Time Series Data Analysis of Mauna Loa Nitrous Oxide Levels from 1997 to 2023

Ethan Weisgerber, Asit Das, Rahul Khatri

[University of Maryland, Baltimore County]
Department: Statistics 417
[December 2024]

Contents

Abstract

This paper presents a comprehensive time series analysis of Nitrous Oxide (N_2O) levels at Mauna Loa, Hawaii, from 1997 to 2023, with a focus on the last decade. By decomposing the time series into trend, seasonality, and residual components, we constructed a robust predictive model incorporating a third-degree polynomial trend, sinusoidal seasonality, and AR(17) residuals. The model demonstrates high predictive accuracy, with a Mean Squared Error (MSE) of 0.025 and a Mean Absolute Percentage Error (MAPE) of 0.039%, making it a reliable tool for forecasting future N_2O levels.

The analysis uncovered a consistent upward trend in N_2O levels, significant seasonal variations, and the absence of autocorrelation in the residuals after modeling. Forecasts for 2024–2026 predict a continued rise in N_2O levels, underscoring the importance of ongoing monitoring and mitigation strategies. Additionally, seasonal peaks in late autumn and early winter suggest potential environmental or anthropogenic influences that warrant further investigation.

Comparative analysis with Holt-Winters exponential smoothing revealed that both methods achieved exceptional accuracy, with Holt-Winters producing slightly lower error metrics. However, the custom model provides greater interpretability, offering valuable insights into the distinct contributions of each component. Future work could expand the dataset, incorporate external environmental factors, and explore advanced modeling techniques to refine accuracy further. This study highlights the critical role of time series modeling in environmental monitoring and its potential to inform climate research and policy.

1 Introduction

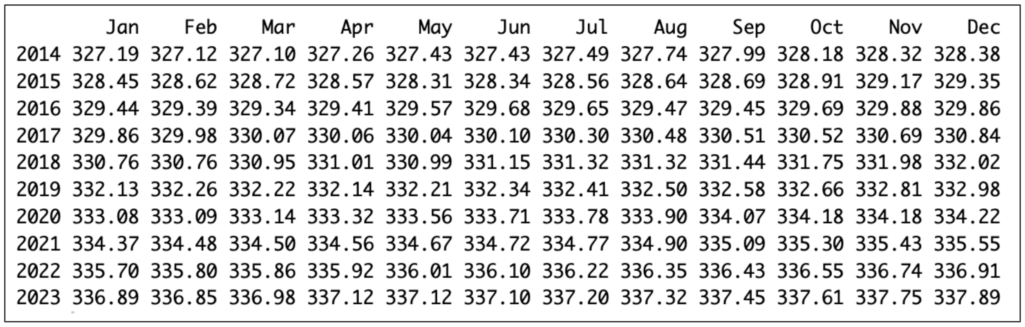
1.1 Data Description

The data for this project consists of monthly Nitrous Oxide (N_2O) levels, measured in parts per billion (ppb), near the Mauna Loa volcano in Hawaii. The dataset spans from May 1997 to December 2023 and is structured originally as a dataframe, with each row containing:

- Year: The calendar year of the measurement.
- Month: The corresponding month as an integer (1 for January, 2 for February, etc.).
- Value: The measured Nitrous Oxide concentration in ppb (parts per billion).

Nitrous Oxide is a potent greenhouse gas, and understanding its trends and patterns can contribute to climate research and environmental monitoring. The Mauna Loa site is an ideal location for this analysis due to its long-standing role in atmospheric observations, with minimal local interference.

For this project, we chose to focus on the last ten years of data, from January 2014 to December 2023, to capture recent trends and reduce potential biases from older data. First, we used the **ts()** function in R to format the dataframe as a time series, and additionally, the **window()** function to select the values starting from January, 2014 and onward. Figure 1 shows the values of the Nitrous Oxide levels as a time series in R:



```
        Jan    Feb    Mar    Apr    May    Jun    Jul    Aug    Sep    Oct    Nov    Dec
2014 327.19 327.12 327.10 327.26 327.43 327.43 327.49 327.74 327.99 328.18 328.32 328.38
2015 328.45 328.62 328.72 328.57 328.31 328.34 328.56 328.64 328.69 328.91 329.17 329.35
2016 329.44 329.39 329.34 329.41 329.57 329.68 329.65 329.47 329.45 329.69 329.88 329.86
2017 329.86 329.98 330.07 330.06 330.04 330.10 330.30 330.48 330.51 330.52 330.69 330.84
2018 330.76 330.76 330.95 331.01 330.99 331.15 331.32 331.32 331.44 331.75 331.98 332.02
2019 332.13 332.26 332.22 332.14 332.21 332.34 332.41 332.50 332.58 332.66 332.81 332.98
2020 333.08 333.09 333.14 333.32 333.56 333.71 333.78 333.90 334.07 334.18 334.18 334.22
2021 334.37 334.48 334.50 334.56 334.67 334.72 334.77 334.90 335.09 335.30 335.43 335.55
2022 335.70 335.80 335.86 335.92 336.01 336.10 336.22 336.35 336.43 336.55 336.74 336.91
2023 336.89 336.85 336.98 337.12 337.12 337.10 337.20 337.32 337.45 337.61 337.75 337.89
```

