1036

p-value of the test statistic is above 0.05. Hence, for a 0.05 significance level, we may accept the null hypothesis and conclude that the residuals are normally distributed.

Based on our residual analysis, we can conclude that the residuals obtained from the estimated model are uncorrelated and normally distributed. From uncorrelatedness, we may conclude that the errors are independently generated i.e. they are not dependent on previously obtained errors. The more correlated the errors are, the more accurately we can model them using autoregression models. Hence, errors are not being sufficiently uncorrelated indicates that the errors are deterministic to some degree, and we have not modelled this deterministic aspect, which means our model is consistently more inaccurate than it could be.

From normality, we may conclude that the errors are not determinined by any external deterministic factors. This is because the residuals are expected to be 0 most of the time, and the probability of obtaining a residual other than 0 must be lesser the further the residual is from 0. The probability must only depend on the distance from 0, and not the sign. All this implies that the residuals must follow a standard normal distribution. Since we have already determinined the the true mean of the residuals is 0, all we need to confirm is normality. If the residuals are not normally distributed, and especially if the distribution is assymmetric, it may indicate the presence of an external deterministic factor affecting the residuals to behave differently than expected.

Combining both the above points, we may conclude that the source of errors (i.e. deviations between the model estimates and the actual values) is not systematic (i.e. there is no practical way to model these errors).

# OBTAINING AND TESTING FORECASTS

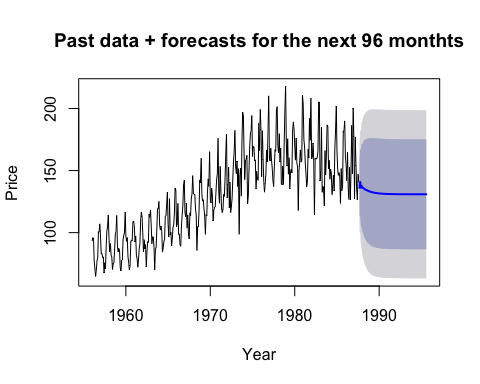
For this, we will compare the forecasts for the testing data’s time periods to the actual testing data from these time periods.

## Forecasts

testingDataLength = fullDataLength - trainingDataLength  
forecasted = forecast(model, h = testingDataLength)

## Plotting forecasts

plot(forecasted,  
 main = "Past data + forecasts for the next 96 monthts",  
 xlab = "Year",  
 ylab = "Price")



## Comparing with plot of actual values

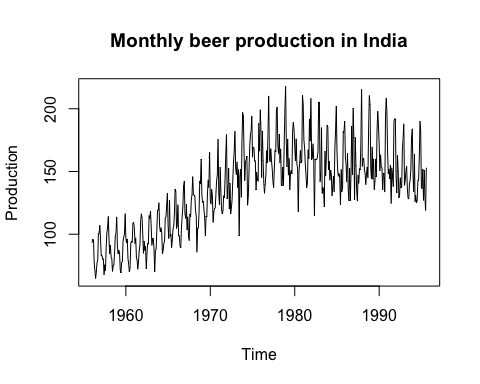
# Converting whole data set into time series object  
data$Month[1] # Minimum date

## [1] 1956-01  
## 476 Levels: 1956-01 1956-02 1956-03 1956-04 1956-05 1956-06 1956-07 ... 1995-08

data$Month[fullDataLength] # Maximum date

## [1] 1995-08  
## 476 Levels: 1956-01 1956-02 1956-03 1956-04 1956-05 1956-06 1956-07 ... 1995-08

full = ts(data$Monthly.beer.production, start=c(1956, 1, 1), end=c(1995, 8, 1), frequency=12)  
ts.plot(full,  
 main="Monthly beer production in India",  
 xlab="Time",  
 ylab="Production")



Through the above two plots, we can observe that the expected forecasts were lower than the moving average of the actual values. This may suggest one or both of the following:

* Inaccurate model, due to missing deterministic factors or insufficient training data
* Extraneous factors that could not have been reliably modelled

However, we saw from our residual analysis that there is a high likelihood that our model takes most, if not all deterministic factors into account. Hence, we may conclude that while our model may not be extremely accurate for forecasting, the inaccuracy for the given test data could be because of extraneous factors. Most likely, the main requirement may be more training data.