INDIAN INSTITUTE OF TECHNOLOGY, KANPUR
DEPARTMENT OF MATHEMATICS AND STATISTICS

Time Series Analysis

MTH442

Project Report

Weather Forecasting

Under the Guidance of: Prof. Amit Mitra
Department of Mathematics and Statistics, IIT Kanpur
amitra@iitk.ac.in

Submitted By: Dasari Charithambika **210302**
Jakkampudi S K L N M Subbarayudu **210462**
Pooja Kumari **210726**
Saranya Pal **210930**
Vanshika Gupta **211146**

IIT KANPUR, NOVEMBER 2023

ABSTRACT

Weather forecasting is a critical aspect of understanding and preparing for the dynamic climate patterns that influence our daily lives. In this project, we focused on predicting daily temperatures for the city of Delhi over a span of five years using time series analysis. Our dataset, meticulously collected over this period, provided a comprehensive view of temperature fluctuations, allowing us to explore underlying trends, seasonality, and potential randomness.

To begin, we identified trends that emerged over time and isolating seasonal patterns inherent in the temperature data. Our investigation involved statistical tests to validate the presence of these components and gain insights into the complexities of Delhi's climate.

Subsequently, we applied the Autoregressive Integrated Moving Average (ARIMA) model, a powerful tool for time series forecasting. By fitting the model to our data, we aimed to capture and predict future temperature trends. The model was fine-tuned through rigorous testing and validation procedures to ensure its reliability in providing accurate forecasts.

Our findings contribute not only to the domain of weather forecasting but also offer valuable insights into the climate dynamics specific to Delhi. The project underscores the importance of leveraging time series analysis techniques, such as ARIMA, for precise predictions in the realm of meteorology.

In conclusion, our project bridges the gap between historical weather data and future predictions, offering a robust methodology for forecasting daily temperatures. The results hold implications for urban planning, resource allocation, and climate resilience, emphasizing the significance of data-driven approaches in addressing the challenges posed by dynamic weather patterns in urban environments like Delhi.

Contents

1 Introduction

In our weather forecasting project for Delhi, we used the R programming language to make sense of five years' worth of daily temperature data. R is like a smart toolbox that helped us clean up our data, draw cool graphs, and predict future temperatures.

With R, we sorted out messy data, fixed mistakes, and made our information easy to understand. We used it to create graphs that showed us how temperatures changed over time in Delhi. These visuals were key in guiding our next steps.

The big star of our project was the ARIMA model, a fancy tool for predicting future temperatures. R made it easy for us to use this model and tweak it to make better predictions. We also used R to check if our predictions were accurate and to test our ideas.

This project is like a puzzle where meteorology (studying weather), data science, Time Series Analysis and programming come together. Through this report, we'll explain how we used R to solve the puzzle, face challenges, and discover important things about Delhi's weather.

2 Libraries Used

These are some inbuilt libraries in R what we have used in our project:

- forecast
- fpp2
- ggplot2
- seasonal
- stats

3 Data Extraction

We aimed to extract historical weather data for New Delhi from the Weather Underground website https://www.wunderground.com/history/monthly/in/new-delhi/VIDD/ covering the years 2016 to 2020.

The final Data set is Time Series Data.csv

Listing 3.1: Code for loading data

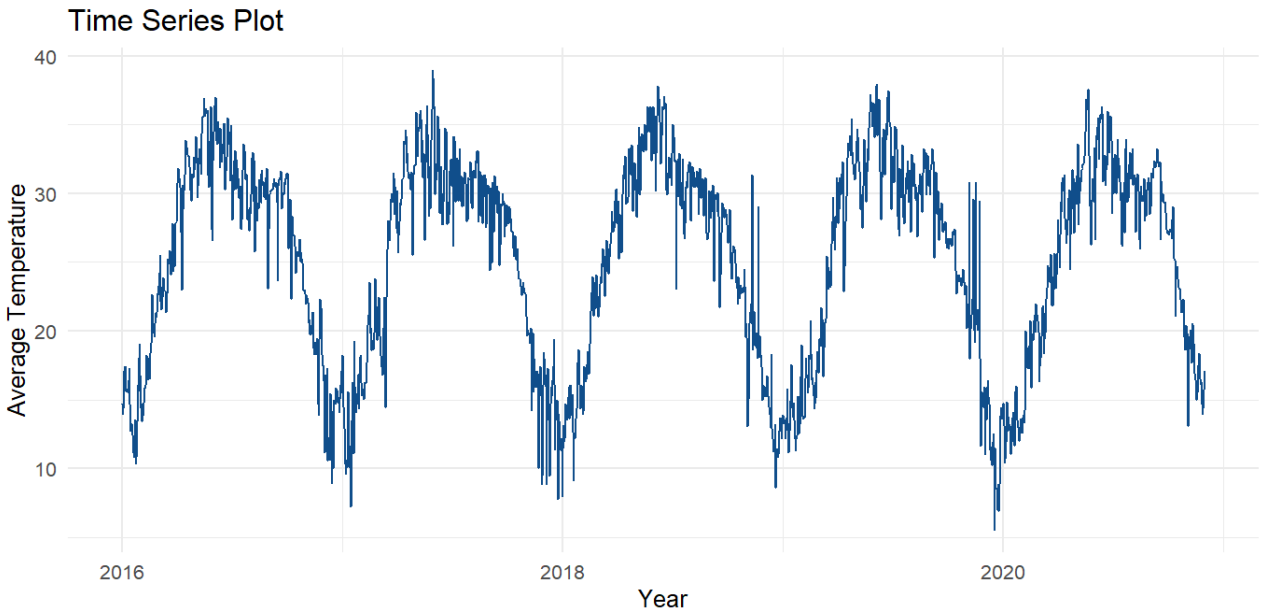
```
data <- read.csv("Time series Data.csv")
head(data)
```

```
        Dates   Max   Avg   Min
1 01-01-2016 23.33 14.67  8.33
2 02-01-2016 21.67 13.94  8.89
3 03-01-2016 22.22 14.39  9.44
4 04-01-2016 22.78 15.72  9.44
5 05-01-2016 22.78 17.06 17.22
6 06-01-2016 24.44 17.44 12.22
```

- Dates: date from 01-01-2016 to 30-11-2023
- Max: maximum temperature on that day in °C
- Avg: average temperature on that day in °C
- Min: minimum temperature on that day in °C

We have divided the Time Series Data into train data(80%) and test data(20%).

