# Forecasting Global Mean Temperature Anomalies

**PSTAT 174W Final Project**

**Nikhita Kalluri**

**UCSB Fall 2024**

## Abstract

This project addresses the impact of global warming by analyzing temperature trends over time on both land and ocean surfaces. Utilizing The Global Land and Ocean-Temperature Anomaly Time Series dataset, which examines anomalies in annual mean temperature from pre-industrial levels, the study aims to forecast future temperature changes. The dataset, sourced from NASA's GISS Surface Temperature Analysis and NOAA National Climatic Data Center, covers the period from 1880 to 2016, although the data used covers from 1905 to 2016, occurring after the pre-industrial so that the data is uniform. The analysis incorporates the assessment of the necessity for time series transformations, such as Box-Cox and log transformations, and incorporates seasonal and trend differencing to achieve stationarity of data. Multiple SARIMA models are fitted for forecasting, with model selection guided by the Akaike Information Criterion (AICc). The chosen model was a tested for normality through the process of diagnostic checking. The chosen model, SARIMA(2,1,2)(1,1,1)12, exhibits invertibility and successfully passes normality tests, qualifying it for forecasting.

It's important to acknowledge that the complexity of the data, influenced by factors like the inherent variability of the climate system, natural disasters, volcanic eruptions, and unpredictable shifts in human contributions to emissions due to technological advances or events such as the COVID-19 pandemic, introduces nuances in the data. Consequently, deviations from the general pattern may occur, as is common with most data. Despite this, the final model slightly deviates from the observations in the actual dataset, although the confidence intervals effectively encompass the values. The chosen final model was:

$$(1 - 0.1619_{(0.5425)}B - 0.2730_{(0.1588)}B^2 - 1.3011_{(0.5740)}B^3 + 0.3244_{(0.5712)}B^4)(1-B)(1-B^{12})Y_t$$

$$= (1 + 1.4288_{(0.1457)}B + 0.3792_{(0.2332)}B^3)Z_t$$

$$sigma^2 = 0.01585233$$

## Introduction

In the face of global warming it is important to examine and understand temperature trends overt time through both land and ocean surfaces. This analysis of trends can allow us to forecast future temperature changes and understand how elements such as greenhouse gases, insolation, and ocean currents can affect temperature.

The specific dataset the will be used in the course of this project is The Global Land and Ocean-Temperature Anamoly Time Series. This examines anomalies (measured in degrees Celsius) in annual mean temperature from the baseline temperature (determined from from pre-industrial global mean temperatures). Based on this information, future changes can be forecasted and determined, important to understanding the scale of global warming and future predictions.

The primary objective of this project is to forecast future global temperature increases concerning the mean baseline temperature, utilizing current data spanning from 1880 to 2016. This data, accessible on datahub.io, is sourced from two entities:

1. GISTEMP, originating from NASA's (GISS) Surface Temperature Analysis, Global Land-Ocean Temperature Index.
2. NOAA National Climatic Data Center (NCDC).

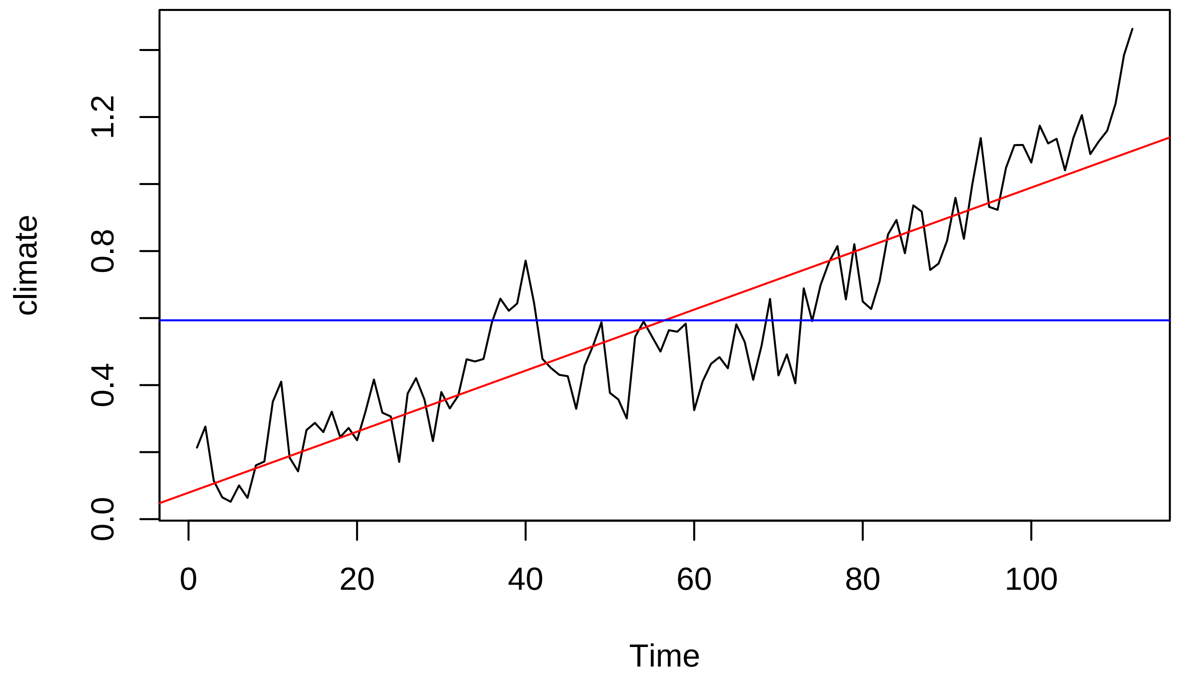
The dataset includes two variables: 'Year,' an integer ranging from 1880 to 2016, and 'Mean,' an integer ranging from -0.5 (indicating a cooling of 0.5 degrees below the pre-industrial global mean temperature) to 1 (indicating a heating of 1 degree Celsius from pre-industrial global mean temperatures).

To forecast future global warming based on past temperature anomalies, the process begins by dividing the data into training and testing sets. After analyzing the training set, the subsequent step involves making the series stationary by transforming the data and eliminating trend and seasonality components. Once the series is stationary, models that align well based on Autocorrelation Functions and Partial Autocorrelation Functions are identified, and the Akaike Information Criterion is employed to determine the most satisfactory model once the process of diagnostic checking shows that the model is satisfactory. Once the model is selected, it can be utilized to forecast future temperature anomaly predictions.

## Read in Data

To begin, the data must be read in as a CSV file. This dataset currently has two mean observations per year. To address this, we can group the data by year and generate a new dataframe with a single mean observation per year. Additionally, the data before the 1900s seems to be flat, not following the linear trend that the rest of the data follows. To accurately forecast the data, I will be working with data only after 1905. It is also importanmt to note that the Means are upshifted by 0.5 because later transformations such as the Box-Cox transformations do not accept negative values. This is in line with pre-industrial temperatures taking 1900 as the baseline for pre-industrial temperature rather than 1860.

```
## # A tibble: 112 x 2
##     Year   Mean
##    <int>  <dbl>
##  1  1905 0.213
##  2  1906 0.276
##  3  1907 0.115
##  4  1908 0.0652
##  5  1909 0.0520
##  6  1910 0.101
##  7  1911 0.0634
##  8  1912 0.161
##  9  1913 0.172
## 10  1914 0.350
## # i 102 more rows
```



Visualizing the time series data as a plot, an upward trend is apparent. Additionally, there is observable seasonality, characterized by cyclic variations in the graph's curvature. Additionally, the variance appears to exhibit only a marginal increase over the range of years. It is important to note the upward peak around the 40th observation which might affect the linear trend.

## Partition Data into Train and Test Sets

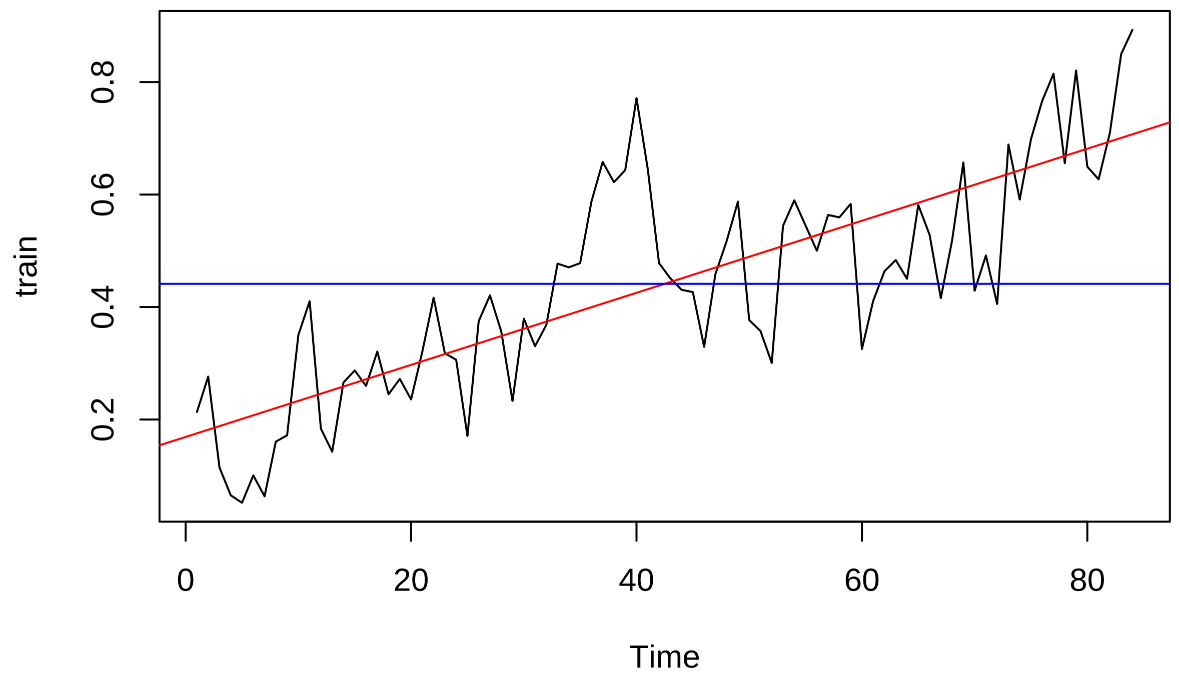
To begin with, we want to create a training and testing data set for the model to be trained on and tested on for performance. The training set will be used to train our models on, while the testing set will serve as new data for our models to be tested on.

I decided to use a 75/30 split, with 75% of the data used for training and 25% of the data used for testing. This will increase the model performance, while still having a decent amount of data left to test on.

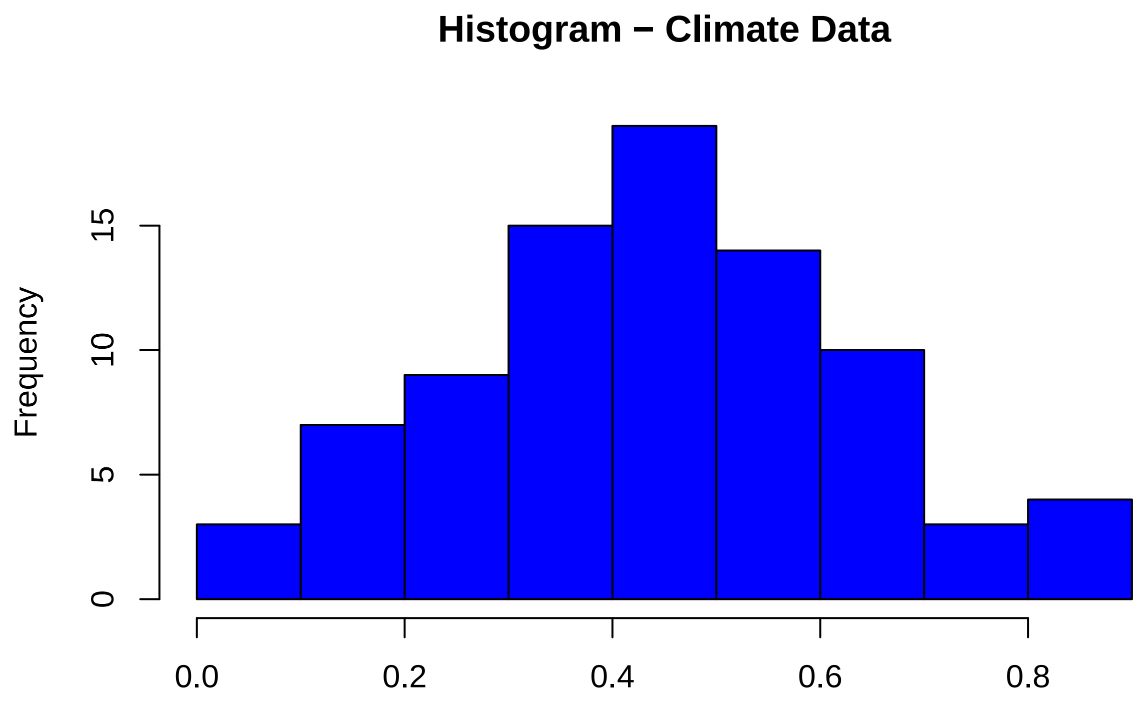
```
# Calculate the index to split the data
split_index <- floor(length(climate) * 0.75)
# Create training and testing sets
train <- window(climate, end = split_index)
test <- window(climate, start = split_index + 1)
```

Now, the training dataset is ready to be explored.

