UPPSALA UNIVERSITET

Forecast of Birth Rates in Singapore

1MS014: Analysis of Time Series Project Report

VT2023

Author:

Kyler Cheong

Table of Contents

1. Abstract

This project report I will attempt to use an Autoregressive Integrated Moving Average (ARIMA) model. I will apply log transformation followed by finding the first difference of the transformed data. Using the first difference, I fit it to an ARIMA model and analyze the results and give a summary of the advantages and disadvantages of the ARIMA model.

2. Introduction

When Singapore first gained independence in 1965, Singapore was a poor, developing country. The fertility rates were very high, at about 5.75 children per female. This means that on average, each family have about five to six children. This was attributed to poor family planning and generally lack of education.

Higher fertility rates means that a larger population which means that the already scarce resources were divided among a larger population. This translates to less education, less food and water and a lower quality of life in each Singaporean. These drove the government to implement policies to reduce the fertility rates.

These policies are extremely effective and within a short span of 25 years, Singapore's fertility rates dipped to an ideal rate of two children per female. However, in the following years till today, Singapore struggled with a much bigger problem – low fertility rates.

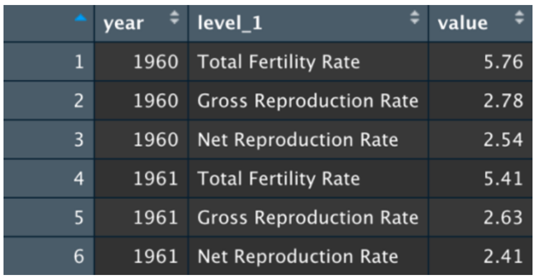
Low fertility rate is a common problem among developed countries like Singapore, Hong Kong and South Korea. Problems such as less young talent in the gene pool and heavier economic burden on the working population to take care of the elderly leads to serious economic consequences.

Hence, I wish to use time series analysis to analyze any trends in the fertility rates in Singapore and attempt to forecast the population growth in Singapore.

3. Data

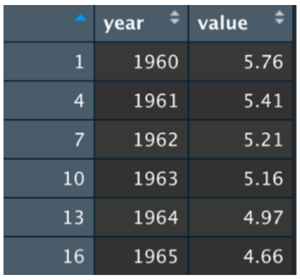
3.1. About the Data

The dataset contains information about the live births occurring within Singapore and its territorial waters as registered under the Registration of Births and Deaths Act. (data.gov) The data contains three columns and 177 rows. There are three types of fertility rates measured here, but for the purpose of this statistical analysis, I will focus only on one type of fertility rate – "Total Fertility Rate". "Total Fertility Rate" refers to, on average, per female how many children she has. This refers to at least one parent who is a resident of Singapore or permanent resident. Below is an example of the observations of the dataset.



	year	level_1	value
1	1960	Total Fertility Rate	5.76
2	1960	Gross Reproduction Rate	2.78
3	1960	Net Reproduction Rate	2.54
4	1961	Total Fertility Rate	5.41
5	1961	Gross Reproduction Rate	2.63
6	1961	Net Reproduction Rate	2.41

Figure 1. Example of an observation.



	year	value
1	1960	5.76
4	1961	5.41
7	1962	5.21
10	1963	5.16
13	1964	4.97
16	1965	4.66

Figure 2. Example of observation used in model.

3.2. Stationarity of raw data

This raw dataset, shows an obvious decaying curve. See Figure 3. The mean and variance of this time series is not constant. This can be seen easily by taking the mean and variance of 2 time periods 1960 to 1970 and 2000 to 2010 and it is easy to observe that it is not constant.

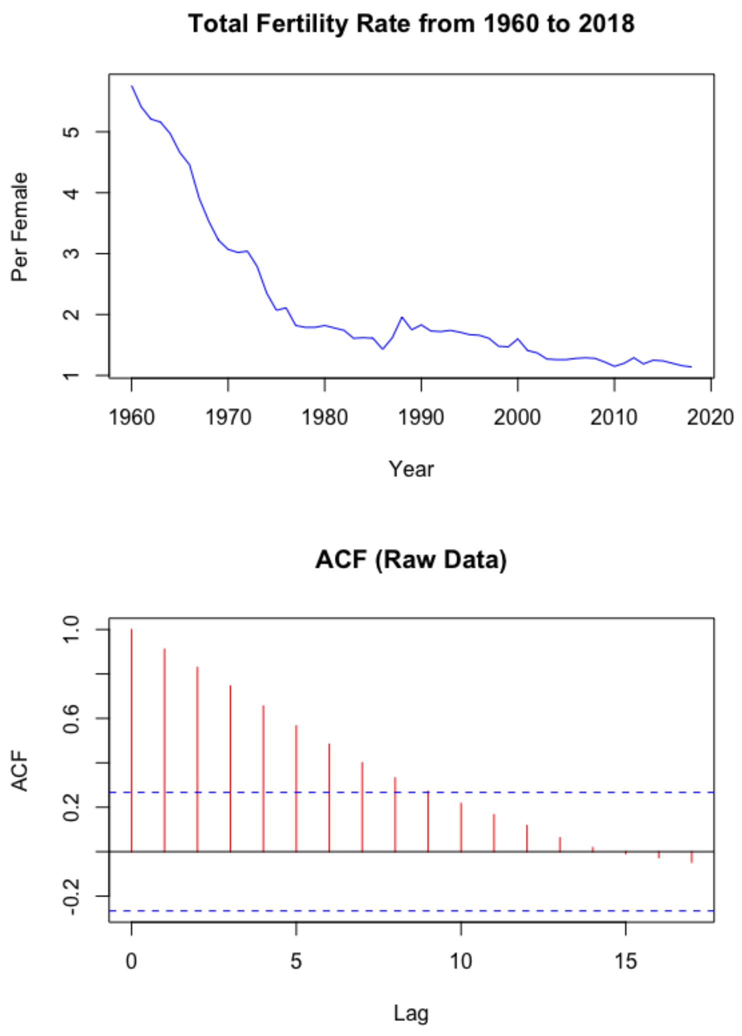


Figure 3. Top, Fertility Rates. Bottom, ACF of raw data.