4 Experiments and Results



4.1 Trend Estimation and Elimination

Relative Ordering test: This is a non-parametric test procedure used for testing the existence of trend components.

H_0: trend is not present v/s H_1: trend is present

Listing 4.1: Testing for trend

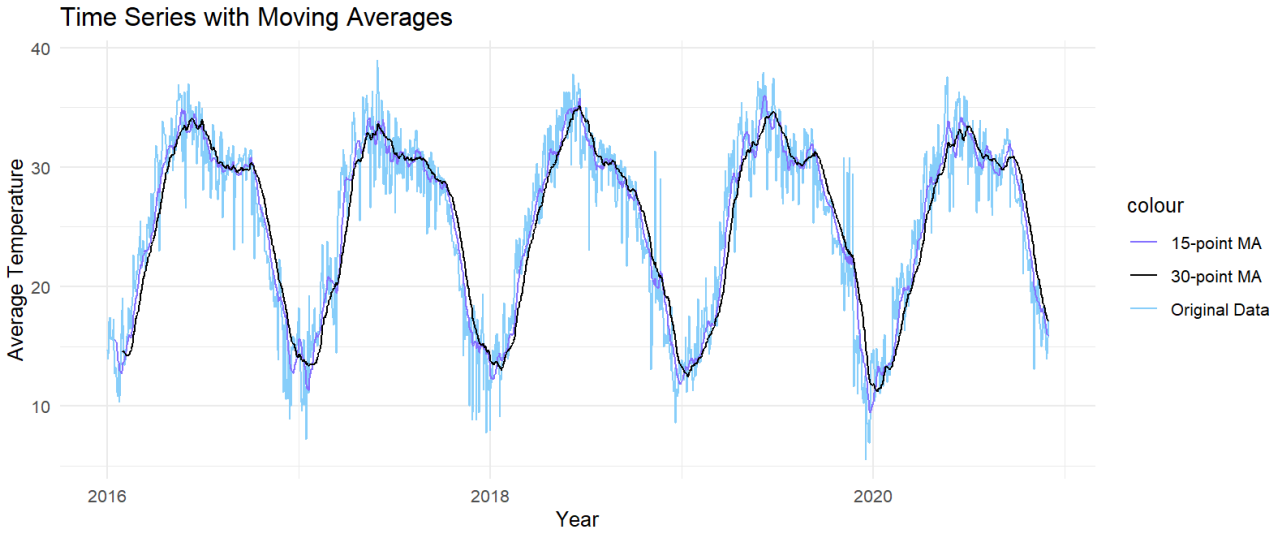
```
Relative_Ordering_Test = function(time_series){
  n = length(time_series)
  Q = 0
  # Total no. of discordants.....
  for (i in 1:n){
    for (j in (i+1):n)
    {
      if (!is.na(time_series[i]) && !is.na(time_series[j]) &&
          time_series[i] > time_series[j])
        Q = Q + 1
    }
  }
  Tau = 1 - 4 * Q / (n * (n - 1))
  Var = 2 * (2 * n + 5) / (9 * n * (n - 1))
  Z = Tau/sqrt(Var)
  #  Let level of significance, alpha=0.05
  alpha = 0.05
  q = qnorm(1-alpha/2,0,1)
  if (Z <= q)
    return (1)
  else
    return (0)
}
```

The output of this function is **1** which means $Z \leq q$, which means there is **no trend** present in the time series model.

4.1.1 Filtering

After conducting a hypothesis test for trend presence, the results indicate that the null hypothesis, suggesting the absence of a trend, is accepted. This implies that there is insufficient evidence to conclude the existence of a discernible trend in the data.

In an effort to enhance the clarity of the underlying patterns, we applied both 15-point and 30-point Moving Average (MA) filtering techniques to smooth the curve. The objective was to discern any potential seasonal components within the data.