## Data Exploration

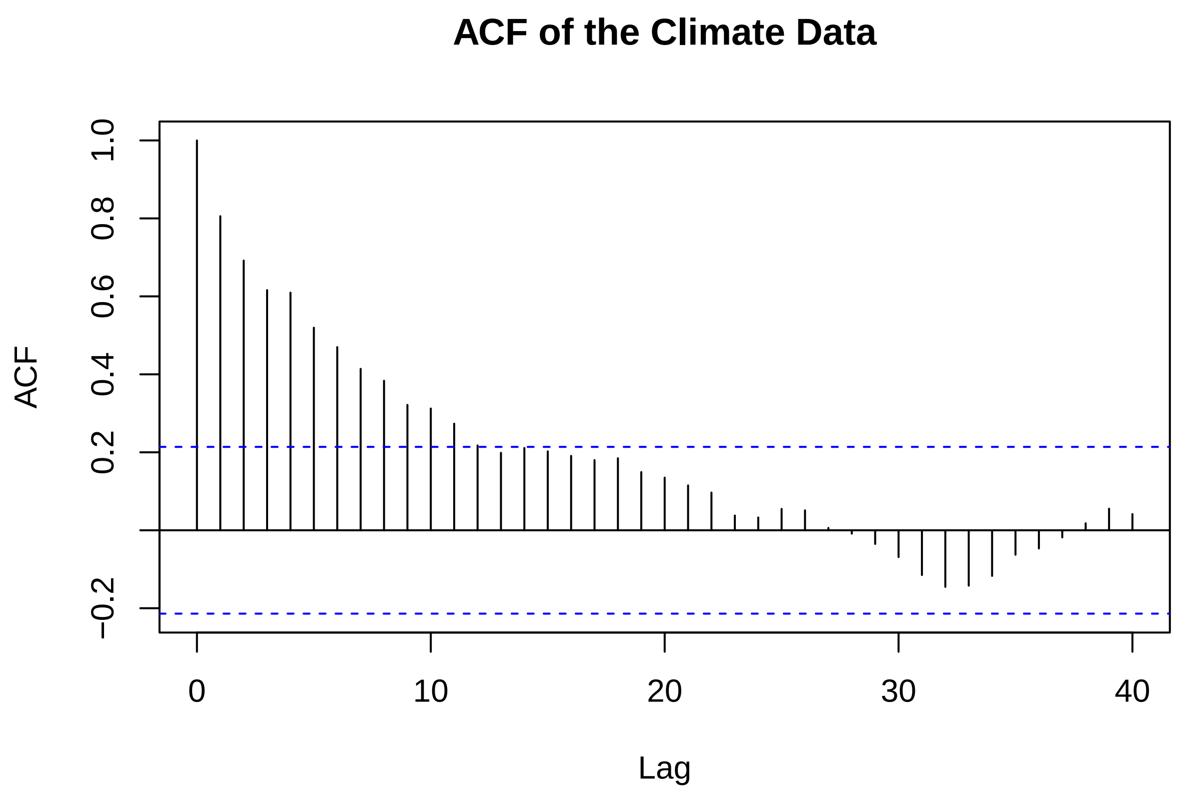


Based on the graph, there is a clear linear upward trend as indicated by the red trend line. Variance seems relatively constant only slightly increasing through the progression of years. A notable feature of the graph is the sharp upward curve in the middle of the graph which might be shifting the trend line upward.



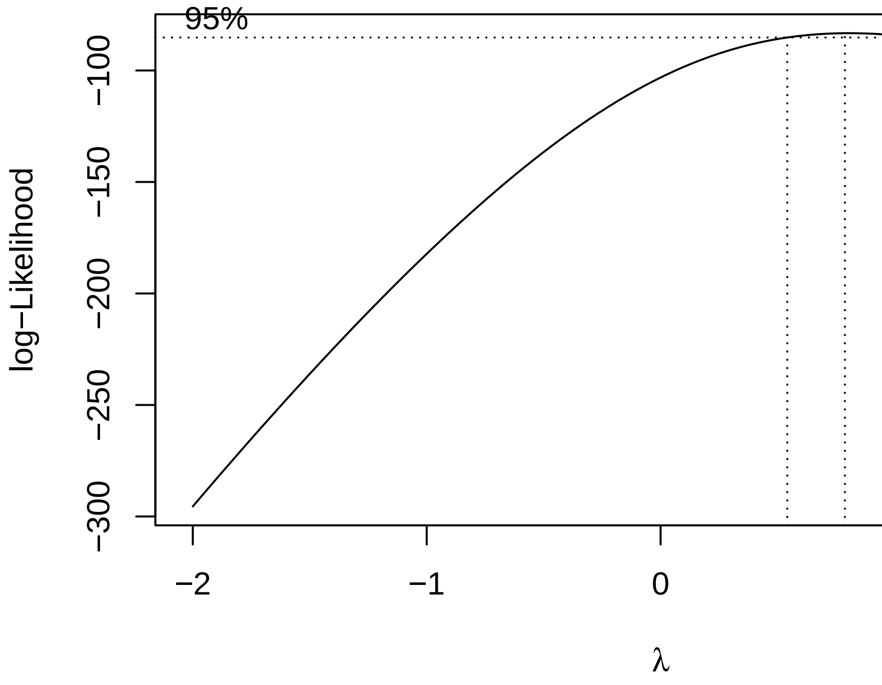
From the histogram, there is a very slight skew of mean observations to the right, suggesting that the data is not entirely Gaussian and may require transformation.

Subsequently, to look at seasonality, we can look at the autocorrelation function of the training set.



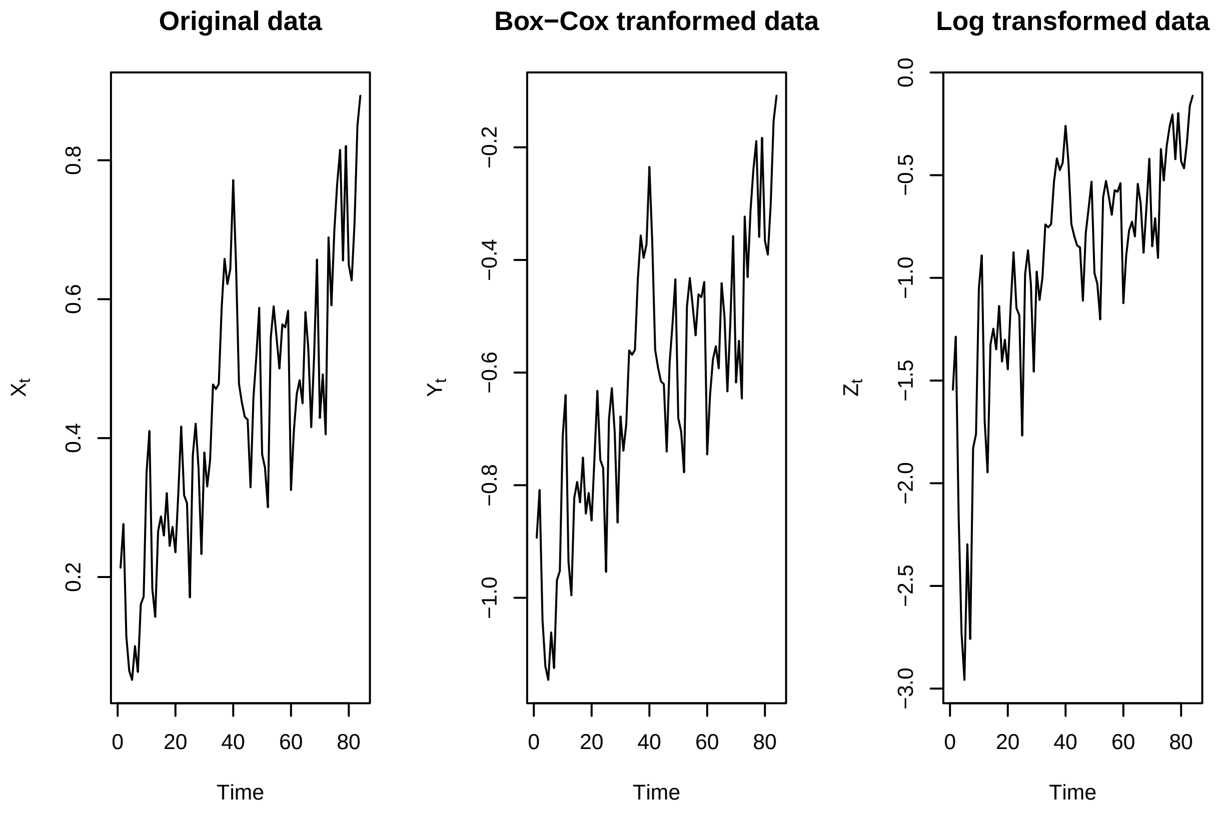
The autocorrelation function exhibits a slow downward trend suggesting that the series is most likely nonstationary. In addition, the ACF is periodic as seen by cyclical curvature, suggesting that there is a seasonal component that needs to be differenced. To make the series stationary, we can explore some transformations.

## Transformations

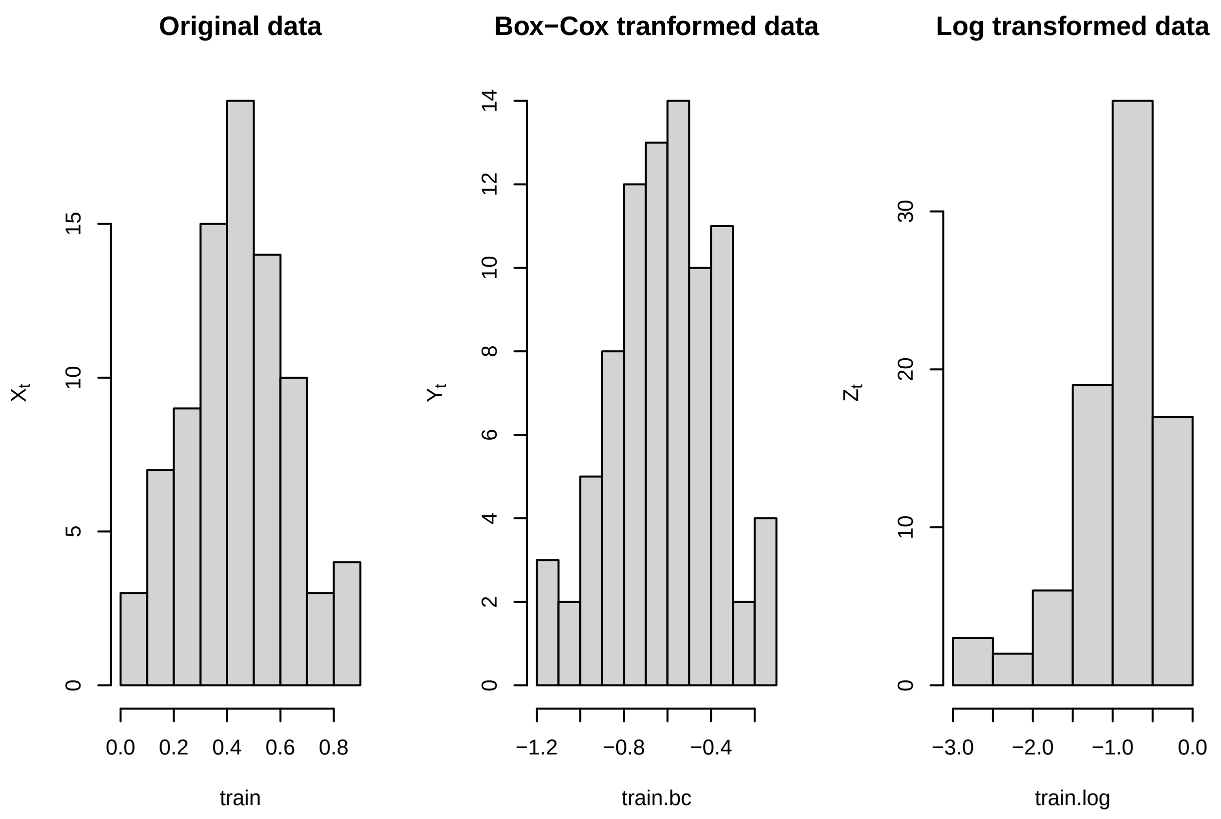


We can start by Box-Cox transforming the data.
The resulting optimal lambda value is 0.5454545. It is important to note that the value 1 lies within the confidence interval, suggesting that a Boc-Cox tansformation is not necessary as will be seen later. Additionally, 0 is within the confidence interval which means that a log transformation is possible. We can test this by taking the log of the training set to log transform the data.

In order to see which model fits the best, I plotted the original data, box-cox transformed data, and log transformed data along with their corresponding histograms.



From the plots, the original data seems to have the least variability with the slowest change in y-values throughout the course of time while the log-transformed data has the highest variance.



Based on the histogram, the original training data has the most normal distribution with most of the values concentrated towards the center and the distribution appearing to be symmetric. The log transformed data has the least normal distribution with a skew to the left and values concentrated to the right.