From observing the ACF of the raw data, the slow decay of the ACF provides evidence that differencing might help. One might assume this is a simple case to make a time series stationary, where detrending the data will suffice or taking first difference will be more than sufficient. However, this is not the case, the statistical stationarity test of the first difference and the detrended data lead to high p-values being obtained, which suggests that the time series is not stationary. See Figure 4.

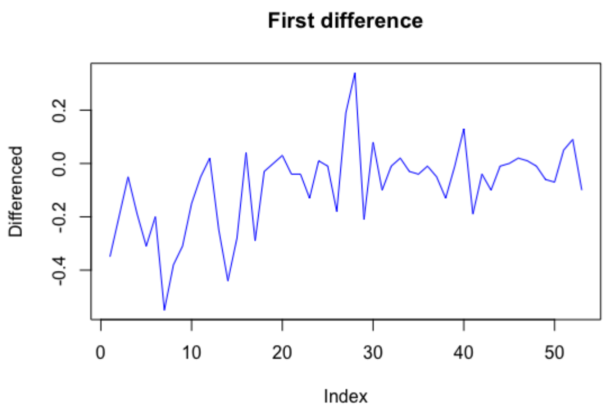
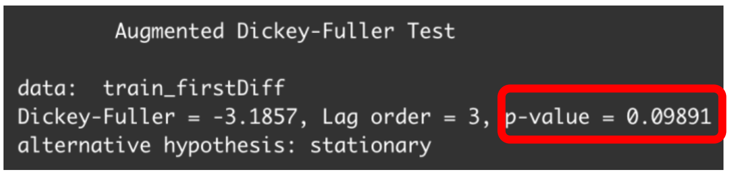


Figure 4. First difference of raw data (non-stationary)

Here, the plot shows that the mean of the first difference has a slightly increasing trend. In addition, the variance of the first 30 indexes is significantly different from that of index 30 to 50. This suggests that this time series might not be stationary. This visual observation can be supported by conducting a statistical test. The Augmented Dickey-Fuller Test on the first differenced time series has a p-value of 0.09891, higher than the p-value of 0.05. This suggests that the first difference time series is not stationary. See Figure 5.



```
        Augmented Dickey-Fuller Test

data:  train_firstDiff
Dickey-Fuller = -3.1857, Lag order = 3, p-value = 0.09891
alternative hypothesis: stationary
```

Figure 5. p-value suggests non-stationarity of time series.

3.3. Data Transformation

Non-stationarity of this time series can be reduced by taking a second difference. This will lead to a p-value of less than 0.01. However, for the simplicity and easy comprehension of the analysis and computation, I did not use the second difference. Instead, I performed a log transformation the original data, to smooth the time series. See Figure 6. From there, I take the first difference of the transformed time series and obtained a stationary time series. See Figure 7.

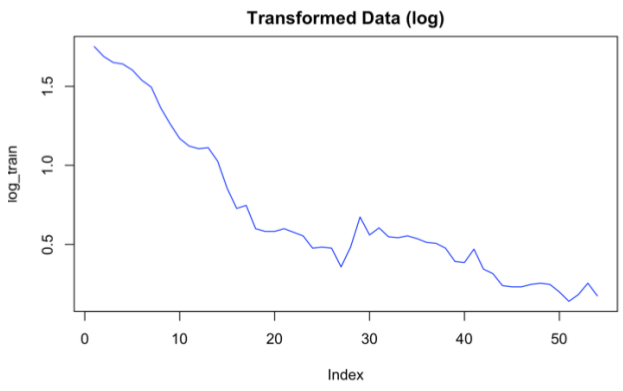


Figure 6. Plot of Transformed Data.

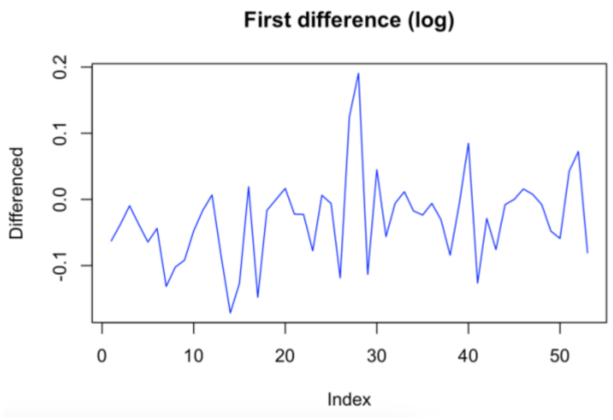
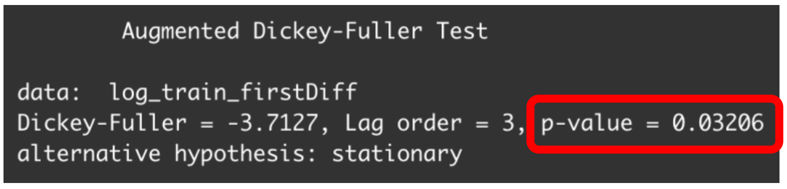


Figure 7. First Difference of log data.