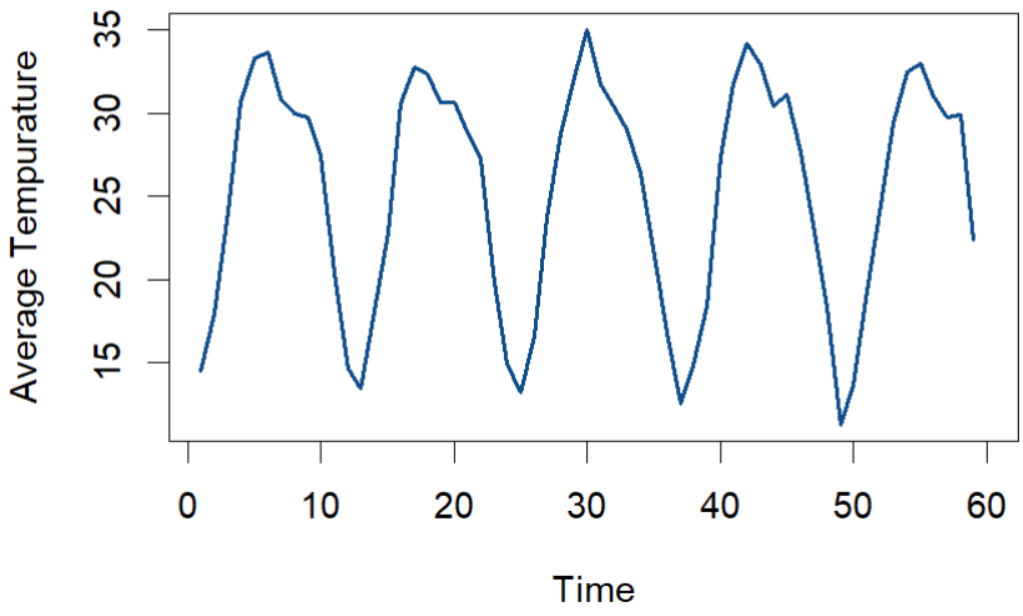
4.2 Seasonal Estimation

Listing 4.2: Testing for seasonality

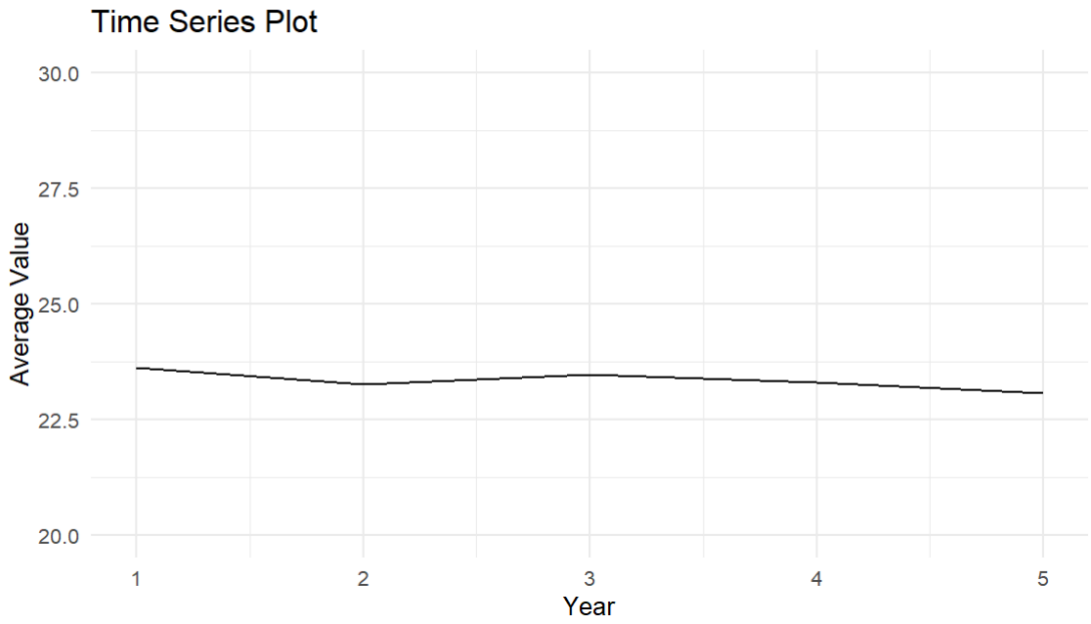
```
# Creating monthly data
 dddd = numeric(60)
 ind1 = 0
 ind2 = 1
 val = 0
 while(ind2+ind1*30 <= 1796)
{
  while(ind2 <= 30)
 {
   val = val + avg_temp[ind2+ind1*30]
   ind2 = ind2 + 1
 }
 ind2 = 1
 ind1 = ind1 + 1
 dddd[ind1] = val/30
 val = 0
}
plot.ts(dddd)

#creating yearly data
 val = 0
 ind1 = 0
 ind2 = 1
```

```
yyyy = numeric(5)
while(ind1*12 +ind2 <= 60)
{
   while(ind2 <= 12 & ind1*12 +ind2 <= 59)
   {
     val = val + dddd[ind1*12 +ind2]
     ind2 = ind2 + 1
   }
   ind1 = ind1 + 1
   yyyy[ind1] = val/ind2
   ind2 = 1
   val = 0
}
plot.ts(yyyy)
```

Seasonal for Monthly data:



Seasonal for Yearly data:



Now in the yearly data plot(~ 23.34), we can see that S_t is similar to all the years we have taken.

4.3 Checking randomness

Turning point test: We will conduct this non-parametric test to test our time series randomness.

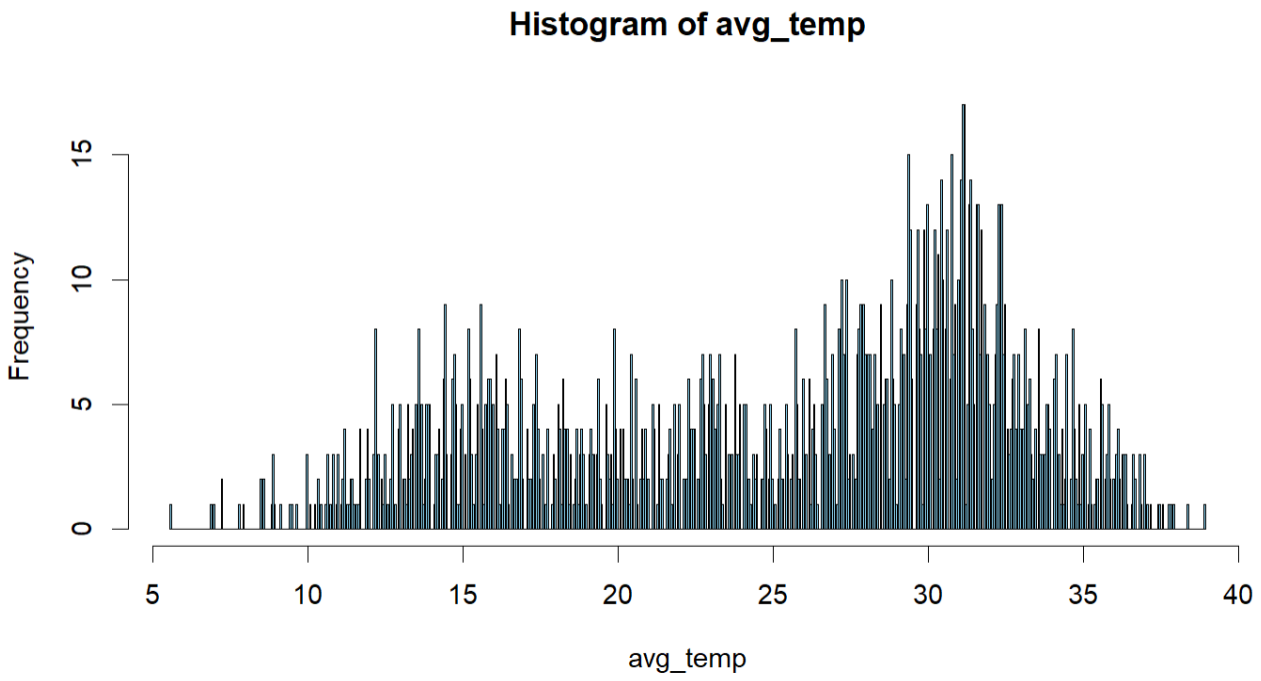
H_0: Series is purely random v/s H_1: H_0 is not true