In order to see if this is the case, we can check the variance of the original data, box-cox transformed data, and log transformed data.

```
# display variance of original data, box-cox transformed data, and log transformed data
var(train)
```

```
## [1] 0.03778555
```

```
var(train.bc)
```

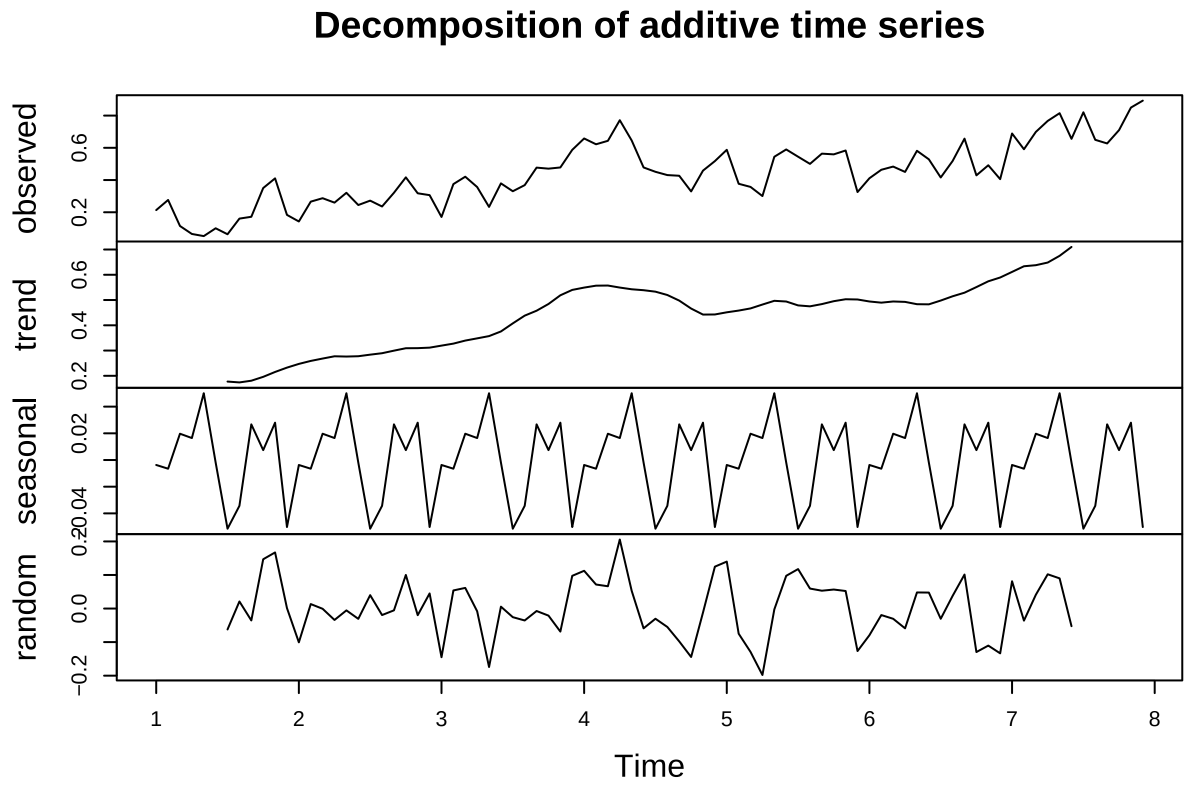
```
## [1] 0.05605993
```

```
var(train.log)
```

```
## [1] 0.3409572
```

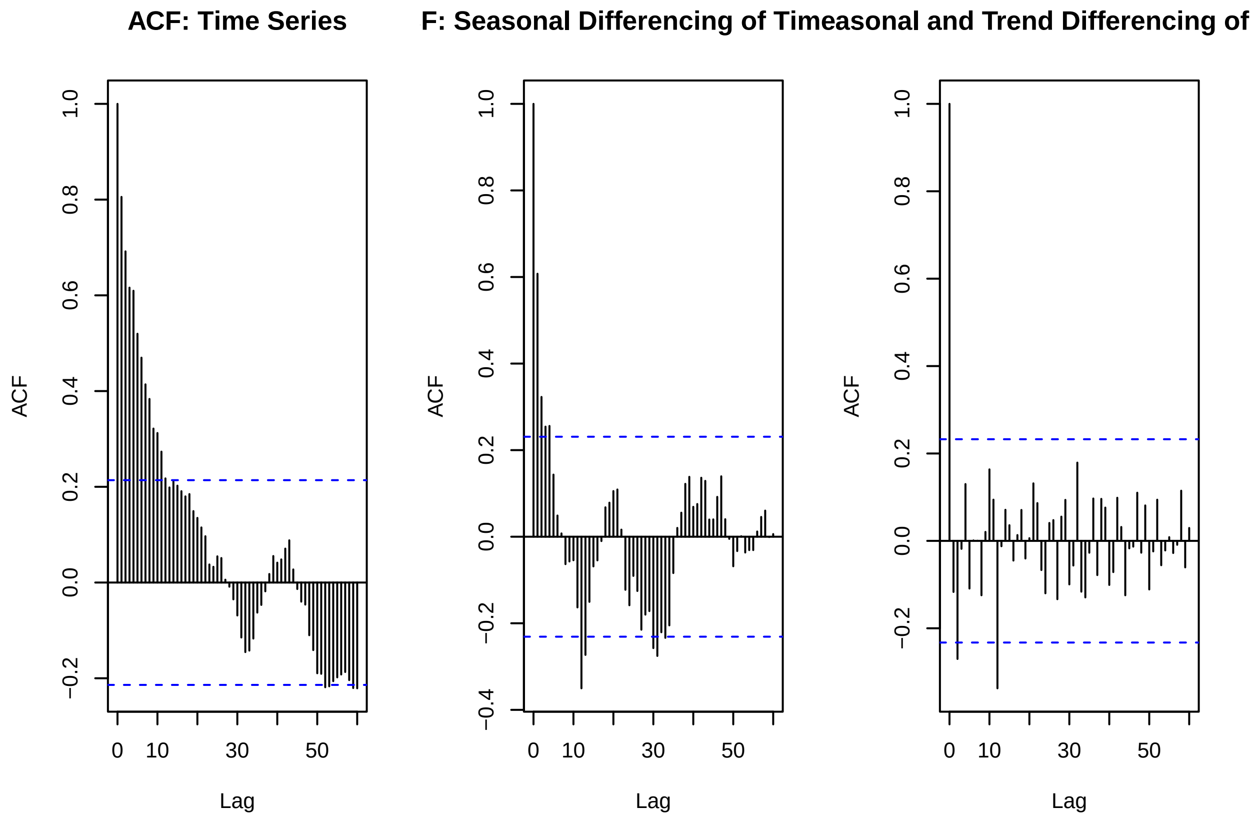
The variance aligns with the observations in the graphs. The training set has the lowest variance, followed by the Box-Cox transformed data, with the log-transformed data exhibiting the highest variance. This suggests that retaining the original data is appropriate, and we can proceed with the differencing process.

Before we move on to the differencing process, it is important to look at the decomposition of the time series to see that it is in line with our observations.



The need for the differencing process is evident through the above decomposition. There is a distinct upward trend and pronounced seasonality. Based on these findings, we can begin differencing the training set. This involves experimenting with both a seasonal difference at lag 12 to eliminate seasonality and a difference at lag 1 to address the linear trend. However, it is important to note that there is a slight shift from linearity in the trend due to an upward shift in data located in the center of the observations. This might potentially impact the accuracy of forecasting with a SARIMA model.

## Analyzing ACF and PACF



Based on the ACF plots above, the ACF of the original time series displays both seasonality with a cyclic period and a downward pattern indicating a trend. It also exhibits a gradual decrease, suggesting nonstationarity. The deseasonalized time series still shows a strong pattern of curvatures indicating trend, along with a gradual decrease in lags below the confidence interval. Employing both seasonal and trend differencing eliminates these components, and there is no slow decay, indicating that the series is stationary.

```
var(train)
```

```
## [1] 0.03778555
```

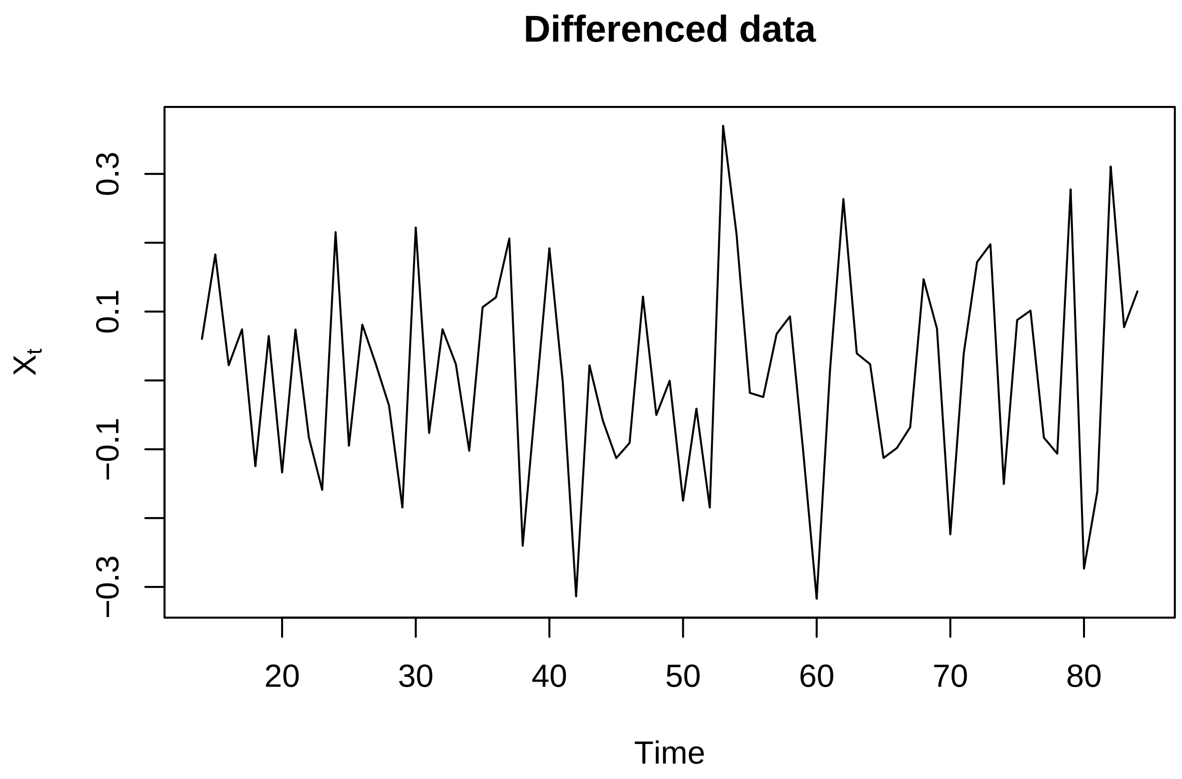
```
var(train_diff_s)
```

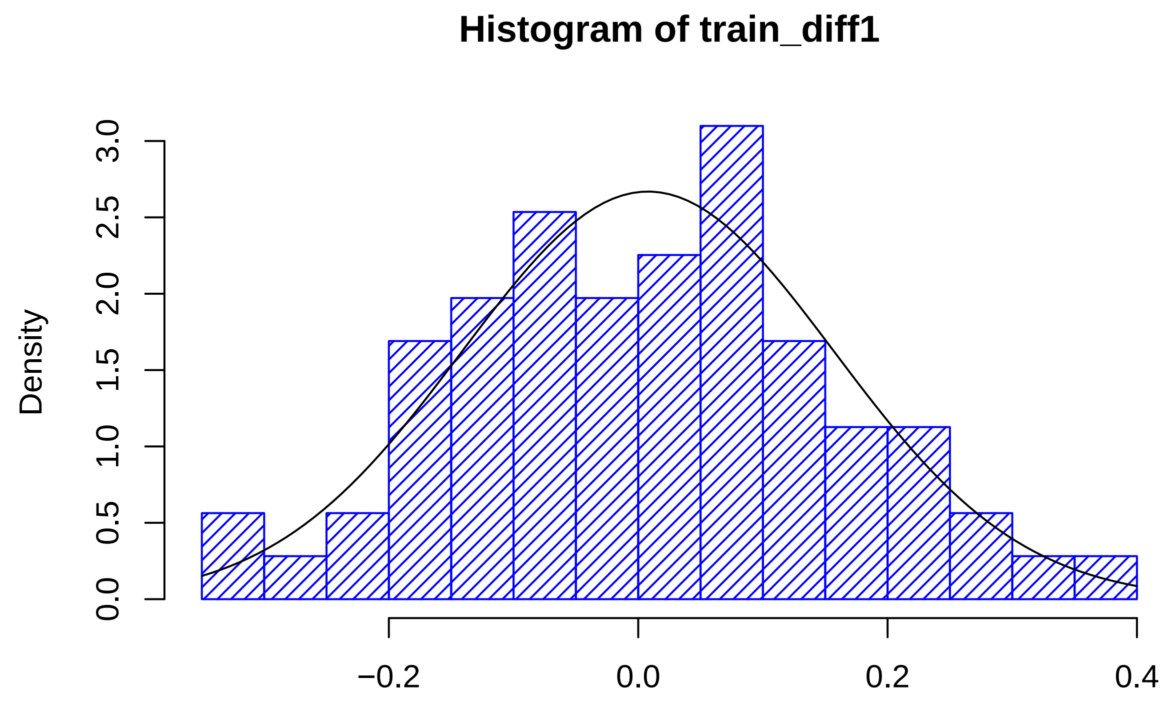
```
## [1] 0.0314217
```

```
var(train_diff1)
```