Visual observation of the first difference of the transformed data has a fairly constant mean but not constant variance. However, the Augmented Dickey-Fuller Test, shows that the time series, is statistically stationary. The Augmented Dickey-Fuller Test has a p-value of 0.0326, smaller than 0.05. This suggests that the first difference of the transformed data is stationary. See Figure 8. Hence, I decide to use this time series for further model evaluation.



```
        Augmented Dickey-Fuller Test

data:  log_train_firstDiff
Dickey-Fuller = -3.7127, Lag order = 3, p-value = 0.03206
alternative hypothesis: stationary
```

Figure 8. Stationary Time series

3.4. Train and Test Sets

I will be using the log transformed data as the new time series that I will build my model on. I split the dataset into training and testing set. The training set consists of approximately 90% of the data and the remaining 10% of the data is testing data. The splitting of the data is done by taking the first 54 years as training data and the last 5 years as testing data. All the above analysis done in this report was using the training data. The testing data is purely for validation purposes.

4. Model

4.1. Overview

I decided to use an ARIMA time series model for this project. I chose this time series model because this is the most suitable time series model that fits the context of my model. I did not use a SARIMA model because the data is generated yearly, thus, there was no observable kind of seasonality that happens on a regular basis. If there is no seasonal component to it, there is no reason to use a SARIMA model.

4.2. ARIMA Model

4.2.1. Motivation

Autoregressive Integrated Moving Average Model (ARIMA Model) is a popular statistical time series model. It is based on the idea that the future values of the time series can be predicted using the past values and past white noises/ error terms in the data. It consists of three components the Autoregressive (AR) component, the Integrated (I) component and the Moving Average (MA) component.

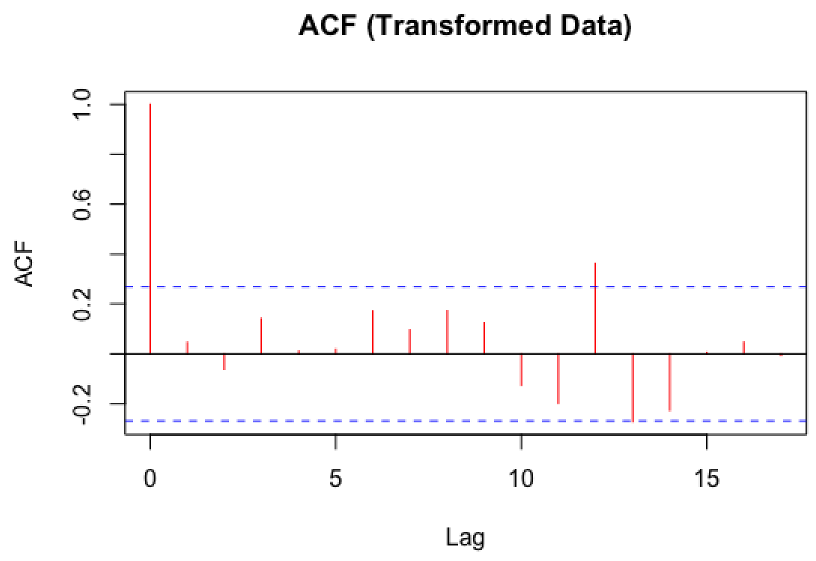
The AR component, of order p, is built on the idea that the future values can be predicted using past values of its own time series and hence the 'autoregressive' nature of the time series model. A p value of 3 means that each future values can be explained as a function of the three past values combined. The I component, of order d, simply means the number of differences it takes to make the time series stationary. A d value of 1 means that first difference is used, etc. The MA component, of order q, is built on the idea that each future values can be predicted using the past white noises/ error terms. Hence, a q value of 3 means that each future value can be predicted using a function of the 3 past white noises/ error terms.

Together, using these three components make up the ARIMA model that is used to predict future values proved to be a powerful statistical tool that can be used for many datasets.

4.2.2. ACF and PACF plots

The PACF and ACF plots give insights on the order values, p, d, and q. Here, we look at the ACF and PACF plots. See Figure 9. Observing the plots of the ACF and the PACF provides a starting point for the orders of the ARIMA model.

Observing Figure 9., the ACF can be seen tailing off, which could possibly suggest that it might be an MA process. To determine a good starting point for the order q, since it is the number of lags that at which the ACF value first dips into the threshold, then for this case, it is obvious that $q = 0$. Looking at the PACF, it is quite difficult to tell from the start what is the order of p. However, most of the values are within the threshold, except for at lag 12 and lag 13. This trend in the data is different from what we normally would expect. But this could be explained just simply as noise in the data or insufficient data points. The data only consists of 54 observations, which might not be sufficient for the model to capture any significant trends and any deviation might cause these spikes in the PACF plots. Hence, I assume that the p is of order 0 by ignoring the spikes at lag 12 and 13 in the PACF. Since I am using the first difference in this time series, the order $d = 1$.



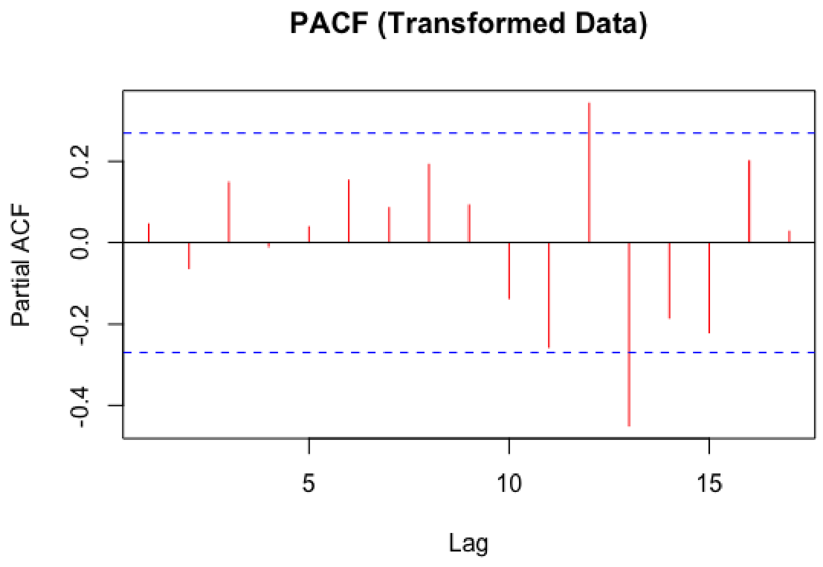
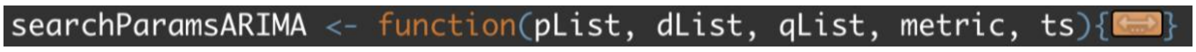


Figure 9. Top: ACF, Bottom: PACF.