Listing 4.3: Testing for randomness

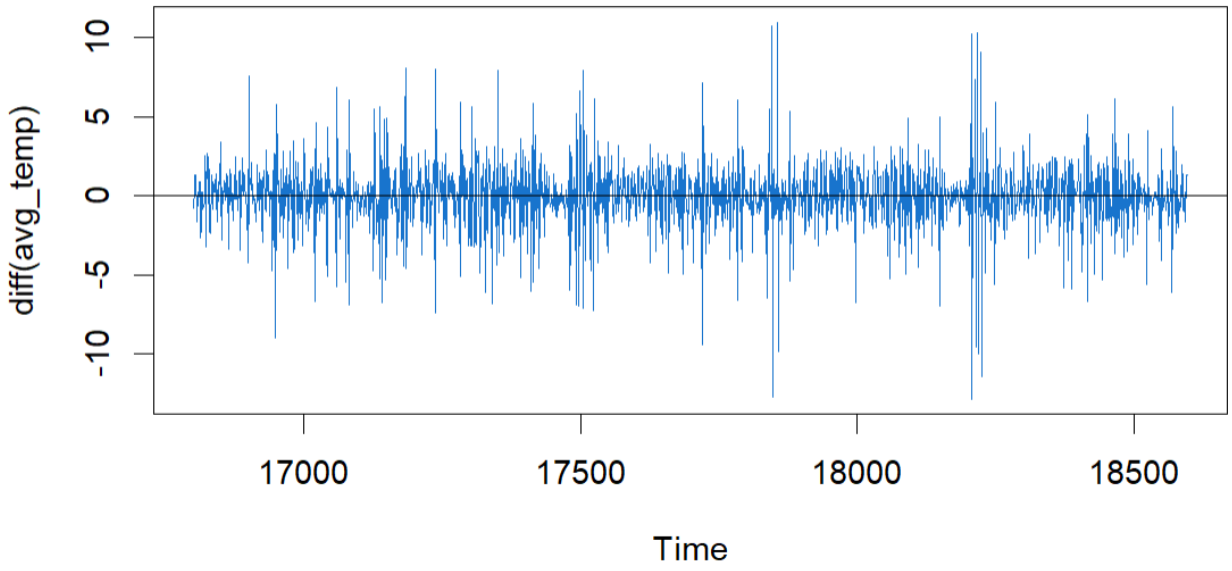
```
Turning_Point_Test = function(time_series)
{
  n = length(time_series)
  P = 0
  for (i in 2:(n-1))
  {
    if (((time_series[i]>time_series[i-1]) && (time_series[i]>
       time_series[i+1])) || ((time_series[i]<time_series[i-1]) &
       & (time_series[i]<time_series[i+1])))
      P = P + 1
  }

  E = 2*(n-2)/3
  Var = (16*n - 29)/90

```

```
  Z = (P - E)/sqrt(Var)

  alpha = 0.05
  q = qnorm(1-alpha/2, 0, 1)

  if (Z <= q)
    return (1)
  else
    return (0)

}
```

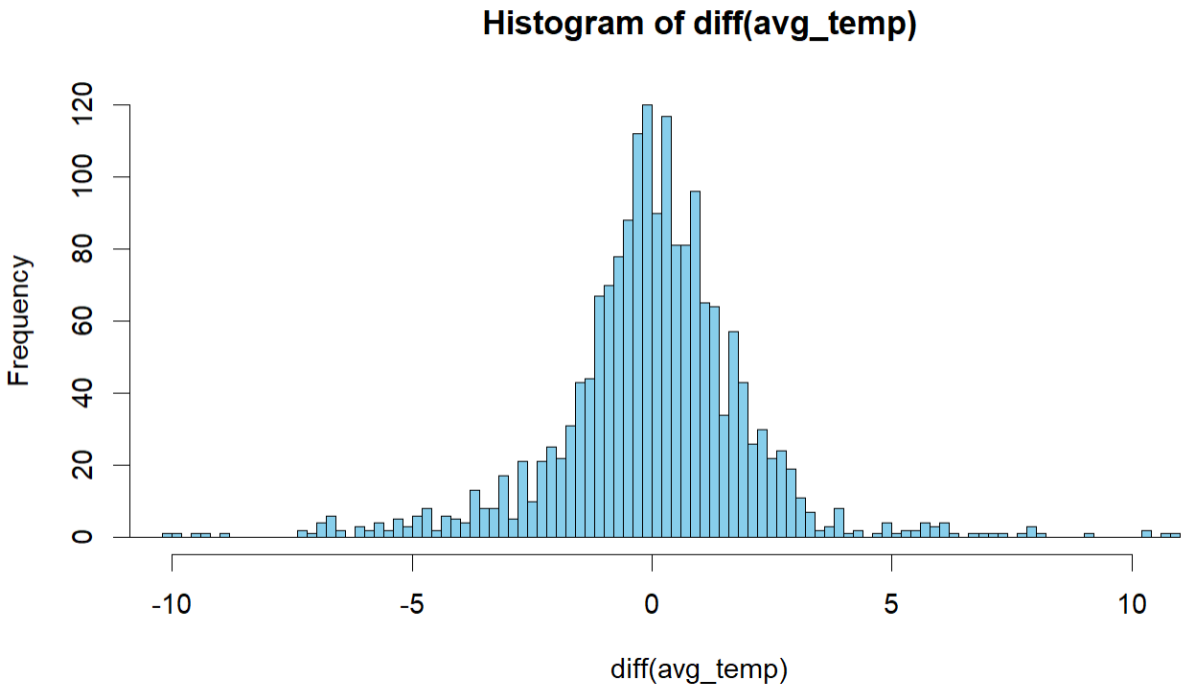
The output of this function came out to be 1 which means that our time series data is purely random.



For this plot, we can observe that the plot is non-stationary.
So we will be doing the first-order difference on avg_temp.



Where diff(avg temp) is the first order difference applied on initial time series data avg temp.



The histogram of diff(avg_temp) gives a stationary plot.

5 Time Models

5.1 ARIMA Model

An autoregressive integrated moving average – ARIMA model is a generalization of a simple autoregressive moving average – ARMA model. According to the name, we can split the model into smaller components as follows:

- **AR:** An AutoregRegressive model which represents a type of random process. The output of the model is linearly dependent on its own previous value i.e. some number of lagged data points or the number of past observations.
- **MA:** A Moving Average model which output is dependent linearly on the current and various past observations of a stochastic term.
- **I:** Integrated here means the differencing step to generate stationary time series data, i.e. removing the seasonal and trend components.

