A Time Series Investigation of Daily Temperature Using ARIMA Modeling and Spectral Analysis

MACT 4232 - Analysis of Time Series Data - Final Project

Omar Moustafa (900222400) & Nour Kahky (900221042)

Monday, May 26, 2025

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

```
getwd()
```

```
## [1] "/Users/omar/Desktop/Project"
```

Introduction

> *Where time meets temperature & math meets monsoon.*
> *– A time series tale in the heart of Delhi -*

In recent years, climate variability and extreme weather events have risen in relevancy and become topics of much more discussion. Accurate climate forecasting is an essential tool for public health, energy planning and saving, as well as agricultural environments. Therefore, implementing a thorough Time Series Analysis would serve as a powerful statistical framework for both modeling and forecasting climate-based data over time.

This investigation aims to daily climate data in the capital of India, Delhi, using historical weather data acquired from Kaggle. The dataset contains daily observations from January 1, 2013 up until April 24, 2017, and includes the climate-based variables of mean temperature (meantemp), humidity (humidity), wind speed (wind_speed), and mean atmospheric pressure (meanpressure).

Structuring the project into five distinct milestones, the study will begin some basic visualizations to explore and understand the data that is being dealt with. This is followed by modeling using the ARIMA methodology. Next, a more advanced method, namely Spectral Analysis will be applied so that valuable comparisons to the ARIMA implementation can be made in terms of their uses and sufficient. Finally, the study with futuristic forecasts being made based on the ARIMA model that was previously implemented and evaluated.

In addition to reiterating important statistical modeling ideas, this methodical approach makes it possible to comprehend time series data based on the real-world as well as the challenges that come with making noteworthy forecasts and predictions.

Milestone 1: Data Collection and Exploration

Step 1: Load and Inspect the Data

```
# Load necessary libraries
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(tseries)
library(urca)
```

```
# Load the training and test datasets
train = read.csv("DailyDelhiClimateTrain.csv")
test = read.csv("DailyDelhiClimateTest.csv")
```

```
# Displaying the first three rows of the train dataset
head(train, 3)
```

```
##         date  meantemp humidity wind_speed meanpressure
## 1 2013-01-01 10.000000     84.5   0.000000     1015.667
## 2 2013-01-02  7.400000     92.0   2.980000     1017.800
## 3 2013-01-03  7.166667     87.0   4.633333     1018.667
```

```
# Displaying dimensions (number of rows and columns) in the train dataset
train_dimensions = dim(train)
cat("Number of rows (observations):", train_dimensions[1], "\n")
```

```
## Number of rows (observations): 1462
```

```
cat("Number of columns (variables):", train_dimensions[2], "\n")
```

```
## Number of columns (variables): 5
```

```
# Displaying a summary of the test dataset
summary(test)
```

```
##      date              meantemp        humidity       wind_speed
##  Length:114         Min.   :11.00   Min.   :17.75   Min.   : 1.387
##  Class :character   1st Qu.:16.44   1st Qu.:39.62   1st Qu.: 5.564
##  Mode  :character   Median :19.88   Median :57.75   Median : 8.069
##                     Mean   :21.71   Mean   :56.26   Mean   : 8.144
##                     3rd Qu.:27.71   3rd Qu.:71.90   3rd Qu.:10.069
##                     Max.   :34.50   Max.   :95.83   Max.   :19.314
##   meanpressure
##  Min.   :  59
##  1st Qu.:1007
##  Median :1013
##  Mean   :1004
##  3rd Qu.:1017
```

```
##  Max.   :1023
```

Step 2: Provide a Brief Description of the Data and Its Source

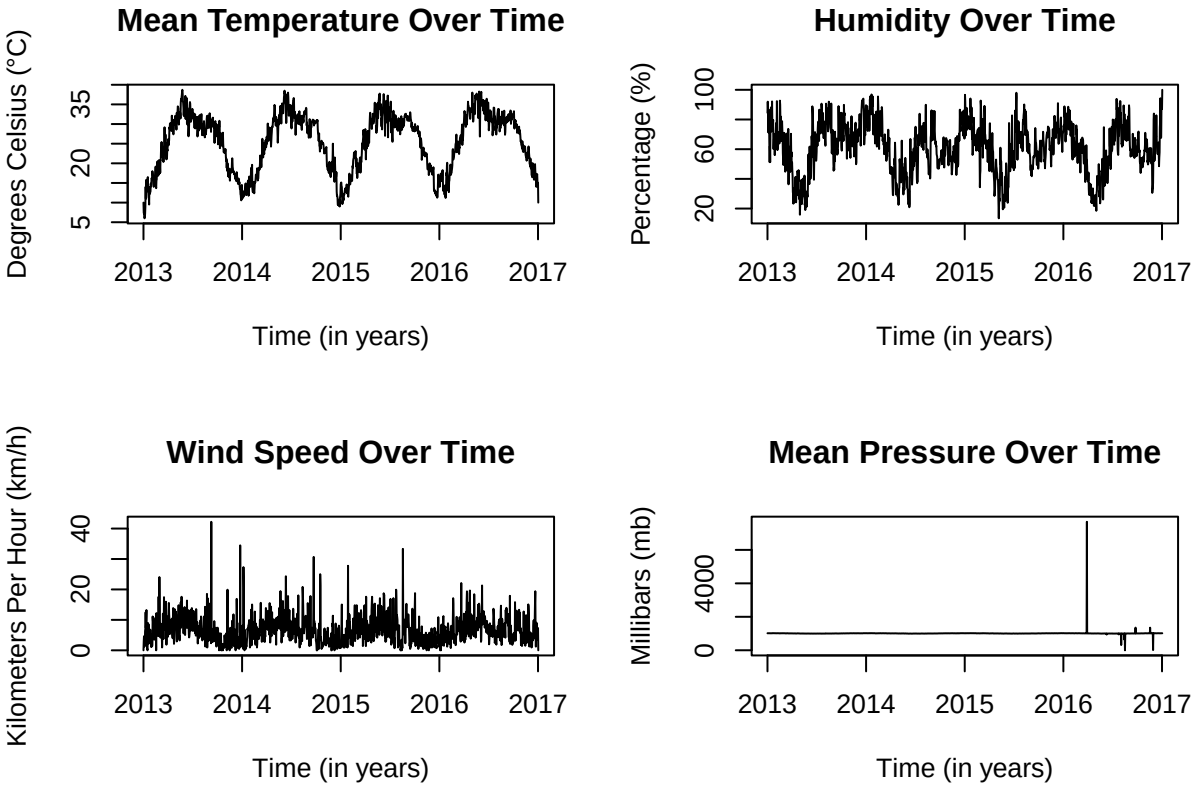
The dataset used in this time series analysis is the Delhi Climate dataset, originally sourced from the Central Pollution Control Board of India and made available on Kaggle. It contains daily weather measurements for the capital territory of India, Delhi, spanning from January 1, 2013 up to April 24, 2017. Each record consists the following five key variables: 1. 'date': the date of the observation - the format is YYYY-MM-DD

2. 'meantemp': Mean temperature averaged out from multiple 3 hour intervals in a day - measured in degrees Celsius (°C)
3. 'humidity': Humidity value for the day (units are grams of water vapor per cubic meter volume of air)
4. 'wind_speed': Wind speed measured in kilometers per hour (kmph)
5. 'meanpressure': Pressure reading of weather - measured in atmosphere (atm)