Figure 1: Nitrous Oxide Levels in ppb (2014-2023)

1.2 Objectives

The primary goal of this analysis is to develop a robust time series model to:

1. Analyze Trends and Seasonality: Examine the patterns and dynamics of Nitrous Oxide levels over the last ten years, including any long-term trends or seasonal fluctuations.
2. Model the Data: Build a predictive model that incorporates the identified trends, seasonality, and residual variations.
3. Forecast Future Levels: Use the model to forecast monthly Nitrous Oxide levels for the next two years (January 2024 to December 2025).
4. Provide Insights: Offer a clear understanding of the observed patterns and the forecasted changes, with implications for environmental monitoring and climate research.

This analysis is particularly relevant in the context of global climate change, as Nitrous Oxide contributes to both greenhouse gas warming and ozone layer depletion. The analysis aims to provide actionable insights aligned with current trends by focusing on recent data.

2 Time Series Data Analysis

2.1 Plotting and Decomposing

The time series data for the monthly Nitrous Oxide (N_2O) levels from January 2014 to December 2023 was plotted to understand its overall behavior (Figure 2). We used R's **stl()** function to decompose the time series into trend, seasonality, and residual components (Figure 3). This revealed key patterns:

- Trend: A steady upward trajectory suggests a long-term increase in Nitrous Oxide levels.
- Seasonality: Cyclical patterns with a periodicity of 12 months indicate seasonal variations.
- Residuals: Random noise remains after removing trend and seasonality.

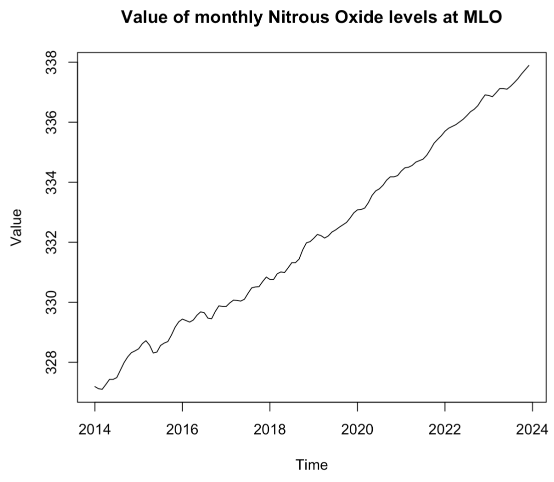


Figure 2: N_2O Levels (2014-2023)

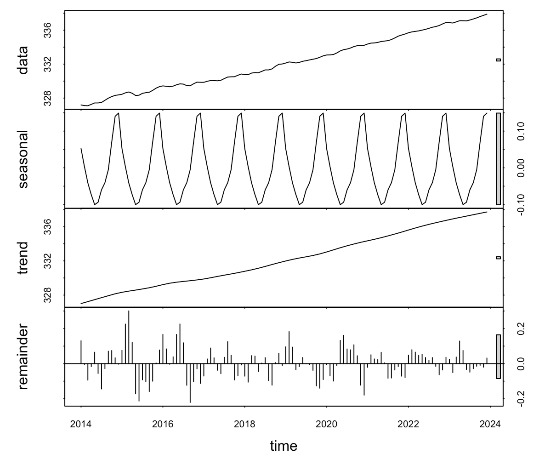


Figure 3: Decomposition of MLO Data

The plots showed clear seasonal patterns, while residual analysis highlighted the need for further modeling. The decomposition also allowed us the save the three components; trend, seasonality, and remainders (or residuals) as separate series defined by appropriately-named variables. The trend relates heavily to the time series plot in Figure 2, but the separate seasonality and residual components can be more closely viewed in the following Figures 4 and 5.

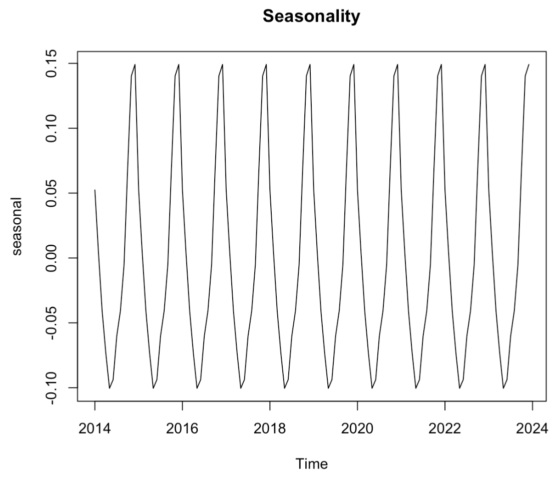


Figure 4: Seasonality of MLO Data

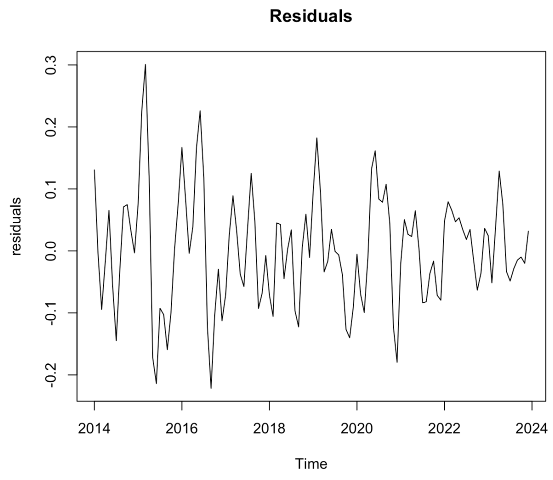


Figure 5: Residuals of MLO Data

The trend and seasonality need to be modeled with separate functions, and the residuals appear to be mean stationary. However, the autocorrelation cannot be determined by the plot, and additional analysis must be conducted.