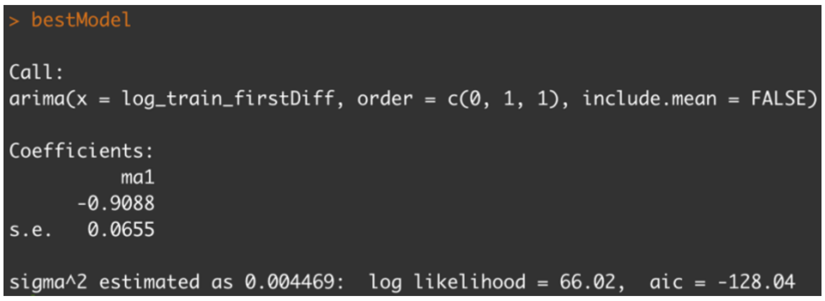
4.2.3. Hyperparameter Tuning

Once the starting points of the orders of the ARIMA model has been selected, in this case, $p = 0$, $d = 1$, $q = 0$, I can start training the model by using 2 different scoring metrics that are commonly used in statistical analysis that measures the performance of the models. I used Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC). Generally, BIC is known to penalize models with a larger number of parameters. The larger the number of parameters, the heavier the penalty as compared to AIC. However, since we are using a small number of parameters in this project, if BIC metric decides that a model is better due to smaller information loss in the model as compared to the AIC metric, then I will cross check using the AIC metric and give more weight to the parameters selected by using AIC as metric. Later in Section 4.2.4, we see that this cross checking will not matter because the different metrics select the same model.

I designed a simple implementation of searching for ideal parameters (See Figure 10). It is a function that takes in, as vector inputs the possible *p*, *d*, *q*, values, a metric (either BIC/ AIC) and the targeted time series, ts. This function will iterate through all possible combinations of the lists given using a simple nested for loop, calculate the best parameters given the metric and prints the best parameters as well as the metric score. A coefficient for the best model is also added. For full implementation of the code, see Section 7, Code.



```
searchParamsARIMA <- function(pList, dList, qList, metric, ts){
```

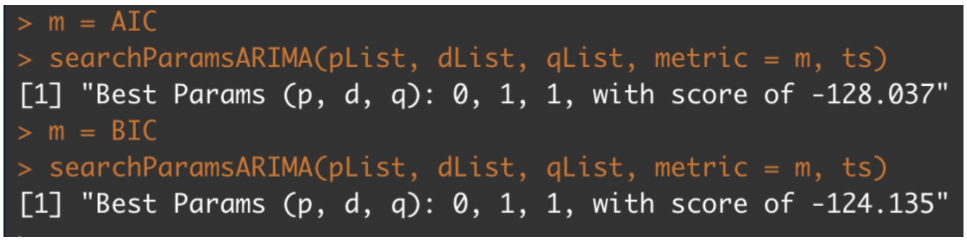


```
> bestModel

Call:
arima(x = log_train_firstDiff, order = c(0, 1, 1), include.mean = FALSE)

Coefficients:
          ma1
      -0.9088
s.e.   0.0655

sigma^2 estimated as 0.004469:  log likelihood = 66.02,  aic = -128.04
```

Figure 10. searchParamsARIMA() function (Top), Coefficients of bestModel (bottom)

4.2.4. Selected ARIMA Model

By iterating through many possible parameters around $p = 0$, $d = 1$, $q = 0$, the best parameters with the lowest possible AIC/ BIC score are when $p = 0$, $d = 1$, $q = 1$. With an AIC and BIC score of -128.037 and -124.135. The selected ARIMA model is ARIMA (0, 1, 1).



```
> m = AIC
> searchParamsARIMA(pList, dList, qList, metric = m, ts)
[1] "Best Params (p, d, q): 0, 1, 1, with score of -128.037"
> m = BIC
> searchParamsARIMA(pList, dList, qList, metric = m, ts)
[1] "Best Params (p, d, q): 0, 1, 1, with score of -124.135"
```

Figure 11. Output showing best parameters for the ARIMA model.

Evaluating the model performance based on the residual diagnostics will show that ARIMA (0, 1, 1) model is the most suitable for this case. See Figure 12 and 13. From Figure 13., we can see the residual diagnostics show that the mean of the residuals is fairly constant across time, the variance of the residuals is also fairly constant except for one outlier around the 28th residual, there is one abnormal spike in the variance and therefore, the residual variance can be treated as constant. ACF of the residuals are not clearly seen in Figure 12., hence we use Figure 13 to evaluate the ACF. It shows that there is little to no autocorrelation among the residuals as all the autocorrelations among the residuals are well within the threshold (blue lines). A more concrete way to determine if the residuals are significantly different from that of white noises/ error terms is using the Ljung Box Statistic. The p-values are well above the threshold of 5% significance. These high p-values indicate that the residuals of the model are no different from white noises/ error terms. This further proves that the ARIMA (0, 1, 1) is a suitable model.