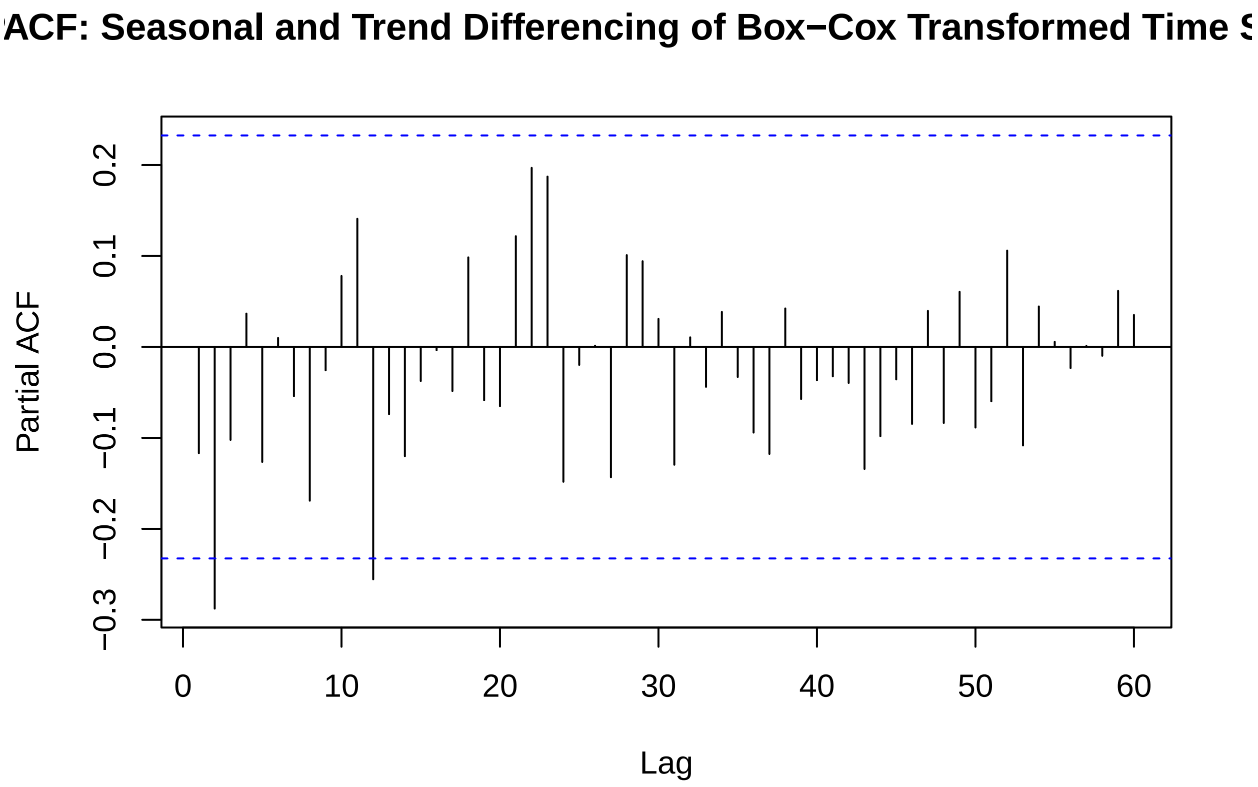
```
## [1] 0.02234191
```

To confirm this, we can examine the variances, which indeed indicate that the series, when both detrended and deseasonalized, exhibits the least variance, making it the most Gaussian and stationary.


Differenced data
$X_t$
Time
0.3
0.1
−0.1
−0.3
20
30
40
50
60
70
80



From the above plots, the differenced data does seem to be random suggesting stationarity and the histogram is normally distributed. Thus, we can choose the series that is differenced at both 1 and 12.

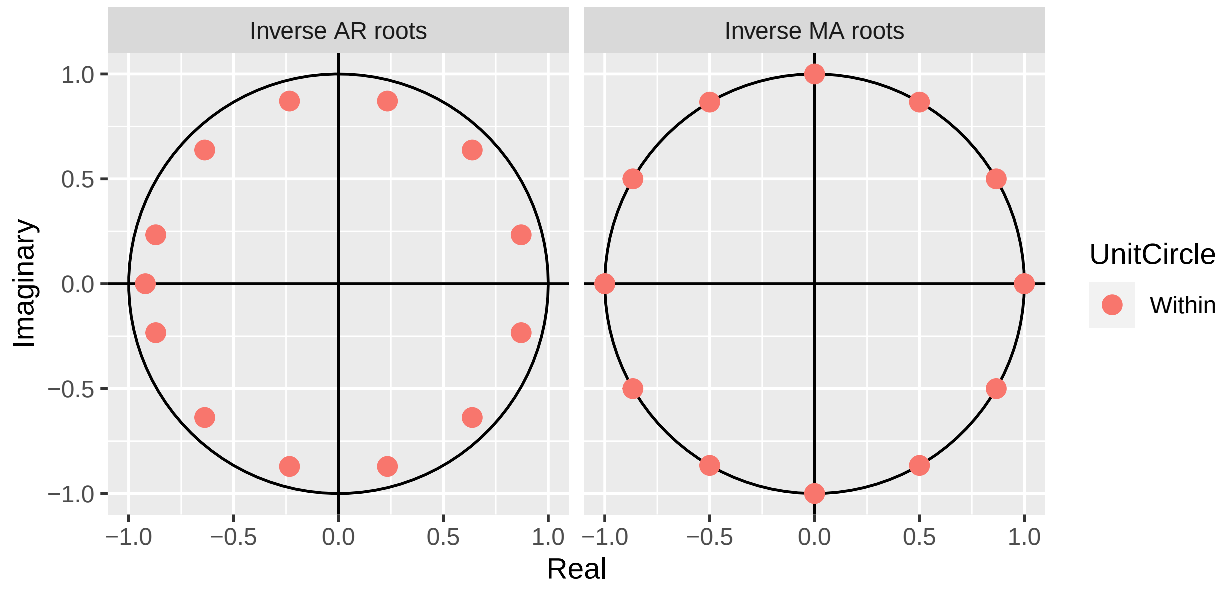


Now, we are ready to begin the process of model selection. From the ACF above, there are 2 MA components outside the confidence interval including either 2 or 3, and 12. This corresponds to the MA component where 2 or 3 is nonseasonal and 12 is seasanal. There are no other multiples of 12 that have significant lags. As a result we can test models with q = 2 or 3, and Q= 1.

Looking at the PACF which corresponds to the AR component, it seems that either at lag 1 or 2 and at lag 12, the corresponding ACF values go outside the confidence interval. This means we can choose p=1 or 2, and P = 1. We differenced 1 time for both seasonlity and trend so choose d = D = 1.

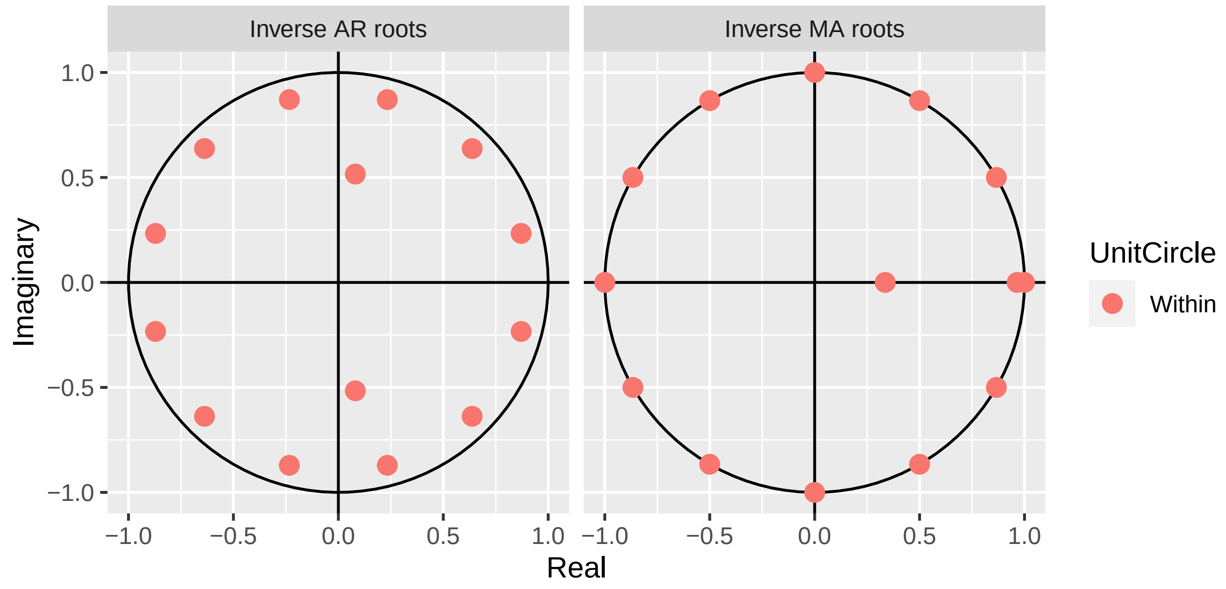
## Model Selection

```
##
## Call:
## arima(x = train_diff1, order = c(1, 1, 2), seasonal = list(order = c(1, 1, 1),
##     period = 12))
##
## Coefficients:
##           ar1      ma1      ma2     sar1     sma1
##       -0.9206  -0.0001  -0.9999  -0.2887  -0.9998
## s.e.   0.0799   0.1641   0.1641   0.1439   0.2513
##
## sigma^2 estimated as 0.02045:  log likelihood = 12.73,  aic = -13.47
##
## Training set error measures:
##                      ME      RMSE        MAE       MPE     MAPE      MASE
## Training set 0.01686115 0.1292559 0.08927081 -84.54658 209.4463 0.4924665
##                      ACF1
## Training set -0.1417163

## [1] -11.82151
```



These are the corresponding result from fitting a SARIMA(1, 1, 2)(1, 1, 1)12 model. The MA results of the autoplot shows that the model is invertible. The AICc is -11.82151.

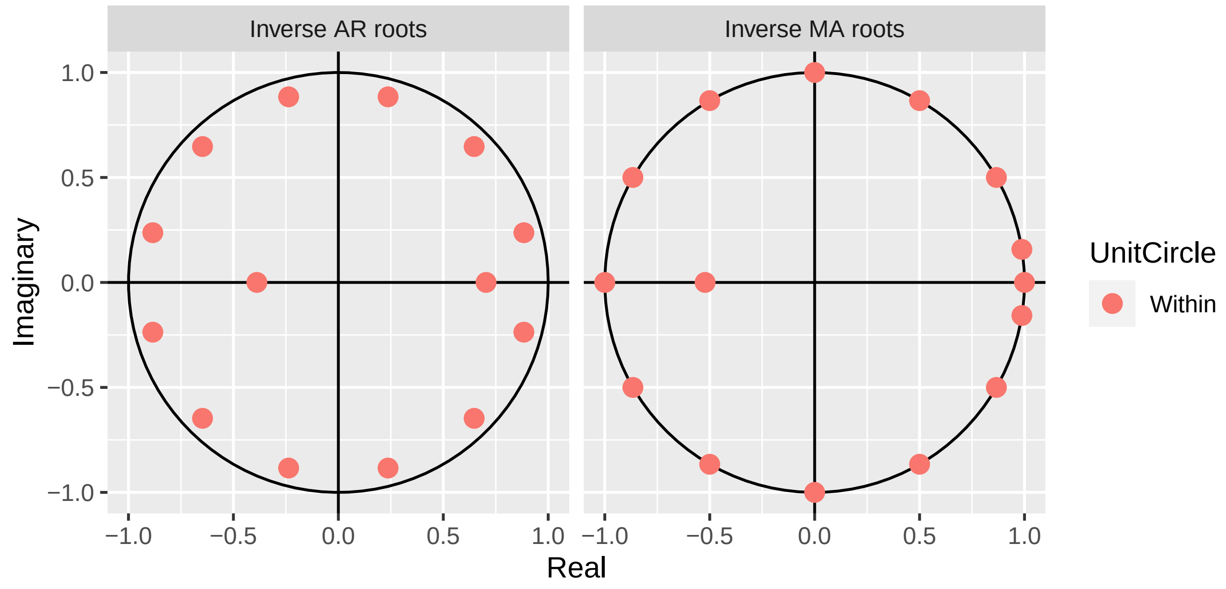
```
##
## Call:
## arima(x = train_diff1, order = c(2, 1, 2), seasonal = list(order = c(1, 1, 1),
##     period = 12))
##
## Coefficients:
##          ar1      ar2      ma1     ma2     sar1     sma1
##       0.1619  -0.2730  -1.3011  0.3244  -0.2893  -0.9999
## s.e.  0.5425   0.1588   0.5739  0.5712   0.1457   0.2332
##
## sigma^2 estimated as 0.01962:  log likelihood = 15.42,  aic = -16.83
##
## Training set error measures:
##                      ME      RMSE        MAE       MPE     MAPE      MASE
## Training set 0.02005199 0.1266141 0.09237646 -180.6916 329.6649 0.5095989
##                     ACF1
## Training set -0.04518867

## [1] -14.59245
```



These are the corresponding result from fitting a SARIMA(2, 1, 2)(1, 1, 1)12 model. The MA results of the autoplot shows that the model is invertible. The AICc is -14.59245.

```
##
## Call:
## arima(x = train_diff1, order = c(2, 1, 3), seasonal = list(order = c(1, 1, 1),
##     period = 12))
##
## Coefficients:
##          ar1     ar2      ma1      ma2     ma3      sar1     sma1
##       0.3158  0.2733  -1.4530  -0.0311  0.5220  -0.3456  -0.9999
## s.e.  0.4814  0.3910   0.4627   0.8870  0.4517   0.1449   0.2413
##
## sigma^2 estimated as 0.01863:  log likelihood = 16.32,  aic = -16.64
##
## Training set error measures:
##                      ME      RMSE        MAE       MPE     MAPE      MASE
## Training set 0.01483763 0.1233756 0.08813802 -24.47164 157.0465 0.4862174
##                     ACF1
## Training set -0.07702827
```

```
## [1] -13.70268
```



These are the corresponding result from fitting a SARIMA(2, 1, 3)(1, 1, 1)12 model. The MA results of the autoplot shows that the model is invertible. The AICc is -13.70268.