2.2 Residual Analysis

The residuals had to be analyzed for randomness to ensure all information from the time series has been captured by the trend and seasonality models. The goal is for the residuals to resemble a white noise process, or one that is mean stationary with a mean of 0, and a constant variance given by σ^2.

When the autocovariance function was plotted using R's **acf()** function, the plot of the residuals indicated that they were not purely random, with significant autocorrelation visible at several lag points. This means there is still some information left in the residuals, and they are not purely random. The acf plot can be seen in Figure 6.

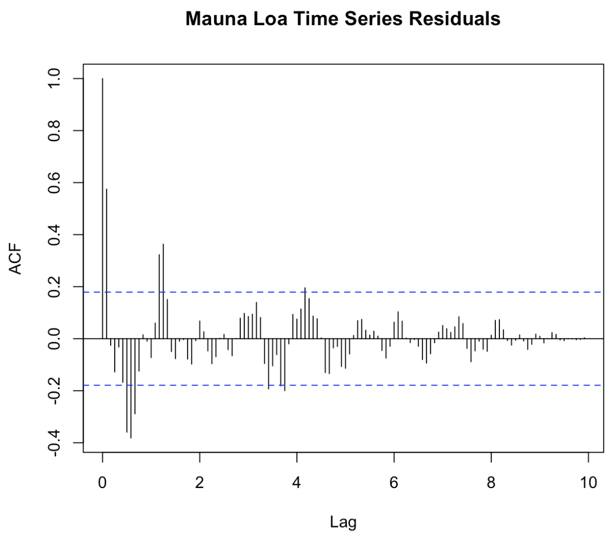
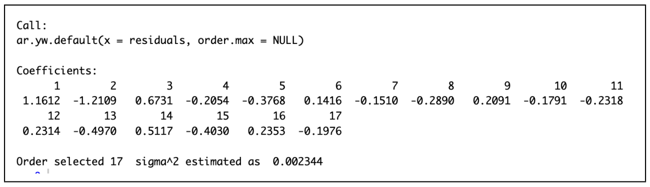


Figure 6: ACF Plot of Residuals from MLO Data

We consider the **ar.yw()** function to fit an AR(p) model where the order p is determined by the Akaike Information Criterion (AIC). An autoregressive model of order 17 (AR(17)) with $\sigma^2 = 0.002344$ was fit to the residuals. This model successfully extracted the remaining information, as confirmed by the ACF plots of the residuals of the AR(17) model and the Ljung-Box test performed on those residuals (p-value of 0.6666).



```
Call:
ar.yw.default(x = residuals, order.max = NULL)

Coefficients:
      1        2        3        4        5        6        7        8        9       10       11
 1.1612  -1.2109   0.6731  -0.2054  -0.3768   0.1416  -0.1510  -0.2890   0.2091  -0.1791  -0.2318
     12       13       14       15       16       17
 0.2314  -0.4970   0.5117  -0.4030   0.2353  -0.1976

Order selected 17  sigma^2 estimated as  0.002344
```

Figure 7: Residual Coefficients AR(17)

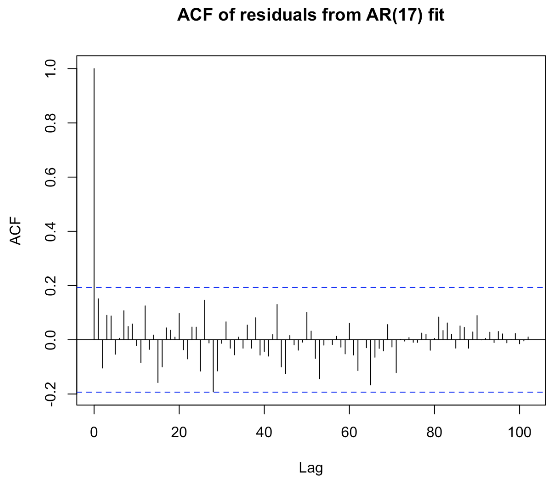


Figure 8: ACF Plot of Residuals from AR(17)

We successfully mapped the residuals to an AR(17) process in which residuals appear to be white noise. Now let's focus on modeling the trend and seasonality components.