ARIMA model is generally denoted as ARIMA(p, d, q), and parameter p, d, q are defined as follow:

- p: the lag order or the number of time lag of autoregressive model AR(p)
- d: degree of differencing or the number of times the data have been subtracted with past value
- q: the order of moving average model MA(q)

Listing 5.1: ARIMA Model

```r
arima.model = auto.arima(avg_temp)
summary(arima.model)
```

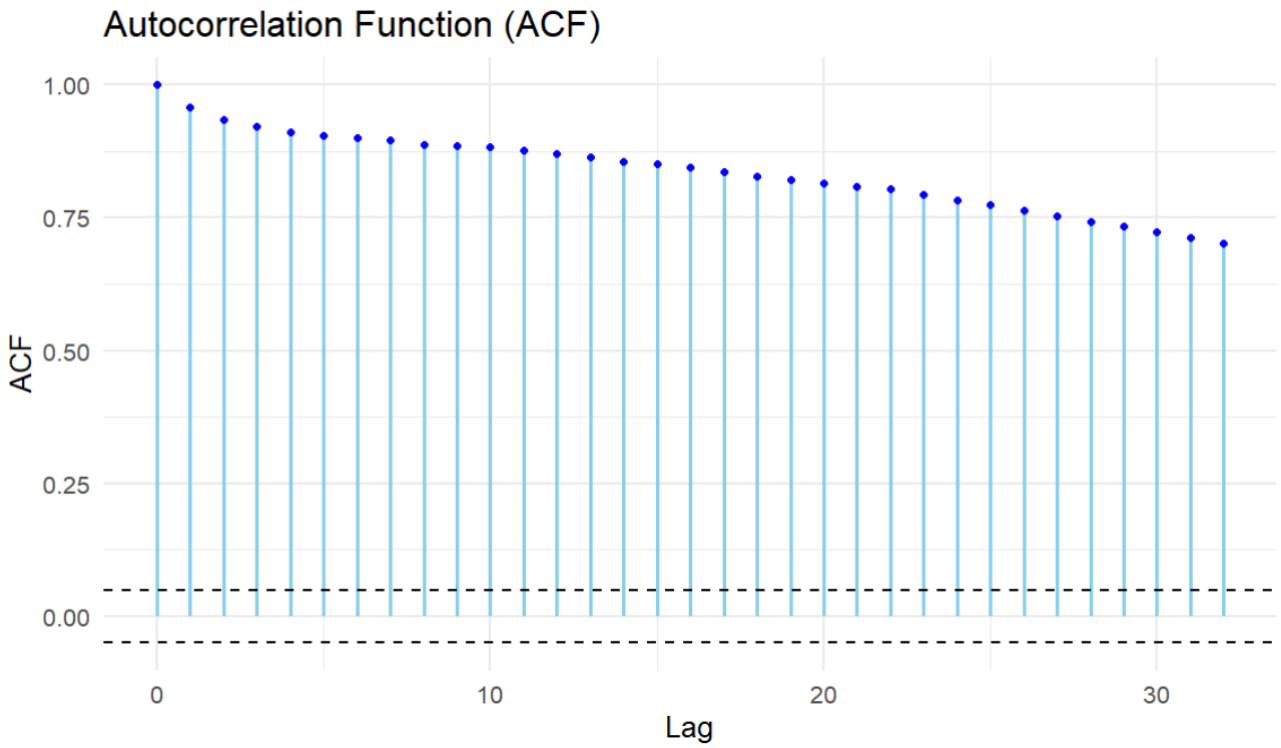
```
Series: avg_temp
ARIMA(5,0,0) with non-zero mean

Coefficients:
         ar1     ar2     ar3     ar4     ar5     mean
      0.6731  0.0766  0.0792  0.0425  0.1083  24.7432
s.e.  0.0234  0.0283  0.0283  0.0283  0.0235   2.2074

sigma^2 = 3.937:  log likelihood = -3777.44
AIC=7568.89   AICc=7568.95   BIC=7607.34

Training set error measures:
                     ME     RMSE      MAE        MPE     MAPE      MASE        ACF1
Training set 0.009288471 1.980839 1.41161 -0.9187474 6.528943 0.9937836 -0.01018635
```

5.2 ACF and PACF plots

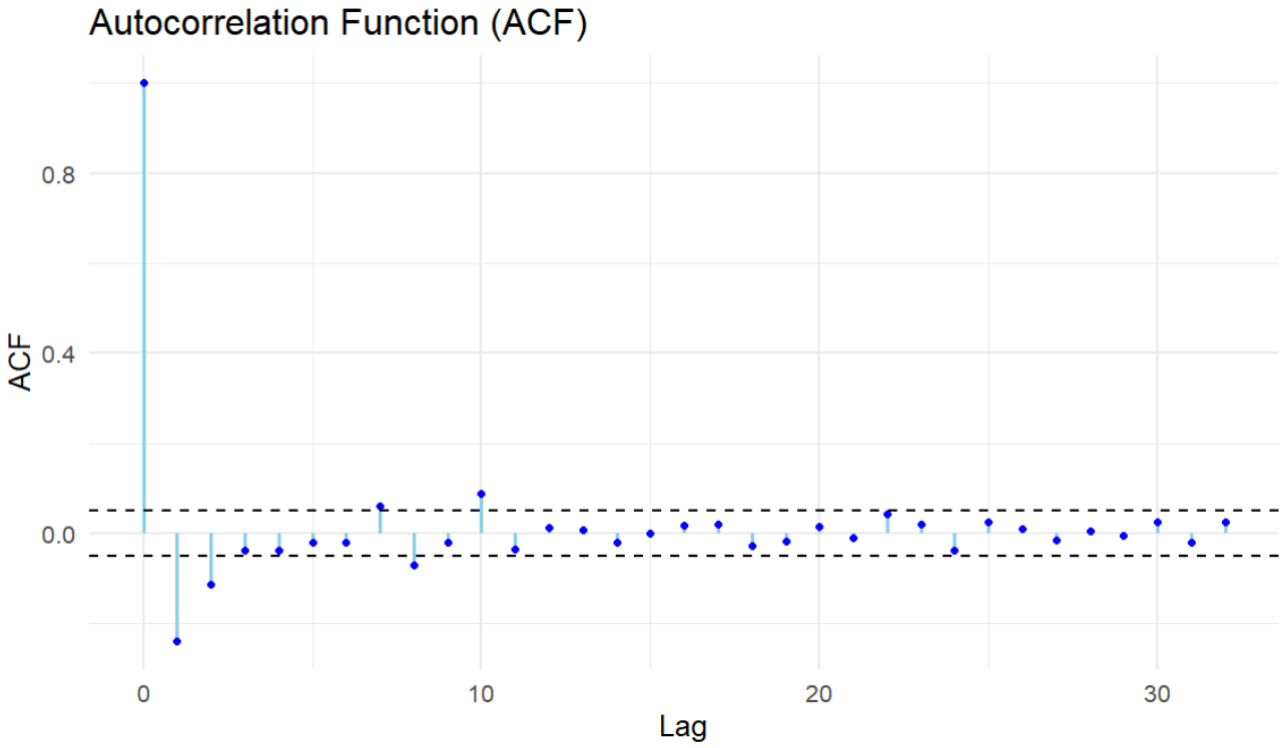
5.2.1 Auto Correlation Function(ACF)

