SARIMA Modelling of Monthly Average of Santa Barbara Municiple Airport Daily Maximum Temperature in Fahrenheit from 1941-2021

Meredith Johnson

```
knitr::opts_chunk$set(message = FALSE, warning = FALSE)
```


Abstract

The goal of this project is to accurately predict the average daily maximum temperature around Goleta, California of the last 12 months of the given data set containing daily maximum temperatures from the Santa Barbara Municipal Airport in California from 1941-2021 in order to confidently predict the average daily maximum temperature of future months. I use a SARIMA model to model this data set and forecast monthly average temperatures. ACF and PACF plots are used in order to create candidate models and the AIC criterion and diagnostic tests are used to choose the best model. A sufficient model is obtained; however, forecasted predictions are not consistently accurate and an alternative method for obtaining more accurate predictions is proposed.

Introduction

Access to weather predictions is essential to the majority of industries as well as the daily life of an average person. I personally check the weather forecast everyday in order to choose appropriate clothing; this inspired me to analyze local temperature data. From the National Centers for Environmental Information, National Oceanic and Atmospheric Administration website, I downloaded a data set containing daily maximum temperatures from the Santa Barbara Municipal Airport in California from 1941-2021. I then averaged the temperatures of each day to obtain monthly average maximum temperatures. I used a SARIMA model, the programming language R, and the Rstudio Integrated Development Environment to model this data set and forecast monthly average maximum temperatures. ACF and PACF plots are used in order to create candidate models and the AIC criterion is used in order to narrow down the best candidate models for diagnostic checking. Histograms, Shapiro-Wilk normality tests, Box-Pierce tests, Box-Ljung tests, and Mcleod-Li tests, Autoregressive residual fitting, residuals plots, Q-Q plots, acf and pacf of the residuals, and acf of the residuals squared are used in model diagnostics. The chosen model is sufficient in regard to all of these diagnostics except for the Mcleod-Li test and the relatively large residual sample mean returned by the autoregressive residual fitting. I believe the model to be sufficient; however, I would use a slightly different technique to forecast future predictions in order to obtain more consistently accurate predictions.

Data Cleaning

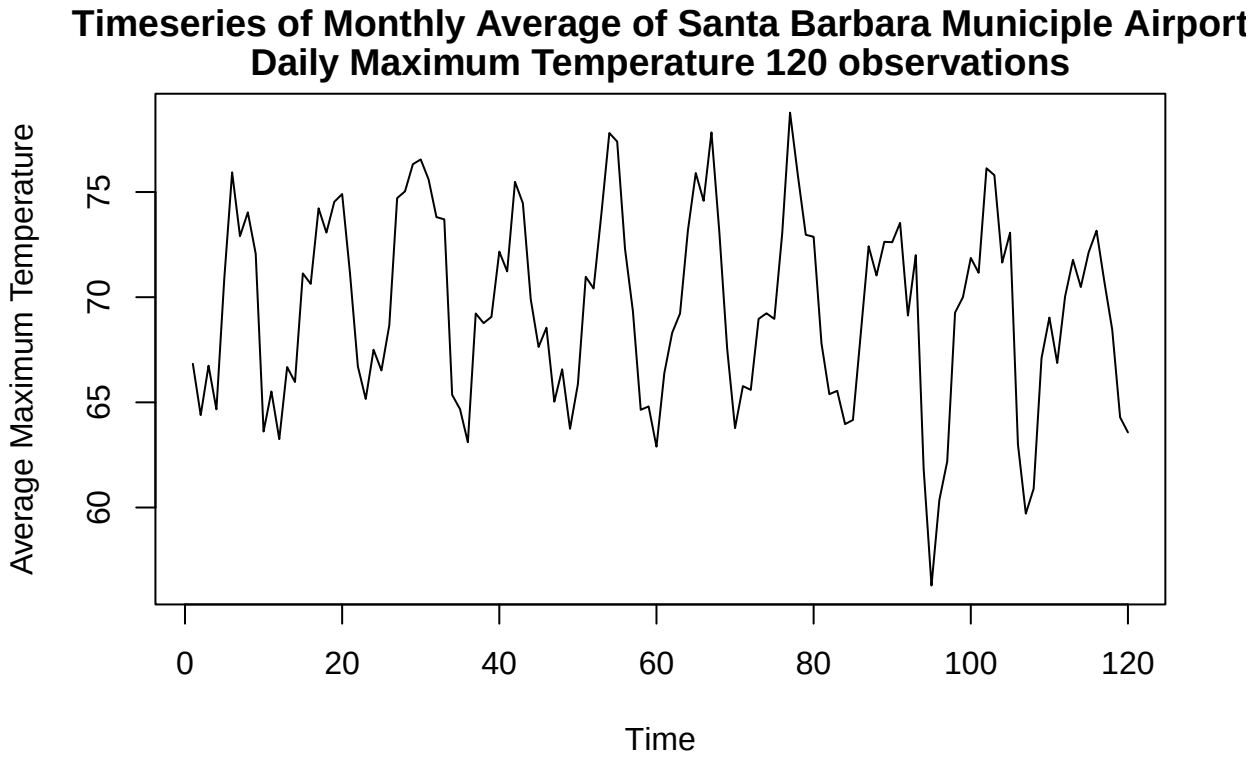
```
library(dplyr)

temp.date =
↪  read.csv("/Users/merej/Downloads/W22_Pstat_174/Final_Project/SBDailyMaxTemp.csv") %>%
↪  #read in CSV file
  select(DATE, TMAX) %>% #select only the date and daily maximum temperature variables
  mutate(DATE = substr(DATE, 1, 7)) %>% #mutate the date variable to only contain year
   ↪  and month
  group_by(DATE) %>% #group the data set by month
```

```
  summarise(MonthlyAverageDailyMaximumTemperature = mean(na.omit(TMAX))) #take the
  ↪  average of the temperatures belonging to each month
temp = unlist(temp.date[2])
head(temp.date)
```

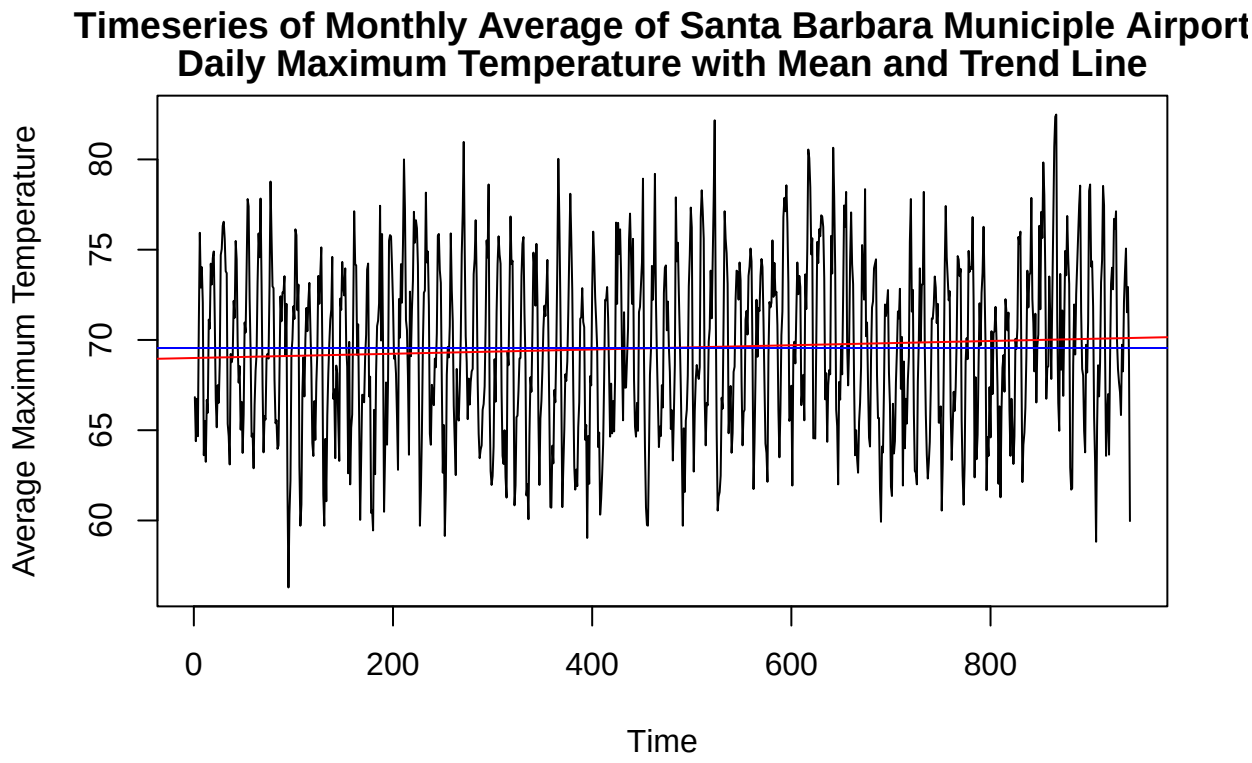
```
## # A tibble: 6 x 2
##   DATE    MonthlyAverageDailyMaximumTemperature
##   <chr>                                   <dbl>
## 1 1941-01                                  66.8
## 2 1941-02                                  64.4
## 3 1941-03                                  66.7
## 4 1941-04                                  64.7
## 5 1941-07                                  70.9
## 6 1941-08                                  75.9
```

Exploratory Analysis

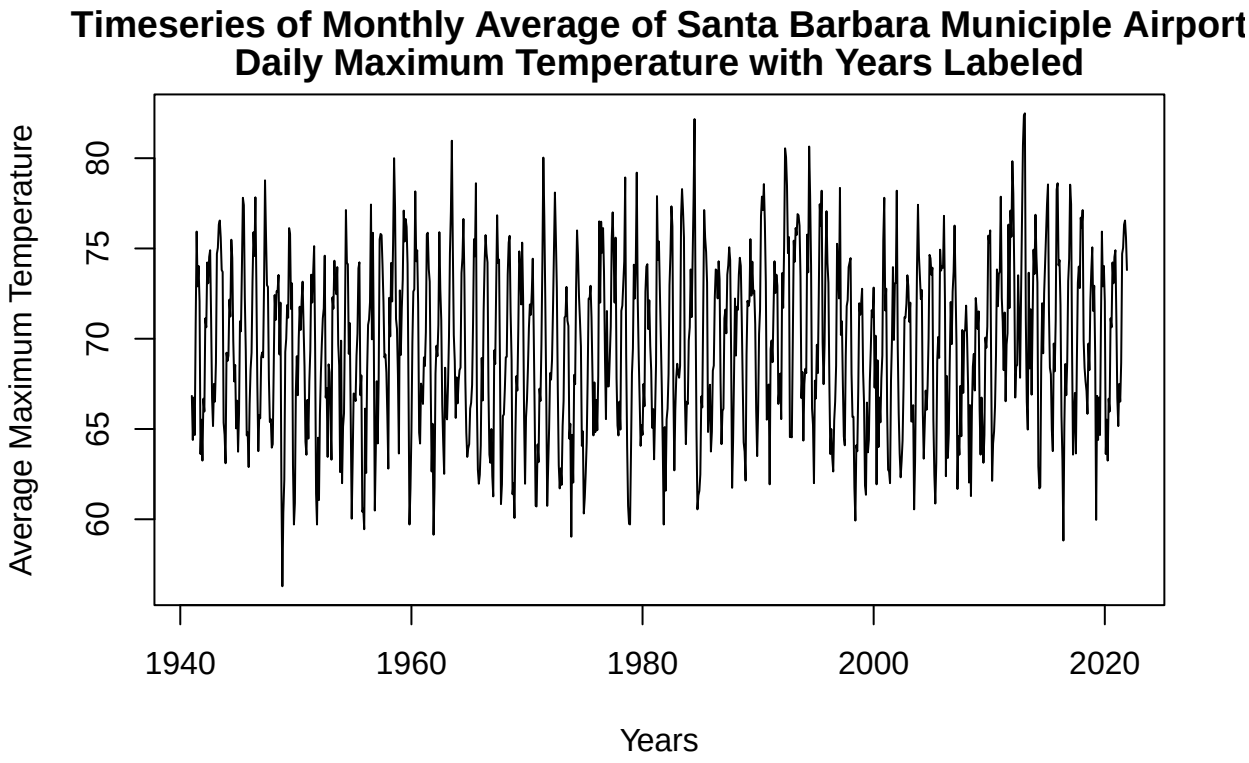
```
#zoomed in plot
plot.ts(temp[c(1: 120)], main = "", ylab = "Average Maximum Temperature")
#Add title in two lines so it doesn't get cut off when knit to pdf
title(main = c("Timeseries of Monthly Average of Santa Barbara Municiple Airport", "Daily
↪  Maximum Temperature 120 observations"), line = c(1, 2))
```



```r
#plot containing all observations
plot.ts(temp, main = "", ylab = "Average Maximum Temperature")
#Add title
title(main = c("Timeseries of Monthly Average of Santa Barbara Municiple Airport", "Daily
↪  Maximum Temperature with Mean and Trend Line"), line = c(1, 2))
#add trend
temp.fit <- lm(temp ~ as.numeric(1:length(temp)))
abline(temp.fit, col = "red")
#add mean
abline(h = mean(temp), col = "blue")
```



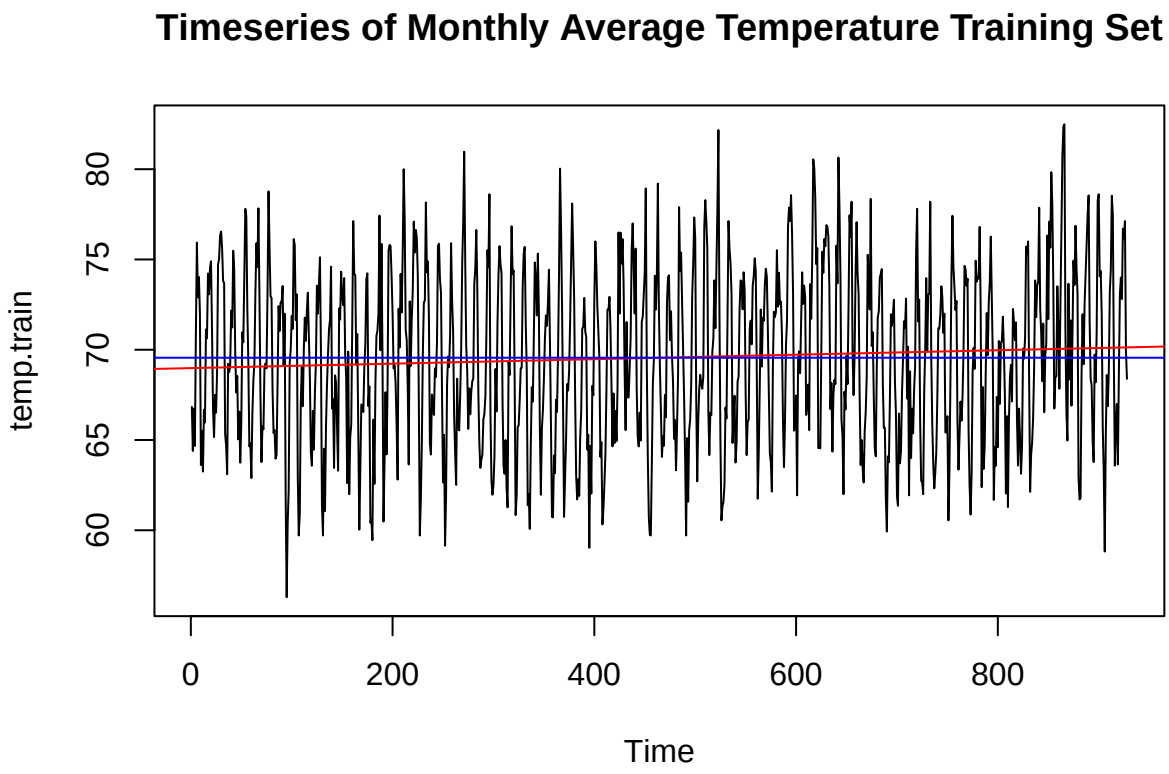

```r
#plot using years
temp.years <- ts(temp, start = c(1941,1), end = c(2021,12), frequency = 12)
ts.plot(temp.years, main = "", xlab = "Years", ylab = "Average Maximum Temperature")
#Add title
title(main = c("Timeseries of Monthly Average of Santa Barbara Municiple Airport", "Daily
↪  Maximum Temperature with Years Labeled"), line = c(1, 2))
```



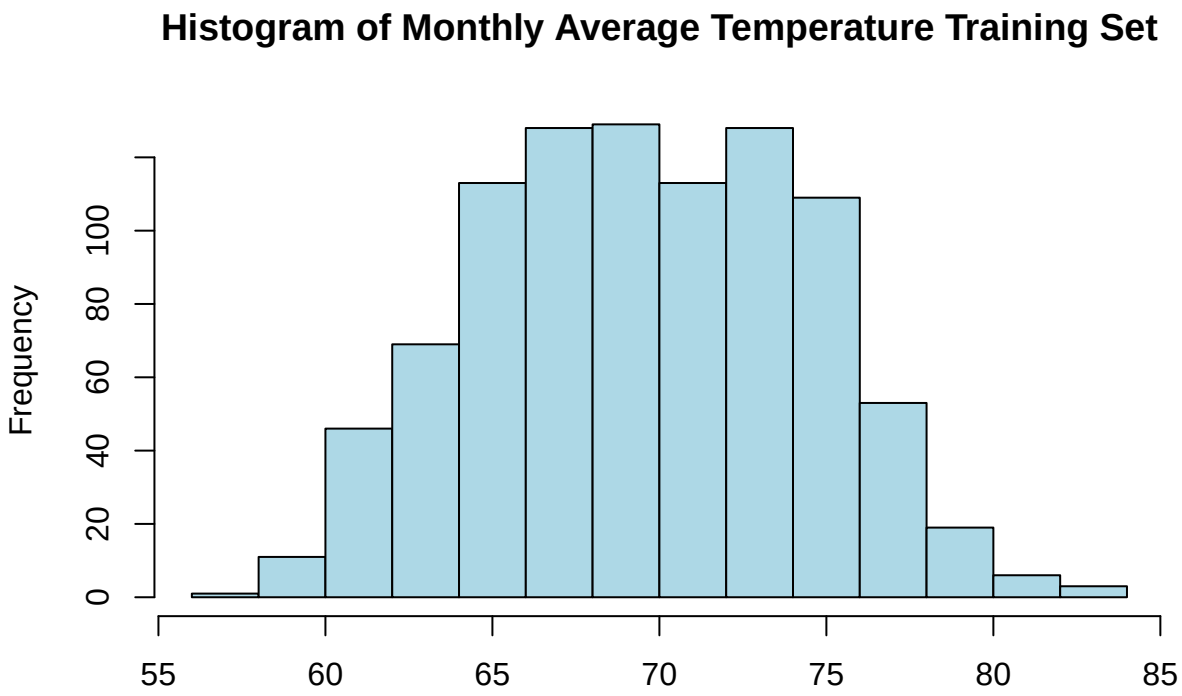
The first plot depicts the seasonality of the data set: the average maximum temperature appears to follow a 12 observation pattern which corresponds to a year. The second plot depicts an overall mean of just below 70 degrees Fahrenheit and a slightly upwards trend. The third plot includes years in the x-axis. There appears to be several sharp changes in behavior. Notable sharp changes include one around 1950 and one around 2007.

```
#Partition data set to two parts for model training and model validation
#training data
temp.train = temp[c(1:928)] #all observations but the last 12
#test data
temp.test = temp[c(929:940)] #only the last 12 observations
```