2.3 Trend Analysis

To model the trend, the first step was to allocate the time series to a time index set from 1 to 120. This made it easier to map linear and exponential models. We then had to decide what function could best map the trend of the MLO data. Several approaches were evaluated:

- Linear Trend: A linear regression model suggested a simple upward trend but failed to capture slight curvatures in the data.
- Polynomial Trend: A third-degree polynomial captured the long-term trend more effectively, with coefficients showing significant contributions from quadratic and cubic terms. The **poly()** function was used for this, with different degrees tested until we found one that worked the best (3rd degree).
- Exponential Trend: An exponential model was tested and showed almost equal alignment to the actual trend line as the polynomial trend model. The **nls()** function, which stands for "non-linear least squares" was used. We had to set the amplitude, a, the initial growth rate, b, and the mean of the data points of the MLO data for the baseline offset, c.

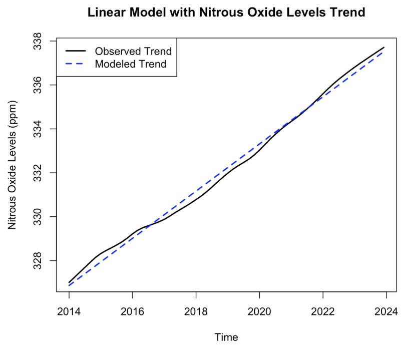


Figure 9: Linear Trend

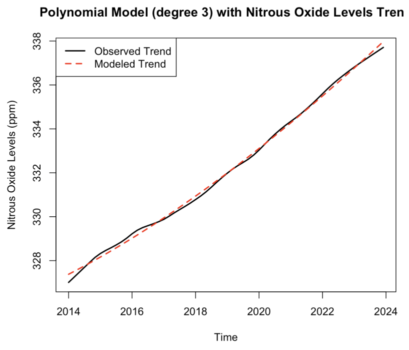


Figure 10: Polynomial Trend

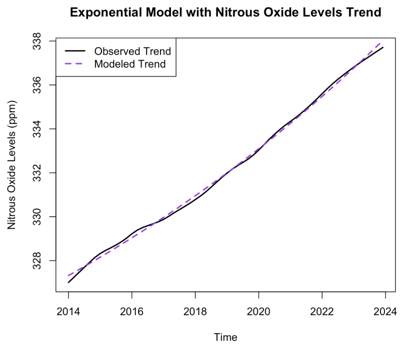


Figure 11: Exponential Trend

Taking a look at the plots of these functions, the polynomial and exponential models seems to capture the data's trend much more accurately than the linear model. Additionally, the linear model had a lower R^2 value (.9917) compared to the polynomial model (.9968) and a higher AIC score (43.8082) compared to the exponential model (-65.44655). Given this, it is likely that the linear model will not be part of the final model we construct, but we will still go ahead and perform testing on it in model fitting (2.5). Next we will turn our attention towards modeling the seasonality.

2.4 Seasonality Analysis

To model the seasonality of our data, the first step was to once again set a time index from 1-120, and to combine with the decomposed seasonality series into a single dataframe. The seasonality was modeled using trigonometric functions (cosine and sine terms) with up to three harmonics. Different levels of harmonics were tested, and three seemed to be the most accurate and effective for our model. The coefficients indicated significant contributions from the first three harmonics, accurately capturing the repeating seasonal pattern. The final seasonality model fitted with the **lm()** function gave a series of coefficients to obtain a final sinusoidal model given by:

$$\text{Seasonality}(t) = 1.172 \times 10^{-9} + 0.1071\cos(a \cdot t) - 0.03107\sin(a \cdot t) + 0.02307\cos(2a \cdot t)$$

$$-0.01183\sin(2a \cdot t) + 0.009294\cos(3a \cdot t) - 0.01536\sin(3a \cdot t)$$

Where $a = \frac{2\pi}{12} = \frac{\pi}{6}$ represents the fundamental frequency of the 12-month periodicity.
This model effectively captured the observed seasonal cycles, as seen in the fitted seasonality plot in Figure 12.

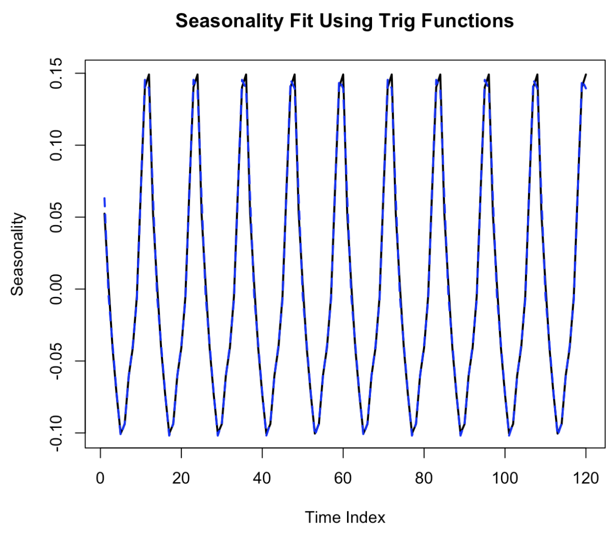
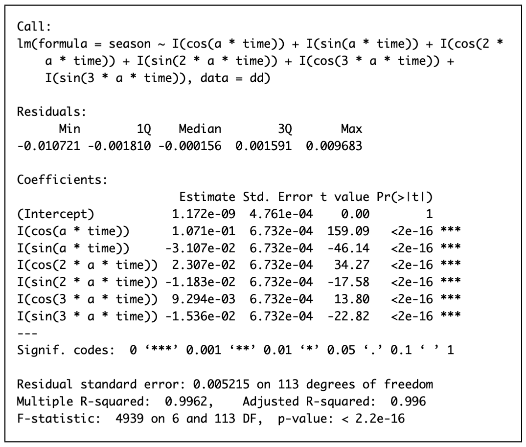