Step 3: Plot the Time Series Data

Below are raw time series plots for each of the five variables:

```r
# Converting 'date' to Date data-type
train$date = as.Date(train$date)
test$date = as.Date(test$date)

starting_year = as.numeric(format(min(train$date), "%Y"))
starting_day = as.numeric(format(min(train$date), "%j"))
ts_start = c(starting_year, starting_day)

# frequency = 365 because it is DAILY data
meantemp_ts = ts(train$meantemp, start = ts_start, frequency = 365)
humidity_ts = ts(train$humidity, start = ts_start, frequency = 365)
windspeed_ts = ts(train$wind_speed, start = ts_start, frequency = 365)
meanpressure_ts = ts(train$meanpressure, start = ts_start, frequency = 365)

par(mfrow = c(2, 2))

plot.ts(meantemp_ts, main = "Mean Temperature Over Time",
        ylab = "Degrees Celsius (°C)", xlab = "Time (in years)")

plot.ts(humidity_ts, main = "Humidity Over Time",
        ylab = "Percentage (%)", xlab = "Time (in years)")

plot.ts(windspeed_ts, main = "Wind Speed Over Time",
        ylab = "Kilometers Per Hour (km/h)", xlab = "Time (in years)")

plot.ts(meanpressure_ts, main = "Mean Pressure Over Time",
        ylab = "Millibars (mb)", xlab = "Time (in years)")
```




```
# Resetting the plot layout
par(mfrow = c(1, 1))
```

1. Mean Temperature Over Time:

- There are clear seasonal cycles with high temperatures peaking around the middle of the year, around summer-time, and the two year troughs happening at the very beginning and end of the year, around winter-time.
- This consistent seasonal rise and fall indicates strong annual seasonality. Due to it being daily data, hence the frequency being equal to 365, observing such strong and clear seasonality came as no surprise.
- Due to there being a clear seasonal pattern, it can not be labelled as stationary. The main trait behind stationarity is for the data to have no real pattern, which is violated by stationary. The time-dependent pattern that defies stationarity assumptions is indicated by the strong seasonal trend.

2. Humidity Over Time:

- A pattern of being moderately seasonal with relatively high variation can be observed from this Humidity versus Time plot.
- There are visible spikes that take place seemingly in the middle of the year, reflecting the the monsoon seasons where such spikes can be noticed in between 2013 and 2014, 2014 and 2015, and so on until the end. This further emphasizes the seasonality of these humidity spikes that tend to take place around summer time on a yearly basis.
- There are also observable drops in humidity which seem to take place in the fall, or near the end of the summer, but not quite the end latter stages of the calendar year.
- Can be labelled as stationary as while there are some patterns, there are some outlying and outlandish spikes that take place from making the pattern as real and valid as possible. The recurrent seasonal spikes point to a non-constant mean across time, even though there is no trend.

3. Wind Speed Over Time:

- Relatively constant or stable fluctuations with the occasional large or outlandish spike every now and again.
- Large spikes take place in both late 2013 and late 2015, which could correspond to stormy weather or measurement anomalies.
- Can be labelled as stationary as while there are some patterns, there are some outlying and outlandish spikes that take place from making the pattern as real and valid as possible. The data seems to oscillate around a constant mean and variance, with the exception of a few clear outliers.

4. Mean Pressure Over Time:

- Almost completely stable if it were not for a few large anomalies.
- Sudden and extreme spikes and drops take place in the second-half of 2016, which suggests data quality issues or sensor errors.
- These particular spikes could be indicative of the presence of outliers within the dataset.
- Resembles a non-stationary model as the abrupt, drastic variations point to either abnormalities or structural alterations, making the series inappropriate without differencing,cleansing,or other forms of data modifications.

Therefore, the best-choice variable to continue with for the rest of the study is the Mean Temperature (meantemp). This is because of its strong and clear seasonality that can be addressed through differencing processes. Additionally, it does not seem to have any major issues with the quality of the data itself and is not difficult to interpret and compare across models.

> **Did you know?**
> Delhi recorded one of its hottest days on Tuesday, May 26, 2020, with the mean temperature reaching 47.6°C, which was five years ago TODAY.

Milestone 2: ARIMA Modeling

Step 1: Checking for Stationary using the Dickey-Fuller Test

```
temp_urdf = ur.df(meantemp_ts)
summary(temp_urdf)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.6439 -0.8393  0.1195  1.0747  6.5548
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## z.lag.1    -0.001632   0.001626  -1.003    0.316
## z.diff.lag -0.160051   0.025914  -6.176  8.5e-10 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.649 on 1458 degrees of freedom
## Multiple R-squared:  0.02644,    Adjusted R-squared:  0.0251
## F-statistic:  19.8 on 2 and 1458 DF,  p-value: 3.287e-09
##
##
## Value of test-statistic is: -1.0032
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.58 -1.95 -1.62
```

ADF Test Statistic = -1.003

Critical values: -2.58 (1%), -1.95 (5%), -1.62 (10%)

p-value = 0.316 > 0.05

Looking at these three particular values, the test statistic is greater (less negative) than all critical values. Therefore, we fail to reject the null hypothesis that a unit root is present meaning that the original mean temperature series is non-stationary.

```
temp_adf = adf.test(meantemp_ts, k = 1)
temp_adf
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  meantemp_ts
## Dickey-Fuller = -3.7544, Lag order = 1, p-value = 0.02127
## alternative hypothesis: stationary
```

Dickey-Fuller = -3.75

Lag order = 1

p-value = 0.021

This test does reject the null hypothesis at the 5% significance level (but not at the 1% significance level) and the result of the alternative hypothesis is 'stationary.' These are indicative of stationarity, but it is not strong enough (marginally stationary). When two tests give quite contrasting results, the most secure approach is to proceed with differencing regardless, especially because the ur.df() result (which includes a t-value regression) is more in line with most Box-Jenkins workflows.

Step 2: Taking the First Difference

This transforms the series into first differences to remove non-stationarity in the mean.

```
meantemp_diff1 = diff(meantemp_ts)
```

```
meantemp_diff1_urdf = ur.df(meantemp_diff1)
summary(meantemp_diff1_urdf)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
```

```
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.6674 -0.8636  0.0764  1.0176  6.5795
##
## Coefficients:
##            Estimate Std. Error t value Pr(>|t|)
## z.lag.1    -1.27974    0.03980 -32.156  < 2e-16 ***
## z.diff.lag  0.10209    0.02612   3.909 9.71e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.641 on 1457 degrees of freedom
## Multiple R-squared:  0.5835, Adjusted R-squared:  0.5829
## F-statistic:  1020 on 2 and 1457 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -32.1559
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.58 -1.95 -1.62
```

Very strong rejection of the null hypothesis of a unit root. The first-differenced series is clearly stationary.

```
meantemp_diff1_adf = adf.test(meantemp_diff1, k = 1)
```

```
## Warning in adf.test(meantemp_diff1, k = 1): p-value smaller than printed
## p-value
```