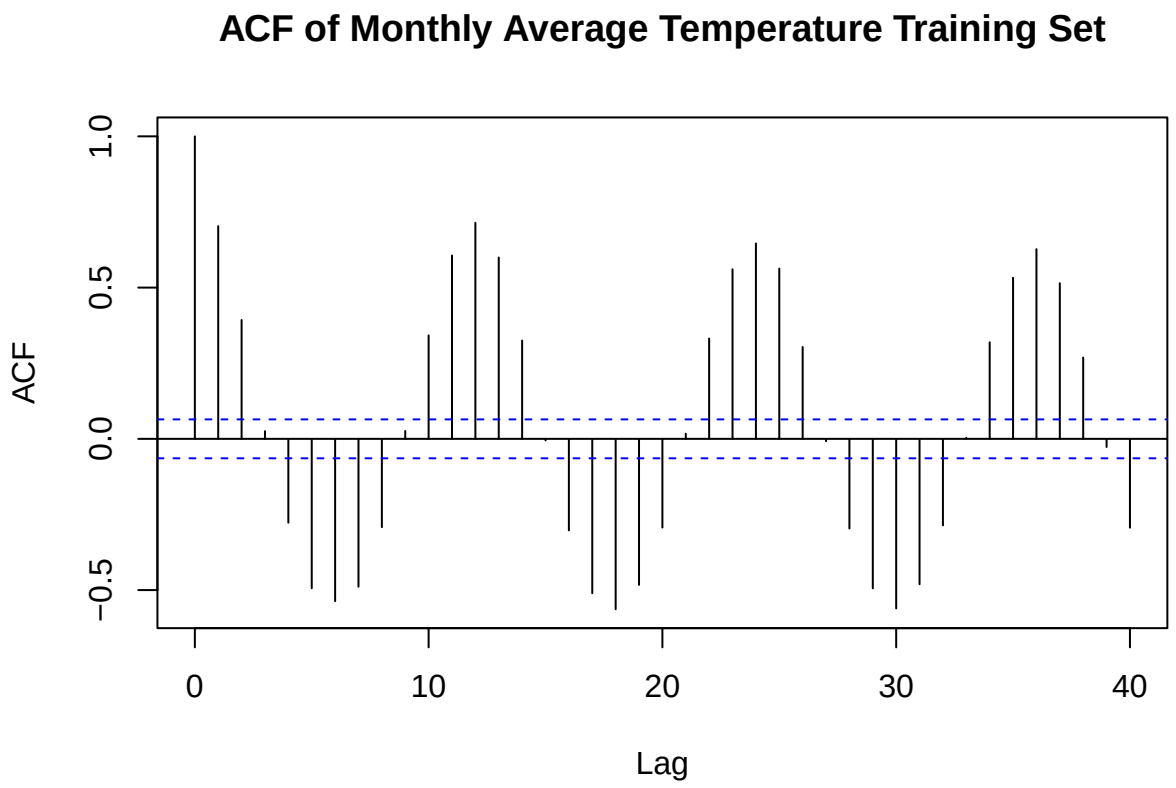
```
#plot training data
plot.ts(temp.train, main = "Timeseries of Monthly Average Temperature Training Set")
temp.train.fit <- lm(temp.train ~ as.numeric(1:length(temp.train)));
abline(temp.train.fit, col="red")
abline(h=mean(temp.train), col="blue")
```




```
#plot histogram
hist(temp.train, col="light blue", xlab="", main = "Histogram of Monthly Average
↪  Temperature Training Set")
```



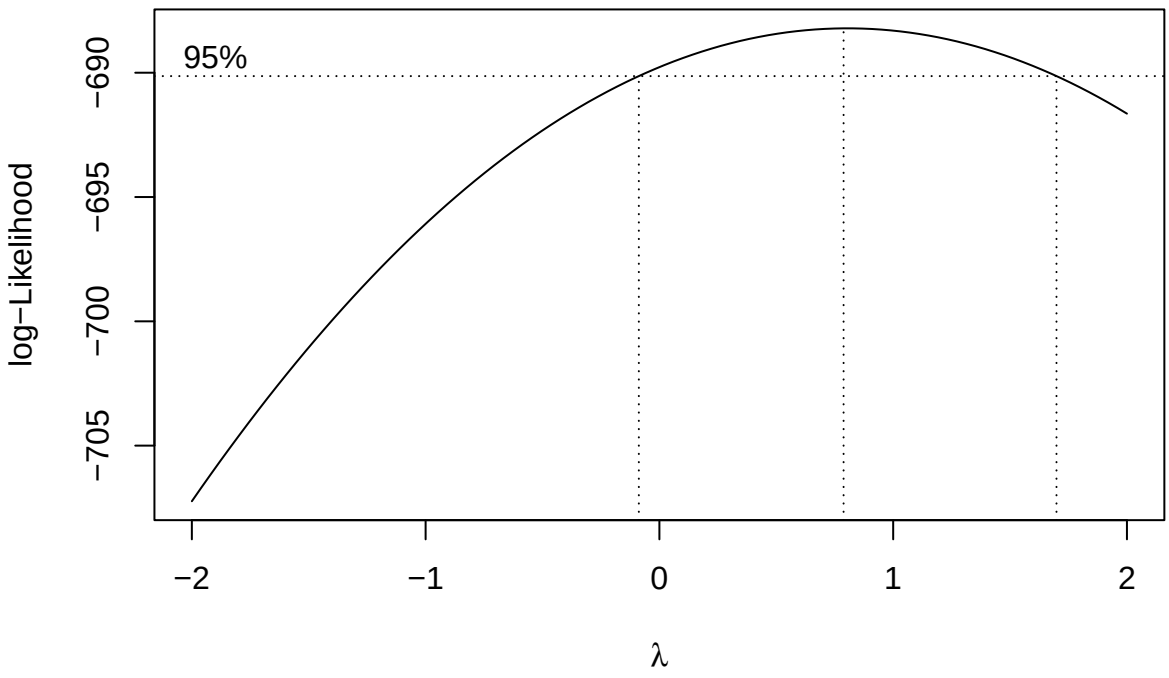

```
#plot acf
acf(temp.train,lag.max=40, main="ACF of Monthly Average Temperature Training Set")
```



The histogram appears to be approximately normal which is an indicator that the data is stationary (has constant variance); therefore, implementing variance-stabilizing transformations is most likely unnecessary. The acf plot depicts seasonality with a period of 12 which indicates that we should difference the data at lag 12 in order to further analyze the data.

Transformations

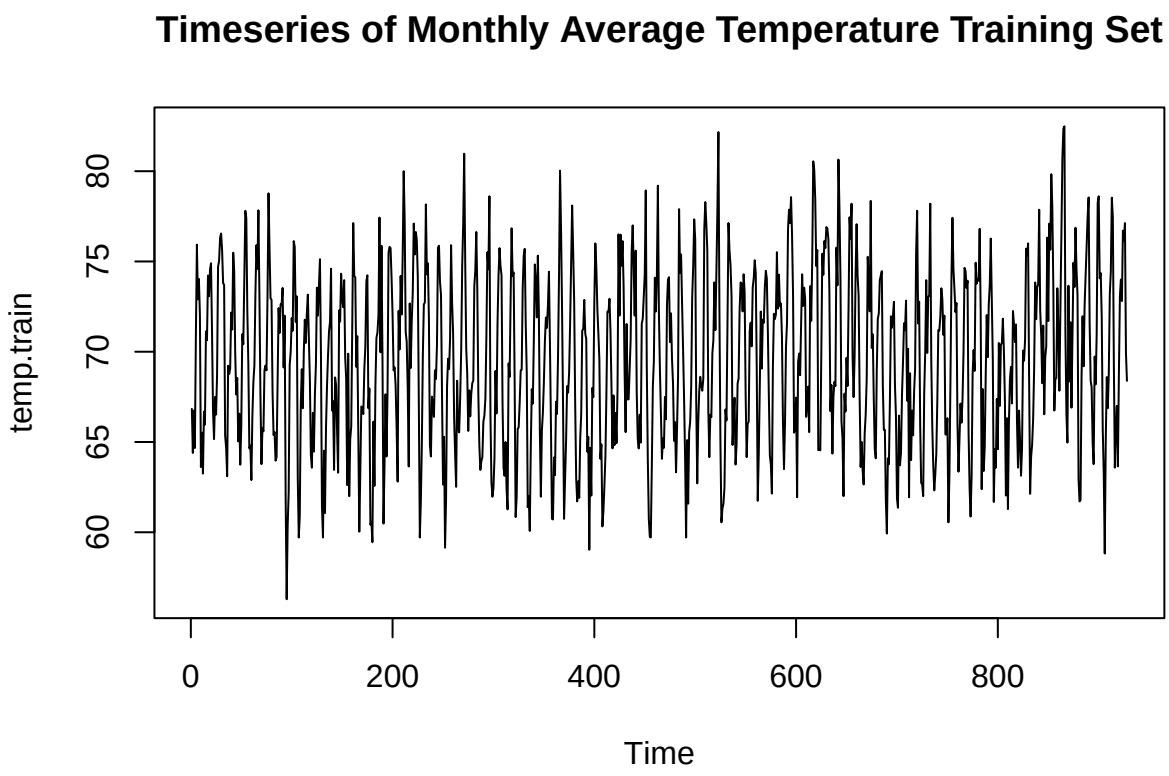
```
#choose lambda for boxcox transformation
#graph to choose parameter lambda of the Box-Cox transformation for data set temp.train:
library(MASS)
bcTransform <- boxcox(temp.train ~ as.numeric(1:length(temp.train)))
```



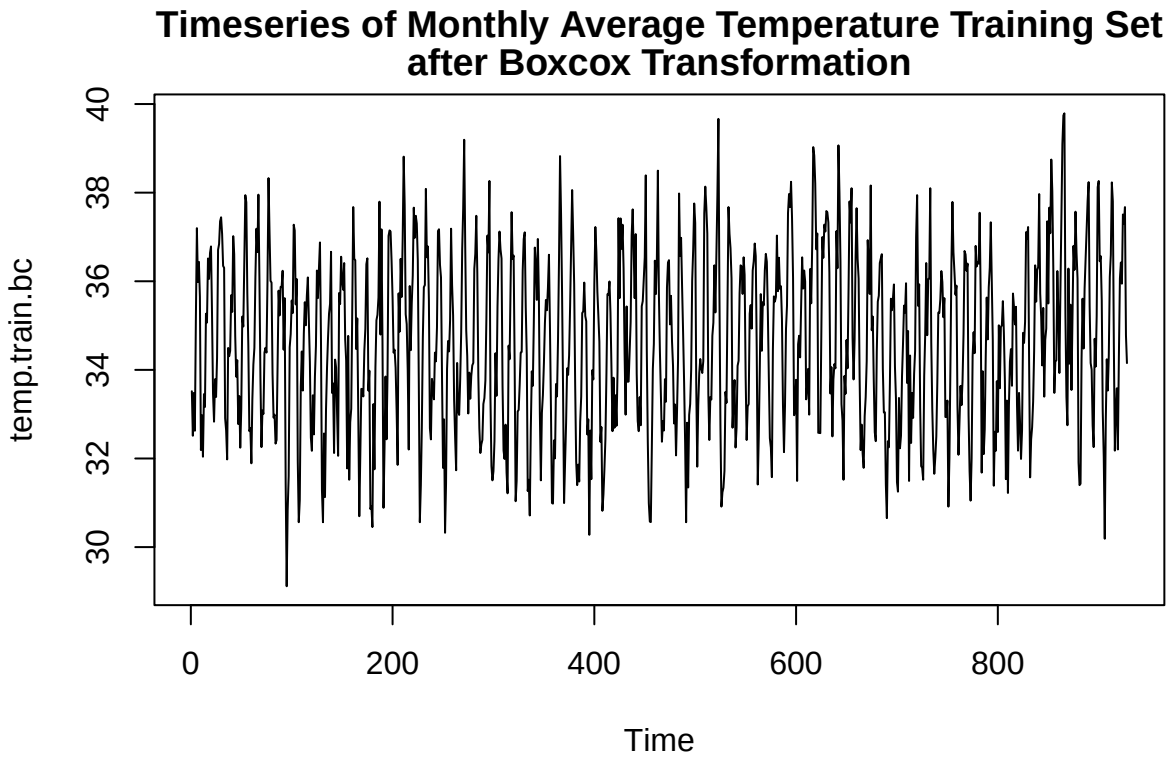

```
#value of lambda
lambda = bcTransform$x[which(bcTransform$y == max(bcTransform$y))]
lambda
```

```
## [1] 0.7878788
```

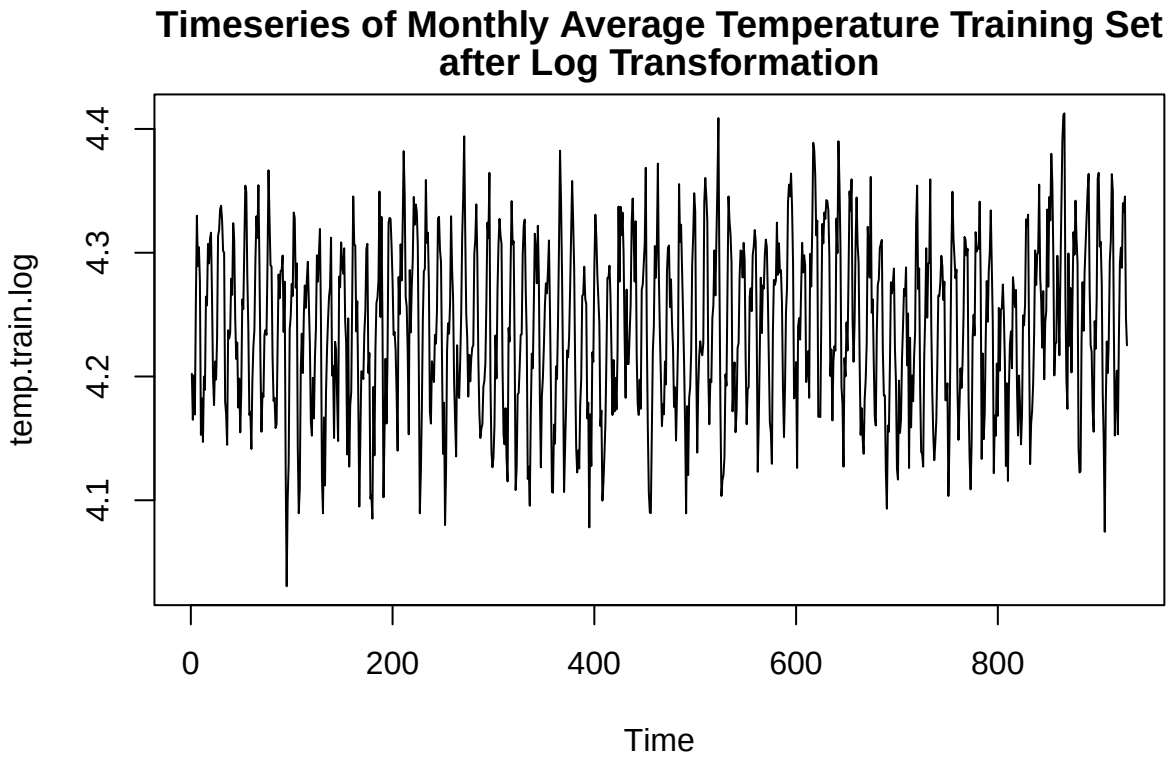
```
#perform boxcox and log transformations
temp.train.bc = (1/lambda)*(temp.train^lambda-1)
temp.train.log <- log(temp.train)

#plot not transformed, boxcox, and log transformed data
plot.ts(temp.train, main = 'Timeseries of Monthly Average Temperature Training Set')
```



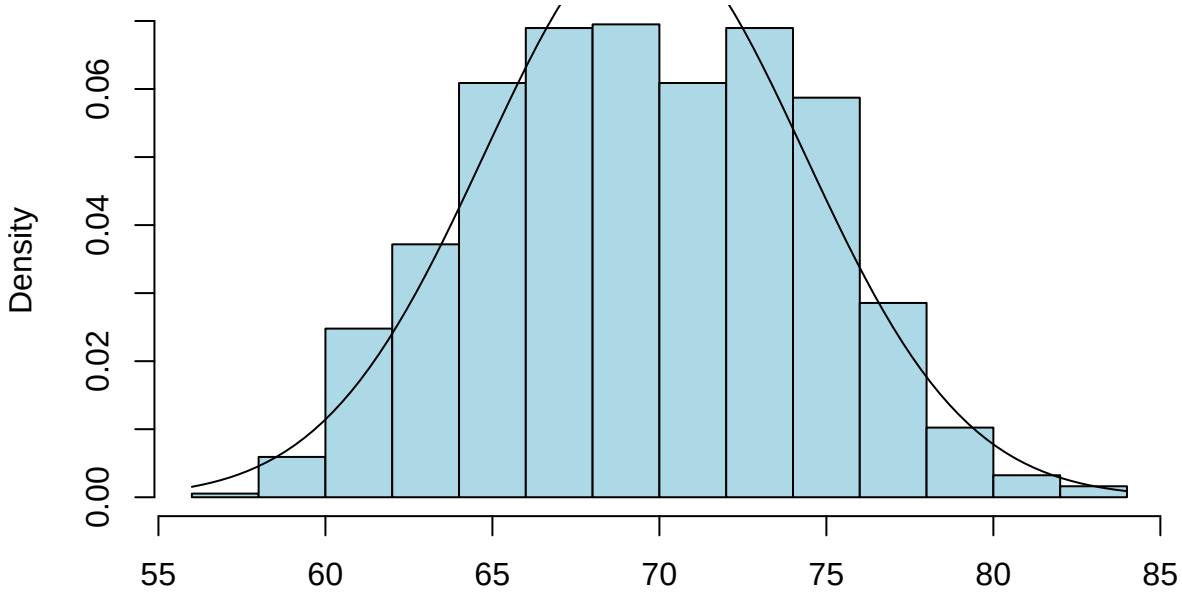

```
plot.ts(temp.train.bc, main = '')
title(main = c("Timeseries of Monthly Average Temperature Training Set", "after Boxcox
↪  Transformation"), line = c(1, 2))
```