Figure 12: Fitted Seasonality

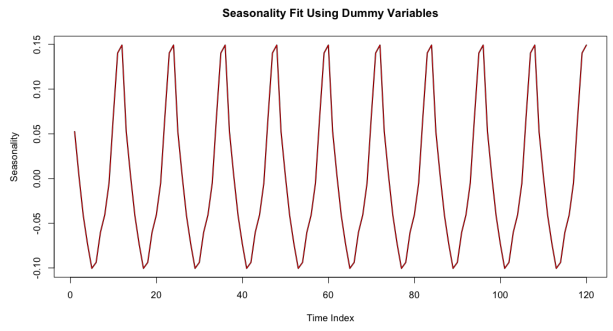


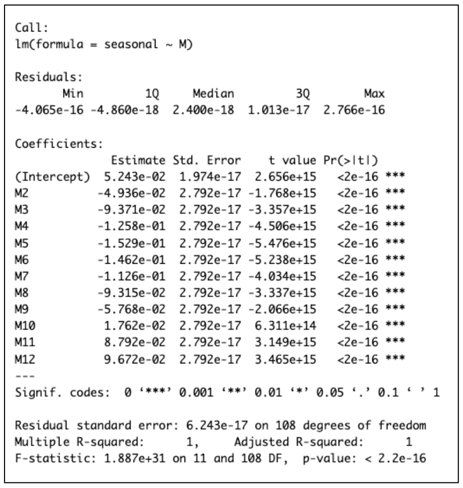
```
Call:
lm(formula = season ~ I(cos(a * time)) + I(sin(a * time)) + I(cos(2 *
    a * time)) + I(sin(2 * a * time)) + I(cos(3 * a * time)) +
    I(sin(3 * a * time)), data = dd)

Residuals:
      Min        1Q    Median        3Q       Max
-0.010721 -0.001810 -0.000156  0.001591  0.009683

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)           1.172e-09  4.761e-04    0.00        1
I(cos(a * time))      1.071e-01  6.732e-04  159.09   <2e-16 ***
I(sin(a * time))     -3.107e-02  6.732e-04  -46.14   <2e-16 ***
I(cos(2 * a * time))  2.307e-02  6.732e-04   34.27   <2e-16 ***
I(sin(2 * a * time)) -1.183e-02  6.732e-04  -17.58   <2e-16 ***
I(cos(3 * a * time))  9.294e-03  6.732e-04   13.80   <2e-16 ***
I(sin(3 * a * time)) -1.536e-02  6.732e-04  -22.82   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.005215 on 113 degrees of freedom
Multiple R-squared:  0.9962,    Adjusted R-squared:  0.996
F-statistic:  4939 on 6 and 113 DF,  p-value: < 2.2e-16
```

Figure 13: Summary of Seasonality Model

Figure 13 shows the summary of this fit, with all of the terms in the model being statistically significant, and a very low p-value. After modeling the trend, seasonality, and residual components, we then must put them together into an additive model and assess the best final model.

We also modeled the seasonality using a dummy variable model, where each dummy variable is associated with a month of the year. Even though the dummy variable model yields a higher R^2 value (1 versus .9962), the dummy variable terms are more difficult to interpret compared to the harmonic terms of the trig model. Hence, we continue with using the trig model in our analysis.





```
Call:
lm(formula = seasonal ~ M)

Residuals:
       Min         1Q     Median         3Q        Max 
-4.065e-16 -4.860e-18  2.400e-18  1.013e-17  2.766e-16 

Coefficients:
              Estimate Std. Error    t value Pr(>|t|)    
(Intercept)  5.243e-02  1.974e-17  2.656e+15   <2e-16 ***
M2          -4.936e-02  2.792e-17 -1.768e+15   <2e-16 ***
M3          -9.371e-02  2.792e-17 -3.357e+15   <2e-16 ***
M4          -1.258e-01  2.792e-17 -4.506e+15   <2e-16 ***
M5          -1.529e-01  2.792e-17 -5.476e+15   <2e-16 ***
M6          -1.462e-01  2.792e-17 -5.238e+15   <2e-16 ***
M7          -1.126e-01  2.792e-17 -4.034e+15   <2e-16 ***
M8          -9.315e-02  2.792e-17 -3.337e+15   <2e-16 ***
M9          -5.768e-02  2.792e-17 -2.066e+15   <2e-16 ***
M10          1.762e-02  2.792e-17  6.311e+14   <2e-16 ***
M11          8.792e-02  2.792e-17  3.149e+15   <2e-16 ***
M12          9.672e-02  2.792e-17  3.465e+15   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.243e-17 on 108 degrees of freedom
Multiple R-squared:      1,	Adjusted R-squared:      1 
F-statistic: 1.887e+31 on 11 and 108 DF,  p-value: < 2.2e-16
```