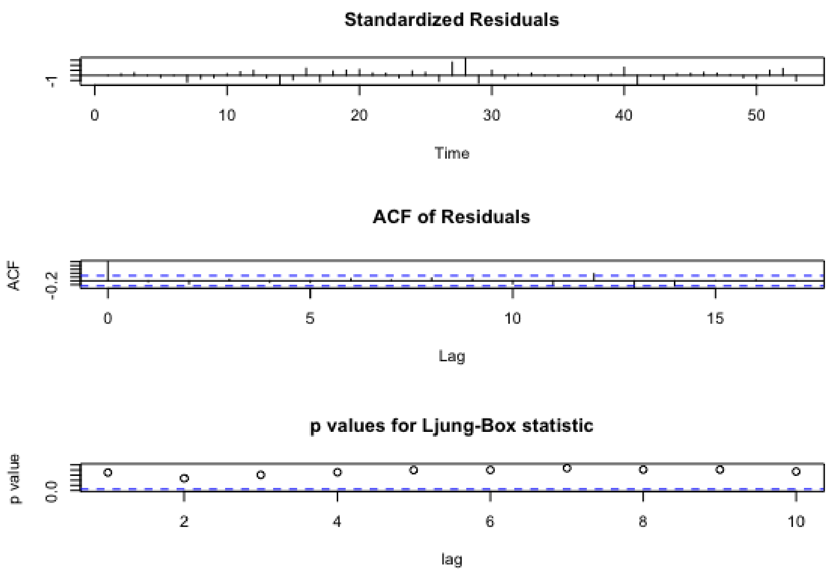


Figure 12. Residual Diagnostics for ARIMA (0, 1, 1).

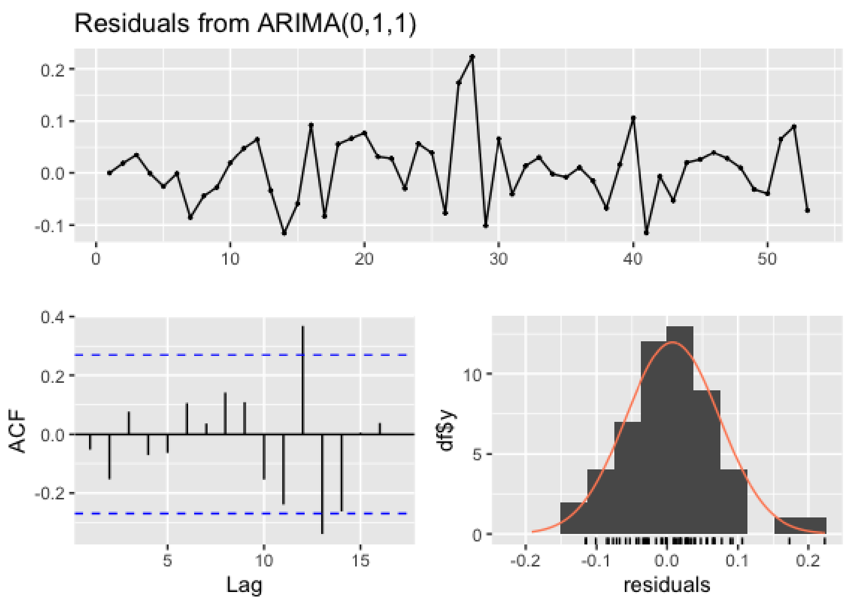


Figure 13. Alternative Residual Diagnostics for ARIMA (0, 1, 1).

By visual observation of the histogram of the residuals and the Q-Q plot of normality for the residuals show that the residuals are normally distributed. This can be seen in Figure 13 that the histogram generally follows a normal distribution, and the Q-Q plot of the residuals is almost a straight line (See Figure 14.), suggesting little to no deviation from a normal distribution.

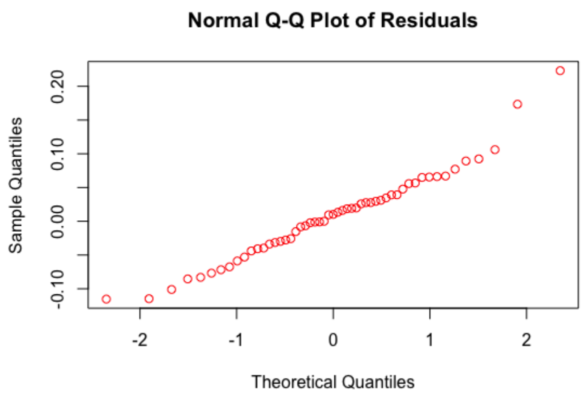


Figure 14. Q-Q plot of Residuals.

All residual diagnostics of the model suggests that the residuals are not correlated to each other (ACF and Ljung-Box Statistic), the residuals have constant mean and variance (Residual Plot and Standardized Residual Plot), and the residuals are normally distributed (Histogram and Q-Q plot). These suggests that the model is capturing all the information and there is no bias in the model. Thus, ARIMA (0, 1, 1) is a well-fitted model.

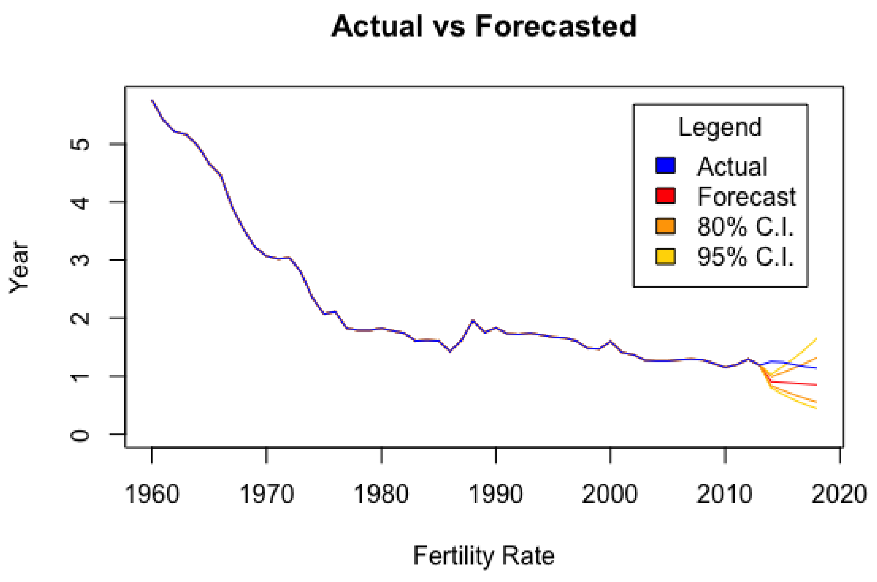
4.3. White Noise Model

The initial guess was supposed to be ARIMA(0, 1, 0) but it seems like ARIMA(0, 1, 1) was more suitable. Here, I attempt to compare it with the white noise model, ARIMA(0, 0, 0) and see if there are any significant differences.