The Auto Correlation Function (ACF) is a statistical tool used in time series analysis to measure the correlation between a time series and its own lagged values. The ACF is often used to identify patterns or dependencies in the data over different time lags.

$$\begin{aligned}\rho_k &= \frac{\text{Cov}(Y_t, Y_{t-k})}{\sqrt{\text{Var}(Y_t) \cdot \text{Var}(Y_{t-k})}} \\ &= \frac{\sum_{t=k+1}^{n}(Y_t - \bar{Y})(Y_{t-k} - \bar{Y})}{\sqrt{\sum_{t=1}^{n}(Y_t - \bar{Y})^2 \cdot \sum_{t=k+1}^{n}(Y_{t-k} - \bar{Y})^2}}\end{aligned}$$



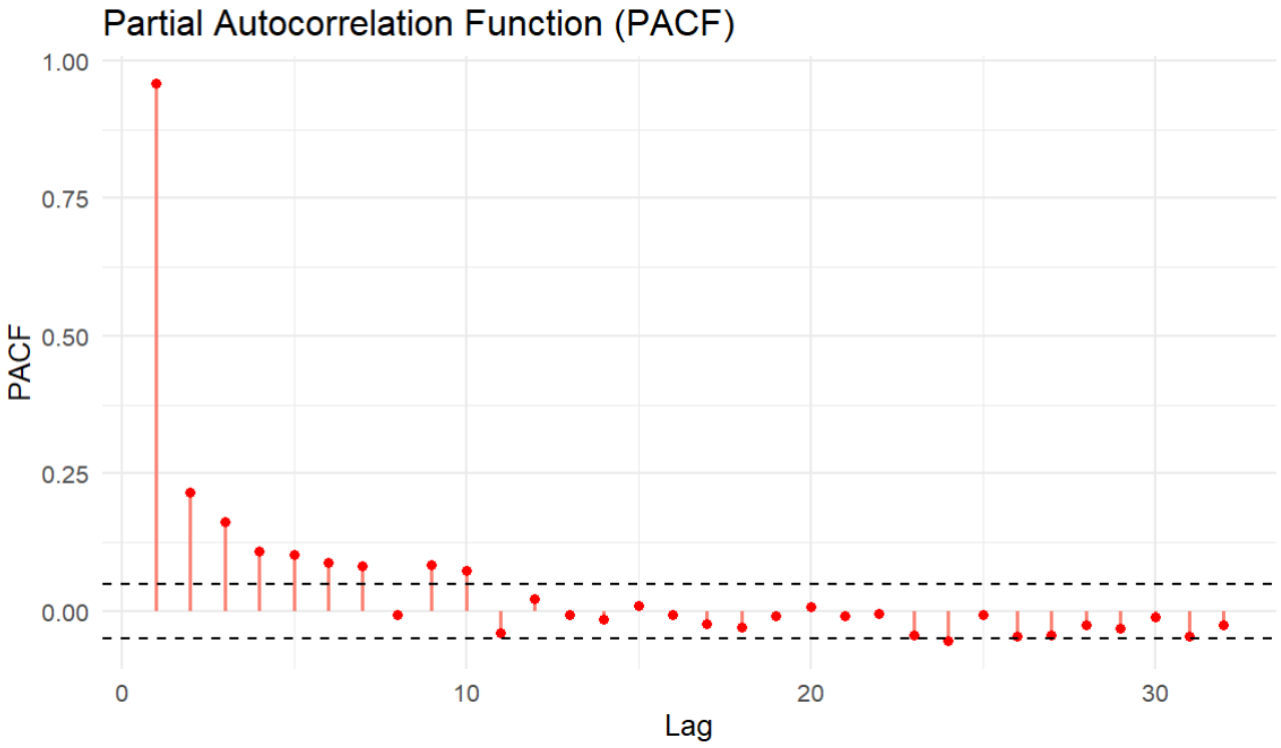
After first order Differencing:



5.2.2 Partial Autocorrelation Function (PACF)

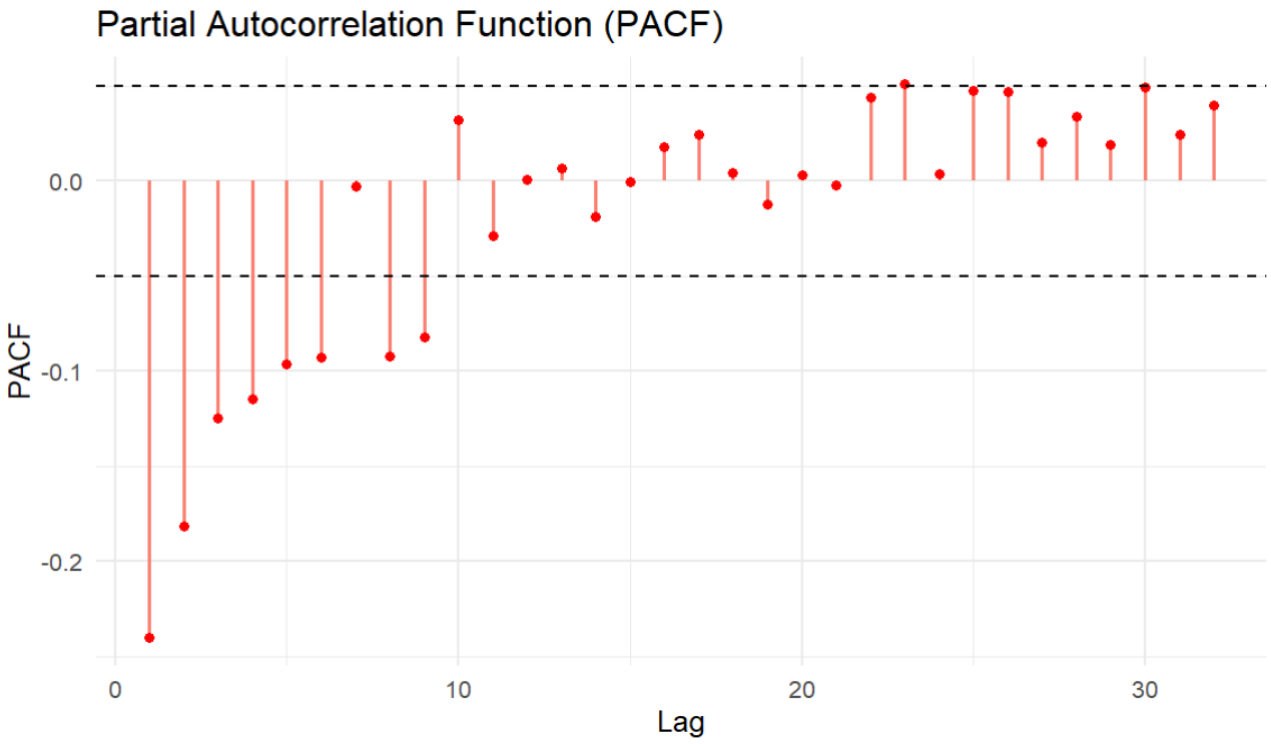
The Partial Autocorrelation Function (PACF) is another statistical tool used in time series analysis. PACF measures the correlation between a time series and its own lagged values while removing the effects of intermediate lags. It helps identify the direct relationship between the observations at two specific time points.