2.5 Model Fitting

The final models combine the three components in an additive model as follows:

$$Y_t = \text{Trend}(t) + \text{Seasonality}(t) + R(t)$$

For the first set of models we built, we generated white noise values using the **rnorm()** function with $\mu = 0$ and σ equaling the standard deviation of the original time series's residuals (obtained using **stl()** earlier).
Model 1: $\hat{y}_1$ consisted of the linear trend model, the seasonality model, and the random white noise residuals. When we observe the plot of this model, it seems as if the model doesn't capture the variation of the data very accurately. While not awful, we know there could be some improvement in the model. Figure 14 shows this model.

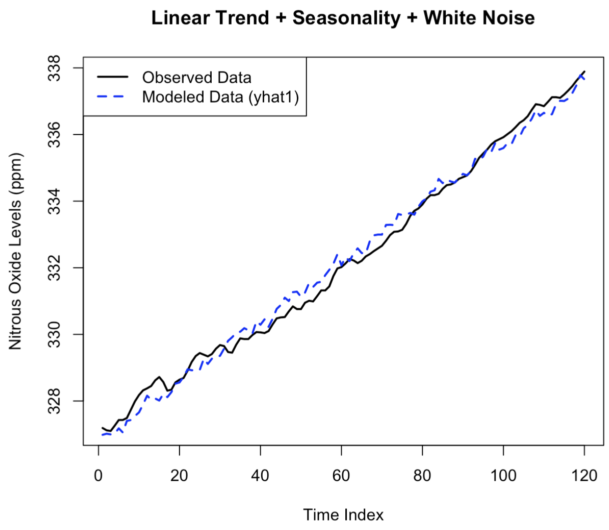


Figure 14: Model 1

Model 2: $\hat{y}_2$ consisted of the exponential trend model, the seasonality model, and the random white noise residuals. This model already appears to be a vast improvement over the previous model (which had the same seasonality/residuals but with a linear trend component). Figure 15 shows this model.

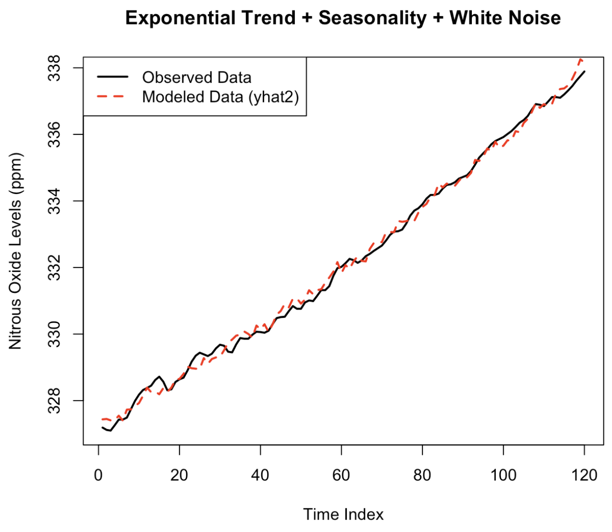


Figure 15: Model 2

Model 3: $\hat{y}_3$ consisted of the third-degree polynomial trend model, the seasonality model, and the random white noise residuals. This is the final model with random white noise residuals, and appears to be very similar in accuracy with $\hat{y}_2$. Figure 16 shows this model.

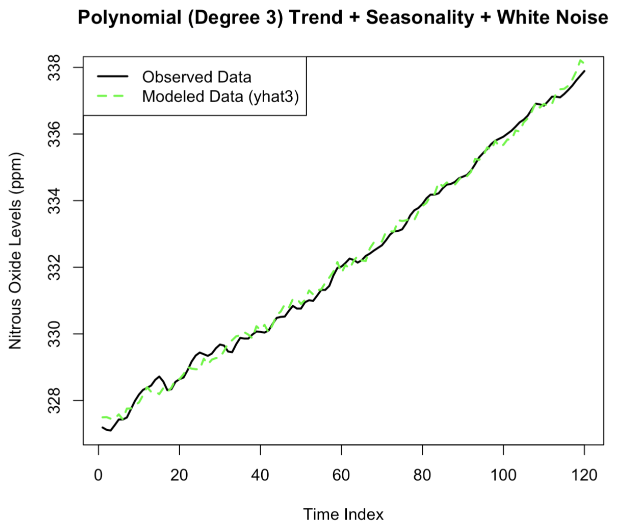


Figure 16: Model 3

After the three white noise models were plotted, we wanted to see how they compared to the same models but with residuals fitted from the AR(17) process. We decide to repeat the preceding process, but only with $\hat{y}_2$ and $\hat{y}_3$, since these seem to be visually superior to $\hat{y}_1$.
The first step was to fit our AR(17) model to extract residuals from. We used the **arima()** function with an order set to (17,0,0). The extracted AIC value was calculated to be -407.1485, which is low. However, we have nothing to compare this value to, so we decided to fit an AR model with order 15 and 20 to check their AIC values. These resulted in -405.9633 and -393.767, respectively, which are both higher in value. This AR(17) model seems to be a good choice for the residuals! Here is a comparison of the autocorrelation function plots of the residuals from the original dataset and the autoregressive model of order 17:

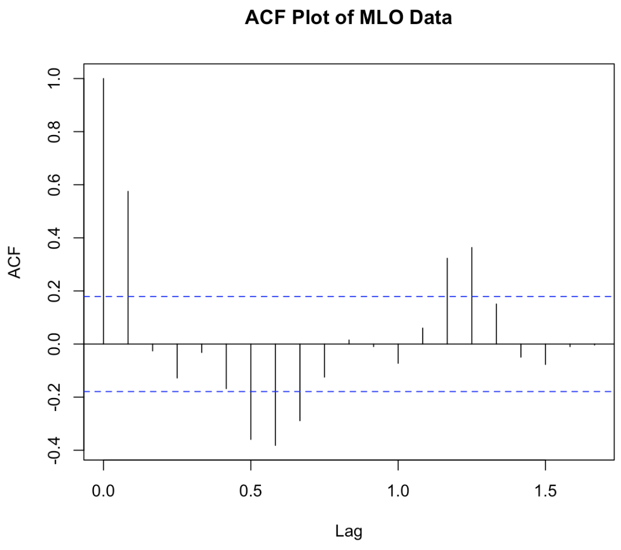


Figure 17: Original ACF

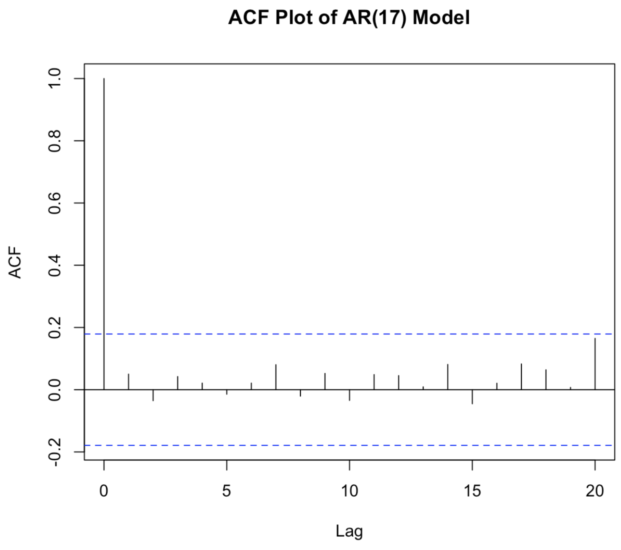


Figure 18: AR(17) ACF

We can see that Figure 17 shows clear signs of autocorrelation in its residuals, while the AR(17) model residuals appears to be white noise. We will proceed in model building, replacing the random white noise residuals with the fitted residuals.

Model 2 with AR(17): $\hat{y}_{2.2}$ consists of the exponential trend model, the seasonality model, and the AR(17) residuals. The resulting model seems to be more smooth to the actual data, and appears to be an improvement over $\hat{y}_2$. Figure 19 shows this model.

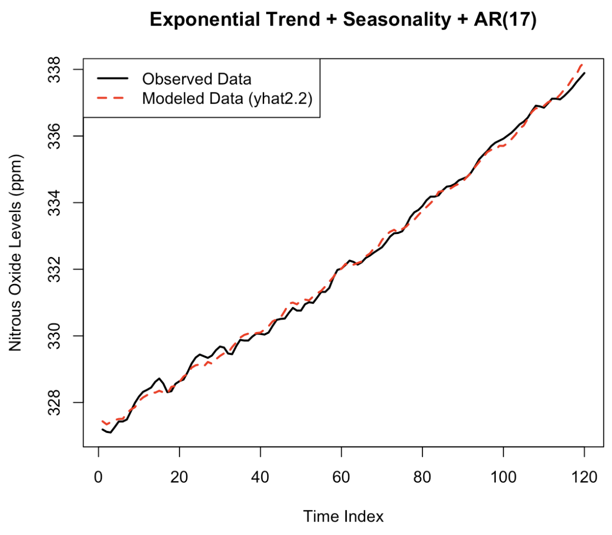


Figure 19: Model 2.2

Model 3 with AR(17): $\hat{y}_{3.2}$ consists of the third-degree exponential trend model, the seasonality model, and the AR(17) residuals. The resulting model also seems to be more smooth to the actual data, and appears to be an improvement over $\hat{y}_3$ as well. Figure 20 shows this model.