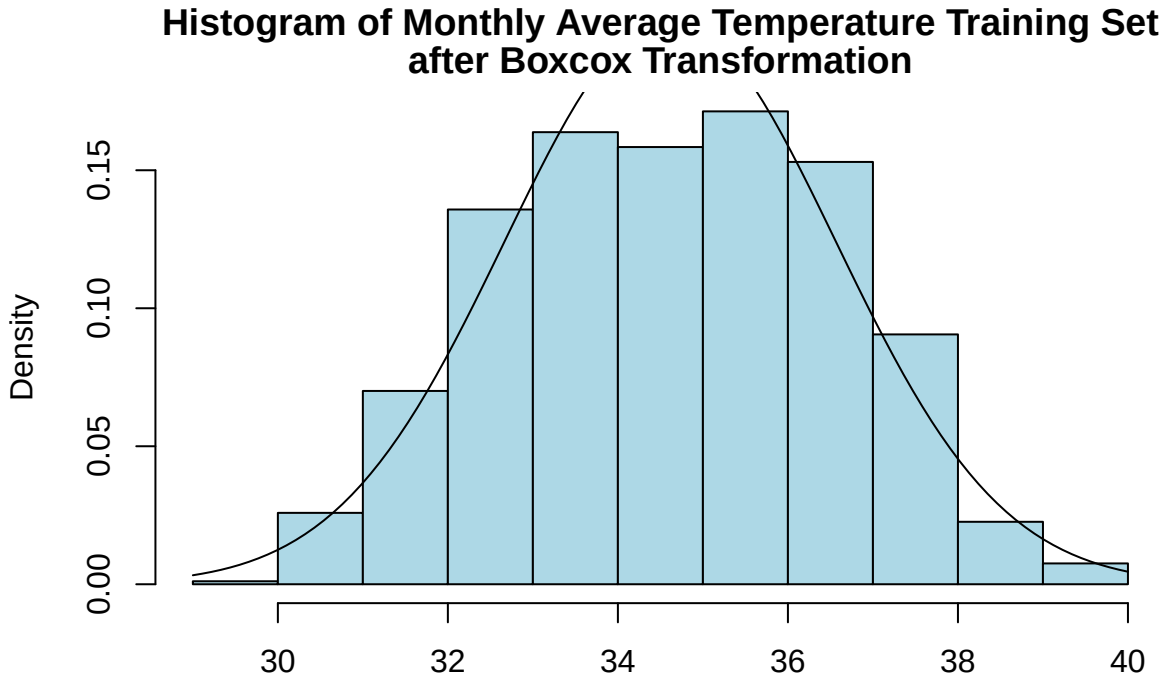

```
plot.ts(temp.train.log, main = '')
title(main = c("Timeseries of Monthly Average Temperature Training Set", "after Log
↪  Transformation"), line = c(1, 2))
```



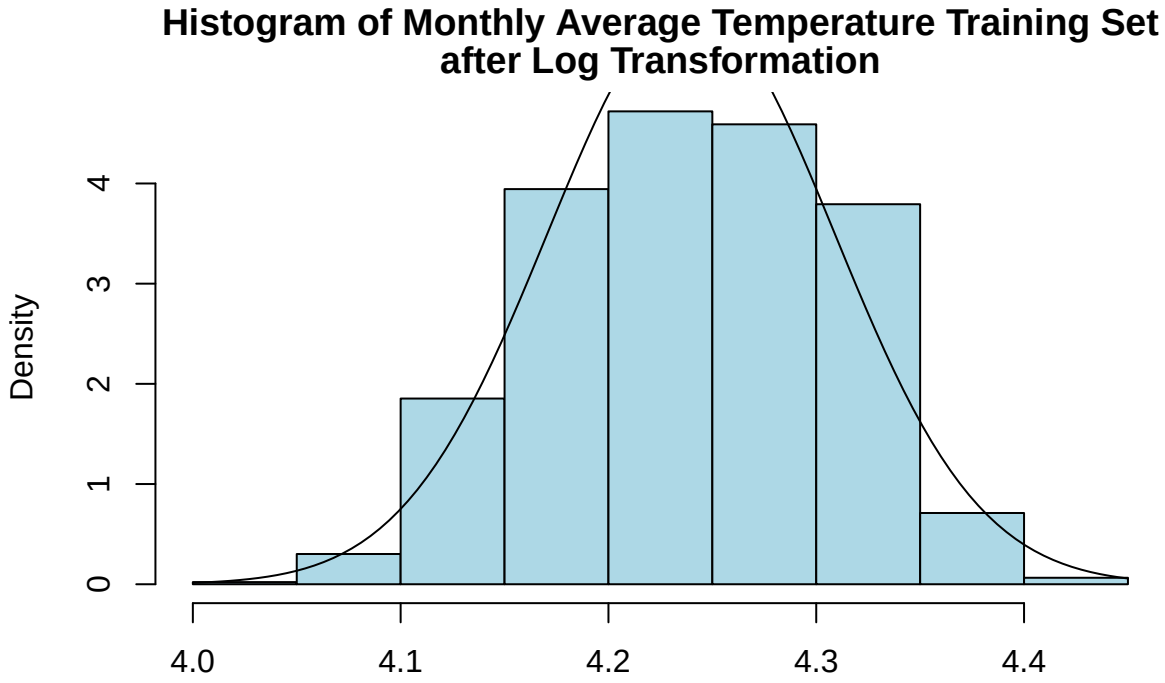

```
#histograms for not transformed, boxcox, and log transformed data
hist(temp.train, col="light blue", xlab="", main = "Histogram of Monthly Average
↪  Temperature Training set", probability = TRUE)
curve(dnorm(x, mean(temp.train), sd(temp.train)), add = TRUE)
```




```
hist(temp.train.bc, col="light blue", xlab="", main="", probability = TRUE)
curve(dnorm(x, mean(temp.train.bc), sd(temp.train.bc)), add = TRUE)
title(main = c("Histogram of Monthly Average Temperature Training Set", "after Boxcox
↪  Transformation"), line = c(1, 2))
```

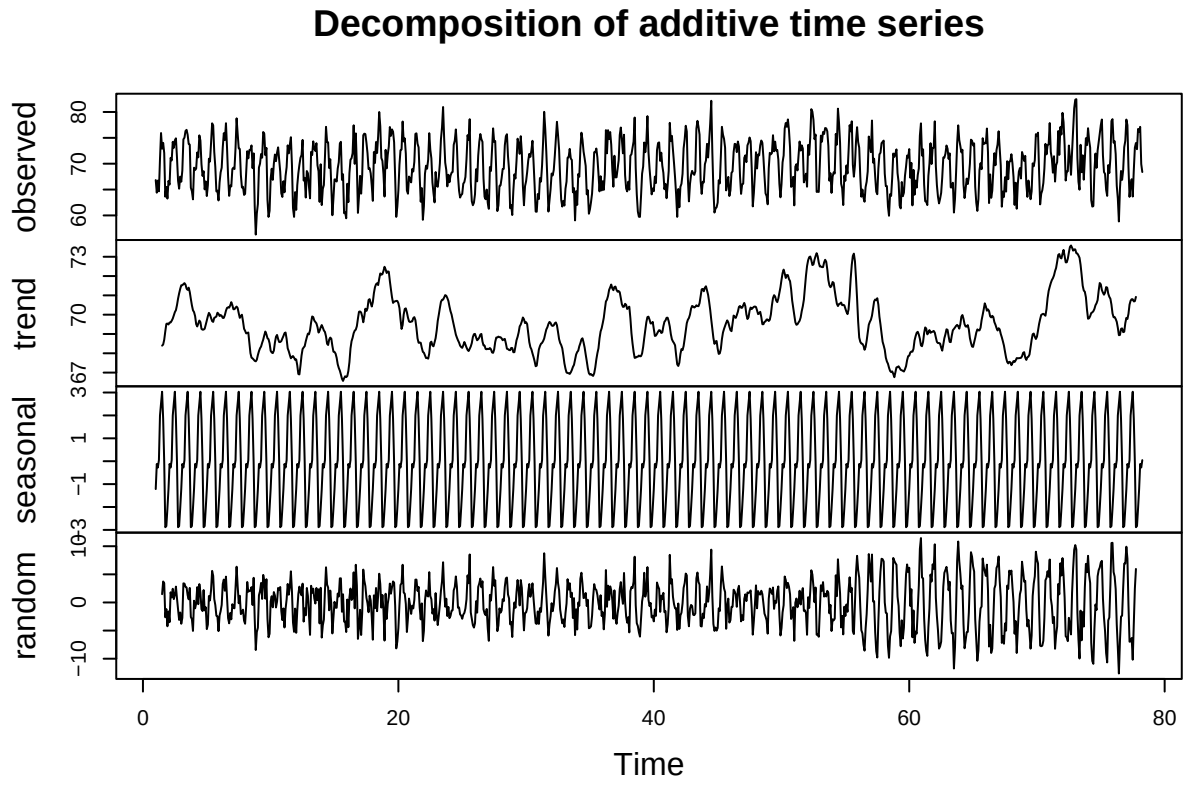



```
hist(temp.train.log, col="light blue", xlab="", main="", probability = TRUE)
curve(dnorm(x, mean(temp.train.log), sd(temp.train.log)), add = TRUE)
title(main = c("Histogram of Monthly Average Temperature Training Set", "after Log
↪  Transformation"), line = c(1, 2))
```

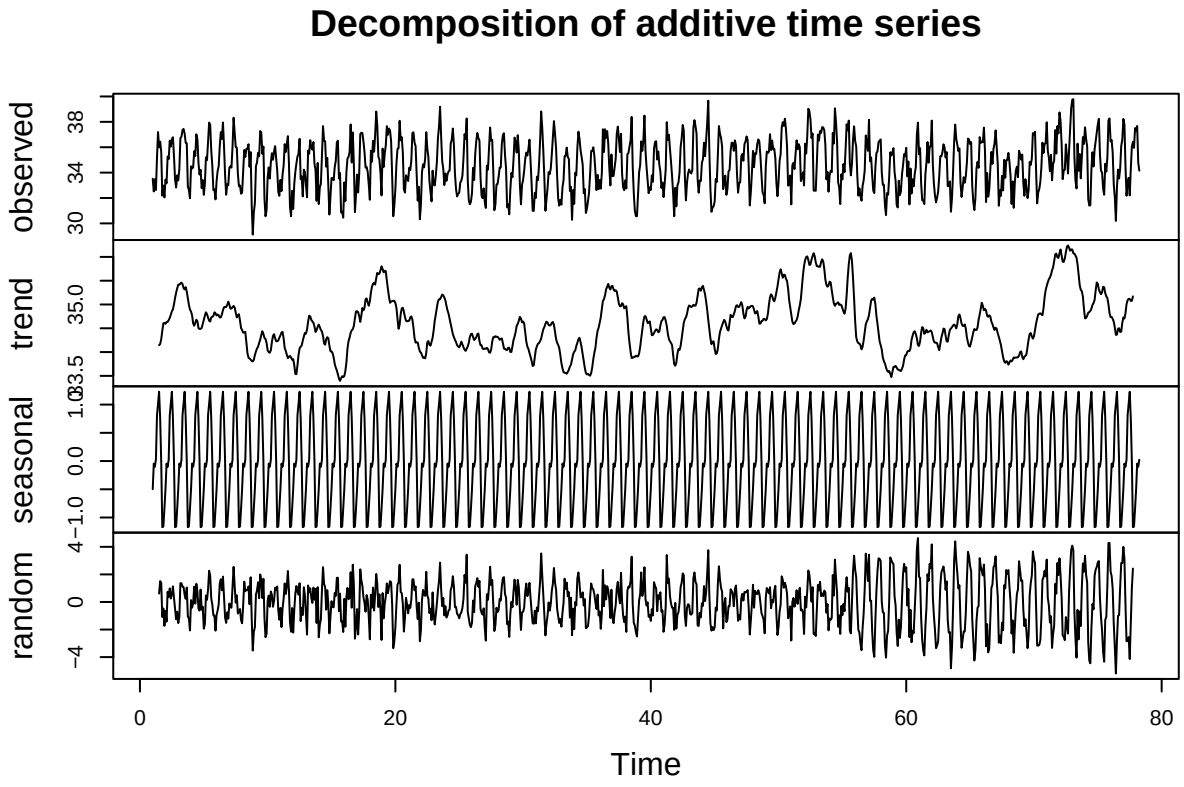


All of the histograms appear approximately normal; therefore, I choose to model the not transformed data.

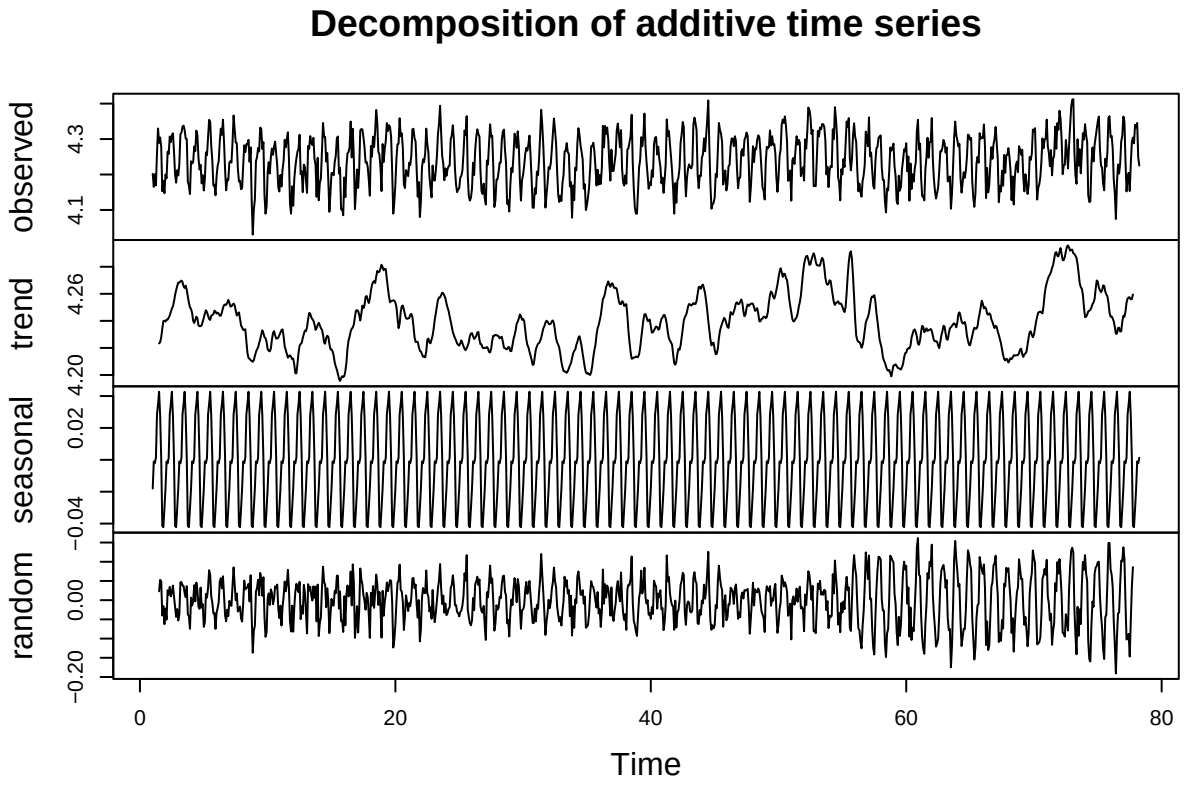
```
# plot the decomposition of not transformed data
y = ts(as.ts(temp.train), frequency = 12)
decomp = decompose(y)
plot(decomp)
```




```
# plot the decomposition of boxcox transformed data
y = ts(as.ts(temp.train.bc), frequency = 12)
decomp = decompose(y)
plot(decomp)
```



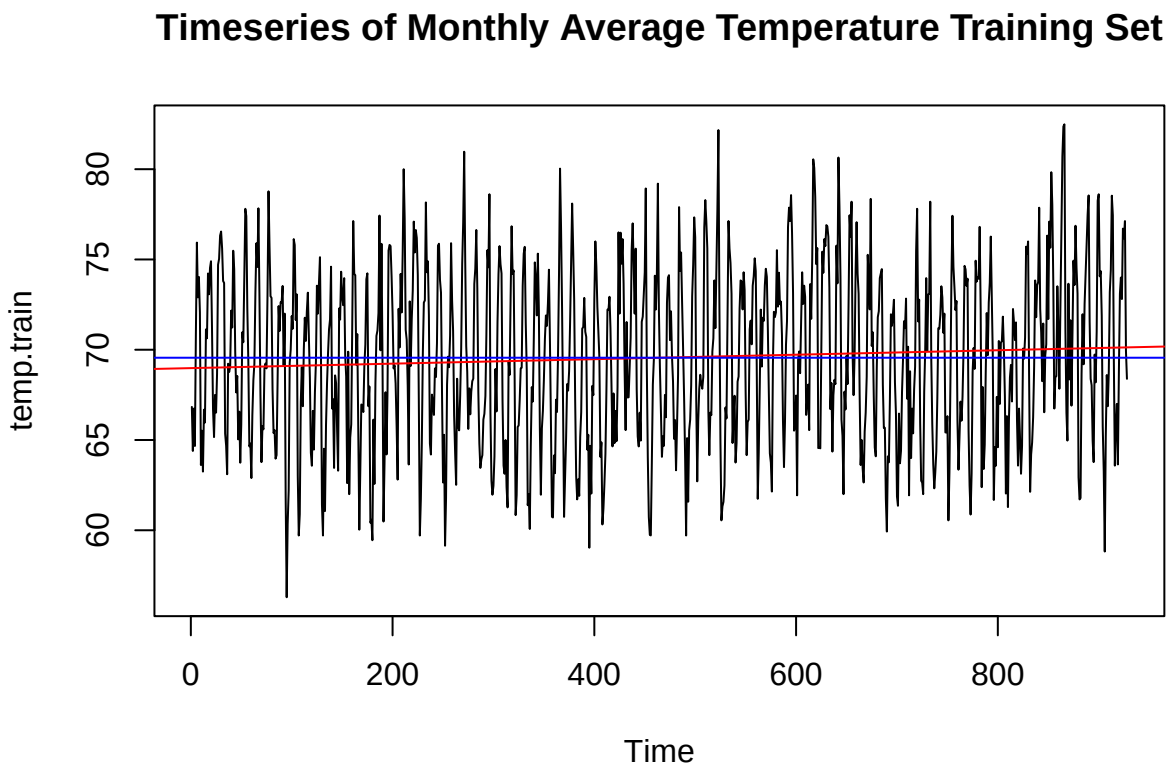

```
# plot the decomposition of log transformed data
y = ts(as.ts(temp.train.log), frequency = 12)
decomp = decompose(y)
plot(decomp)
```