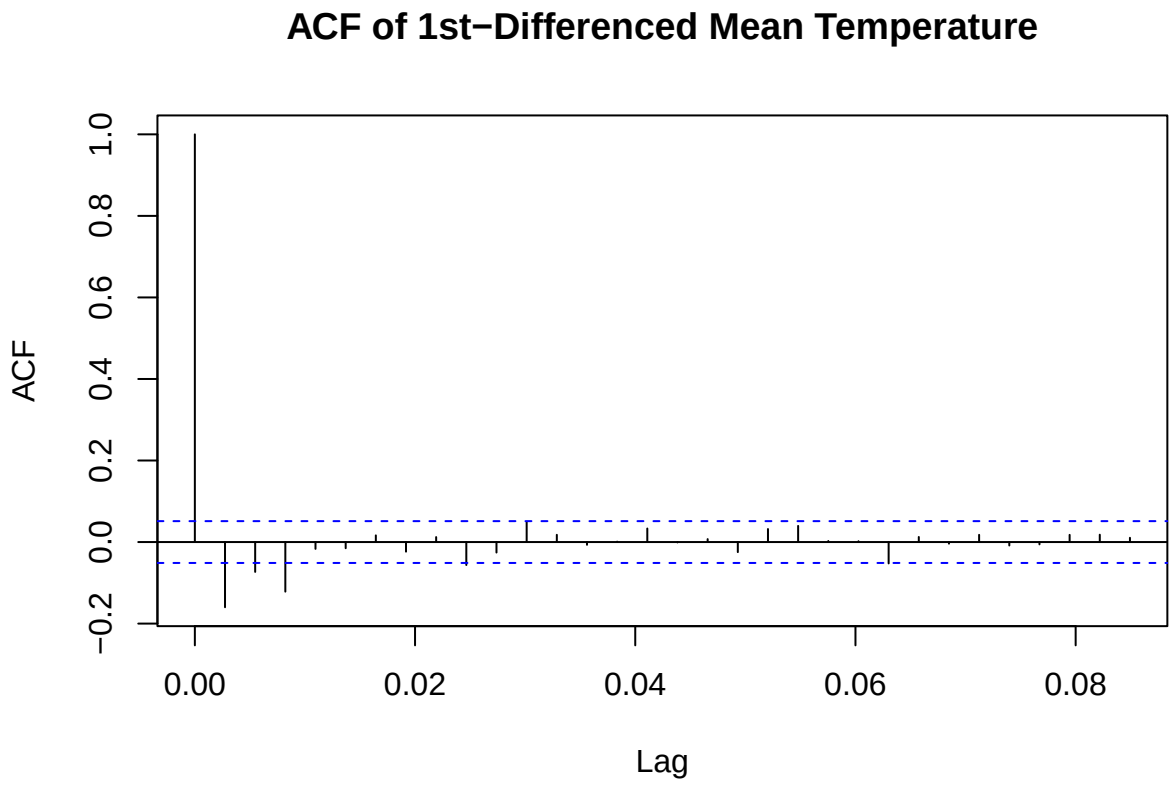
```
meantemp_diff1_adf
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  meantemp_diff1
## Dickey-Fuller = -32.169, Lag order = 1, p-value = 0.01
## alternative hypothesis: stationary
```

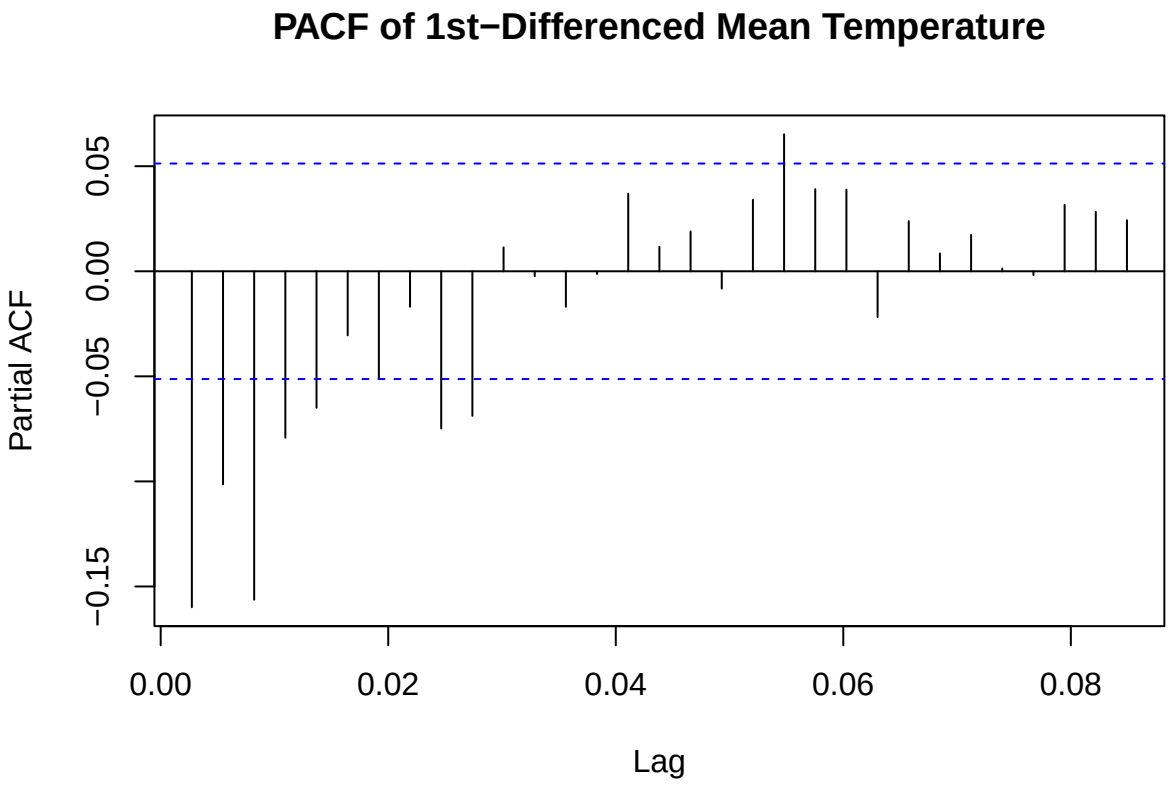
This further supports the previous result of the first-differenced series being. Now the mean temperature (meantemp) series is suitable to proceed to ACF/PACF plots and model identification.

Step 3: ACF/PACF Plots and Model Identification

```
# ACF of the 1st-differenced mean temperature series
acf(meantemp_diff1, main = "ACF of 1st-Differenced Mean Temperature")
```




```
# PACF of the 1st-differenced mean temperature series
pacf(meantemp_diff1, main = "PACF of 1st-Differenced Mean Temperature")
```