5. Forecasting

5.1. ARIMA Model

Forecasting was done on the last 5 years in this dataset. Specially from 2014 to 2018. See Figure 15. Figure 15. shows the actual values (blue line), followed by the forecasted values (red line) and the 80% and 95% confidence intervals in which the values are predicted to fall within (yellow and orange lines respectively).



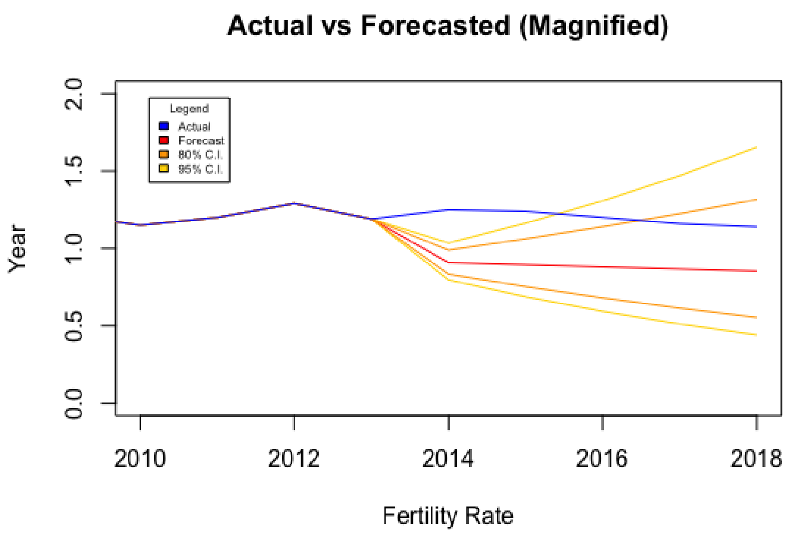


Figure 15. Top: Original Forecast. Bottom: Magnified Forecast.

The forecast plot shows that the model accurately predicts the fertility rates from 2016 to 2018 (using the 95% confidence interval). Therefore, I believe that this model is valid.

5.2. White Noise Model

White Noise model shows similar trends as that to the ARIMA(0, 1, 1) model. See Figure 16 for the magnified version. White noise model shows that only 2014 was predicted wrongly but the next 4 years are all valid. This means that purely counting how many years ARIMA(0, 0, 0) predicted within its range, it seems like the white noise model is doing better than ARIMA(0, 1, 1).

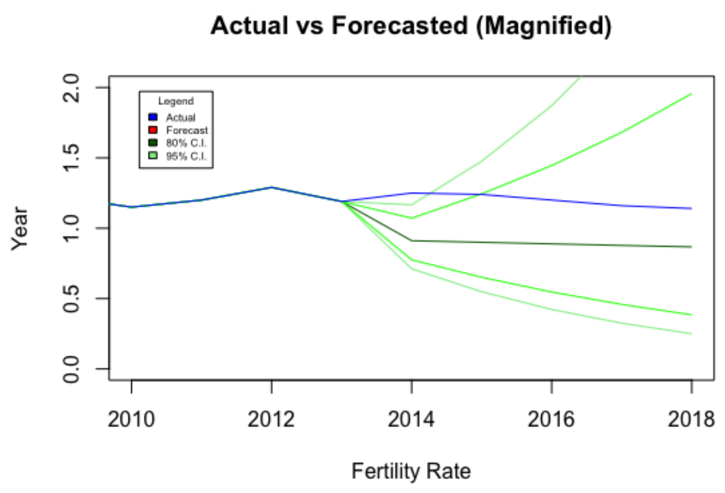


Figure 16. Magnified view of forecasts for white noise model.

5.3. ARIMA Model vs White Noise Model

Comparing both the ARIMA and White Noise Model side by side, it seems as like these 2 models are both good at predicting the forecasts for the next 5 years. See Figure 17.

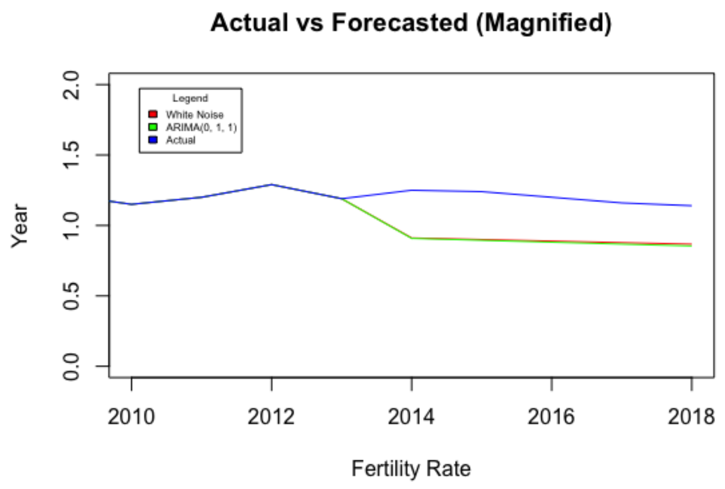


Figure 17. White noise model vs ARIMA(0, 1, 1) model.