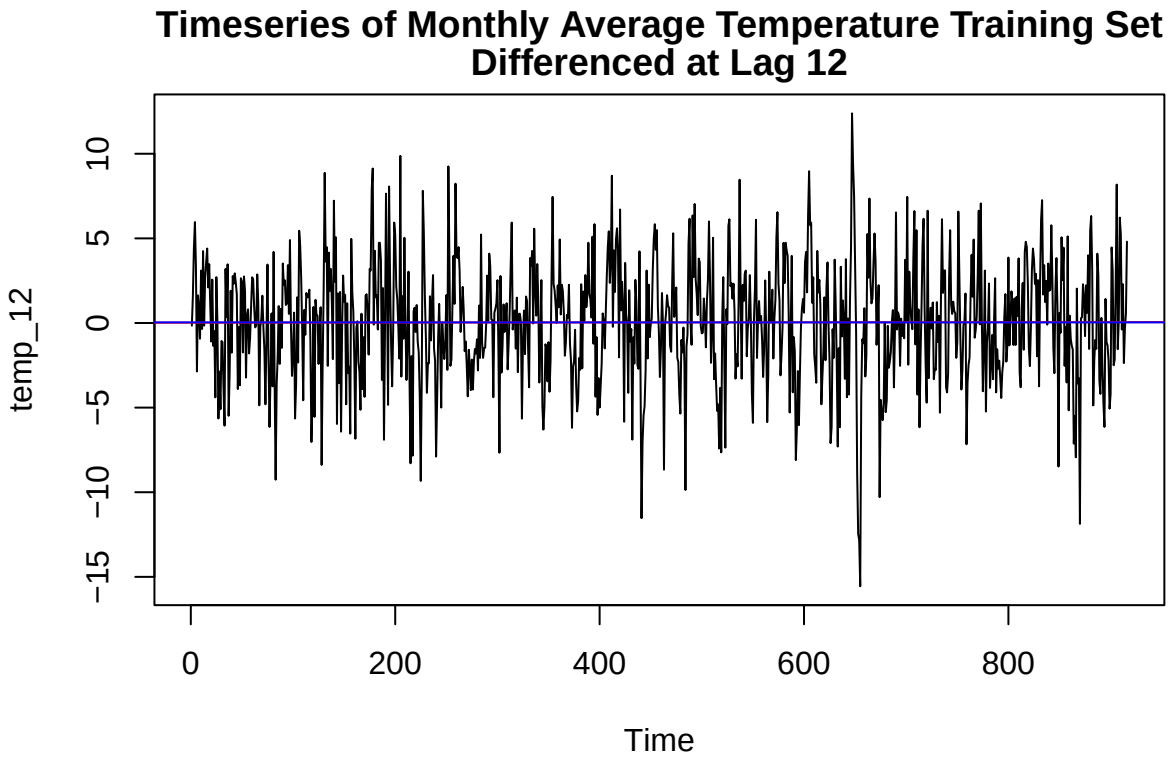
There appears to be a trend but it does not look to be modelable by a polynomial. There seems to be a seasonal pattern. The variance in the random plot for the not transformed, boxcox, and log transformed data appears to increase between 40 and 60; however the increase is not large and the variance remains constant before and after the increase.

Differencing

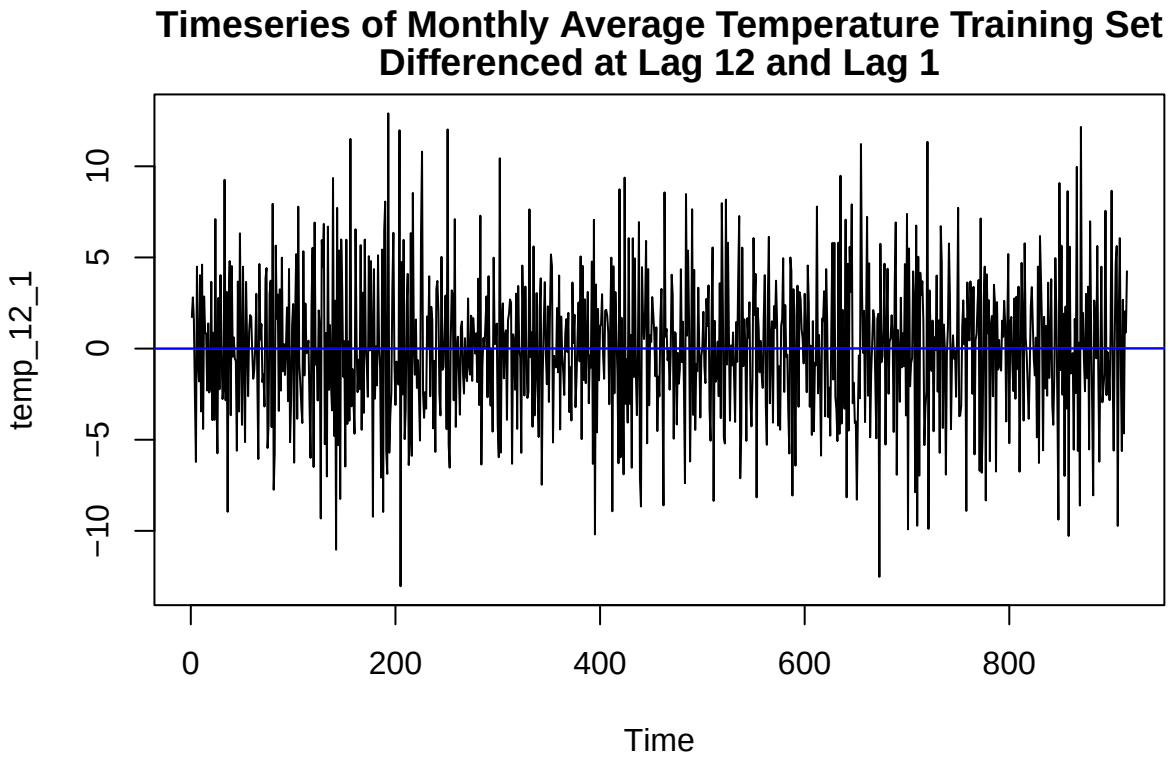
```
#time series plot with trend and mean for not differenced data
plot.ts(temp.train, main = "Timeseries of Monthly Average Temperature Training Set")
temp.train.fit <- lm(temp.train ~ as.numeric(1:length(temp.train)));
abline(temp.train.fit, col="red")
abline(h=mean(temp.train), col="blue")
```




```
#time series plot with trend and mean for differenced at lag 12 data
temp_12 = diff(temp.train, lag = 12)
plot.ts(temp_12, main = "")
temp_12.fit = lm(temp_12 ~ as.numeric(1:length(temp_12)))
abline(temp_12.fit, col = "red")
abline(h = mean(temp_12), col="blue")
title(main = c("Timeseries of Monthly Average Temperature Training Set", "Differenced at
↪  Lag 12"), line = c(1, 2))
```




```
#time series plot with trend and mean for differenced at lag 12 and 1 data
temp_12_1 = diff(temp_12, lag = 1)
plot.ts(temp_12_1, main = "")
temp_12_1.fit = lm(temp_12_1 ~ as.numeric(1:length(temp_12_1)))
abline(temp_12_1.fit, col = "red")
abline(h = mean(temp_12_1), col="blue")
title(main = c("Timeseries of Monthly Average Temperature Training Set", "Differenced at
↪  Lag 12 and Lag 1"), line = c(1, 2))
```




```
#means of not differenced, differenced at lag 12, and differenced at lag 12 and 1 data
print(paste("Monthly Average Temperature Training Set Mean:", mean(temp.train)))
```

```
## [1] "Monthly Average Temperature Training Set Mean: 69.5562285065058"
```

```
print(paste("Monthly Average Temperature Training Set Differenced at Lag 12 Mean:",
↪  mean(temp_12)))
```

```
## [1] "Monthly Average Temperature Training Set Differenced at Lag 12 Mean: 0.035617698415397"
```

```
cat("Monthly Average Temperature Training Set Differenced at Lag 12 and Lag 1 Mean:",
↪  mean(temp_12_1), sep="\n")
```

```
## Monthly Average Temperature Training Set Differenced at Lag 12 and Lag 1 Mean:
## 0.00542335
```

```
#variances of not differenced, differenced at lag 12, and differenced at lag 12 and 1
↪  data
print(paste("Monthly Average Temperature Training Set Variance:", var(temp.train)))
```

```
## [1] "Monthly Average Temperature Training Set Variance: 23.0059338731091"
```

```r
cat("Monthly Average Temperature Training Set Differenced at Lag 12 Variance:",
↪  var(temp_12), sep="\n")
```

```
## Monthly Average Temperature Training Set Differenced at Lag 12 Variance:
## 12.77494
```

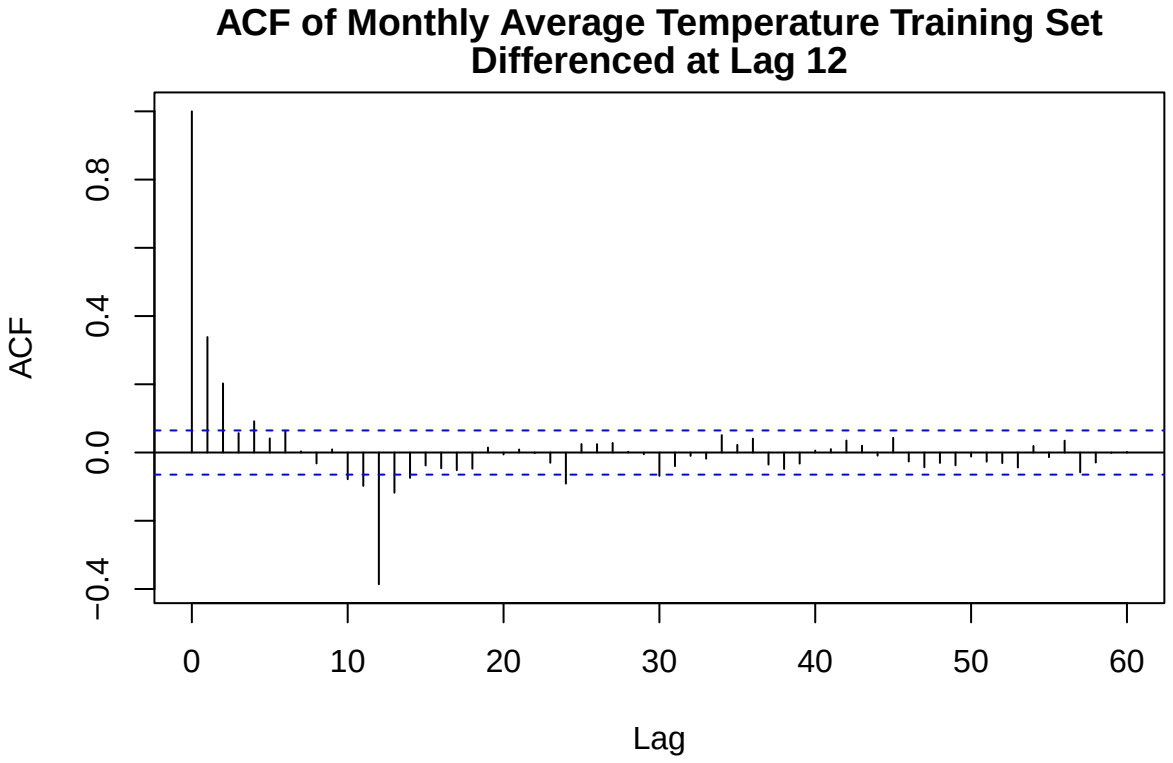
```r
cat("Monthly Average Temperature Training Set Differenced at Lag 12 and Lag 1 Variance:",
↪  var(temp_12_1), sep="\n")
```

```
## Monthly Average Temperature Training Set Differenced at Lag 12 and Lag 1 Variance:
## 16.89707
```

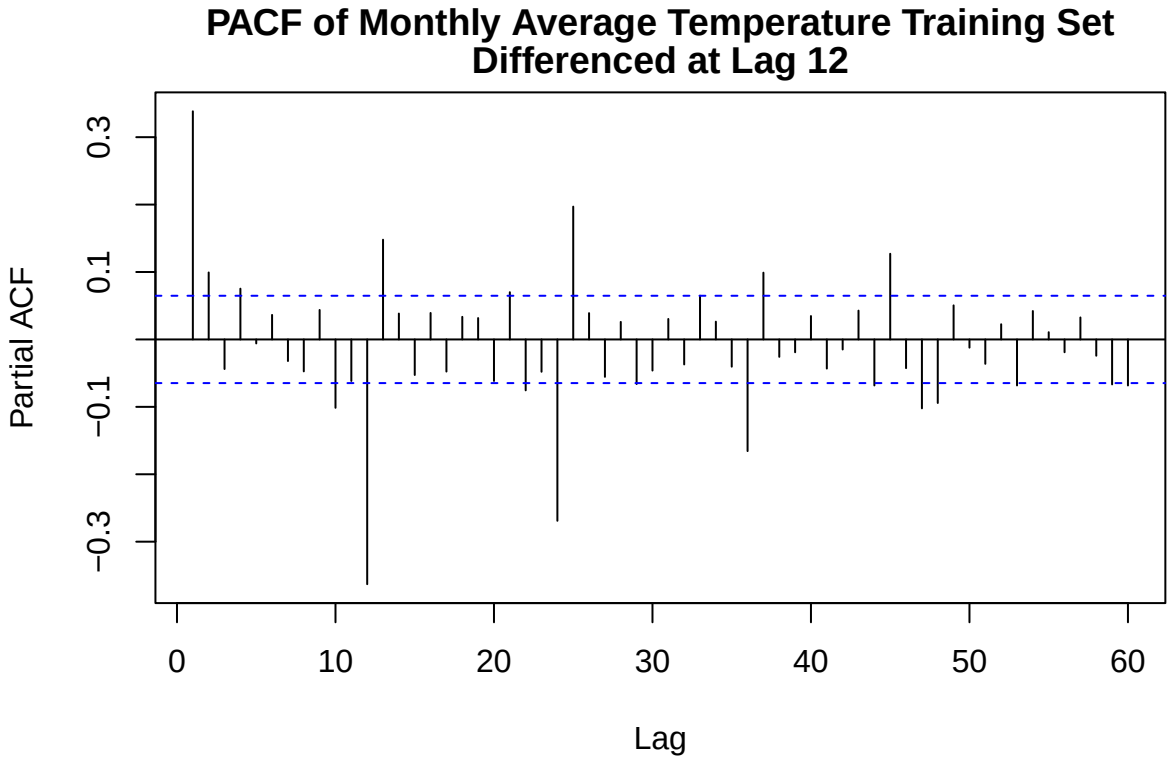
We first difference at lag 12 because the data is periodic in groups of 12 and then again at lag 1 because there is no apparent seasonality other than in groups of 12. Differencing at lag 12 significantly decreases the variance and moves the mean significantly closer to 0; differencing at lag 12 and 1 increases the variance and moves the mean slightly closer to 0 compared to the data set differenced only at lag 12. Therefore, I choose to only difference at lag 12 in order to maintain low variance, avoid over-differencing, and make the data stationary.