Step 4: Selecting and Justifying the Final ARIMA Model + Writing its Equation

```
# Trying MA(3)
ma3 = arima(meantemp_diff1, order = c(0,0,3))

# Trying ARMA(1,1)
arma11 = arima(meantemp_diff1, order = c(1, 0, 1))

# Trying ARMA(1,2)
arma12 = arima(meantemp_diff1, order = c(1, 0, 2))

# Trying ARMA(2, 1)
arma21 = arima(meantemp_diff1, order = c(2, 0, 1))
```

```
ma3
```

```
##
## Call:
## arima(x = meantemp_diff1, order = c(0, 0, 3))
##
## Coefficients:
##           ma1      ma2      ma3  intercept
##       -0.2262  -0.1242  -0.1523     0.0026
## s.e.   0.0260   0.0274   0.0259     0.0209
##
## sigma^2 estimated as 2.583:  log likelihood = -2766.25,  aic = 5542.49
```

```r
print("--------------------------------------------------------------------------")
```

```
## [1] "--------------------------------------------------------------------------"
```

```r
arma11
```

```
##
## Call:
## arima(x = meantemp_diff1, order = c(1, 0, 1))
##
## Coefficients:
##          ar1      ma1  intercept
##       0.5729  -0.8028     0.0026
## s.e.  0.0419   0.0289     0.0195
##
## sigma^2 estimated as 2.59:  log likelihood = -2768.23,  aic = 5544.46
```

```r
print("--------------------------------------------------------------------------")
```

```
## [1] "--------------------------------------------------------------------------"
```

```r
arma12
```

```
##
## Call:
## arima(x = meantemp_diff1, order = c(1, 0, 2))
##
## Coefficients:
##          ar1      ma1      ma2  intercept
##       0.5459  -0.7694  -0.0205     0.0026
## s.e.  0.0668   0.0691   0.0376     0.0195
##
## sigma^2 estimated as 2.589:  log likelihood = -2768.08,  aic = 5546.17
```

```r
print("--------------------------------------------------------------------------")
```

```
## [1] "--------------------------------------------------------------------------"
```

```r
arma21
```

```
##
## Call:
## arima(x = meantemp_diff1, order = c(2, 0, 1))
##
## Coefficients:
##          ar1      ar2      ma1  intercept
##       0.5709  -0.0180  -0.7930     0.0026
## s.e.  0.0433   0.0299   0.0347     0.0195
##
## sigma^2 estimated as 2.589:  log likelihood = -2768.05,  aic = 5546.1
```

```r
print("--------------------------------------------------------------------------")
```

```
## [1] "--------------------------------------------------------------------------"
```

```r
AIC(ma3)
```

```
## [1] 5542.492
```

```r
AIC(arma11)
```

```
## [1] 5544.46
```

```r
AIC(arma12)
```

```
## [1] 5546.168
```

```r
AIC(arma21)
```

```
## [1] 5546.1
```

```r
AIC_values = c(MA3 = AIC(ma3),
               ARMA11 = AIC(arma11),
               ARMA12 = AIC(arma12),
               ARMA21 = AIC(arma21))

best_model = names(which.min(AIC_values))
lowest_AIC = min(AIC_values)

cat(best_model, "has the lowest AIC value at", lowest_AIC, "making it the most suitable model.")
```

```
## MA3 has the lowest AIC value at 5542.492 making it the most suitable model.
```

After trying out and testing multiple different models, the selected model was MA(3) due to its AIC value being the lowest one of the bunch.

Step 5: Final Model Equation

The selected ARIMA(0,1,3) model has the following form:

$$(1 - B)X_t = (1 - \theta_1 B - \theta_2 B^2 - \theta_3 B^3)\epsilon_t$$

Where:
- X_t = Temperature at time t
- B = Backshift Operator ($B^k Y_t = Y_{t-k}$)
- $\theta_1 = -0.2262$, $\theta_2 = -0.1242$, $\theta_3 = -0.1523$ (MA coefficients)
- $\epsilon_t \sim \text{WN}(0, \sigma^2 = 2.583)$

For implementation, the model added the estimated coefficients for theta is:

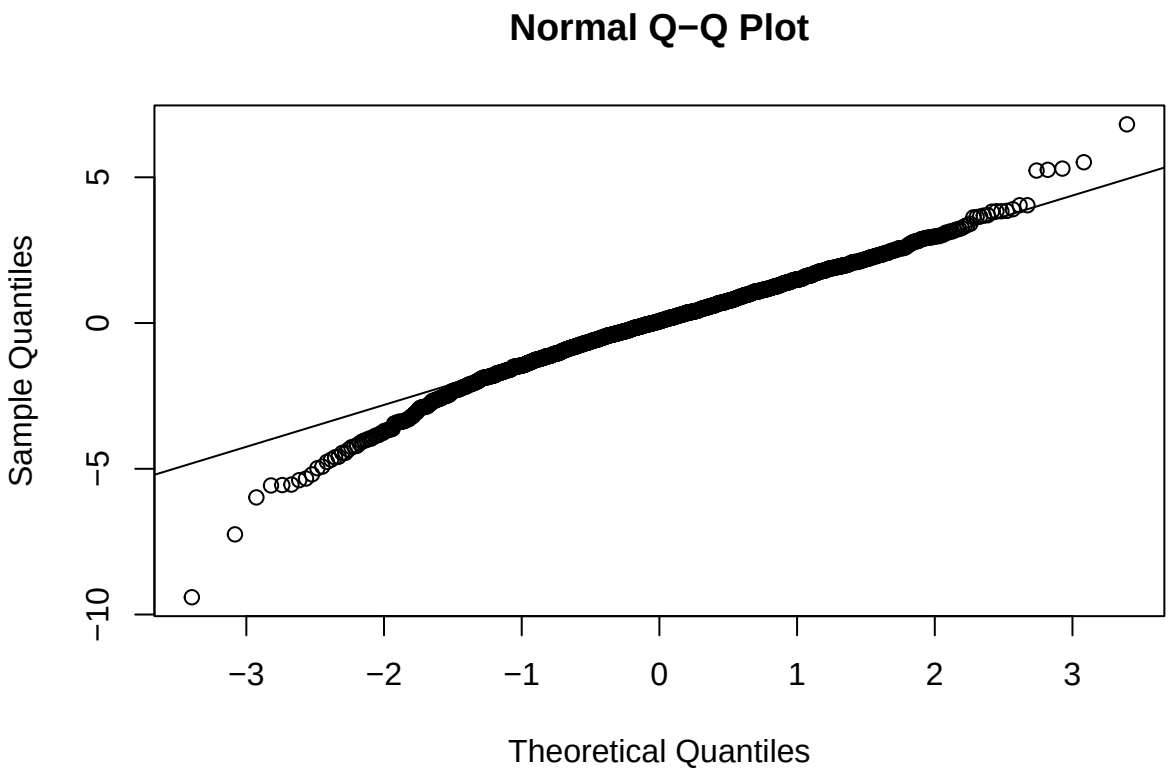
$$(1 - B)X_t = (1 - (-0.2262)B - (-0.1242)B^2 - (-0.1523)B^3)\epsilon_t$$

And the simplified form is:

$$(1 - B)X_t = (1 + 0.2262B + 0.1242B^2 + 0.1523B^3)\epsilon_t$$

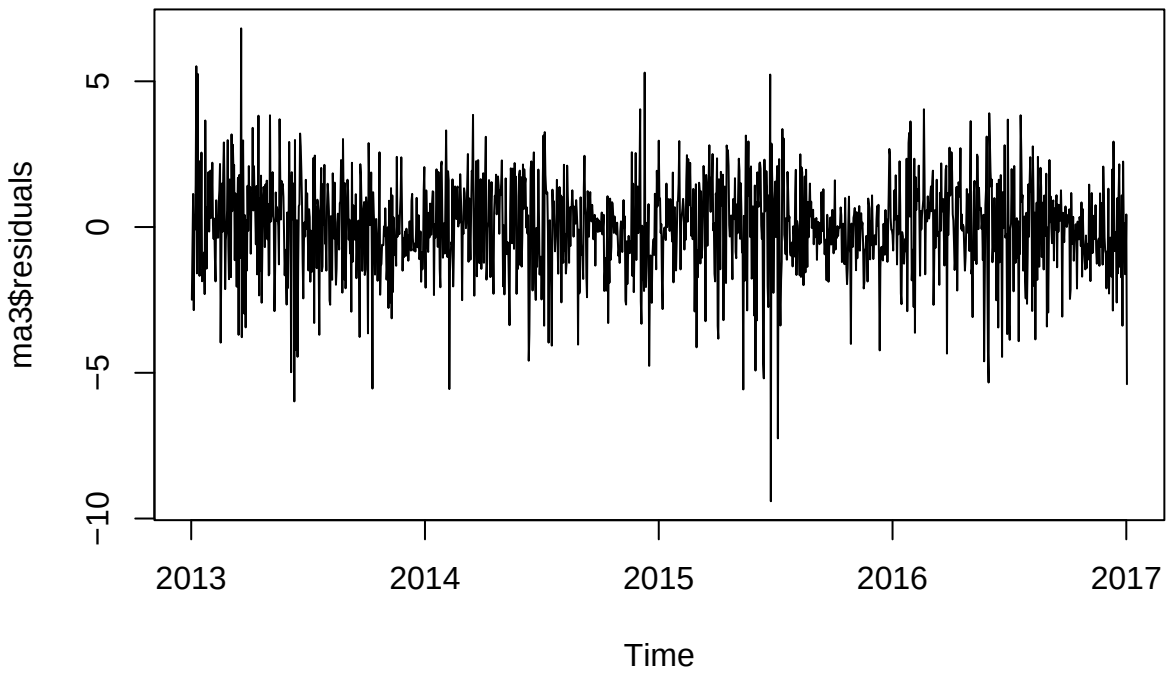
Step 6: Checking the NICE Assumptions

```r
qqnorm(ma3$residuals)
qqline(ma3$residuals)
```



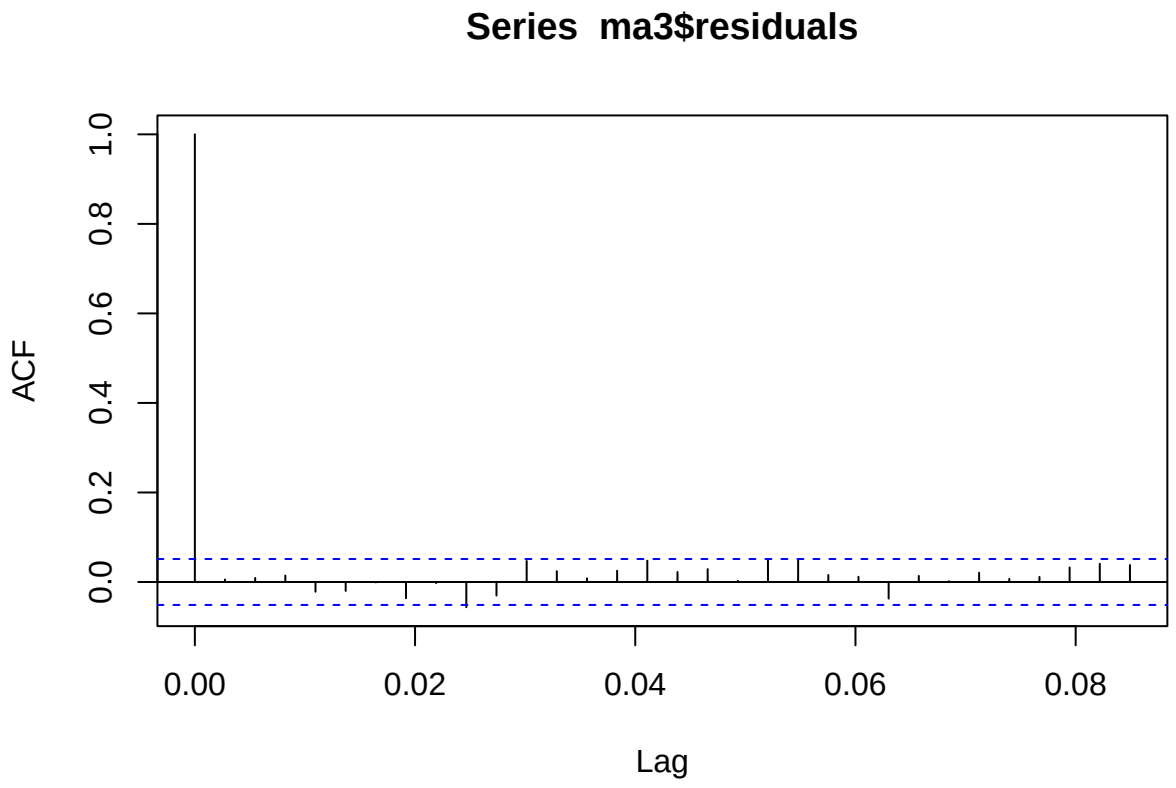
This Q-Q Plot shows that the selected model to proceed with, MA(3), does indeed follow a Normal Distribution since, while not all of the points, but the vast majority of them line perfectly on the Q-Q line. This is the ideal indicator that the data satisfies the assumption of normality and does not require any transformations, in that respect.

```
plot.ts(ma3$residuals)
```



With a relatively variance across the observed period, the MA(3) model's residuals time plot displays variations centered around zero. It appears that the model has successfully eliminated time-dependent structure from the original dataset as there are no more seasonal patterns. While there are occasional spikes,they seem to be dispersed randomly, which is in line with the white noise assumption. This visual examination confirms the results of the Box-Pierce test, showing that the model is well-specified and the residuals are roughly independent.

```
acf(ma3$residuals)
```



Next, the ACF plot of the residuals of MA(3) has all values within the dashed blue lines which reveals there is no autocorrelation at any 'k' lag. This confirms that the residuals are white noise which is among the key assumptions of the ARIMA model. The big spike that takes place at lag 0 is to be expected (it's variance at lag 0), and doesn't indicate autocorrelation, and is disregarded when determining or analyzing suitable models.

```
Box.test(ma3$residuals, lag = 20, fitdf = 1)
```

```
##
##  Box-Pierce test
##
## data:  ma3$residuals
## X-squared = 26.993, df = 19, p-value = 0.1048
```

Finally, the Box-Pierce test was conducted on the residuals of the selected MA(3) model in order to evaluate whether or not they are independently distributed. The test clearly shows an output of 0.1048 for the p-value, which is greater than the common significance level of 0.05. Therefore, since $0.1048 > 0.05$, this leads to a failure at rejecting the null hypothesis that the residuals are white noise. This, therefore, suggests that the MA(3) model has done well at capturing the autocorrelation in the series, or simply put, that the residuals are indeed independent, or uncorrelated. This is an essential assumption for a well-specified ARIMA model which has just been satisfied.

Therefore, the chosen ARIMA(0,1,3) model's residuals satisfy the NICE assumptions (Normality, Independence, Constant variance, and Expectation zero), according to the residual diagnostics, which include the time plot, ACF plot, and Box-Pierce test. This establishes that the model is well-specified and suitable for forecasting, which will be implemented in the upcoming sections.