All of these models are invertible. However, the AICc is lowest for fit 2, followed by fit3. Looking at the coefficients for Model 2, both the ar1 and ma2 coefficients contain 0 in the confidence intervals. Additionally, for the next best fit- fit3, ar1, ar2, ma2 contain 0 in the confidence intervals. To check for a better model, we can set these coefficients to 0 and check the resulting AICc.

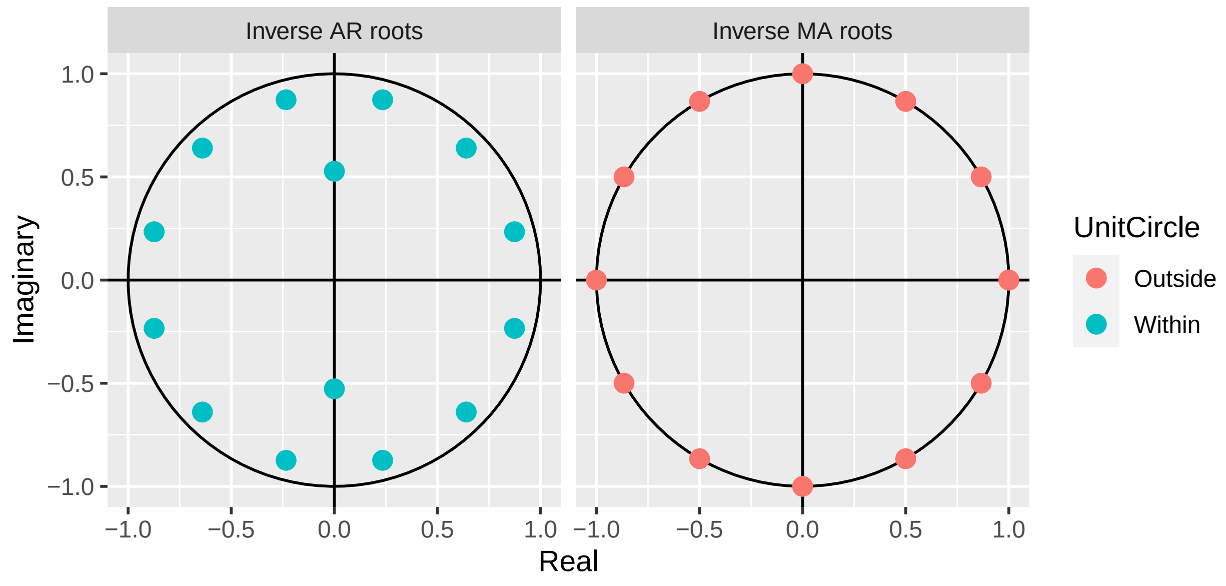
```
modelA <- arima(train_diff1, order = c(2, 1, 2), seasonal = list(order =
c(1, 1, 1), period = 12), fixed = c(0, NA, NA, 0, NA, NA), method="ML")
summary(modelA)
```

```
##
## Call:
## arima(x = train_diff1, order = c(2, 1, 2), seasonal = list(order = c(1, 1, 1),
##     period = 12), fixed = c(0, NA, NA, 0, NA, NA), method = "ML")
##
## Coefficients:
##       ar1      ar2      ma1  ma2     sar1     sma1
##         0  -0.2782  -1.0000    0  -0.3003  -1.0000
## s.e.    0   0.1282   0.1431    0   0.1453   0.2378
##
## sigma^2 estimated as 0.01961:  log likelihood = 14.73,  aic = -19.46
##
## Training set error measures:
##                      ME      RMSE        MAE       MPE     MAPE      MASE
## Training set 0.01849895 0.1265658 0.09065198 -216.6488 353.1686 0.5000858
##                       ACF1
## Training set -0.1785414
```

```
npar <- length(modelA$coef) + 1
nstar <- length(modelA$residuals) - modelA$arma[6]-modelA$arma[7]*modelA$arma[5]
aiccA <- modelA$aic + 2 * npar * (nstar/(nstar - npar - 1) - 1)
aiccA
```

```
## [1] -17.22379
```

```
autoplot(modelA)
```



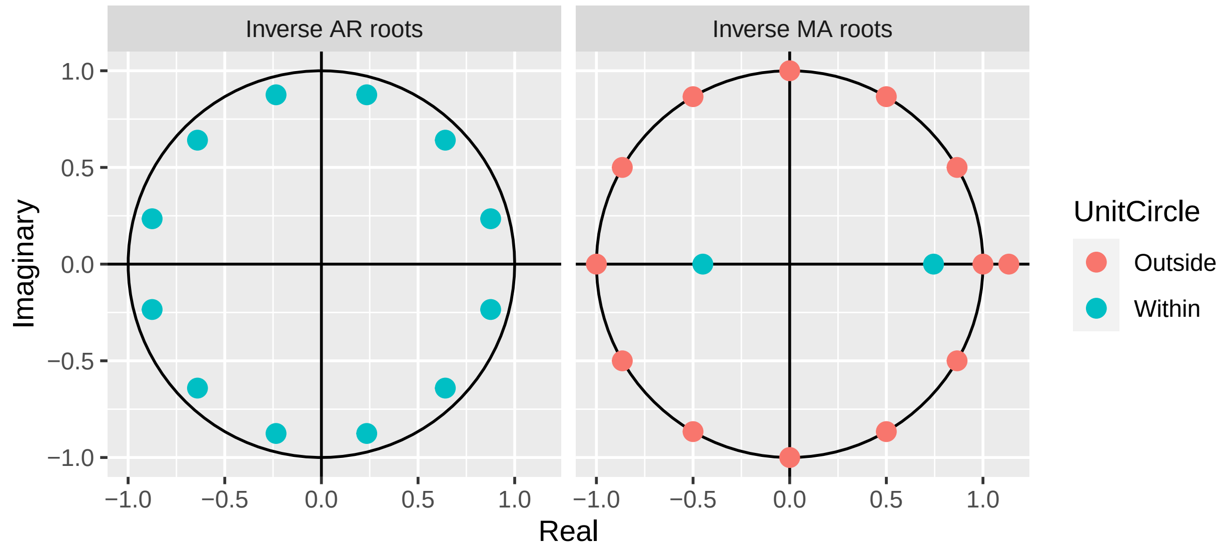
```
modelB <- arima(train_diff1, order = c(2, 1, 3), seasonal = list(order =
c(1, 1, 1), period = 12), fixed = c(0, 0, NA, 0, NA, NA, NA), method="ML")
summary(modelB)
```

```
##
## Call:
## arima(x = train_diff1, order = c(2, 1, 3), seasonal = list(order = c(1, 1, 1),
##     period = 12), fixed = c(0, 0, NA, 0, NA, NA, NA), method = "ML")
##
## Coefficients:
##       ar1  ar2      ma1  ma2     ma3     sar1     sma1
##         0    0  -1.4288    0  0.3792  -0.3085  -1.0000
## s.e.    0    0   0.5825    0  0.5203   0.1508   0.2344
##
## sigma^2 estimated as 0.01551:  log likelihood = 15.37,  aic = -20.74
##
## Training set error measures:
##                      ME      RMSE        MAE       MPE     MAPE     MASE
## Training set 0.01687239 0.1125551 0.08171925 -149.5435 271.6447 0.450808
##                     ACF1
## Training set 0.001820772
```

```
npar <- length(modelB$coef) + 1
nstar <- length(modelB$residuals) - modelB$arma[6]-modelB$arma[7]*modelB$arma[5]
aiccB <- modelB$aic + 2 * npar * (nstar/(nstar - npar - 1) - 1)
aiccB
```

```
## [1] -17.80436
```