Model Identification

```r
#plot acf for data differenced at lag 12
acf(temp_12,lag.max=60, main="")
title(main = c("ACF of Monthly Average Temperature Training Set", "Differenced at Lag
↪  12"), line = c(1, 2))
```



```r
#plot pacf for data differenced at lag 12
pacf(temp_12,lag.max=60, main="")
title(main = c("PACF of Monthly Average Temperature Training Set", "Differenced at Lag
↪  12"), line = c(1, 2))
```



There are spikes in the acf at lags 0, 1, 2, 4, 10, 11, 12, 13, and 24; and spikes in the pacf at lags 1, 2, 4, 10, 12, 13, 24, 25, 36, 37, 45, 47 and 48. In the context of SARIMA models, the non-seasonal difference component d = 0 and the seasonal difference component D = 1 because the data is differenced only at lag 12. Candidate values for the non-seasonal AR component include p = 1, 2, 3, 13, 25, and 37; candidate values for the non-seasonal MA component include q = 0, 1, 2, 4, 10, 11, and 13; candidate values for the seasonal AR component include P = 1, 2, 3, 4; and candidate values for the seasonal MA component include Q = 1, 2. All combinations of p = 1, 2, 3, 13; q = 0, 1, 2, 4, 10, 11, 13; P = 1, 2, 3, 4; and Q = 1, 2 were evaluated. The five models possessing the lowest AICs are displayed.

```r
fit.i = arima(x = temp.train, order = c(13, 0, 0), seasonal = c(2, 1, 1, period = 12),
↪  method = "ML")
fit.i
```

```
##
## Call:
## arima(x = temp.train, order = c(13, 0, 0), seasonal = c(2, 1, 1, period = 12),
##     method = "ML")
##
## Coefficients:
##           ar1      ar2      ar3      ar4      ar5      ar6      ar7      ar8
```

```
##       -0.5513  -0.4573  -0.5310  -0.5008  -0.5618  -0.4689  -0.5032  -0.5793
## s.e.   0.0329   0.0365   0.0376   0.0386   0.0392   0.0382   0.0381   0.0390
##           ar9     ar10     ar11     ar12     ar13     sar1     sar2     sma1
##       -0.4978  -0.4649  -0.3897   0.2662   0.0802  -0.1264  -0.1505  -0.3664
## s.e.   0.0385   0.0386   0.0395   0.0485   0.0331   0.1005   0.0523   0.1220
##
## sigma^2 estimated as 6.671:  log likelihood = -2198.46,  aic = 4430.93
```

```
fit.ii = arima(x = temp.train, order = c(13, 0, 2), seasonal = c(4, 1, 2, period = 12),
↪  method = "ML")
fit.ii
```

```
##
## Call:
## arima(x = temp.train, order = c(13, 0, 2), seasonal = c(4, 1, 2, period = 12),
##     method = "ML")
##
## Coefficients:
##           ar1     ar2      ar3      ar4      ar5      ar6      ar7      ar8
##       -0.4503  -0.196  -0.3569  -0.3177  -0.3919  -0.2979  -0.3297  -0.4167
## s.e.   0.0132     NaN   0.0202   0.0150      NaN   0.0032   0.0370   0.0155
##           ar9     ar10     ar11     ar12     ar13      ma1      ma2     sar1
##       -0.2844  -0.2414  -0.1804   0.3154   0.1403  -0.0953  -0.2025  -0.1656
## s.e.      NaN      NaN   0.0124   0.0171   0.0103   0.0556      NaN   0.0156
##         sar2    sar3    sar4     sma1     sma2
##       0.6054  0.3156  0.2036  -0.2028  -0.7261
## s.e.     NaN  0.0038     NaN      NaN      NaN
##
## sigma^2 estimated as 6.586:  log likelihood = -2193.94,  aic = 4431.87
```

```
fit.iii = arima(x = temp.train, order = c(13, 0, 0), seasonal = c(2, 1, 2, period = 12),
↪  method = "ML")
fit.iii
```

```
##
## Call:
## arima(x = temp.train, order = c(13, 0, 0), seasonal = c(2, 1, 2, period = 12),
##     method = "ML")
##
## Coefficients:
##           ar1      ar2      ar3      ar4      ar5      ar6      ar7      ar8
##       -0.5484  -0.4545  -0.5302  -0.4973  -0.5614  -0.4679  -0.5018  -0.5801
## s.e.   0.0331   0.0367   0.0376   0.0389   0.0392   0.0383   0.0382   0.0390
##           ar9     ar10     ar11    ar12    ar13    sar1     sar2     sma1
##       -0.4933  -0.4603  -0.3831   0.257  0.0797  0.2423  -0.1944  -0.7237
## s.e.   0.0389   0.0391   0.0406   0.050  0.0332  0.3730   0.0686   0.3646
##         sma2
##       0.2302
## s.e.  0.2101
##
## sigma^2 estimated as 6.665:  log likelihood = -2198.07,  aic = 4432.15
```

```
fit.iii.0 = arima(x = temp.train, order = c(13, 0, 0), seasonal = c(2, 1, 2, period =
↪  12), fixed = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 0, NA, NA, NA),
↪  method = "ML")
fit.iii.0
```

```
##
## Call:
## arima(x = temp.train, order = c(13, 0, 0), seasonal = c(2, 1, 2, period = 12),
##     fixed = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 0, NA, NA,
##         NA), method = "ML")
##
## Coefficients:
##           ar1      ar2      ar3      ar4      ar5      ar6      ar7      ar8
##       -0.5496  -0.4560  -0.5307  -0.4988  -0.5614  -0.4682  -0.5023  -0.5796
## s.e.   0.0331   0.0366   0.0377   0.0388   0.0392   0.0382   0.0382   0.0391
##           ar9     ar10     ar11     ar12     ar13  sar1     sar2     sma1    sma2
##       -0.4955  -0.4625  -0.3868   0.2624   0.0799     0  -0.1755  -0.4886  0.0898
## s.e.   0.0387   0.0388   0.0399   0.0489   0.0332     0   0.0645   0.0567  0.0615
##
## sigma^2 estimated as 6.668:  log likelihood = -2198.25,  aic = 4430.5
```

```
fit.iv = arima(x = temp.train, order = c(13, 0, 1), seasonal = c(2, 1, 1, period = 12),
↪  method = "ML")
fit.iv
```

```
##
## Call:
## arima(x = temp.train, order = c(13, 0, 1), seasonal = c(2, 1, 1, period = 12),
##     method = "ML")
##
## Coefficients:
##           ar1      ar2      ar3      ar4      ar5     ar6      ar7      ar8
##       -0.4367  -0.3966  -0.4752  -0.4356  -0.5002  -0.399  -0.4451  -0.5177
## s.e.      NaN      NaN      NaN      NaN      NaN     NaN      NaN   0.0044
##           ar9     ar10     ar11     ar12     ar13      ma1     sar1     sar2
##       -0.4257  -0.4031  -0.3304   0.3124   0.0544  -0.1153  -0.1310  -0.1510
## s.e.      NaN      NaN      NaN   0.0201   0.0251      NaN   0.0305   0.0299
##          sma1
##       -0.3577
## s.e.   0.0589
##
## sigma^2 estimated as 6.671:  log likelihood = -2198.43,  aic = 4432.86
```

The first coefficient for the SAR component pertaining to fit.iii possesses a confidence interval that includes 0. I fix the first coefficient for the SAR component of that model to 0 in an attempt to lower the AIC of that model and name the new model fit.iii.0. The new model does have a lower AIC. The models that possess the lowest AICs are fit.i and fit.iii.0.

Diagnostic Checking

```r
#install.packages("UnitCircle")
library(UnitCircle)

par(mfrow = c(2, 2))
#fit.i AR
uc.check(pol_ = c(1, 0.5513, 0.4573, 0.5310, 0.5008, 0.5618, 0.4689, 0.5032, 0.5793,
↪  0.4978, 0.4649, 0.3897, -0.2662, -0.0802), plot_output = TRUE)
```

```
##         real   complex outside
## 1   0.508440  0.880397    TRUE
## 2  -0.930605  0.515851    TRUE
## 3  -0.500812 -0.903812    TRUE
## 4   0.867375 -0.501615    TRUE
## 5   0.010827  1.055502    TRUE
## 6  -1.045676  0.000000    TRUE
## 7   0.010827 -1.055502    TRUE
## 8   0.867375  0.501615    TRUE
## 9  -0.500812  0.903812    TRUE
## 10 -0.930605 -0.515851    TRUE
## 11  0.508440 -0.880397    TRUE
## 12  2.026420  0.000000    TRUE
## 13 -4.210397  0.000000    TRUE
## *Results are rounded to 6 digits.
```

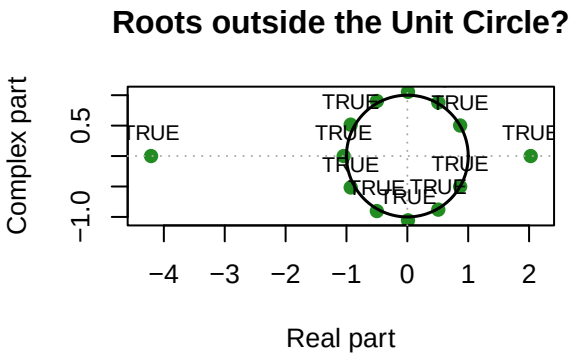
```r
#fit.i SAR
uc.check(pol_ = c(1, 0.1264, 0.1505), plot_output = TRUE)
```

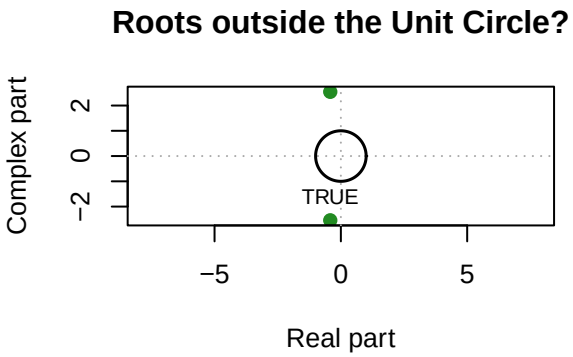
```
##        real   complex outside
## 1 -0.419934  2.543261    TRUE
## 2 -0.419934 -2.543261    TRUE
## *Results are rounded to 6 digits.
```

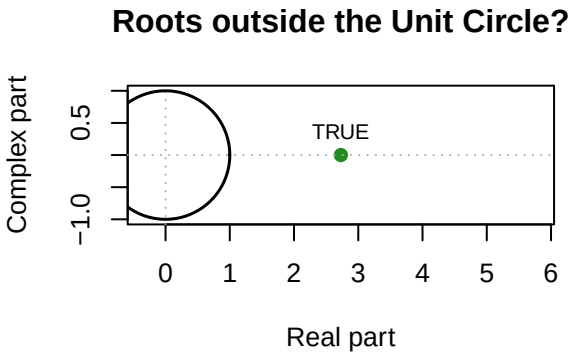
```r
#fit.i SMA
uc.check(pol_ = c(1, -0.3664), plot_output = TRUE)
```

```
##       real complex outside
## 1 2.729258       0    TRUE
## *Results are rounded to 6 digits.
```

```r
par(mfrow = c(2, 2))
```








```
#fit.iii.0 AR
uc.check(pol_ = c(1, 0.5496, 0.4560, 0.5307, 0.4988, 0.5614, 0.4682, 0.5023, 0.5796,
↪  0.4955, 0.4625, 0.3868, -0.2624, -0.0799), plot_output = TRUE)
```

```
##         real   complex outside
## 1   0.508795  0.881118    TRUE
## 2  -0.932222  0.516046    TRUE
## 3  -0.500649 -0.904849    TRUE
## 4   0.867489 -0.501724    TRUE
## 5   0.011283  1.056595    TRUE
## 6  -1.047162  0.000000    TRUE
## 7   0.011283 -1.056595    TRUE
## 8   0.867489  0.501724    TRUE
## 9  -0.500649  0.904849    TRUE
## 10 -0.932222 -0.516046    TRUE
## 11  0.508795 -0.881118    TRUE
## 12  2.030392  0.000000    TRUE
## 13 -4.176727  0.000000    TRUE
## *Results are rounded to 6 digits.
```

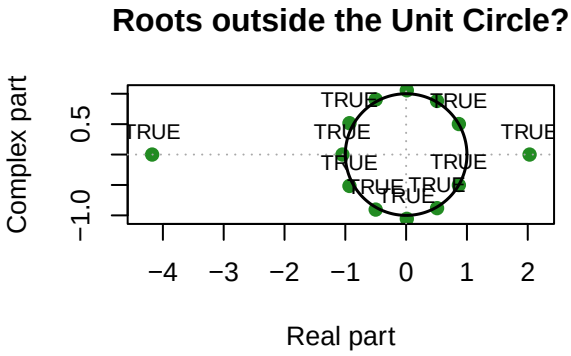
```
#fit.iii.0 SAR
uc.check(pol_ = c(1, 0, 0.1755), plot_output = TRUE)
```

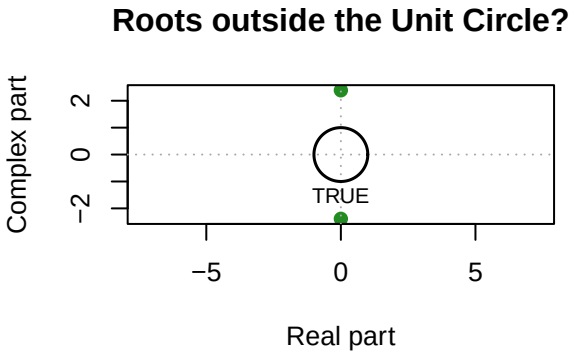
```
##   real  complex outside
```

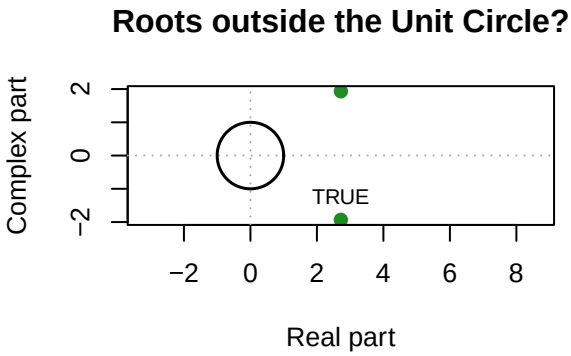
```
## 1    0  2.38705    TRUE
## 2    0 -2.38705    TRUE
## *Results are rounded to 6 digits.
```

```
#fit.iii.0 SMA
uc.check(pol_ = c(1, -0.4886, 0.0898), plot_output = TRUE)
```

```
##      real   complex outside
## 1 2.72049  1.932561    TRUE
## 2 2.72049 -1.932561    TRUE
## *Results are rounded to 6 digits.
```



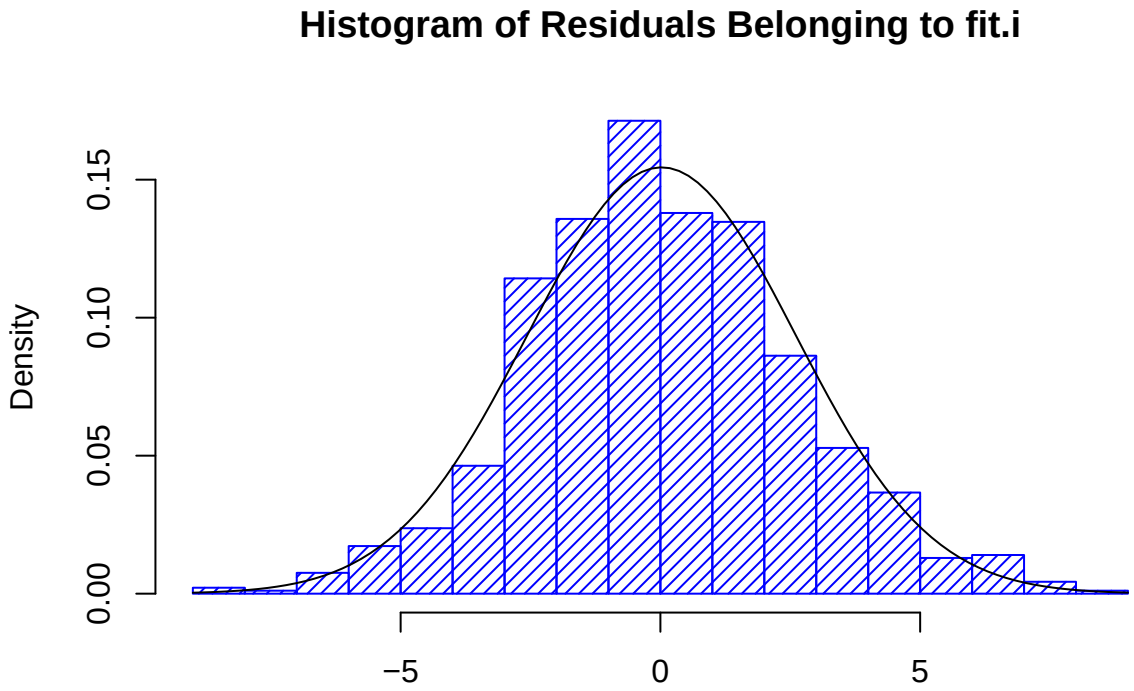




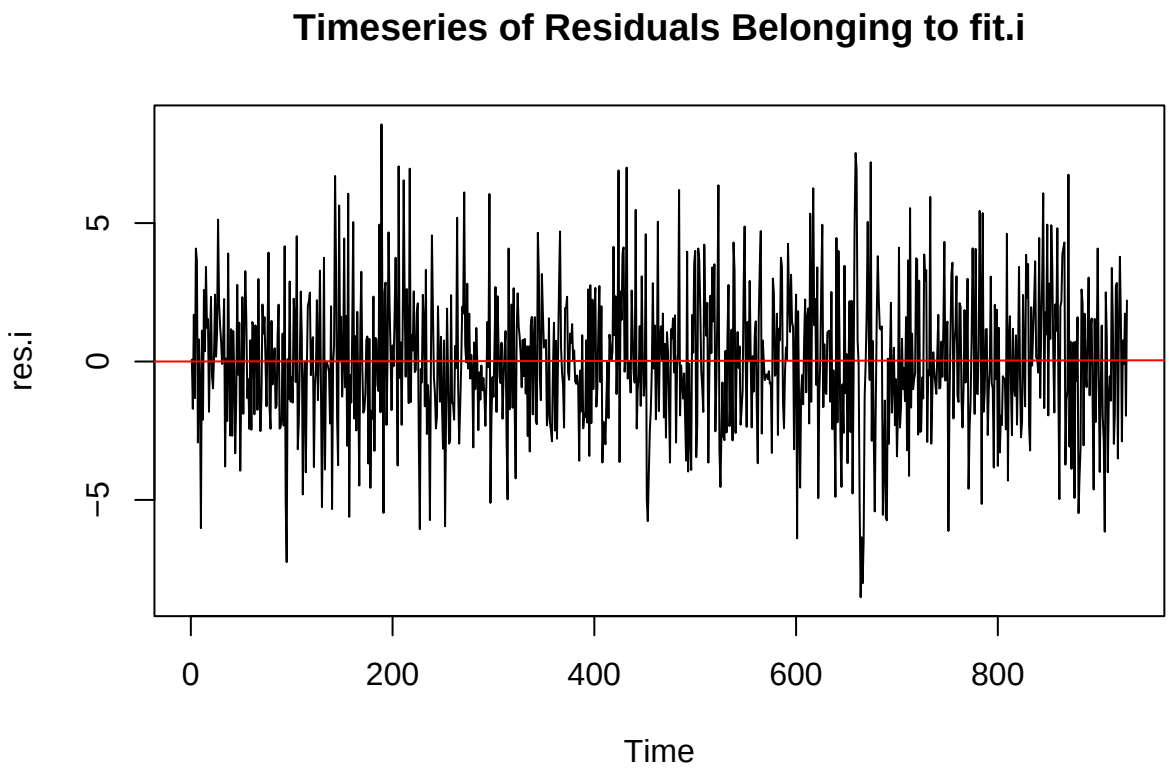
If the roots of the AR and SAR polynomials of a model lie outside of the unit circle, then that model is stationary. If the roots of the MA and SMA polynomials of a model lie outside of the unit circle, then that model is invertible. All of the roots for all of the components of both fit.i and fit.iii lie outside of the unit circle, therefore, both models are stationary and invertible. No MA component is checked for either model because neither model possesses an MA component.