$$\begin{aligned}\phi_{kk} &= \mathrm{Corr}(Y_t, Y_{t-k} \mid Y_{t-1}, Y_{t-2}, \ldots, Y_{t-k+1}) \\ &= \frac{\det(\mathbf{R}_{kk})}{\det(\mathbf{R})}\end{aligned}$$



where $\mathbf{R}_{kk}$ is the auto-covariance matrix of lag k and $\mathbf{R}$ is the auto-covariance matrix of the entire time series.

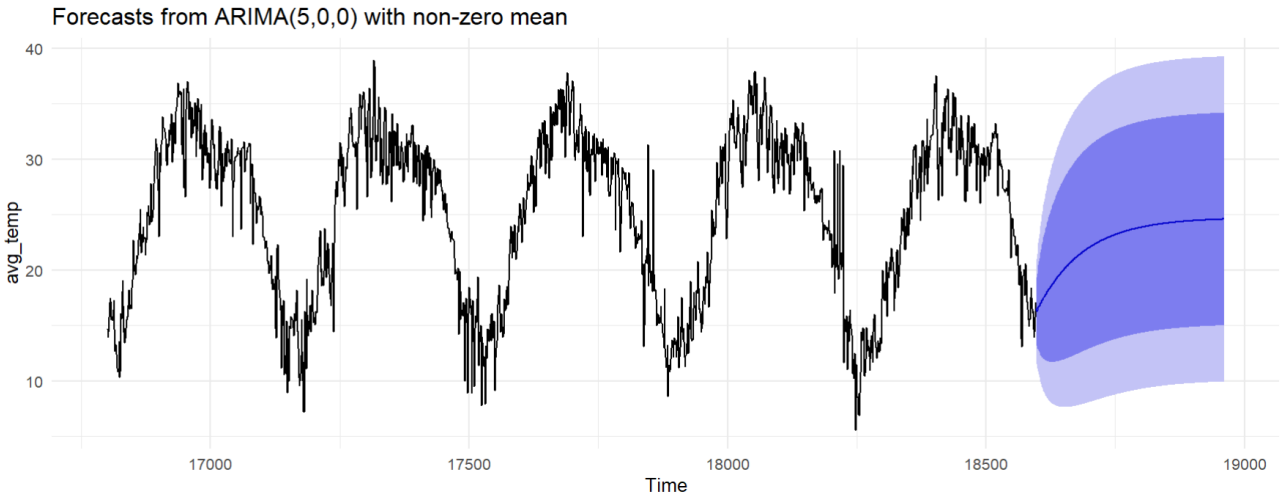
After first order Differencing:



6 Prediction

Listing 6.1: Plot

```
arima.forecast = forecast(arima.model, h=365)
autoplot(arima.forecast)
```



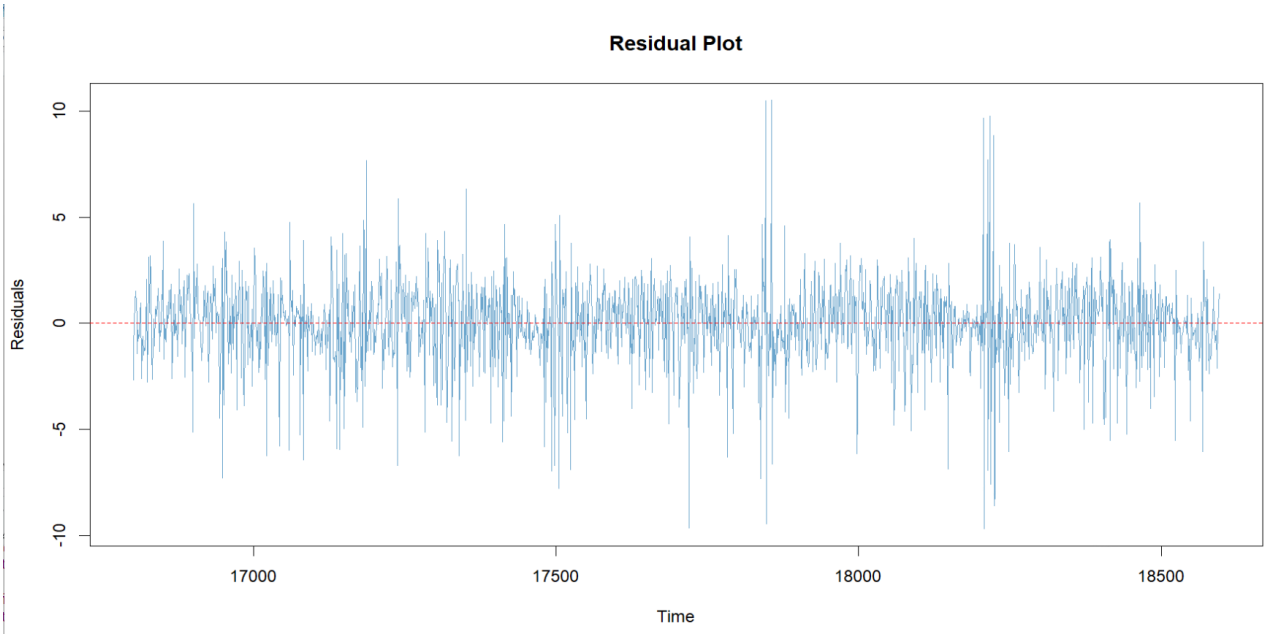
6.1 Residuals

In the context of time series analysis, residuals refer to the differences between the observed values of a time series and the values predicted by a model. These differences represent the unexplained or leftover variability in the data after the model has been applied.