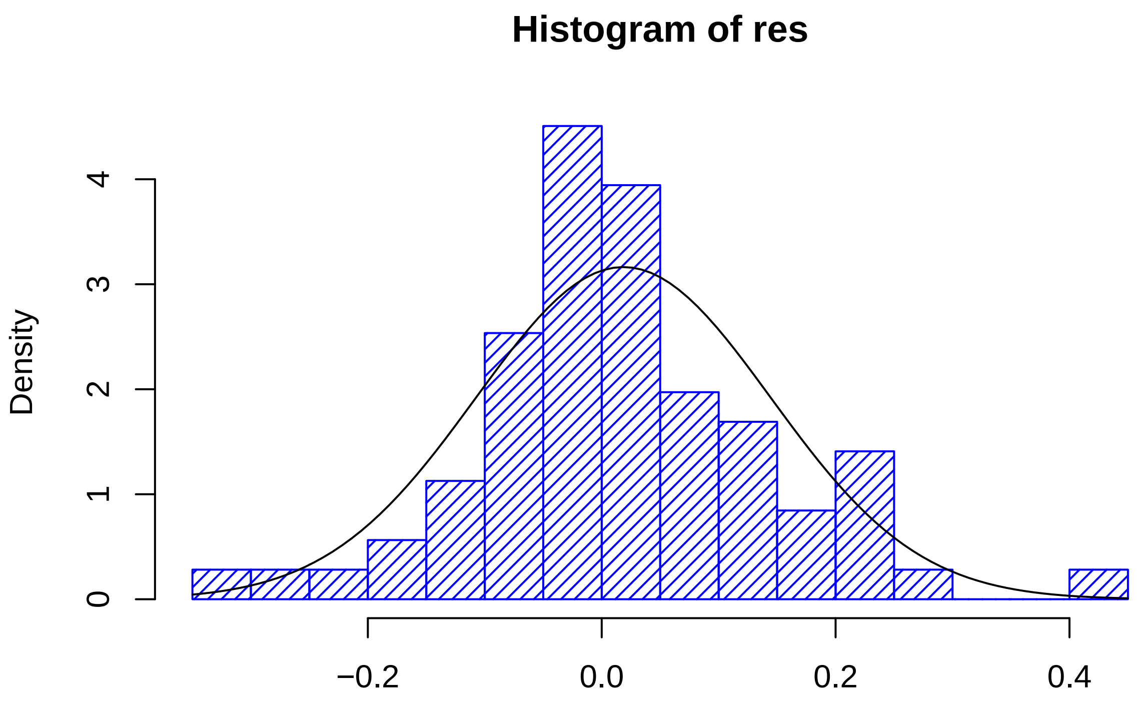
```
autoplot(modelB)
```

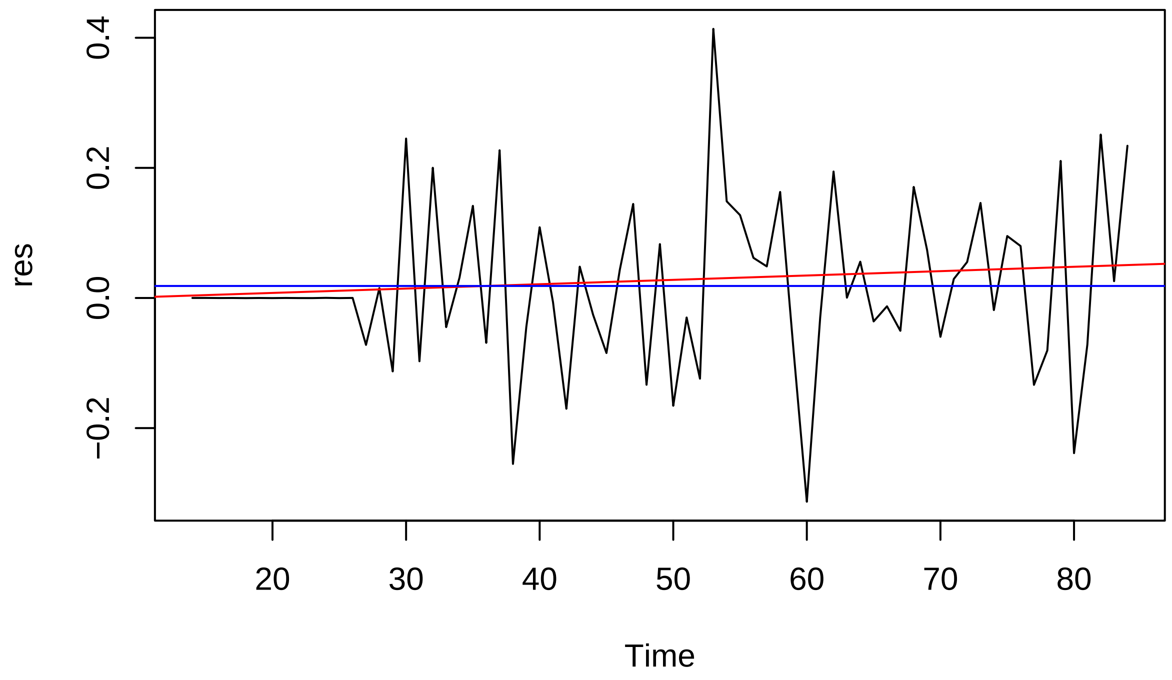


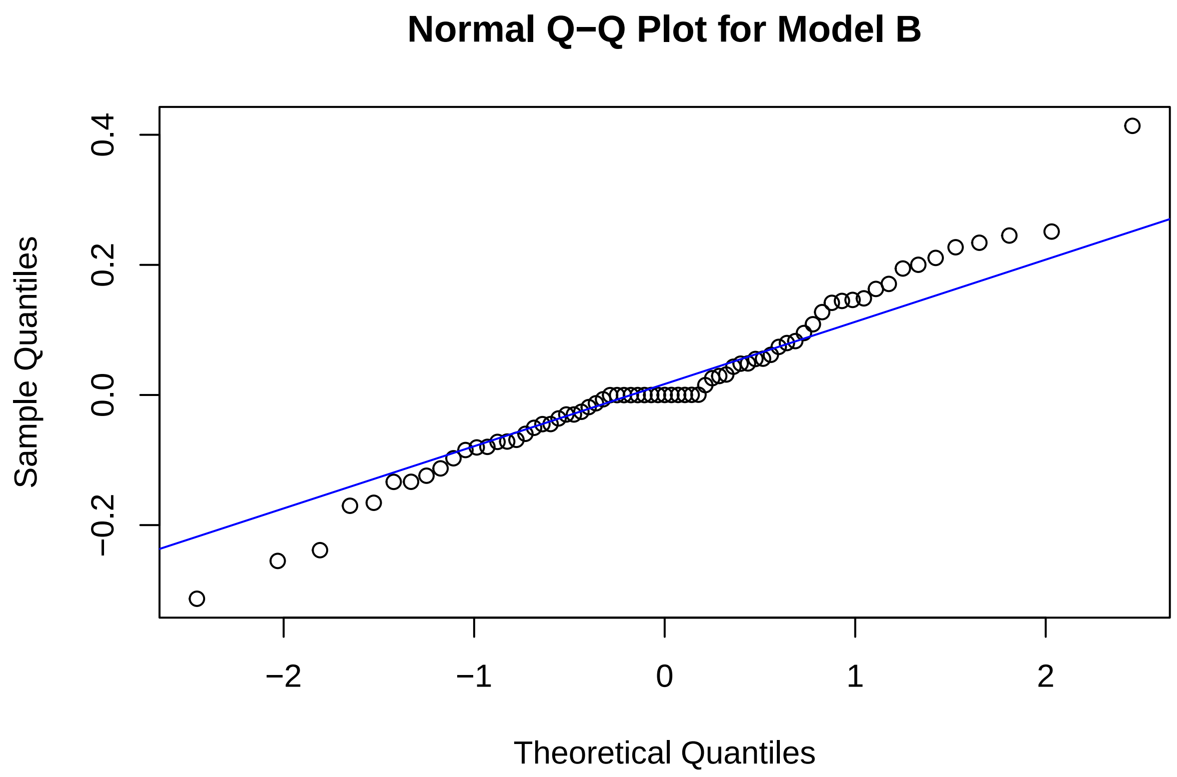
Fixing the parameters at 0 does decrease the model's AICc. However, Model B is not invertible so we cannot use this to perform diagnostic testing. The two models we will then choose are Model A and fit2, which both have the lowest AICc and are invertible.

## Diagnostic Checking

First, we begin by working with modelA. We can check both the histogram of resiuals to make sure that it is normal as well as the plot of residuals to ensure that there is no trend. Additionally the Q-Q plot can tell us if it is normal based if the observations follow a straight line.




0.4
0.2
0.0
−0.2
res
20
30
40
50
60
70
80
Time



From the above plots, the histogram and Q-Q plot display normality. However, the plot of residuals shows that the trend line slightly strays from the blue mean line. We can run diagnostic checks to check the results.

```
shapiro.test(res)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res
## W = 0.97406, p-value = 0.1484
```

```
Box.test(res, lag = 12, type = c("Box-Pierce"), fitdf = 5)
```

```
##
##  Box-Pierce test
##
## data:  res
## X-squared = 6.1035, df = 7, p-value = 0.5277
```

```
Box.test(res, lag = 12, type = c("Ljung-Box"), fitdf = 5)
```

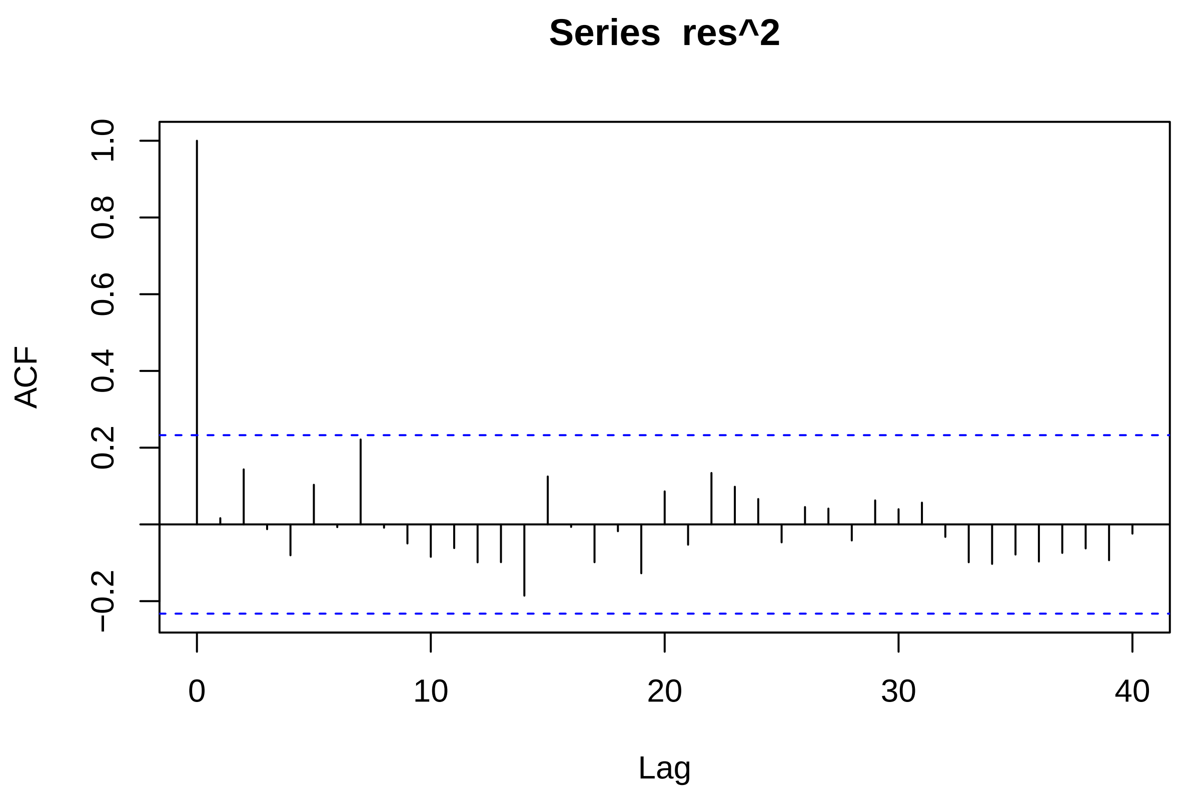
```
##
##  Box-Ljung test
##
```

```
## data:  res
## X-squared = 6.8509, df = 7, p-value = 0.4446
```

```
Box.test(res^2, lag = 12, type = c("Ljung-Box"), fitdf = 0)
```

```
##
##  Box-Ljung test
##
## data:  res^2
## X-squared = 8.8984, df = 12, p-value = 0.7116
```