Tests include Shapiro-Wilk normality test, Box-Pierce test, Box-Ljung test, and Mcleod-Li for normality, correlation (linear dependence), and nonlinear relationships between the residuals. Lag = 31 because $\sqrt{n} \approx 31$.

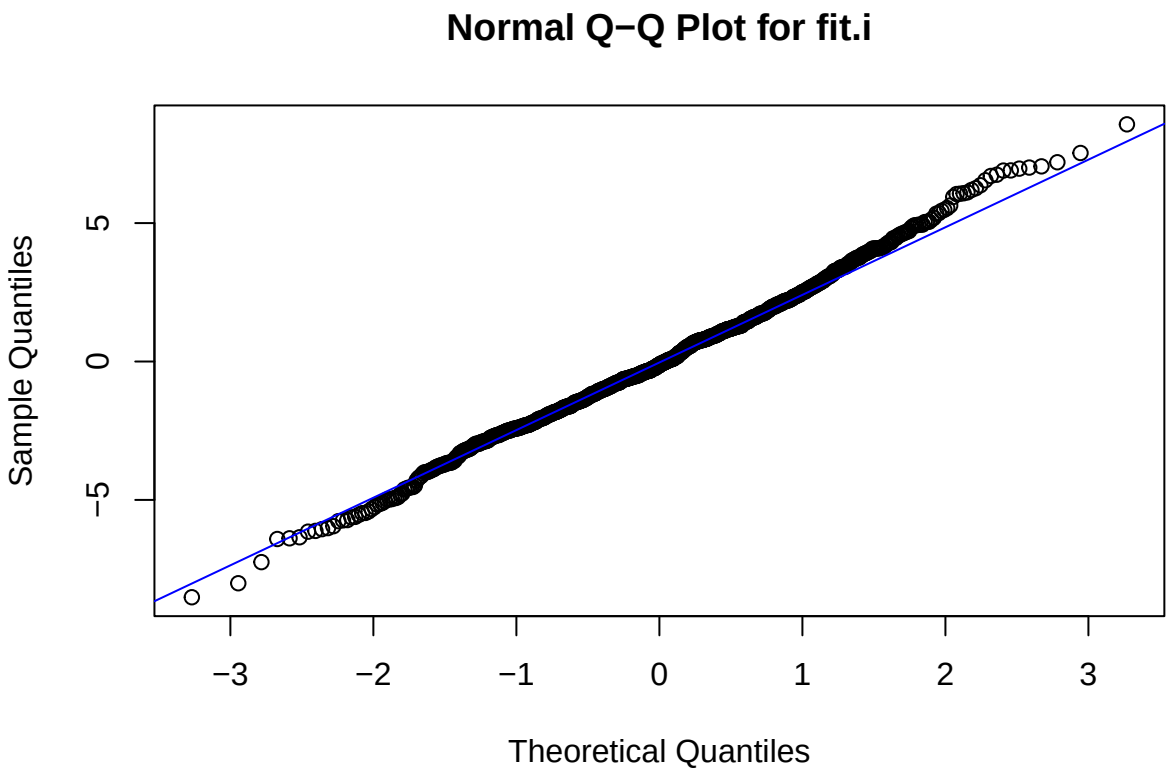
```
#fit.i
res.i = residuals(fit.i)
hist(res.i,density=20,breaks=20, col="blue", xlab="", prob=TRUE, main = "Histogram of
↪  Residuals Belonging to fit.i")
curve(dnorm(x, mean(res.i), sd(res.i)), add=TRUE )
```



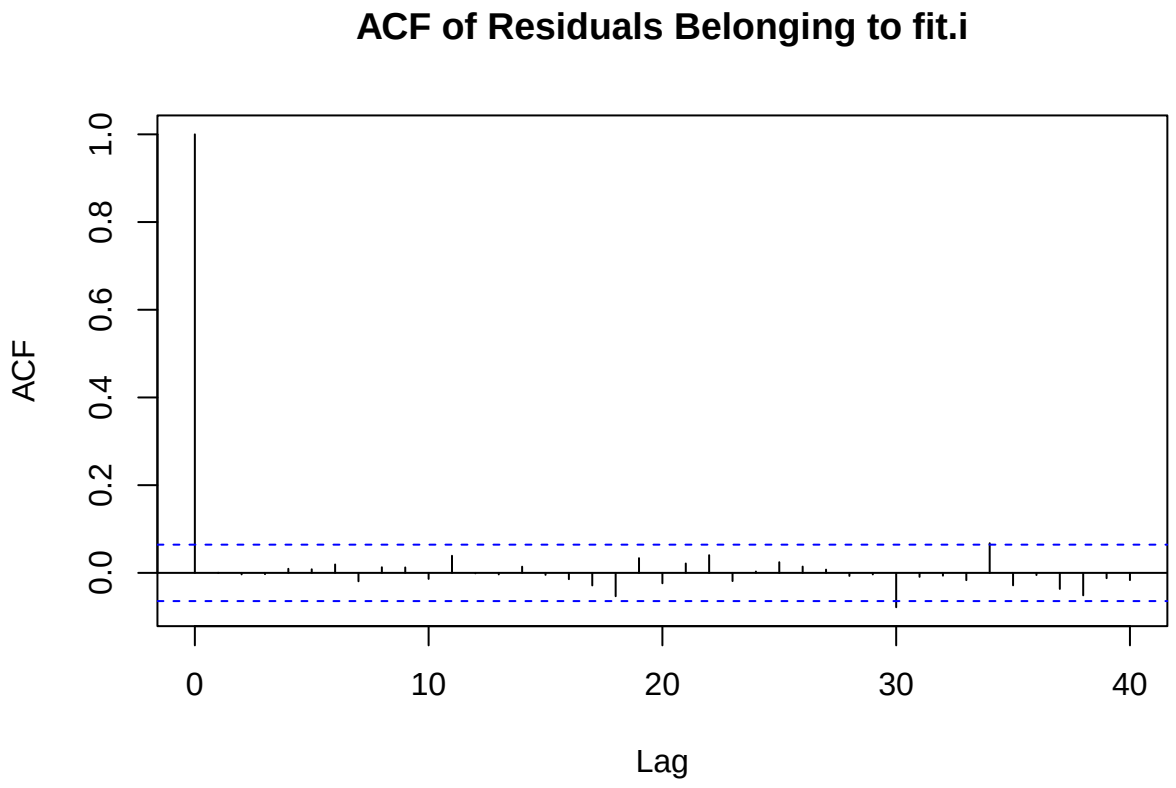

```
plot.ts(res.i, main = "Timeseries of Residuals Belonging to fit.i")
res.i.fit <- lm(res.i ~ as.numeric(1:length(res.i)))
abline(res.i.fit, col="red")
abline(h=mean(res.i.fit), col="blue")
```



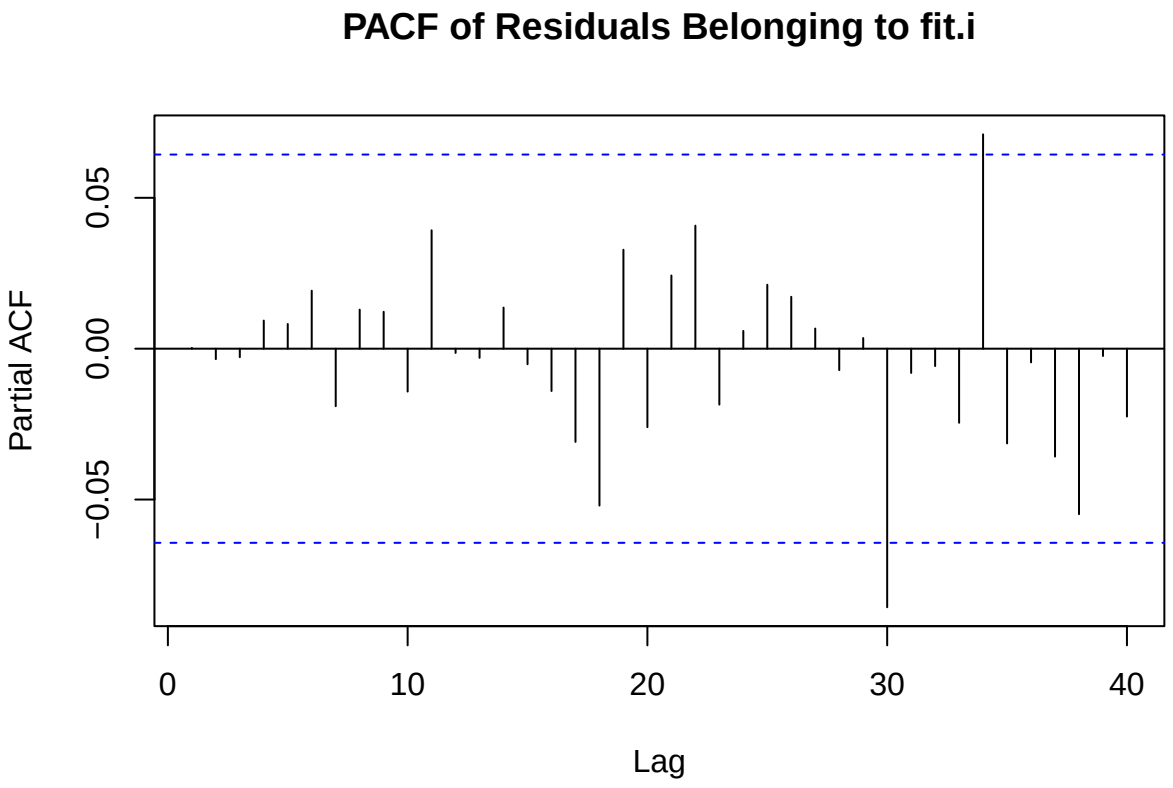

```
qqnorm(res.i,main= "Normal Q-Q Plot for fit.i")
qqline(res.i,col="blue")
```




```
acf(res.i, lag.max=40, main = "ACF of Residuals Belonging to fit.i")
```




```
pacf(res.i, lag.max=40, main = "PACF of Residuals Belonging to fit.i")
```




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

```
##
##  Shapiro-Wilk normality test
##
## data:  res.i
## W = 0.99671, p-value = 0.05144
```

```
Box.test(res.i, lag = 31, type = c("Box-Pierce"), fitdf = 16)
```

```
##
##  Box-Pierce test
##
## data:  res.i
## X-squared = 16.976, df = 15, p-value = 0.3203
```

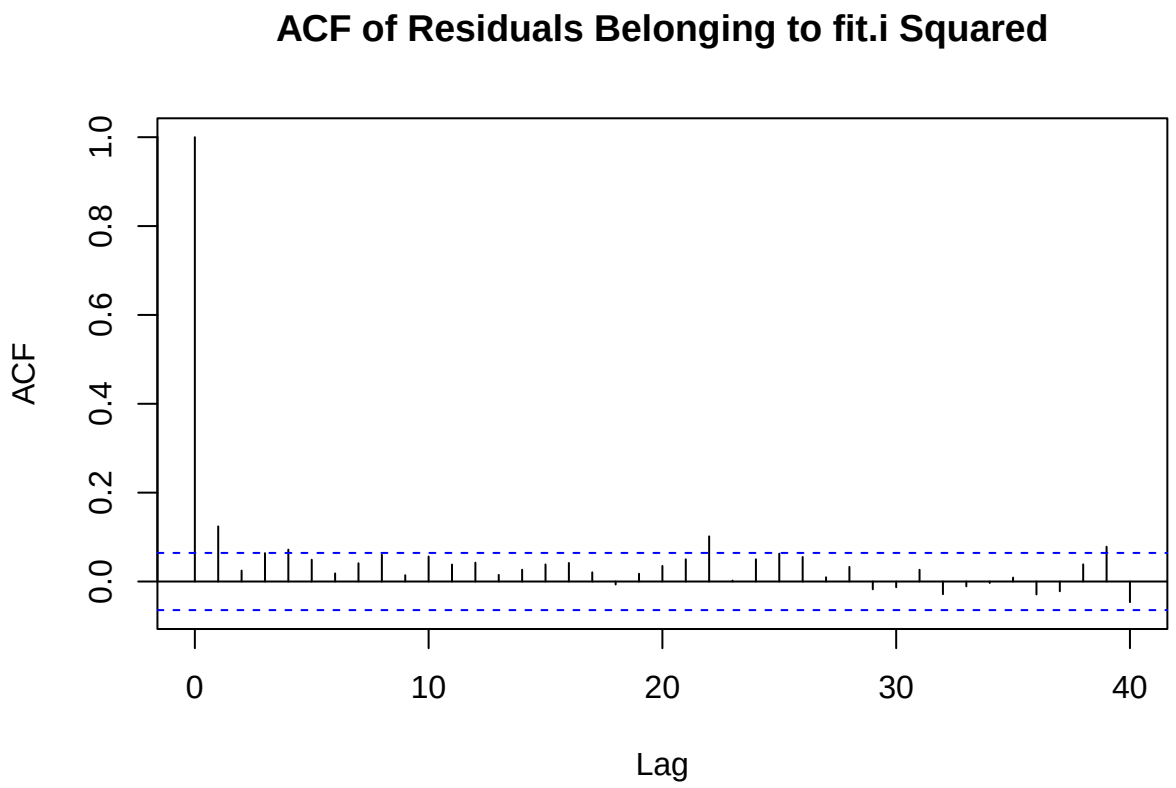
```
Box.test(res.i, lag = 31, type = c("Ljung-Box"), fitdf = 16)
```

```
##
##  Box-Ljung test
##
## data:  res.i
## X-squared = 17.419, df = 15, p-value = 0.2944
```

```
Box.test(res.i^2, lag = 31, type = c("Ljung-Box"), fitdf = 0)
```

```
##
##  Box-Ljung test
##
## data:  res.i^2
## X-squared = 66.598, df = 31, p-value = 0.0002099
```

```
acf(res.i^2, lag.max=40, main = "ACF of Residuals Belonging to fit.i Squared")
```



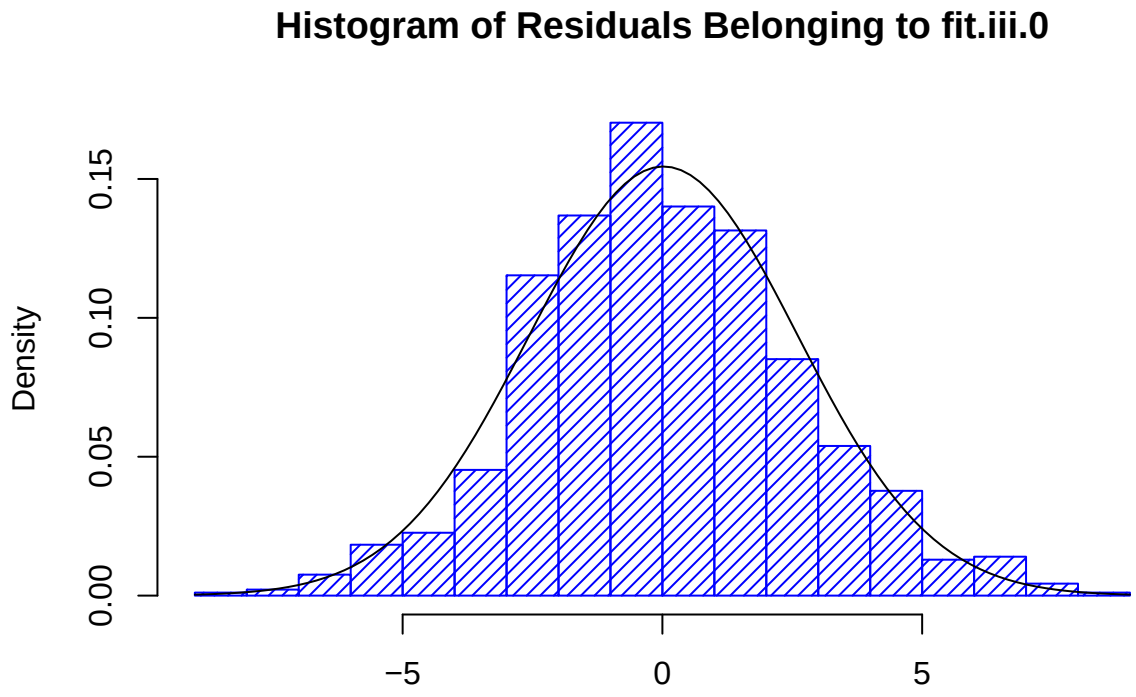

```
ar(res.i, aic = TRUE, order.max = NULL, method = c("yule-walker"))
```

```
##
## Call:
## ar(x = res.i, aic = TRUE, order.max = NULL, method = c("yule-walker"))
##
##
## Order selected 0  sigma^2 estimated as  6.67
```

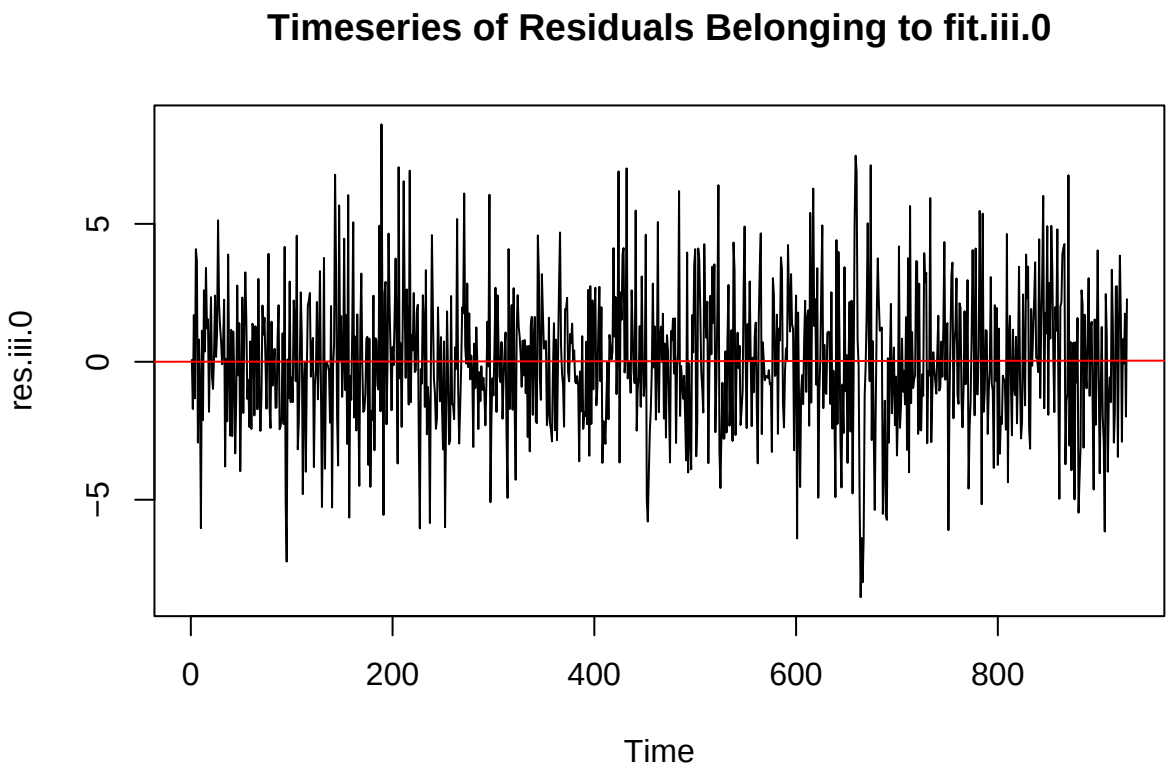
The histogram of the residuals is approximately normally distributed. fit.i passes the Shapiro-Wilk normality test, Box-Pierce test, and Box-Ljung test with a p-value > 0.05 ; however, it does not pass the Mcleod-Li test for autoregressive conditional heteroskedasticity. This may be due to the slight increase in variance referenced earlier in the decomposition of the data or due to the sharp changes in behavior referenced in

the exploratory analysis. The residuals follow an AR(0) process with a sample mean of 6.67, which would ideally be close to 0. The graph of the residuals appears stationary with a mean of 0. The Q-Q Plot is approximately a straight line which is an indication of normality. The ACF of the residuals, PACF of the residuals, and ACF of the residuals squared posses lags slightly outside of the confidence interval; Bartlett's theorem states that confidence intervals provided by R are very conservative. Since the lags are only slightly outside of the confidence interval, we may take them to resemble Gaussian White noise.