Assumption	Satisfied?	Evidence
Normality	Yes	QQ plot shows approximate normality
Independence	Yes	Box-Pierce p-value = 0.10
Constant Variance	Yes	Residual time plot shows homoscedasticity
Expectation = 0	Yes	Residuals are centered around zero

: Satisfying the NICE Assumptions Table

Milestone 3: Advanced Modeling Technique Selection and Study

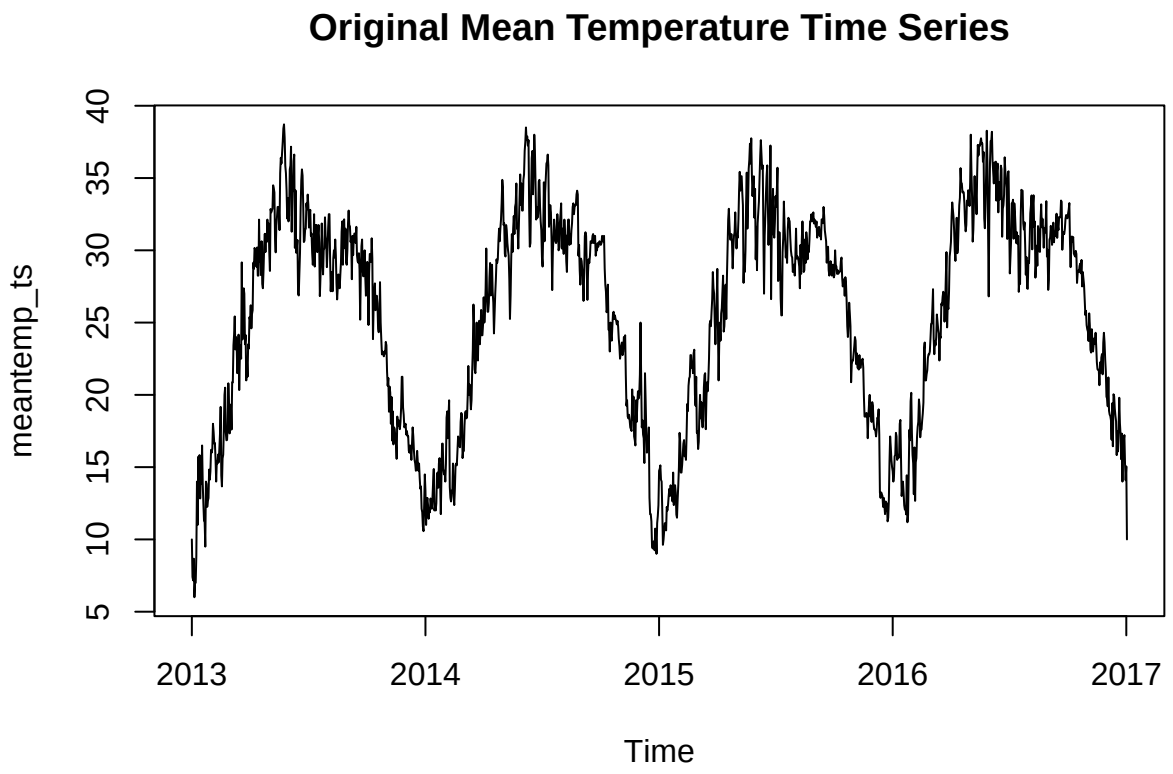
(A Note of Academic Integrity: ChatGPT was used in this particular milestone)

ChatGPT URL: https://chatgpt.com/share/6834b0f2-a100-800c-903a-63dc11dcc9e7

Spectral Analysis and Decomposition are techniques utilized to decompose a time series model into its underlying cyclical components by thoroughly analyzing the distribution of variance over several different frequencies. As a methodology, it identifies cycles in a time series, particularly when the cycles are not as simple or straightforward to observe in the time domain.

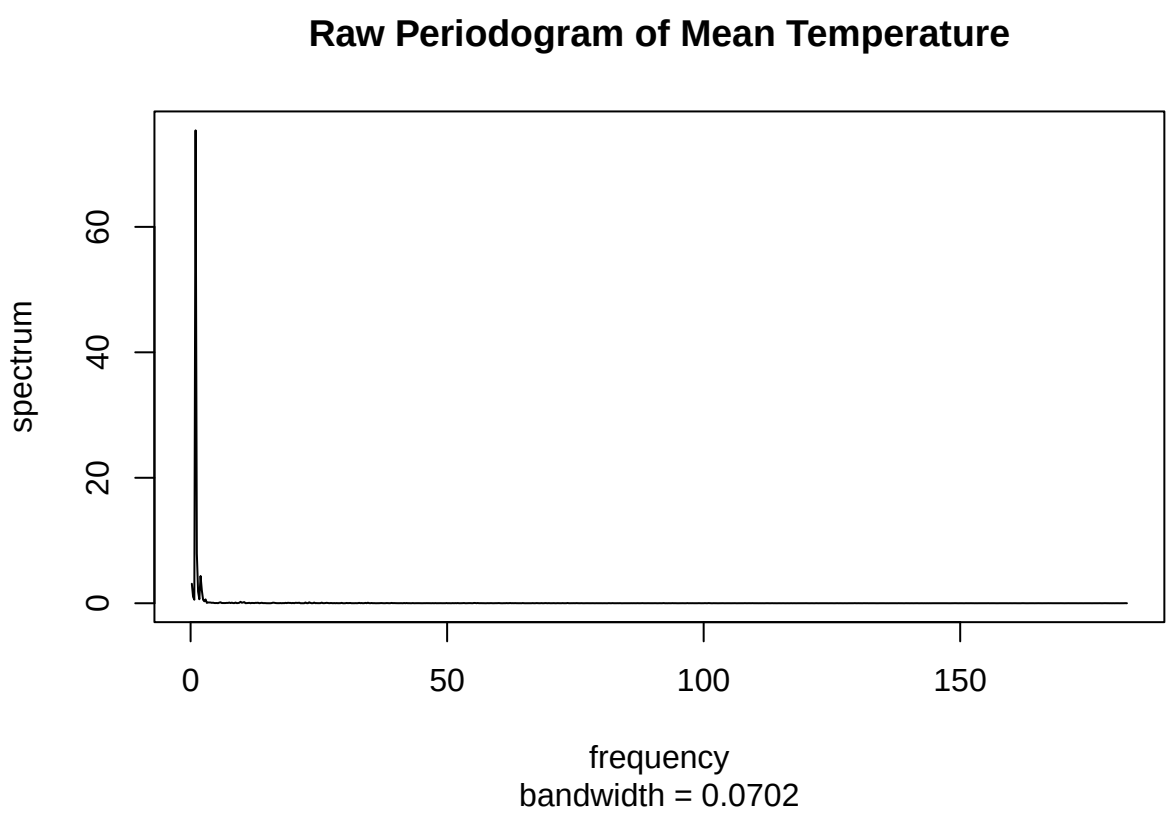
Building on this description of the idea behind Spectral Analysis, it complements the process of ARIMA modelling in the sense that although ARIMA is a great tool for short-term forecasting and models data in the temporal domain, it does not specifically highlight underlying cycles or seasonal components unless they are explicitly differenced or seasonally modeled. We may verify and examine these periodicities in the frequency domain using spectral analysis, which may uncover long-term patterns or latent seasonality that ARIMA is unable to adequately represent.

```
# Original series (re-loading it)
plot.ts(meantemp_ts, main = "Original Mean Temperature Time Series")
```