Earlier, it seems as though the white noise model is more suitable because of the number of years it predicts correctly (within range). However, the white noise model's range varies a lot more than the ARIMA(0, 1, 1). This suggests that it might not be most ideal because due to its high range of possible values, making an accurate prediction might not be easy. Thus ARIMA(0, 1, 1) will still be preferred.

6. Conclusion

In conclusion, I have explored the possibility of using ARIMA (0, 1, 1) model for time series analysis. The aim of this project is to analyze the trends of Singapore's Fertility Rates and be able to forecast how the fertility rates in the future will change. I believe that the ARIMA (0, 1, 1) model is useful for short-term forecasting but could not be used for long term forecasting.

The advantages of using the ARIMA (0, 1, 1) model is that the model is easy to interpret and understand. For this model specifically, each future values can be predicted using only the error term/ white noise from the previous value. this is a simple and straight-forward model. This model is proven to be suitable from a statistical standpoint. All the residual diagnostics, stationarity tests and model selection based on different metrics indicate that ARIMA (0, 1, 1) model is suitable for this dataset.

However, there are a few disadvantages. Firstly, the fundamental disadvantage of all the time series models lies in the failure to capture the contextual knowledge (e.g. political situation) of the problem. In this case, the time series model failed to capture the fact that Singapore is putting in place many policies aimed to increase the fertility rates in Singapore. This includes policies like cheaper housing options, baby bonuses and longer paternal and maternal leave. These are just a few of the whole slew of policies that have been implement by the Singapore government which unfortunately, could not be captured in this dataset.

In conclusion, although I think that an ARIMA model is valid for this case, I think that if other variables come into play, then this model is not suitable. However, analyzing this time series is a good practice for starters like myself.