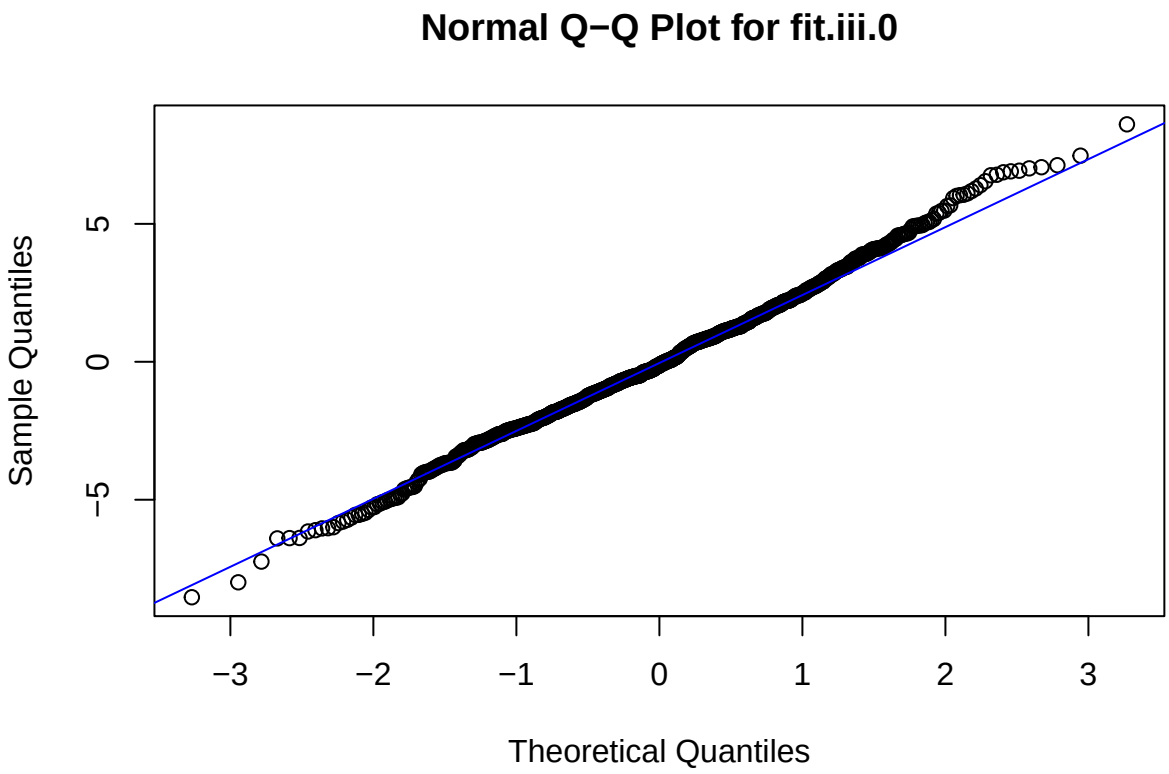
```
#fit.iii.0
res.iii.0 = residuals(fit.iii.0)
hist(res.iii.0,density=20,breaks=20, col="blue", xlab="", prob=TRUE, main = "Histogram of
↪  Residuals Belonging to fit.iii.0")
curve(dnorm(x, mean(res.iii.0), sd(res.iii.0)), add=TRUE )
```



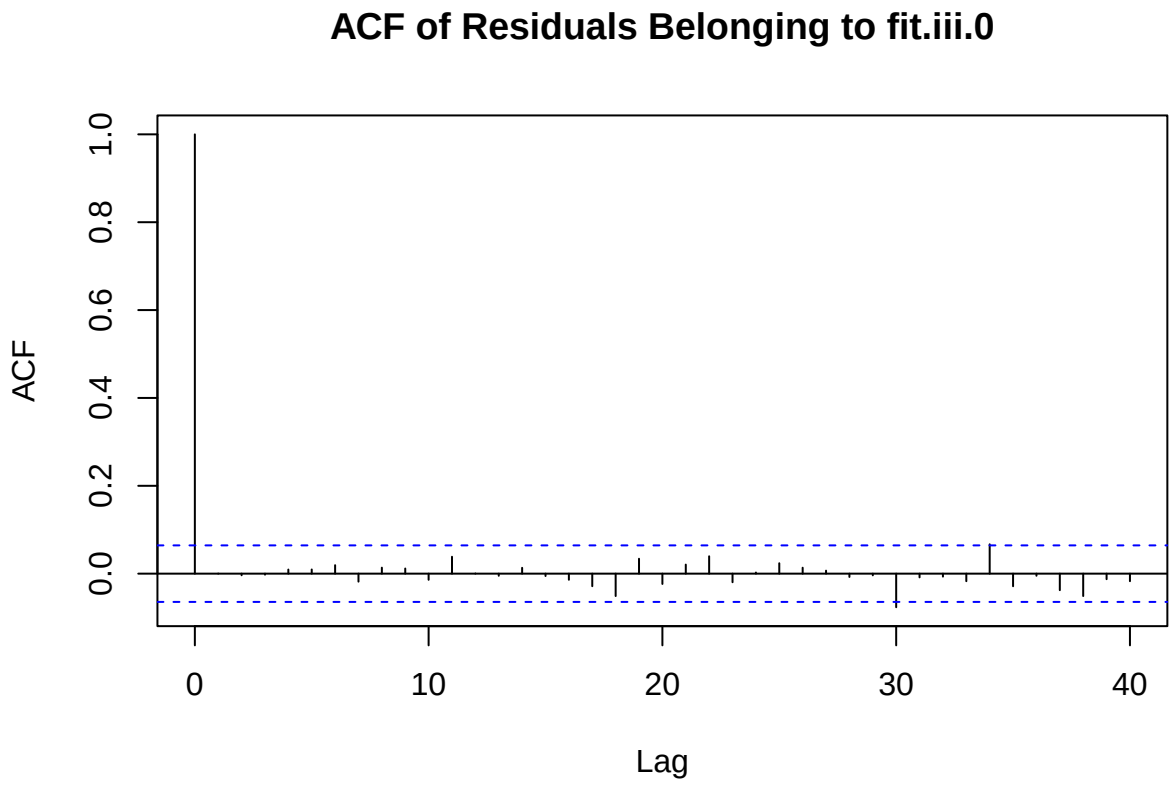

```
plot.ts(res.iii.0, main = "Timeseries of Residuals Belonging to fit.iii.0")
res.iii.0.fit <- lm(res.iii.0 ~ as.numeric(1:length(res.iii.0)))
abline(res.iii.0.fit, col="red")
abline(h=mean(res.iii.0.fit), col="blue")
```



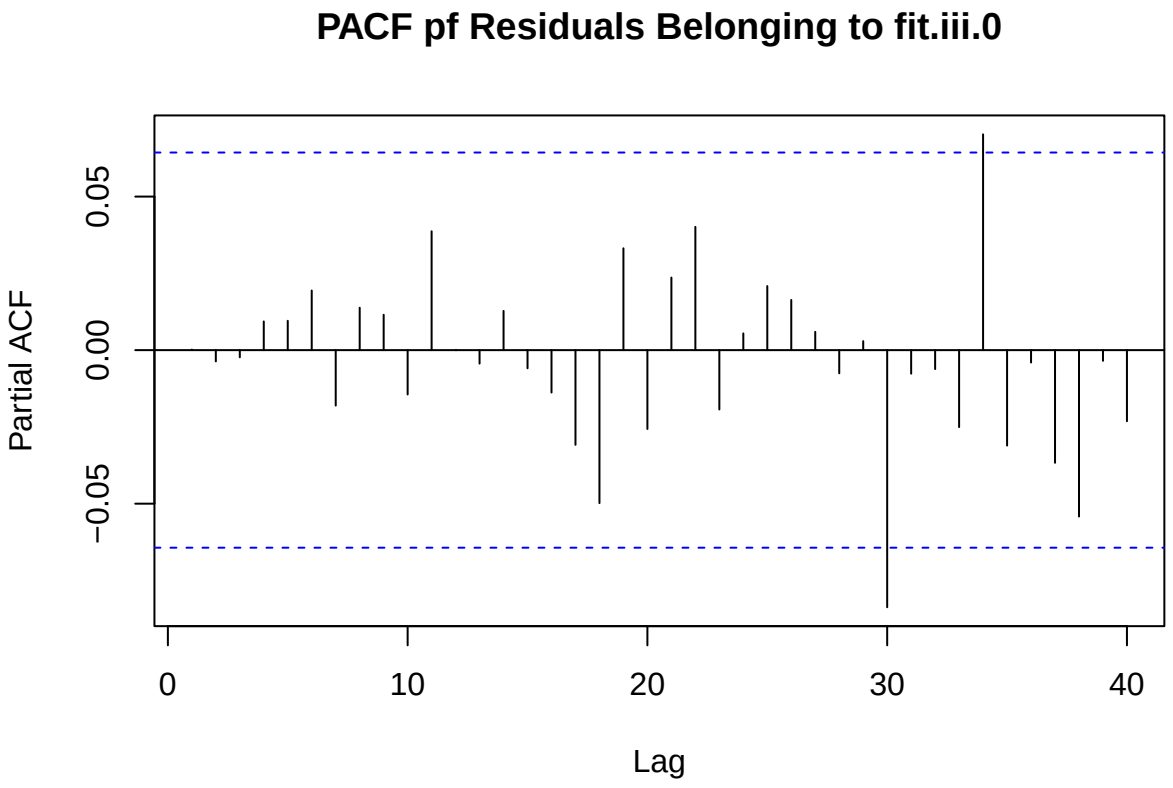

```
qqnorm(res.iii.0,main= "Normal Q-Q Plot for fit.iii.0")
qqline(res.iii.0,col="blue")
```




```
acf(res.iii.0, lag.max=40, main = "ACF of Residuals Belonging to fit.iii.0")
```




```
pacf(res.iii.0, lag.max=40, main = "PACF pf Residuals Belonging to fit.iii.0")
```




```
shapiro.test(res.iii.0)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.iii.0
## W = 0.99661, p-value = 0.04368
```

```
Box.test(res.iii.0, lag = 31, type = c("Box-Pierce"), fitdf = 16)
```

```
##
##  Box-Pierce test
##
## data:  res.iii.0
## X-squared = 16.348, df = 15, p-value = 0.3593
```

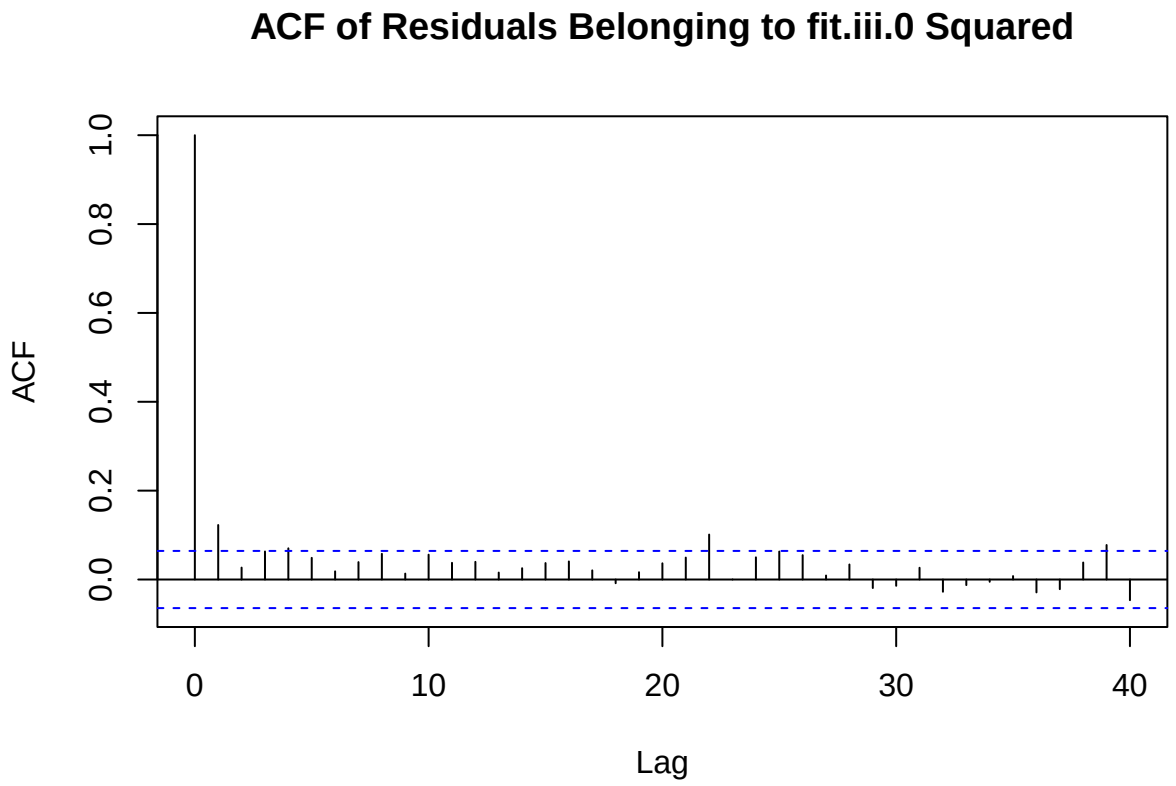
```
Box.test(res.iii.0, lag = 31, type = c("Ljung-Box"), fitdf = 16)
```

```
##
##  Box-Ljung test
##
## data:  res.iii.0
## X-squared = 16.773, df = 15, p-value = 0.3326
```

```
Box.test(res.iii.0^2, lag = 31, type = c("Ljung-Box"), fitdf = 0)
```

```
##
##  Box-Ljung test
##
## data:  res.iii.0^2
## X-squared = 65.24, df = 31, p-value = 0.0003114
```

```
acf(res.iii.0^2, lag.max=40, main = "ACF of Residuals Belonging to fit.iii.0 Squared")
```




```
ar(res.iii.0, aic = TRUE, order.max = NULL, method = c("yule-walker"))
```

```
##
## Call:
## ar(x = res.iii.0, aic = TRUE, order.max = NULL, method = c("yule-walker"))
##
##
## Order selected 0  sigma^2 estimated as  6.667
```

The histogram of the residuals is approximately normally distributed. fit.iii.0 passes the Box-Pierce test and Box-Ljung test with a p-value > 0.05 ; however, it does not pass the Shapiro-Wilk normality test and Mcleod-Li test for autoregressive conditional heteroskedasticity. This may be due to the slight increase in variance referenced earlier in the decomposition of the data, the sharp changes in behavior referenced in

the exploratory analysis, or the model fitting the data poorly. The residuals follow an AR(0) process with a sample mean of 6.667, which would ideally be close to 0. The graph of the residuals appears stationary with a mean of 0. The Q-Q Plot is approximately a straight line however, it is less straight than that of fit.i. The ACF of the residuals, PACF of the residuals, and ACF of the residuals squared posses lags slightly outside of the confidence interval; Bartlett's theorem states that confidence intervals provided by R are very conservative. Since the lags are only slightly outside of the confidence interval, we may take them to resemble Gaussian White noise.

fit.i appears to be the better fitting model on account of it's ability to pass the Shapiro-Wilk normality test and Q-Q Plot forming a slightly straighter line than that of fit.iii.0 (normally distributed residuals). Both the chosen model, fit.i, and the model possesing the second highest AIC consist of AR, MA, SAR, and SMA components suggested by the ACF and PACF; however, neither of these models posses an MA component and I would have assumed that the models with the lowest AIC posses an MA component due to the results of the ACF.

fit.i in algebraic form: $(1+0.5513B+0.4573B^2+0.5310B^3+0.5008B^4+0.5618B^5+0.4689B^6+0.5032B^7+0.5793B^8+0.4978B^9+0.4649B^{10}+0.3897B^{11}-0.2662B^{12}-0.0802B^{13})(1+0.1264B^{12}+0.1505B^{24})X_t = (1-0.3664B^{12})Z_t$

I conclude from the analysis of residuals that my model is satisfactory but not ideal due to its inability to pass the Mcleod-Li test, for autoregressive conditional heteroskedasticity, its large residual sample mean of 6.67, and its sufficient performance regarding all other diagnostic criteria.