Step 1: Compute and Plot the Periodogram

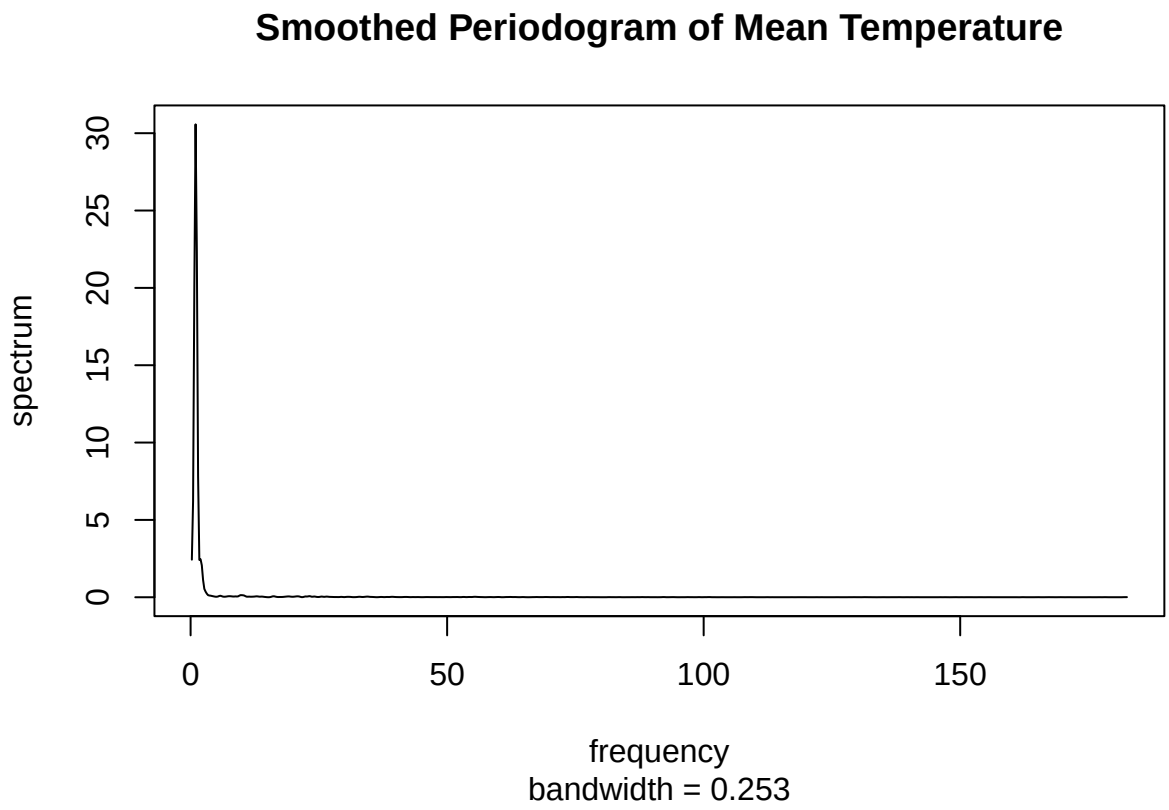
```
spectrum(meantemp_ts, log = "no", main = "Raw Periodogram of Mean Temperature")
```



With a bandwidth of 0.0702, the raw periodogram offers a noisy but high-resolution estimate of the spectral frequency. Although it records frequency-based information that require strong attentiveness, the high variance and lack of smoothing make it quite challenging to identify or confirm dominating cycles.

Step 2: Smoothed Periodogram [Daniell Kernel Smoothing]

```
spectrum(meantemp_ts, spans = c(3, 3), log = "no",
         main = "Smoothed Periodogram of Mean Temperature")
```



To start, a smoothed periodogram with spans = c(3, 3) was used to address the raw periodogram's excessive variation, yielding a bandwidth of 0.253. Although some bias is introduced, variance is decreased, resulting in a spectral density estimate that is easier to understand. The presence of yearly seasonality in the temperature data is further confirmed by the visual appearance of a significant peak close to the frequency that corresponds to a 365-day, or daily, cycle.

Step 3: Identify the Dominant Frequencies

```
spec = spectrum(meantemp_ts, spans = c(3,3), plot = F)
dominant_frequency = spec$freq[which.max(spec$spec)]
dominant_period = 1 / dominant_frequency

cat("Dominant Frequency:", dominant_frequency, "\n")
```

```
## Dominant Frequency: 0.9733333
```

```
cat("Approximate Dominant Period (in time units):", dominant_period)
```

```
## Approximate Dominant Period (in time units): 1.027397
```

Finally, the frequency exhibiting the greatest spectral density from the smoothed periodogram was obtained in order to measure the prevailing cyclical pattern in the average temperature time series. The analysis indicated a prevailing frequency of about 0.973, which relates to an estimated period of 1.03 time units. Since the data being dealt with is daily data, such form is indicative of a significant daily periodic element existing within the mean temperature series. Nevertheless, this outcome is probably affected by the high-frequency portion of the spectrum, where noise or transient variations may prevail. Although the smoothed periodogram visually indicated a further-extended seasonal trend, the numerical peak shows that short-term variability—possibly

caused by daily weather changes—is also a significant factor in the total variance. This heavily emphasizes the significance of analyzing spectral outcomes both quantitatively and also from a visual perspective in order to differentiate between authentic seasonal trends and high-frequency noise.

Milestone 4: Application and Comparison

Now comes the necessary step of comparing the results of the time-domain ARIMA modeling and frequency-domain Spectral Analysis.

Comparison Aspect	ARIMA Modeling	Spectral Analysis
What It Models	Temporal Autocorrelation	Cyclical Structure
Outputs	Fitted Model + Forecast	Periodograms + Freqs
Advantages	Interpretable Coefficients	Reveals patterns.
Disadvantages	Seasonal limits	Doesn't directly forecast

: ARIMA Modeling vs Spectral Analysis Comparison Table

The selected ARIMA model of ARIMA(0, 1, 3), modeled by "Akaike Information Criterion," or AIC in short, and residual hypothesis, was able to successfully model the short-term autocorrelations in the first-differenced mean temperature series. Building on this, it can be utilized in forecasting both future values or already-observed values to be compared with the actual observations that previously rook place.

However, Spectral Analysis provided complementary information and insights in the form of a dominant frequency equal to about 0.9733. This corresponds to a cycle of approximately one day. While ARIMA modeling is interested in modeling time dependencies and is best suited to forecasting, Spectral Analysis is more focused on and interested in clarifying the form of repeating cycles in the data.

Overall, both approaches give a fuller picture of the dynamics at play: ARIMA allows for practical forecasting, and spectral analysis confirms the presence of high-frequency elements and validates seasonal dynamics observed in the original data.