7. Code

```
############################
# Code                     #
#    1. EDA                #
#    2. Data Preprocessing #
#    3. AMIRA Model        #
#    4. Forecasting        #
############################

#######
# EDA #
#######

# Read Data
data = read.csv("./data/data.csv")

#install.packages("ggplot2")
library(ggplot2)
ggplot(data, aes(x = year, y = value, group = level_1, color = level_1)) +
  geom_line() +
  labs(title = "Line Plot of Three Categories", x = "Year", y = "Value", color = 'Type')

# Fertility rate
fertilityRate = subset(data, level_1 == "Total Fertility Rate")
fertilityRate = fertilityRate[, c(1,3)]
plot(x = fertilityRate$year, y = fertilityRate$value, type = "l",
     main = "Total Fertility Rate from 1960 to 2018",
     xlab = "Year",
     ylab = "Per Female",
     col = "blue")

# ACF and PACF of raw data
acf(train[, c(2)], main = "ACF (Raw Data)", col = "red")
pacf(train[, c(2)], main = "PACF (Raw Data)", col = "red")

######################
# Data Preprocessing #
######################

# first difference
train_firstDiff = diff(train[, c(2)], difference = 1)
plot(train_firstDiff, type = "l", main = "First difference", ylab = "Differenced", col =
"blue")

# log transformation
log_train = log(train$value)
log_train_firstDiff = diff(log_train, difference = 1)
plot(log_train_firstDiff, type = "l", main = "First difference (log)", ylab = "Differenced",
col = "blue")

# ACF and PACF of first difference (log)
acf(log_train_firstDiff, main = "ACF (Transformed Data)", col = "red")
pacf(log_train_firstDiff, main = "PACF (Transformed Data)", col = "red")

# check stationarity
#install.packages("tseries")
library(tseries)
adf.test(log_train_firstDiff)

###############
# ARIMA Model #
###############

# Function to search for optimal parameters
searchParamsARIMA <- function(pList, dList, qList, metric, ts){
  bestScore <- 0
  bestParams <- c(0,0,0)
  metric = metric
  for (p in pList){
    for (d in dList){
      for (q in qList){
        params = c(p, d, q)
        model = arima(ts, order = params, include.mean = FALSE, method = "ML")
        # change AIC() for different metrics
        score = metric(model)
        if (score < bestScore) {
          bestScore = score
          bestParams = c(p,d,q)
        }
      }
    }
  }
  msg <- sprintf("Best Params (p, d, q): %d, %d, %d, with score of %0.3f", bestParams[1],
bestParams[2], bestParams[3], bestScore)
  print(msg)
}

# search for best parameters
pList = c(0:2)
dList = c(1)
qList = c(0:2)
ts = log_train_firstDiff

# metrics
m = AIC
searchParamsARIMA(pList, dList, qList, metric = m, ts)
m = BIC
searchParamsARIMA(pList, dList, qList, metric = m, ts)

# Model Evaluation
bestModel = arima(log_train_firstDiff, order = c(0,1,1), include.mean = FALSE)

acf(bestModel$residual ,main = "ACF of Residuals", col='red')
pacf(bestModel$residual ,main = "PACF of Residuals", col='red')
tsdiag(bestModel)
checkresiduals(bestModel)
hist(bestModel$residual, main = "Histogram of Residuals", col='orange')
qqnorm(bestModel$residual, main = "Normal Q-Q Plot of Residuals", col='red')

###############
# Forecasting #
###############

# install.packages("forecast")
library(forecast)
forecasts = forecast(bestModel, h = 5)

# Plot forecasts
x1 = fertilityRate$year
y1 = fertilityRate$value

fore0 = exp(tail(log_train_firstDiff, 1) + cumsum(forecasts$mean))
y2 = c(y1[1:length(train$value)], fore0)

fore97.5 = exp(tail(log_train_firstDiff, 1) + cumsum(forecasts$upper[, c(2)]))
fore2.5 = exp(tail(log_train_firstDiff, 1) + cumsum(forecasts$lower[, c(2)]))
y3 = c(y1[1:length(train$value)], fore97.5)
y4 = c(y1[1:length(train$value)], fore2.5)

fore90 = exp(tail(log_train_firstDiff, 1) + cumsum(forecasts$upper[, c(1)]))
fore10 = exp(tail(log_train_firstDiff, 1) + cumsum(forecasts$lower[, c(1)]))
y5 = c(y1[1:length(train$value)], fore90)
y6 = c(y1[1:length(train$value)], fore10)

plot(x1, y2, ylim = c(0,max(y1)), type = "l", col = "red",
     main = "Actual vs Forecasted",
     xlab = "Fertility Rate",
     ylab = "Year")
lines(x1, y3, type = "l", col = "gold")
lines(x1, y4, type = "l", col = "gold")
lines(x1, y5, type = "l", col = "orange")
lines(x1, y6, type = "l", col = "orange")
lines(x1, y1, type = "l", col = "blue")

legend(x = "topright",
       inset = 0.05,
       title = "Legend",
       legend = c("Actual", "Forecast", "80% C.I.", "95% C.I."),
       fill = c("blue", "red", "orange", "gold"))

# plot from 2010 onwards (magnified view)
```

```r
plot(x1, y2, xlim = c(2010, 2018), ylim = c(0,2), type = "l", col = "red",
     main = "Actual vs Forecasted (Magnified)",
     xlab = "Fertility Rate",
     ylab = "Year")
lines(x1, y3, type = "l", col = "gold")
lines(x1, y4, type = "l", col = "gold")
lines(x1, y5, type = "l", col = "orange")
lines(x1, y6, type = "l", col = "orange")
lines(x1, y1, type = "l", col = "blue")

legend(x = "topleft",
       cex = 0.5,
       inset = 0.05,
       title = "Legend",
       legend = c("Actual", "Forecast", "80% C.I.", "95% C.I."),
       fill = c("blue", "red", "orange", "gold"))

#####################
# White Noise Model #
#####################

wn_mean = mean(log_train_firstDiff)
wn_sd = sd(log_train_firstDiff)
wn_model = arima.sim(model = list(order = c(0, 0, 0)), n = 5, mean = wn_mean, sd = wn_sd)
#plot(wn_model)

forecasts_wn = forecast(wn_model, h = 5)

# Plot forecasts
x1 = fertilityRate$year
y1 = fertilityRate$value

fore0 = exp(tail(log_train_firstDiff, 1) + cumsum(forecasts_wn$mean))
y2 = c(y1[1:length(train$value)], fore0)

fore97.5 = exp(tail(log_train_firstDiff, 1) + cumsum(forecasts_wn$upper[, c(2)]))
fore2.5 = exp(tail(log_train_firstDiff, 1) + cumsum(forecasts_wn$lower[, c(2)]))
y3 = c(y1[1:length(train$value)], fore97.5)
y4 = c(y1[1:length(train$value)], fore2.5)

fore90 = exp(tail(log_train_firstDiff, 1) + cumsum(forecasts_wn$upper[, c(1)]))
fore10 = exp(tail(log_train_firstDiff, 1) + cumsum(forecasts_wn$lower[, c(1)]))
y5 = c(y1[1:length(train$value)], fore90)
y6 = c(y1[1:length(train$value)], fore10)

# plot from 2010 onwards (magnified view)
plot(x1, y2, xlim = c(2010, 2018), ylim = c(0,2), type = "l", col = "darkgreen",
     main = "Actual vs Forecasted (Magnified)",
     xlab = "Fertility Rate",
     ylab = "Year")
lines(x1, y3, type = "l", col = "lightgreen")
lines(x1, y4, type = "l", col = "lightgreen")
lines(x1, y5, type = "l", col = "green")
lines(x1, y6, type = "l", col = "green")
lines(x1, y1, type = "l", col = "blue")

legend(x = "topleft",
       cex = 0.5,
       inset = 0.05,
       title = "Legend",
       legend = c("Actual", "Forecast", "80% C.I.", "95% C.I."),
       fill = c("blue", "red", "darkgreen", "lightgreen"))


#################################
# WN compared to ARIMA(0, 1, 1) #
#################################

forecasts_wn = forecast(wn_model, h = 5)
forecasts_arima = forecast(bestModel, h = 5)

# Plot forecasts
x1 = fertilityRate$year
y1 = fertilityRate$value

fore_wn = exp(tail(log_train_firstDiff, 1) + cumsum(forecasts_wn$mean))
y2 = c(y1[1:length(train$value)], fore_wn)
```

```
fore_arima = exp(tail(log_train_firstDiff, 1) + cumsum(forecasts_arima$mean))
y3 = c(y1[1:length(train$value)], fore_arima)

plot(x1, y2, ylim = c(0,max(y1)), type = "l", col = "red",
     main = "Actual vs Forecasted",
     xlab = "Fertility Rate",
     ylab = "Year")
lines(x1, y3, type = "l", col = "green")
lines(x1, y1, type = "l", col = "blue")

legend(x = "topright",
       cex = 0.5,
       inset = 0.05,
       title = "Legend",
       legend = c("White Noise", "ARIMA(0, 1, 1)", "Actual"),
       fill = c("red", "green", "blue"))

# plot magnified forecasts
plot(x1, y2, xlim = c(2010, 2018), ylim = c(0,2), type = "l", col = "red",
     main = "Actual vs Forecasted (Magnified)",
     xlab = "Fertility Rate",
     ylab = "Year")
lines(x1, y3, type = "l", col = "green")
lines(x1, y1, type = "l", col = "blue")

legend(x = "topleft",
       cex = 0.5,
       inset = 0.05,
       title = "Legend",
       legend = c("White Noise", "ARIMA(0, 1, 1)", "Actual"),
       fill = c("red", "green", "blue"))
```