Forecasting

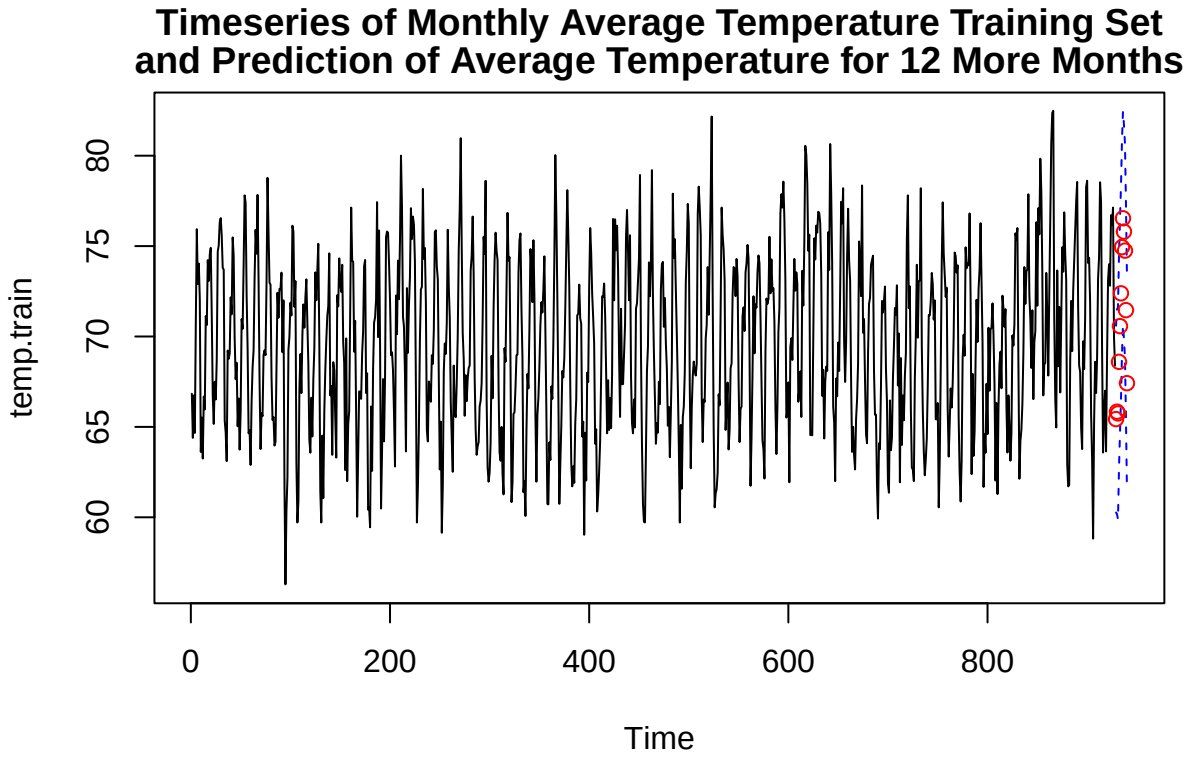
```
#install.packages("forecast")
library(forecast)
head(forecast(fit.i))
```

```
## $method
## [1] "ARIMA(13,1,0)(2,0,1)[12]"
##
## $model
##
## Call:
## arima(x = temp.train, order = c(13, 0, 0), seasonal = c(2, 1, 1, period = 12),
##     method = "ML")
##
## Coefficients:
##           ar1      ar2      ar3      ar4      ar5      ar6      ar7      ar8
##       -0.5513  -0.4573  -0.5310  -0.5008  -0.5618  -0.4689  -0.5032  -0.5793
## s.e.   0.0329   0.0365   0.0376   0.0386   0.0392   0.0382   0.0381   0.0390
##           ar9     ar10     ar11     ar12     ar13     sar1     sar2     sma1
##       -0.4978  -0.4649  -0.3897   0.2662   0.0802  -0.1264  -0.1505  -0.3664
## s.e.   0.0385   0.0386   0.0395   0.0485   0.0331   0.1005   0.0523   0.1220
##
## sigma^2 estimated as 6.671:  log likelihood = -2198.46,  aic = 4430.93
##
## $level
## [1] 80 95
##
## $mean
## Time Series:
## Start = 929
## End = 952
```

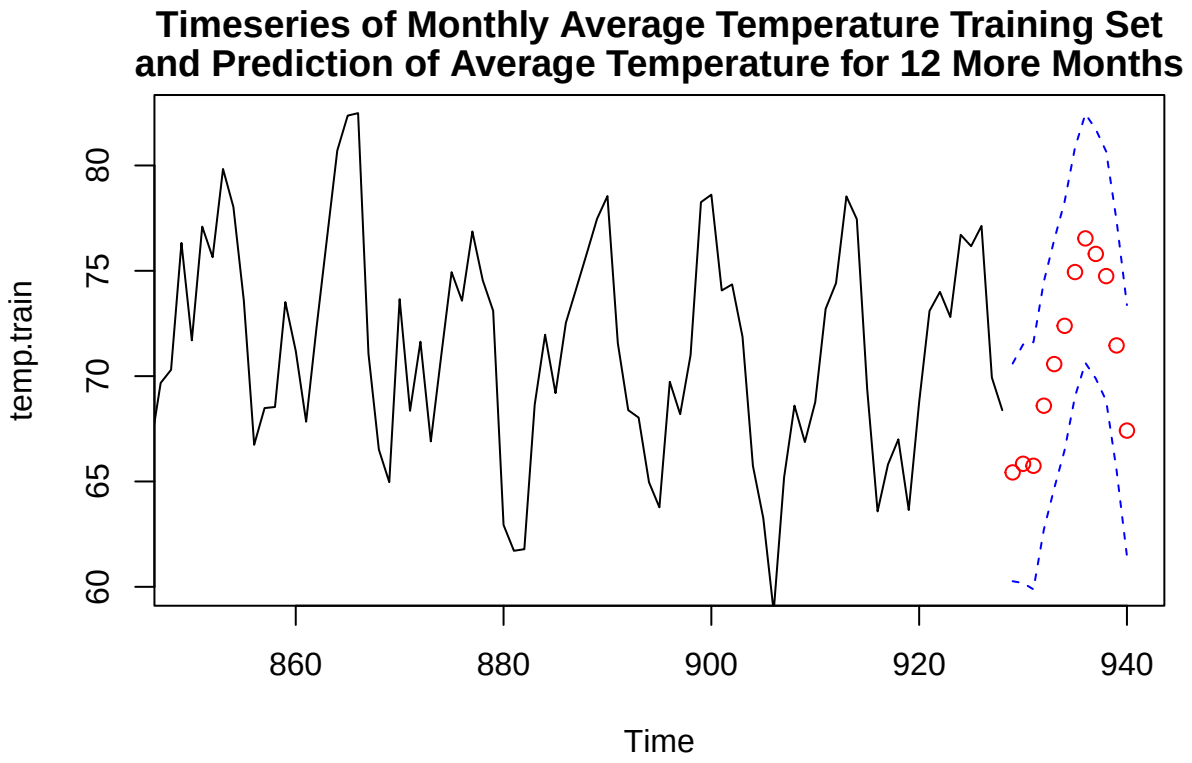
```
## Frequency = 1
##  [1] 65.43137 65.84019 65.74462 68.59643 70.57217 72.38771 74.94095 76.53772
##  [9] 75.80118 74.74822 71.45733 67.41629 65.05329 64.68125 65.96925 68.30663
## [17] 69.98723 71.98679 74.78304 76.11249 75.99804 74.42313 71.23520 67.12494
##
## $lower
## Time Series:
## Start = 929
## End = 952
## Frequency = 1
##          80%      95%
## 929 62.12135 60.36913
## 930 62.21223 60.29171
## 931 61.98731 59.99832
## 932 64.82429 62.82744
## 933 66.79288 64.79224
## 934 68.60727 66.60602
## 935 71.15513 69.15103
## 936 72.75169 70.74748
## 937 72.01079 70.00428
## 938 70.95757 68.95091
## 939 67.66253 65.65369
## 940 63.59817 61.57698
## 941 61.12833 59.05058
## 942 60.65152 58.51832
## 943 61.90112 59.74759
## 944 64.23216 62.07526
## 945 65.91060 63.75256
## 946 67.90922 65.75068
## 947 70.70409 68.54483
## 948 72.03349 69.87419
## 949 71.91691 69.75650
## 950 70.34196 68.18152
## 951 67.14793 64.98426
## 952 63.01192 60.83462
##
## $upper
## Time Series:
## Start = 929
## End = 952
## Frequency = 1
##          80%      95%
## 929 68.74138 70.49360
## 930 69.46814 71.38866
## 931 69.50192 71.49092
## 932 72.36857 74.36542
## 933 74.35146 76.35210
## 934 76.16815 78.16940
## 935 78.72678 80.73087
## 936 80.32376 82.32796
## 937 79.59158 81.59809
## 938 78.53887 80.54552
## 939 75.25213 77.26098
## 940 71.23441 73.25560
```

```
## 941 68.97825 71.05600
## 942 68.71097 70.84418
## 943 70.03737 72.19090
## 944 72.38110 74.53800
## 945 74.06385 76.22189
## 946 76.06437 78.22291
## 947 78.86198 81.02124
## 948 80.19149 82.35079
## 949 80.07916 82.23958
## 950 78.50429 80.66473
## 951 75.32248 77.48615
## 952 71.23796 73.41526
```

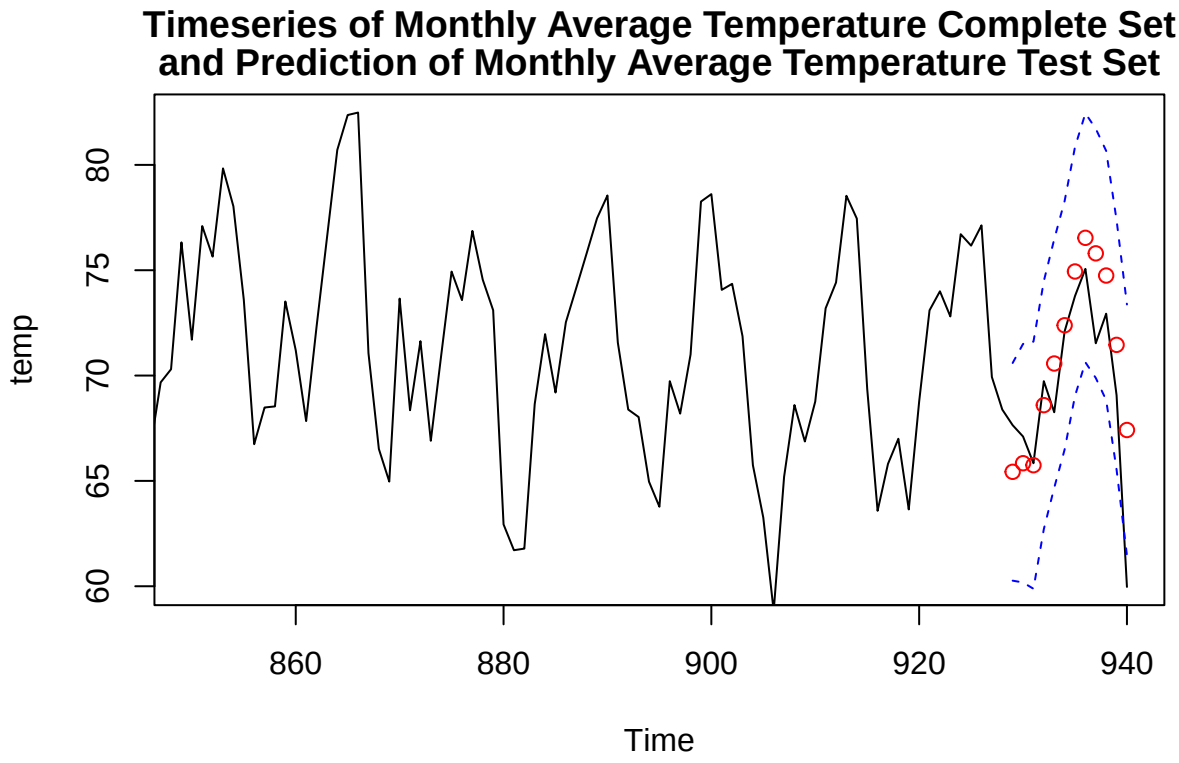
```
pred <- predict(fit.i, n.ahead = 12) #predict the average daily maximum temperature of
↪  the next 12 months
U = pred$pred + 2*pred$se #% upper bound of prediction interval calculation
L = pred$pred - 2*pred$se #% lower bound calculation
ts.plot(temp.train, xlim=c(1,length(temp.train)+12), ylim = c(min(temp.train),max(U)),
↪  main = "") #plot training data
title(main = c("Timeseries of Monthly Average Temperature Training Set", "and Prediction
↪  of Average Temperature for 12 More Months"), line = c(1, 2))
lines(U, col="blue", lty="dashed") #% upper bound of prediction interval
lines(L, col="blue", lty="dashed") #% lower bound
points((length(temp.train)+1):(length(temp.train)+12), pred$pred, col="red") #plot
↪  predictions
```



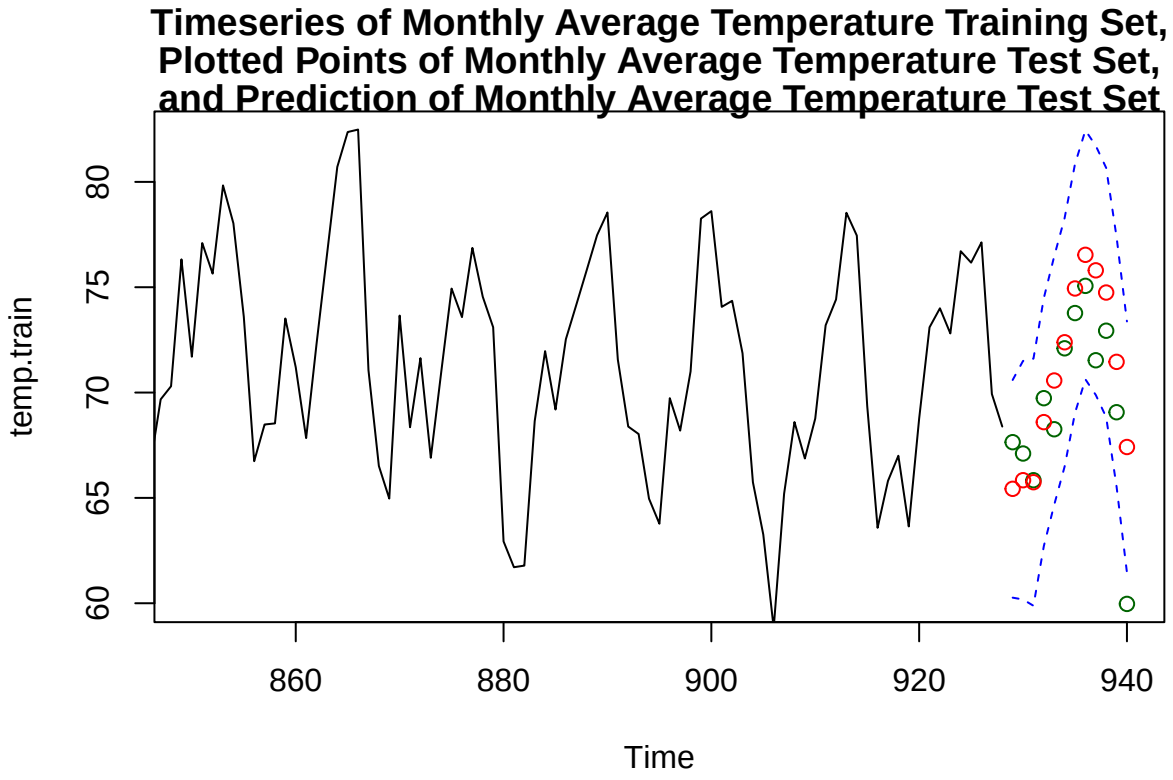
```
#time series prediction plot without test data
ts.plot(temp.train, xlim = c(850,length(temp.train)+12), ylim = c(60,max(U)), main = "")
title(main = c("Timeseries of Monthly Average Temperature Training Set", "and Prediction
↪  of Average Temperature for 12 More Months"), line = c(1, 2))
lines(U, col="blue", lty="dashed") #% upper bound of prediction interval
lines(L, col="blue", lty="dashed") #% lower bound
points((length(temp.train)+1):(length(temp.train)+12), pred$pred, col="red")
```




```
#time series prediction plot with time series test data
ts.plot(temp, xlim = c(850,length(temp.train)+12), ylim = c(60,max(U)), main = "")
title(main = c("Timeseries of Monthly Average Temperature Complete Set", "and Prediction
↪  of Monthly Average Temperature Test Set"), line = c(1, 2))
lines(U, col="blue", lty="dashed")
lines(L, col="blue", lty="dashed")
points((length(temp.train)+1):(length(temp.train)+12), pred$pred, col="red")
```




```
#time series prediction plot with test data points
ts.plot(temp.train, xlim = c(850,length(temp.train)+12), ylim = c(60,max(U)), main = "")
title(main = c("Timeseries of Monthly Average Temperature Training Set,", "Plotted Points
↪  of Monthly Average Temperature Test Set,", "and Prediction of Monthly Average
↪  Temperature Test Set"), line = c(1, 3))
lines(U, col="blue", lty="dashed")
lines(L, col="blue", lty="dashed")
points((length(temp.train)+1):(length(temp.train)+12), temp.test, col="dark green")
points((length(temp.train)+1):(length(temp.train)+12), pred$pred, col="red")
```



Conclusion

chosen model formula: $(1+0.5513B+0.4573B^2+0.5310B^3+0.5008B^4+0.5618B^5+0.4689B^6+0.5032B^7+0.5793B^8+0.4978B^9+0.4649B^{10}+0.3897B^{11}-0.2662B^{12}-0.0802B^{13})(1+0.1264B^{12}+0.1505B^{24})X_t = (1-0.3664B^{12})Z_t$

My initial goal was to accurately predict the average daily maximum temperature of the last 12 months of the given data set in order to confidently predict the average daily maximum temperature of future months. I was able to forecast the test data within the the prediction intervals and accurately predict the average daily maximum temperature of several months; however, the largest difference in prediction and test data is approximately 7 degrees Fahrenheit. Due to the significance of this difference, I would not promote future predictions using the chosen model to industry or the general public as consistently accurate; however, I believe that more consistently accurate predictions could be made using a similar model obtained by modeling the daily maximum temperature and then taking the average of those predictions to acquire a monthly average daily maximum temperature. I believe this strategy would provide more accurate predictions because the local variance of individual days is likely smaller which would facilitate more accurate daily predictions and thus more accurate monthly averages.

Acknowledgements

Thank you to Dr. Raya Feldman for providing all direction and statistical knowledge utilized in this project.