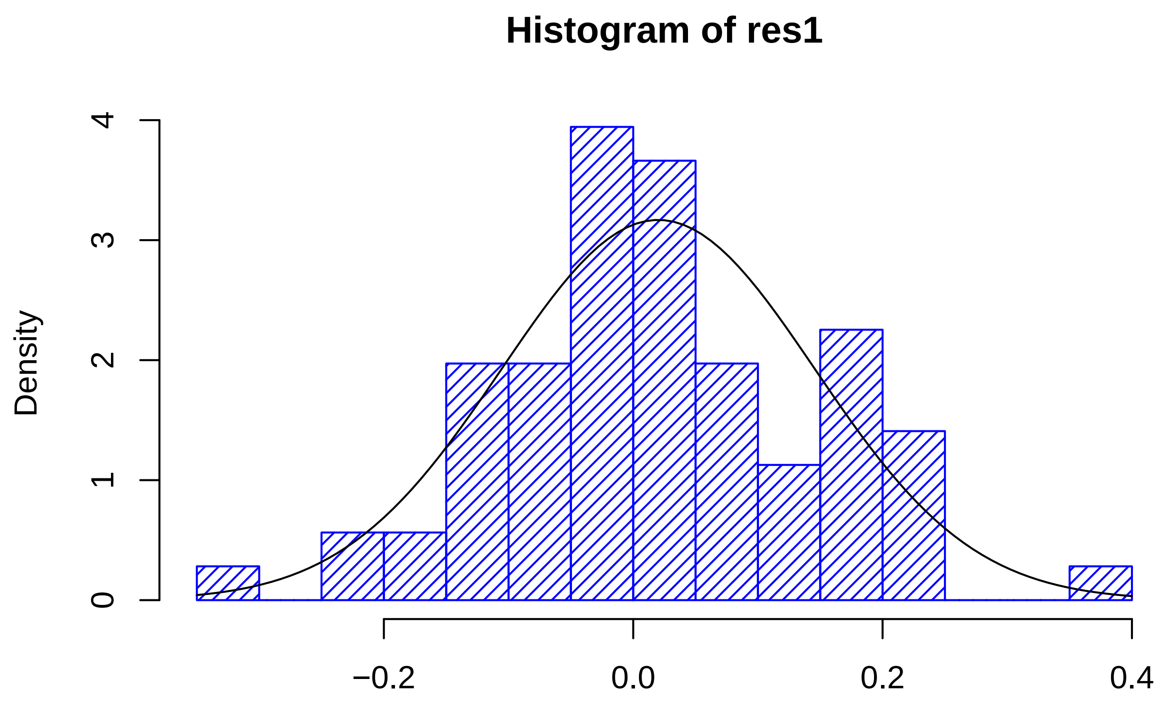
```
acf(res^2, lag.max=40)
```

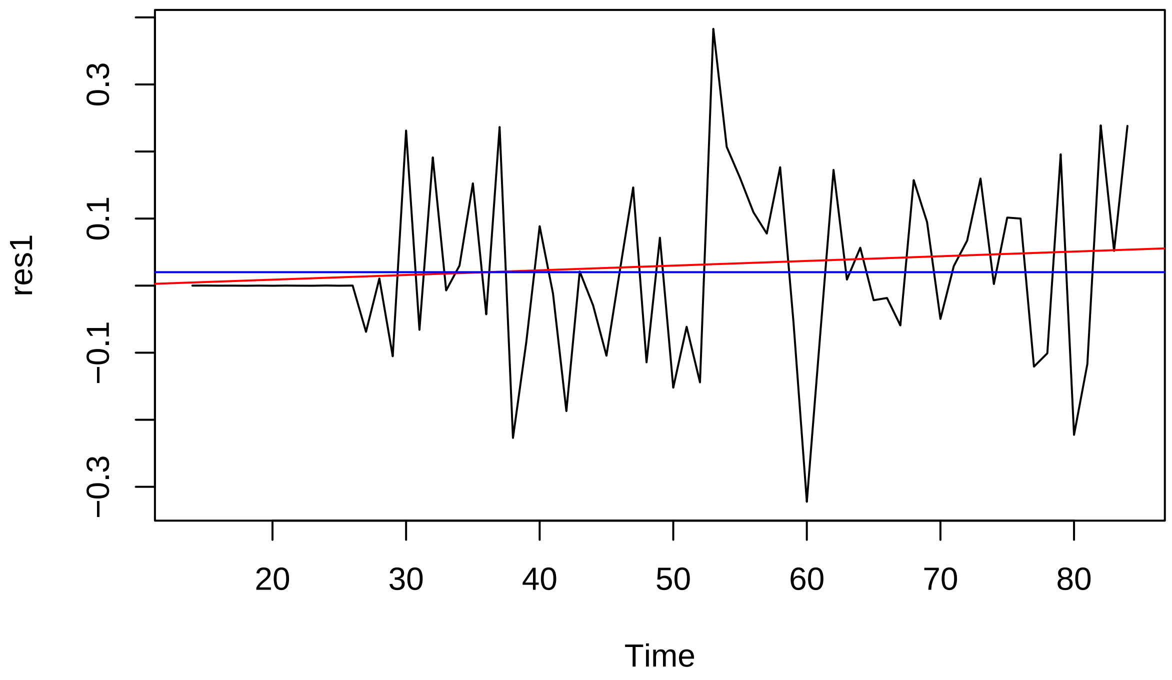


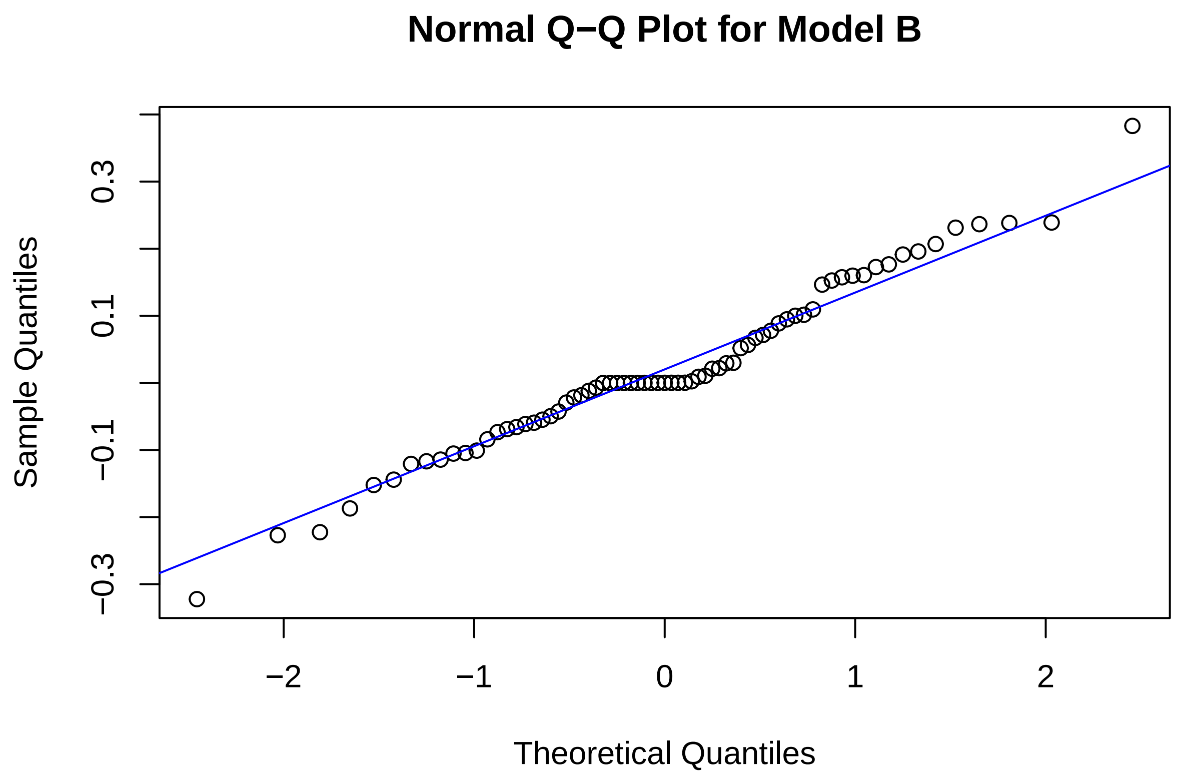
```
ar(res, aic = TRUE, order.max = NULL, method = c("yule-walker"))
```

```
##
## Call:
## ar(x = res, aic = TRUE, order.max = NULL, method = c("yule-walker"))
##
## Coefficients:
##       1
## -0.1785
##
## Order selected 1  sigma^2 estimated as  0.01562
```

This model passes diagnostic checking as all test are have a p-value well above 0.05. However, the ar model selected a model of order 1, which suggests that it is not entirely normal. Additionally, not all points are on the linear trend line for the Q-Q plot, and the trend line slightly strays from the blue mean line as observed earlier. We can move on to model modelB to see if it passes all tests of normality.




res1
0.3
0.1
−0.1
−0.3
20
30
40
50
60
70
80
Time



From the above plots corresponding to fit2, the histogram shows normality. However, it is important to note that the Q-Q plot for this model seems to exhibit a slight dip around the 0th theoretical quantile that strays from complete the straight line which means that there might be a slight stray in normality at some points. The red trend line also shows this very slight stray from the mean.

```
shapiro.test(res1)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res1
## W = 0.9783, p-value = 0.2561
```

```
Box.test(res1, lag = 12, type = c("Box-Pierce"), fitdf = 5)
```

```
##
##  Box-Pierce test
##
## data:  res1
## X-squared = 4.9373, df = 7, p-value = 0.6676
```

```
Box.test(res1, lag = 12, type = c("Ljung-Box"), fitdf = 5)
```

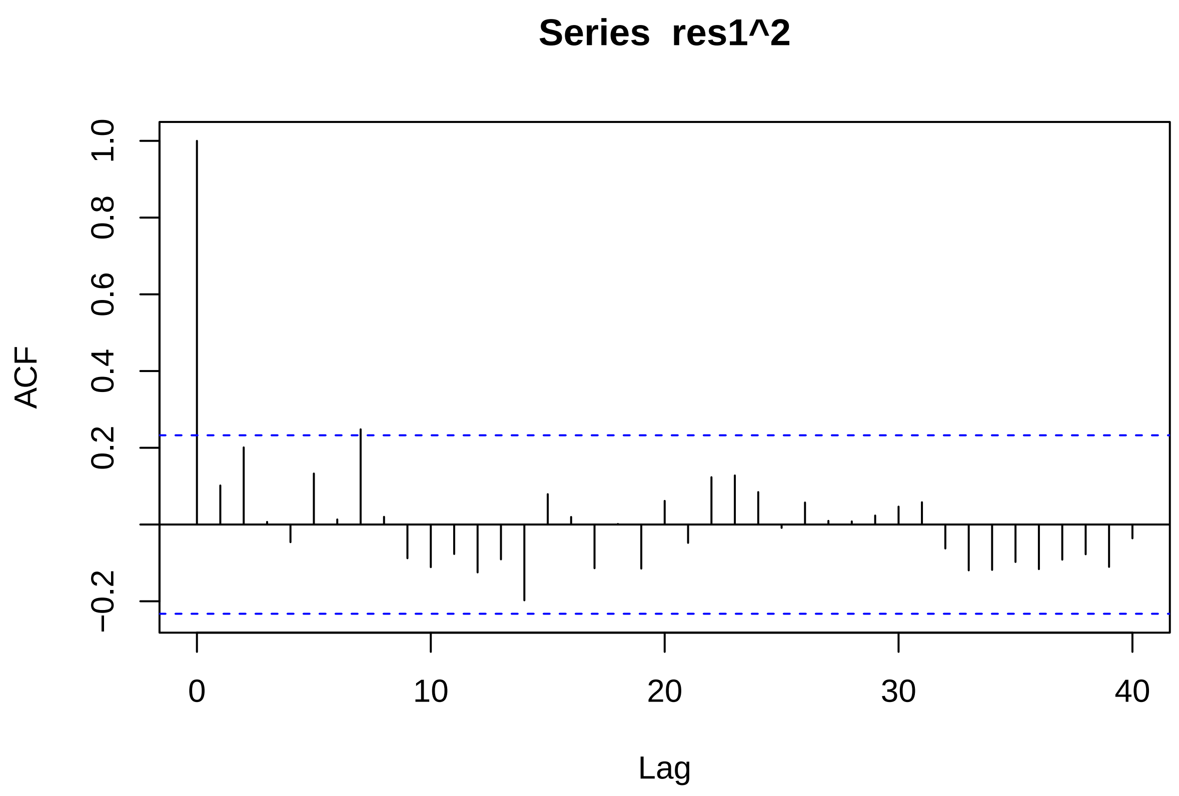
```
##
##  Box-Ljung test
```

```
##
## data:  res1
## X-squared = 5.729, df = 7, p-value = 0.5717
```

```
Box.test(res1^2, lag = 12, type = c("Ljung-Box"), fitdf = 0)
```

```
##
##  Box-Ljung test
##
## data:  res1^2
## X-squared = 13.966, df = 12, p-value = 0.3029
```

```
acf(res1^2, lag.max=40)
```



```
ar(res1, aic = TRUE, order.max = NULL, method = c("yule-walker"))
```

```
##
## Call:
## ar(x = res1, aic = TRUE, order.max = NULL, method = c("yule-walker"))
##
##
## Order selected 0  sigma^2 estimated as  0.01585
```

From the normality tests run on fit2, it can be seen that all p-values are well above 0.05 suggesting normality. Aditionally, the ar model selected order 0, which means that the data is normal and passes diagnostic checking. We will choose model fit2 and begin the process of forecasting our data with this model.

**Algebraic Form**

Final Model for the original data: follows SARIMA(2,1,2)(1,1,1)12 model:

$$(1 - 0.1619_{(0.5425)}B - 0.2730_{(0.1588)}B^2 - 1.3011_{(0.5740)}B^3 + 0.3244_{(0.5712)}B^4)(1-B)(1-B^{12})Y_t$$
$$= (1 + 1.4288_{(0.1457)}B + 0.3792_{(0.2332)}B^3)Z_t$$

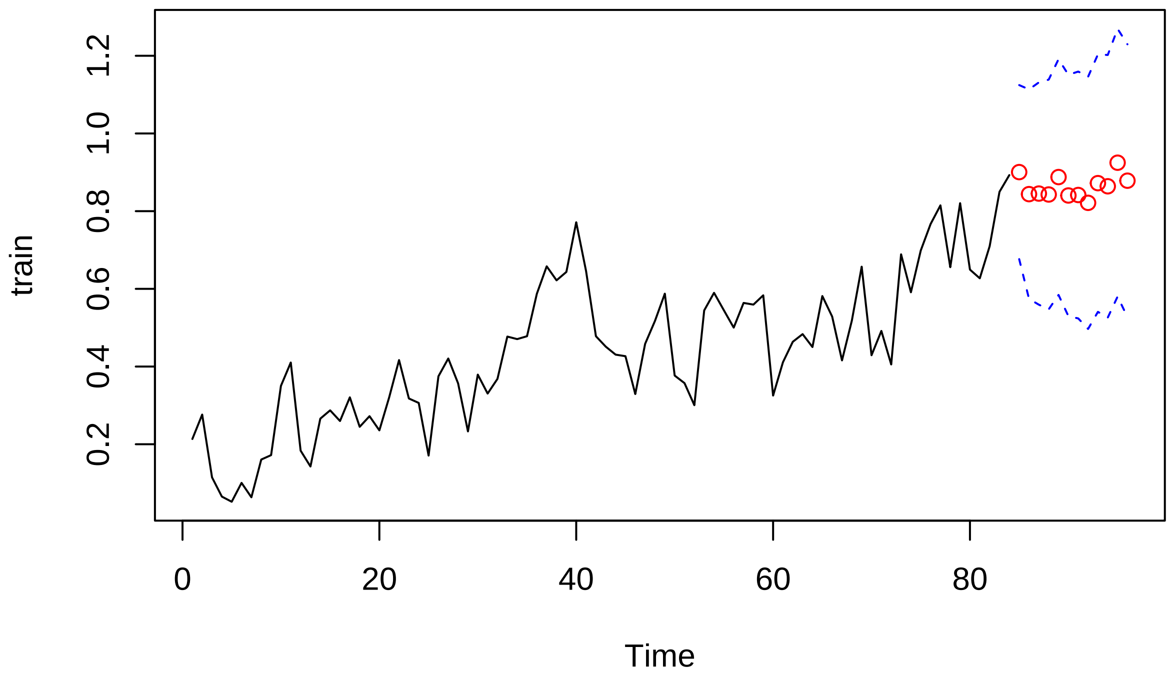
$$sigma^2 = 0.01585233$$

Now that we have the final model that will be used, we can begin the process of forecasting.

## Forecasting

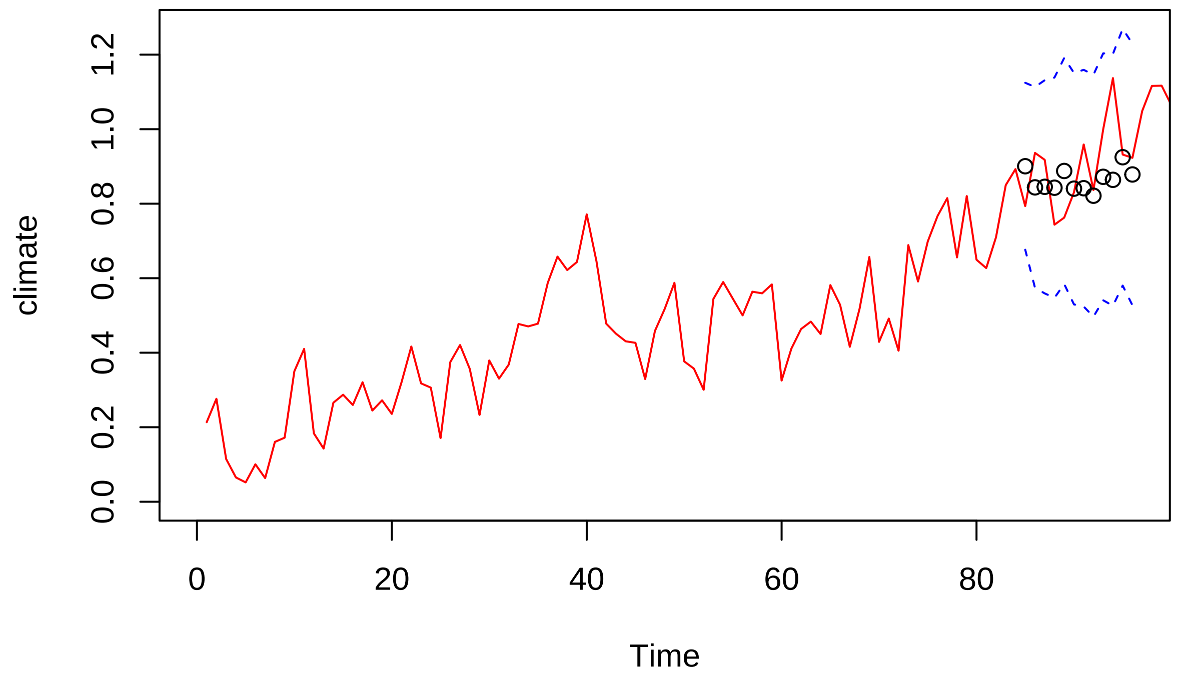
We begin by fitting the model on the training data to predict the next 12 values.

```
##     Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
##  85      0.9005192 0.7569899 1.044048 0.6810101 1.120028
##  86      0.8438089 0.6710531 1.016565 0.5796017 1.108016
##  87      0.8452947 0.6619685 1.028621 0.5649214 1.125668
##  88      0.8428837 0.6534421 1.032325 0.5531577 1.132610
##  89      0.8877822 0.6933610 1.082203 0.5904406 1.185124
##  90      0.8403346 0.6412605 1.039409 0.5358771 1.144792
##  91      0.8415929 0.6380033 1.045182 0.5302295 1.152956
##  92      0.8213578 0.6133506 1.029365 0.5032382 1.139477
##  93      0.8721460 0.6598084 1.084484 0.5474037 1.196888
##  94      0.8640522 0.6474603 1.080644 0.5328034 1.195301
##  95      0.9247172 0.7039242 1.145510 0.5870434 1.262391
##  96      0.8784146 0.6534509 1.103378 0.5343623 1.222467
##  97      0.9464171 0.7088986 1.183936 0.5831639 1.309670
##  98      0.9288892 0.6828681 1.174910 0.5526324 1.305146
##  99      0.9435115 0.6911388 1.195884 0.5575408 1.329482
## 100      0.9443955 0.6864782 1.202313 0.5499450 1.338846
## 101      0.9901472 0.7269838 1.253311 0.5876735 1.392621
## 102      0.9439847 0.6757186 1.212251 0.5337071 1.354262
## 103      0.9435401 0.6702730 1.216807 0.5256141 1.361466
## 104      0.9248565 0.6466778 1.203035 0.4994188 1.350294
## 105      0.9763980 0.6933894 1.259407 0.5435736 1.409222
## 106      0.9673606 0.6795902 1.255131 0.5272539 1.407467
## 107      1.0272021 0.7347084 1.319696 0.5798715 1.474533
## 108      0.9799725 0.6827495 1.277196 0.5254091 1.434536
```



The points forecasted from the plot above follow the trend of the graph although they seem to be a little flatter than the slope of the curve. Additionally the confidence interval is moderately large.