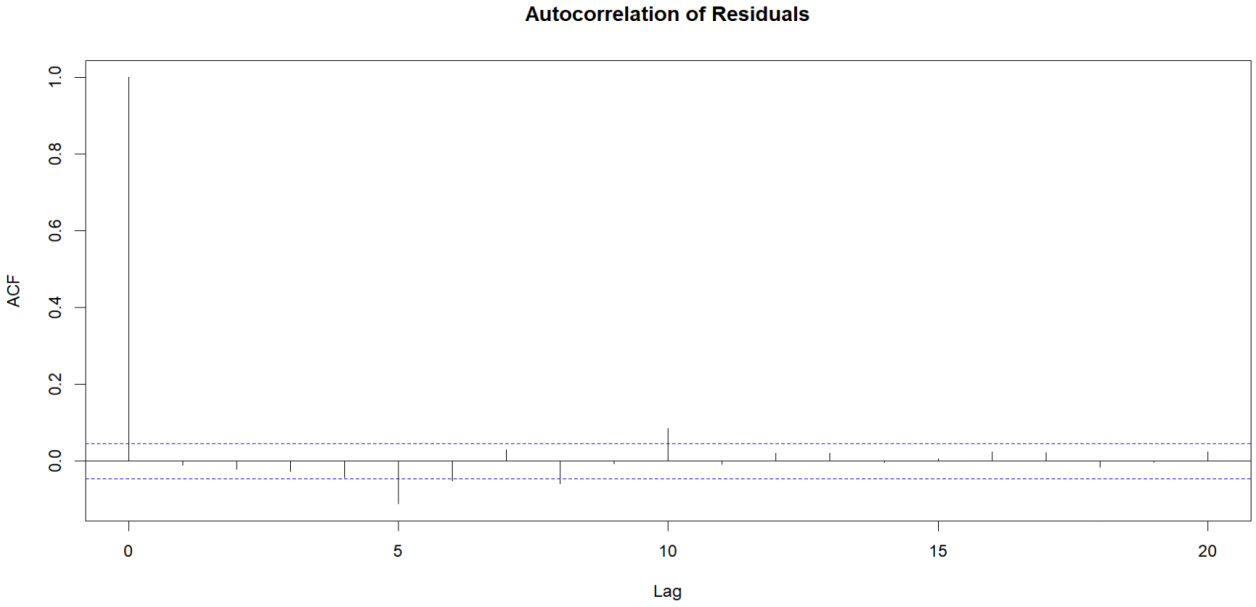
When you fit a time series model, such as an ARIMA (AutoRegressive Integrated Moving Average) model, to historical data, the model aims to capture the underlying patterns and structure in the data. However, there are often fluctuations, noise, or unexplained variations that the model does not capture perfectly. The residuals quantify these unexplained variations.

$$e_t = Y_t - \hat{Y}_t$$

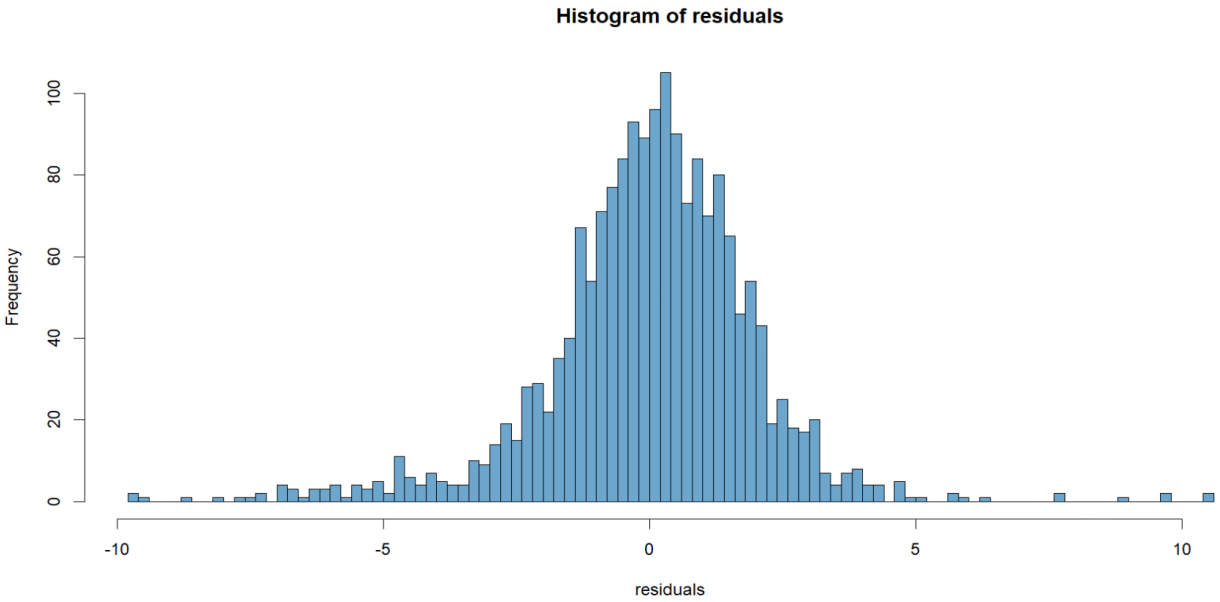
Residual Plots:



ACF of residual:



Histogram of residual:



7 Conclusion

In this project, we explored time series analysis methods to forecast the temperature in Delhi based on historical data spanning five years. We began by thoroughly examining the data, identifying trends, seasonality, and randomness. Utilizing R programming language, we employed ARIMA modeling, decomposition techniques, and statistical tests to gain insights into the underlying patterns.

Our analysis included the fitting of ARIMA models, visualizing the time series components, and evaluating the model performance. We leveraged tools such as ACF and PACF plots, and moving averages to diagnose and refine our models.