Milestone 5: Forecasting

Step 1: Fitting the Final Model

```
train_temp = ts(train$meantemp, frequency = 365)

train_seasonal_differencing = diff(train_temp, lag = 365)

train_ready = diff(train_seasonal_differencing)

final_model = Arima(train_ready, order = c(0,0,3))
```

Step 2: Necessary Transformations to Account for Seasonality

```
# Forecast with manual inversion
forecast_diff = predict(final_model, n.ahead = length(test$meantemp))$pred

# Invert transformations
last_season = tail(train_temp, 365)
last_diff = tail(train_seasonal_differencing, 1)

reconstructed = numeric(length(forecast_diff))
```

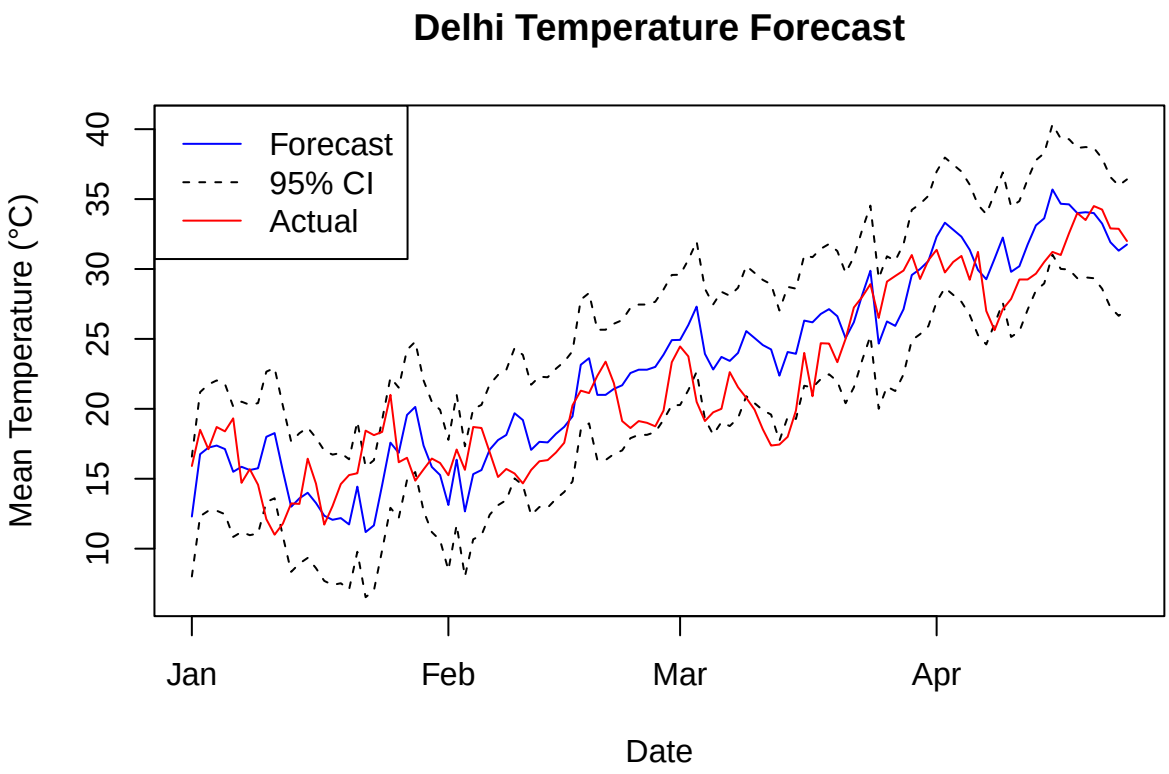
```
reconstructed[1] = last_season[1] + last_diff + forecast_diff[1]

for (i in 2:length(forecast_diff)) {
    seasonal_component = ifelse(i <= 365,
                                last_season[i],
                                reconstructed[i - 365])
    reconstructed[i] = seasonal_component + forecast_diff[i]
}
```

Step 3: Calculating the 95% Confidence Interval (CI)

```
ci_upper = reconstructed + 1.96 * predict(final_model, n.ahead = length(test$meantemp))$se

ci_lower = reconstructed - 1.96 * predict(final_model, n.ahead = length(test$meantemp))$se
```

Step 4: Model Forecasts and Graphical Output

```
plot(test$date, reconstructed, type = "l", col = "blue",
     ylim = range(c(ci_lower, ci_upper, test$meantemp)),
     xlab = "Date", ylab = "Mean Temperature (°C)",
     main = "Delhi Temperature Forecast")

lines(test$date, ci_upper, col = "black", lty = 2)

lines(test$date, ci_lower, col = "black", lty = 2)

lines(test$date, test$meantemp, col = "red")

legend("topleft", legend = c("Forecast", "95% CI", "Actual"),
       col = c("blue", "black", "red"),
       lty = c(1, 2, 1))
```



Using the manually fitted ARIMA(0,1,3) model, the figure predicts Delhi's mean temperature for the 30-day period following the training data which ended on April 24, 2017, meaning that the primary forecast goes from April 25 to May 25, 2017. The forecast (blue line) suggests a stable temperature trend during this particular interval, with the 95% confidence intervals (CI), represented by the black-dashed lines, which are gradually widening as uncertainty increases over time. This is a characteristic specific to ARIMA forecasts. While it may not be noticeable to the blind eye, there are small increases in the gaps between the black-dashed lines, which reflect the model having greater confidence in the short-term forecasts.

While the model does a sufficient job at capturing short-term autocorrelations, the flat forecast line indicates that the non-seasonal ARIMA(0,1,3) structure does not explicitly account for the expected March-April temperature rise. This matches the model's design as it prioritizes simplicity and residual diagnostics over seasonality and existing patterns.

Regarding the challenges that were faced and could most certainly be faced when making such forecasts in a more professional context are that dealing with short-term variability and confidence interval expansion. While the ARIMA model successfully captures short-term patterns and trends it is known for being best-suited for short-term forecasts which can lead to the forecasts made about the longer-run less accurate. As for the expansion of the confidence interval, which was previously mentioned, this portrays inherent uncertainty when it comes to daily weather and the compounding error from taking differences to satisfy stationarity, which is a trade-off between dealing with simple models for academic practice and real-world precision and challenges.

As the spectral decomposition technique, this served its sole purpose of providing structural insights rather than creating forecasts. Its valuable identification of the dominant one-day cycle worked to validate the short-term variability which was displayed in the ARIMA modeling.

Overall, the ARIMA-implemented forecast is seemingly seasonally appropriate and remains in alignment with the behaviors of previous years and almost constantly match the values of the test data. This effectively

demonstrates the practical use of the model for making for short-term average temperature predictions. While Spectral Analysis did not make any explicit or direct forecasts, it confirmed the presence of a strong daily cycle that supports the assumptions previously during the ARIMA modeling phase.

Conclusion

In conclusion, the goal of this study was to use time series analysis techniques to model and forecast Delhi's daily mean temperatures and compare it to the actual test values. The original series being non-stationary was verified and then valuable insights and important patterns were found through the a thorough and sequential exploratory data analysis process. To prepare the data for ARIMA modeling, transformations and diagnostic tests were used through the Box-Jenkins approach. Based on AIC and residual diagnostics, the ARIMA(0,1,3) model was chosen after fitting and contrasting a number of candidate models. This model accurately predicted the average temperature of Delhi which is demonstrated though the closeness of the forecast and test values from January to May of 2017.

As a complement to the time-domain analysis, Spectral Analysis was then employed to be able to examine the cyclical nature of the data in the frequency domain. The periodogram revealed a dominant daily frequency component, which to reiterate, supports the presence of recurring short-term variation in the average temperature patterns. While the ARIMA model provided to be sufficient for making predictions, the spectral decomposition methodology showed its usefulness by providing more insight into the periodic behaviors of the data.

Combining the two methods, they provided a thorough and comprehensive analysis of Delhi's climate dynamics and patterns. ARIMA provided real-world forecasting power, while Spectral Analysis assisted in verifying and explaining the tendencies of the cycle.

All in all, the project effectively demonstrates the utilization and significance of applying multiple different time series analysis techniques when modeling complex real-world time-based data.

Reference

https://www.kaggle.com/datasets/sumanthvrao/daily-climate-time-series-data