Now, we can check how well it works on the original data by plotting the training values on the original climate data.



From the plot above, it is evident that the confidence interval does capture the essence of the dataset. However, the individual data points deviate slightly from the fitted line. This deviation can be attributed, in part, to the presence of a pronounced curve in the central region of the original dataset, posing a challenge for accurate representation by a SARIMA model. Consequently, eliminating all noise from this model proves to be challenging.

This model is better employed for predicting ranges within which future observations are likely to fall, rather than providing precise point predictions. This limitation stems from various factors, including inherent variability in the climate system, fluctuations in anthropogenic emission levels during significant events like the COVID-19 pandemic, and natural disruptions such as volcanic eruptions. While the model can forecast future points with a certain level of confidence, exact predictions become challenging within a dataset that encompasses numerous factors and intricacies.

## Conclusion

The project aimed to fit a SARIMA model for accurate predictions of global mean temperature changes, and, for the most part, this goal was achieved. The selected model, a SARIMA(2,1,2)(1,1,1)12, determined through differencing and diagnostic checking, is represented by the following equation:

$$(1 - 0.1619_{(0.5425)}B - 0.2730_{(0.1588)}B^2 - 1.3011_{(0.5740)}B^3 + 0.3244_{(0.5712)}B^4)(1 - B)(1 - B^{12})Y_t$$
$$= (1 + 1.4288_{(0.1457)}B + 0.3792_{(0.2332)}B^3)Z_t$$

$$sigma^2 = 0.01585233$$

While the model effectively captures the general trend and seasonality, its performance is constrained by factors such as noise in the data, stemming from complexities in climate-affecting factors that were challenging to accurately capture. Although individual data points may not precisely match the original data, the confidence interval successfully encompasses it, indicating a satisfactory level of accuracy.

The forecasting accuracy proves suitable for predicting temperature ranges, highlighting the inherent uncertainty in climate predictions. In the future more sophisticated models can be explored to accurately capture the noise and variations within the data.

## References

"Global Temperature Time Series." DataHub, datahub.io/core/global-temp.

GISTEMP: NASA Goddard Institute for Space Studies (GISS) Surface Temperature Analysis, Global Land-Ocean Temperature Index.

NOAA National Climatic Data Center (NCDC), global component of Climate at a Glance (GCAG).

## Appendix

```r
# import necessary libraries
library(dplyr)
library(stringr)
library(xts)
library(tidyr)
library(astsa)
library(na.tools)
library(MASS)
library(ggplot2)
library(ggfortify)
library(tsdl)
library(forecast)

# read in data from csv file
data <- read.csv("global_temp.csv")
new_dat = subset(data, select = -c(Source) )

# find mean of the two samples collected for each month and group them together so that there is one Me
new_dat <- new_dat %>%
  group_by(Year) %>%
  summarise(Mean = mean(Mean))


#since there are negative mean values which are not suitable for later Box Cox transformations shift me
new_dat$Mean = new_dat$Mean + 0.5

new_dat <- new_dat[26: 137,]
new_dat
#convert to time series object
climate <- ts(new_dat$Mean)

# plot time series and add mean line and trend line
ts.plot(climate)
fit <- lm(climate ~ as.numeric(1:length(climate)))
```

```r
abline(fit, col="red")
abline(h=mean(climate), col="blue")
```

```r
# Calculate the index to split the data
split_index <- floor(length(climate) * 0.75)
split_index
# Create training and testing sets
train <- window(climate, end = split_index)
test <- window(climate, start = split_index + 1)

# work with training data#
# plot the time series with mean and trend
ts.plot(train)
fit <- lm(train ~ as.numeric(1:length(train)))
abline(fit, col=" red")
abline(h=mean(train), col="blue")
```

```r
# plot the histogram
hist(train, col="blue", xlab="", main="Histogram - Climate Data")
```

```r
# plot acf of train data
acf(train,lag.max=40, main="ACF of the Climate Data")
```

```r
#box cox transformation
t = 1:length(train)
fit = lm(train ~ t)
bcTransform = boxcox(train ~ t,plotit = TRUE)
```

```r
lambda = bcTransform$x[which(bcTransform$y == max(bcTransform$y))]
train.bc = (1/lambda)*(train^lambda-1)
lambda

#log transformation
train.log = log(train)

#plot original data, box-cox transformed data, and log transformed data
ts.plot(train,main = "Original data",ylab = expression(X[t]))
```

```r
ts.plot(train.bc,main = "Box-Cox tranformed data", ylab = expression(Y[t]))
```

```r
ts.plot(train.log, main = "Log transformed data",ylab = expression(Z[t]))
```

```r
#show histogram of original data, box-cox transformed data, and log transformed data
hist(train,main = "Original data",ylab = expression(X[t]))
```

```r
hist(train.bc,main = "Box-Cox tranformed data", ylab = expression(Y[t]))
```

```r
hist(train.log,main = "Log transformed data", ylab = expression(Z[t]))
```

```r
# display variance of original data, box-cox transformed data, and log transformed data
var(train)
var(train.bc)
var(train.log)

# decomposition of time series
y <- ts(as.ts(train), frequency = 12)
decomp <- decompose(y)
plot(decomp)
```

```r
# ACF and variance of undifferenced train data
acf(train, lag.max = 60, main = "ACF: Time Series")
```

```r
var(train)

# ACF and variance of de-seasonalized train data
train_diff_s = diff(train, 12)
acf(train_diff_s, lag.max = 60, main = "ACF: Seasonal Differencing")
```

```r
var(train_diff_s)

# ACF and variance of de-seasonalized and det-rended train data
train_diff1 = diff(train_diff_s, 1)
acf(train_diff1, lag.max = 60, main = "ACF: Seasonal and Trend Differencing")
```

```r
var(train_diff1)

# plot the differenced data
ts.plot(train_diff1,main = "Differenced data",ylab = expression(X[t]))
```

```r
# check histogram of differenced data
hist(train_diff1, density=20,breaks=20, col="blue", xlab="", prob=TRUE)
m<-mean(train_diff1)
std<- sqrt(var(train_diff1))
curve(dnorm(x,m,std), add=TRUE)
```

```r
# check pacf of differenced data
pacf(train_diff1, lag.max=60, main="PACF: Seasonal/Trend Differencing of Box-Cox Transformed Time Serie
```

```r
#try fitting models
# SARIMA(1, 1, 2)(1, 1, 1)12
fit1 <- arima(train_diff1, order = c(1, 1, 2), seasonal = list(order = c(1, 1, 1), period = 12))
summary(fit1)
# AICC  of fit1
npar <- length(fit1$coef) + 1
nstar <- length(fit1$residuals) - fit1$arma[6] - fit1$arma[7] * fit1$arma[5]
aicc1 <- fit1$aic + 2 * npar * (nstar/(nstar - npar - 1) - 1)
aicc1
#check invertibility of fit2
autoplot(fit1)
```

```r
# SARIMA(2, 1, 2)(1, 1, 1)12
fit2 <- arima(train_diff1, order = c(2, 1, 2), seasonal = list(order = c(1, 1, 1), period = 12))
summary(fit1)
# AICC  of fit2
npar <- length(fit2$coef) + 1
nstar <- length(fit2$residuals) - fit2$arma[6] - fit2$arma[7] * fit2$arma[5]
aicc2 <- fit2$aic + 2 * npar * (nstar/(nstar - npar - 1) - 1)
aicc2
#check invertibility of fit 2
autoplot(fit2)
```

```r
# SARIMA(2, 1, 3)(1, 1, 1)12
fit3 <- arima(train_diff1, order = c(2, 1, 3), seasonal = list(order = c(1, 1, 1), period = 12))
summary(fit3)
# AICC  of fit3
npar <- length(fit3$coef) + 1
nstar <- length(fit3$residuals) - fit3$arma[6] - fit3$arma[7] * fit3$arma[5]
aicc3 <- fit3$aic + 2 * npar * (nstar/(nstar - npar - 1) - 1)
aicc3
#check invertibility of fit3
autoplot(fit3)
```

```r
#use fit2 to fix parameters
modelA <- arima(train_diff1, order = c(2, 1, 2), seasonal = list(order =
c(1, 1, 1), period = 12), fixed = c(0, NA, NA, 0, NA, NA), method="ML")

summary(modelA)
#check invertiblity of modelA
autoplot(modelA)
```

```r
#use fit3 to fix parameters
modelB <- arima(train_diff1, order = c(2, 1, 3), seasonal = list(order =
c(1, 1, 1), period = 12), fixed = c(0, 0, NA, 0, NA, NA, NA), method="ML")

summary(modelB)
#check invertiblity of modelB
autoplot(modelB)
```

```r
#check residual plot
plot.ts(res)
fitt <- lm(res ~ as.numeric(1:length(res)))
abline(fitt, col="red")
abline(h=mean(res), col="blue")
```

```r
#normal Q-Q plot for model B
qqnorm(res, main= "Normal Q-Q Plot for Model B")
qqline(res, col="blue")
```

```r
#normality tests
shapiro.test(res)
Box.test(res, lag = 12, type = c("Box-Pierce"), fitdf = 5)
Box.test(res, lag = 12, type = c("Ljung-Box"), fitdf = 5)
Box.test(res^2, lag = 12, type = c("Ljung-Box"), fitdf = 0)
acf(res^2, lag.max=40)
```

```r
ar(res, aic = TRUE, order.max = NULL, method = c("yule-walker"))

#now move onto fit2
res1 <- residuals(fit2)
hist(res1, density=20,breaks=20, col="blue", xlab="", prob=TRUE)
m <- mean(res1)
std <- sqrt(var(res1))
curve(dnorm(x,m,std), add=TRUE )
```

```r
#check residual plot
plot.ts(res1)
fitt <- lm(res1 ~ as.numeric(1:length(res1)))
abline(fitt, col="red")
abline(h=mean(res1), col="blue")
```

```r
#normal Q-Q plot for fit2
qqnorm(res1, main= "Normal Q-Q Plot for Model B")
qqline(res1, col="blue")
```

```r
#normality tests
shapiro.test(res1)
Box.test(res1, lag = 12, type = c("Box-Pierce"), fitdf = 5)
Box.test(res1, lag = 12, type = c("Ljung-Box"), fitdf = 5)
Box.test(res1^2, lag = 12, type = c("Ljung-Box"), fitdf = 0)
acf(res1^2, lag.max=40)
```

```r
ar(res1, aic = TRUE, order.max = NULL, method = c("yule-walker"))
var(res1)

# fit on train data
fit.A <- arima(train, order = c(2, 1, 2), seasonal = list(order =
c(1, 1, 1), period = 12), method="ML")

forecast(fit.A)

#predict on  train data
pred.tr <- predict(fit.A, n.ahead = 12)
U.tr= pred.tr$pred + 2*pred.tr$se
L.tr= pred.tr$pred - 2* pred.tr$se
ts.plot(train, xlim=c(1,length(train)+12), ylim = c(min(train), max(U.tr)))
lines(U.tr, col="blue", lty="dashed")
lines(L.tr, col="blue", lty="dashed")
points((length(train)+1):(length(train)+12), pred.tr$pred, col="red")
```

```r
# fit on original data
pred.orig <- (pred.tr$pred)
U= (U.tr)
L= (L.tr)
ts.plot(climate, xlim = c(0,length(train)+12), ylim = c(0,max(U)), col="red")
```

```r
lines(U, col="blue", lty="dashed")
lines(L, col="blue", lty="dashed")
points((length(train)+1):(length(train)+12), pred.orig, col="black")
```