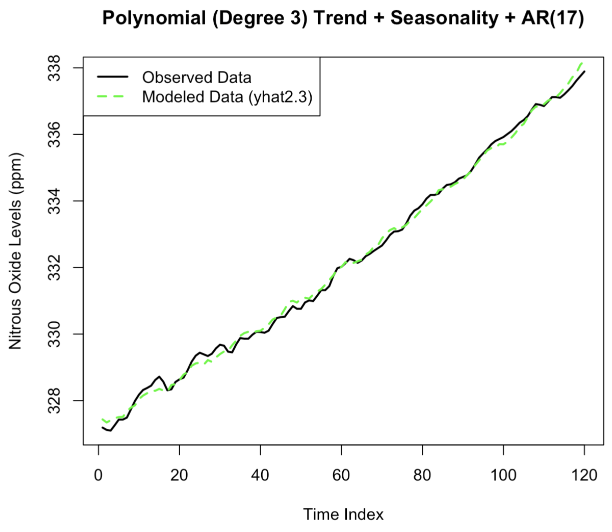


Figure 20: Model 3.2

After plotting the models, it seems as if $\hat{y}_{2.2}$ or $\hat{y}_{3.2}$ will be our final model. However, it makes sense to check the errors from the original data to see mathematically which model is superior. We will compute the Mean Squared Error (MSE) and the Mean Absolute Percentage Error (MAPE) for all five models for comparison reasons. For this criteria, the lowest value, or the value closest to zero, for either criteria will contain the smallest error among the models. The formulas used are as follows:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \quad \text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100$$

The results are calculated (in order of MSE, MAPE) in Table 1:

Model	**MSE**	**MAPE**
$\hat{y}_1$	0.091	0.078%
$\hat{y}_2$	0.041	0.050%
$\hat{y}_3$	0.040	0.049%
$\hat{y}_{2.2}$	0.026	0.040%
$\hat{y}_{3.2}$	**0.025**	**0.039%**

Table 1: Performance Metrics for Each Model

$\hat{y}_{3.2}$ came out to be the best model with the lowest MSE and MAPE values, only a fraction smaller than that of $\hat{y}_{2.2}$. These models would probably both be sufficient to use with such small error values, but we will choose the polynomial $\hat{y}_{3.2}$ to be our final model for the Nitrous Oxide levels at Mauna Loa, Hawaii. This was the model with a third degree polynomial trend, the sinusoidal seasonality, and the residuals from an AR(17) process.
We can show the final model for this MLO data is given by

$$Y_t = \text{Trend}(t) + \text{Seasonality}(t) + R(t)$$

Where:

- $Y(t)$: Observed Nitrous Oxide level
- Trend(t): Third-degree polynomial trend.
- Seasonality(t): Sinusoidal seasonalily model.

- $R(t)$: Residuals modeled using AR(17).

The polynomial model's equation for the trend is as follows:

$$\text{Trend}(t) = 332.1947 + 33.9708 \cdot t + 2.4221 \cdot t^2 - 0.1191 \cdot t^3$$

With the seasonality function as before, and AR(17) residuals. Before we progress, we wanted to check the residuals for the final model after it is combined with the other two components. We used the **checkresiduals()** function to plot the acf as well as conduct a Ljung-Box Test, both of which can be seen below:

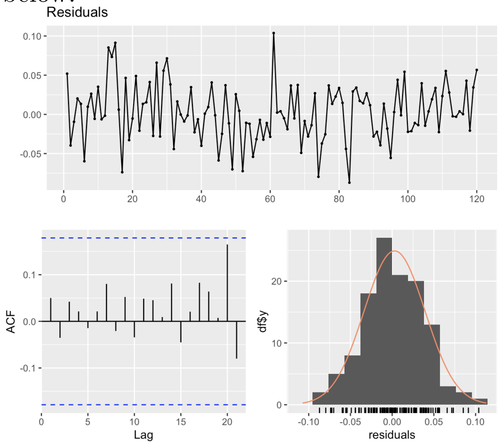
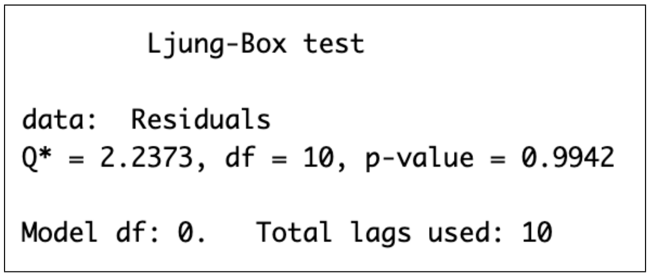


Figure 21: Final Model Residual Analysis



```
        Ljung-Box test

data:  Residuals
Q* = 2.2373, df = 10, p-value = 0.9942

Model df: 0.   Total lags used: 10
```

Figure 22: Ljung-Box Test for Final Model

The ACF plot appears to resemble white noise, which is confirmed by a high p-value in the Ljung-Box Test at 0.9942, which indicates that there is no significant evidence of autocorrelation in the residuals of the time series model.
Finally, we renamed $\hat{y}_{3.2}$ to an appropriately named final model variable, and before evaluating this model, it is almost ready for forecasting.

2.6 Model Evaluation

Prior to forecasting, we wanted to validate the accuracy of our model in the most recent years to ensure our future forecasted values will be as accurate as possible. We decided to test our model up against the last two years of the actual N_2O levels.
We split the data into a training set (2014-2021) and a testing set (2022-2023) using the **window()** function, where the final model is applied to the training set, and the forecast is compared with the actual test data from 2022-2023. We plotted the actual data for the last two years as well as the model's forecast of the last two years. That plot can be seen in Figure 23.

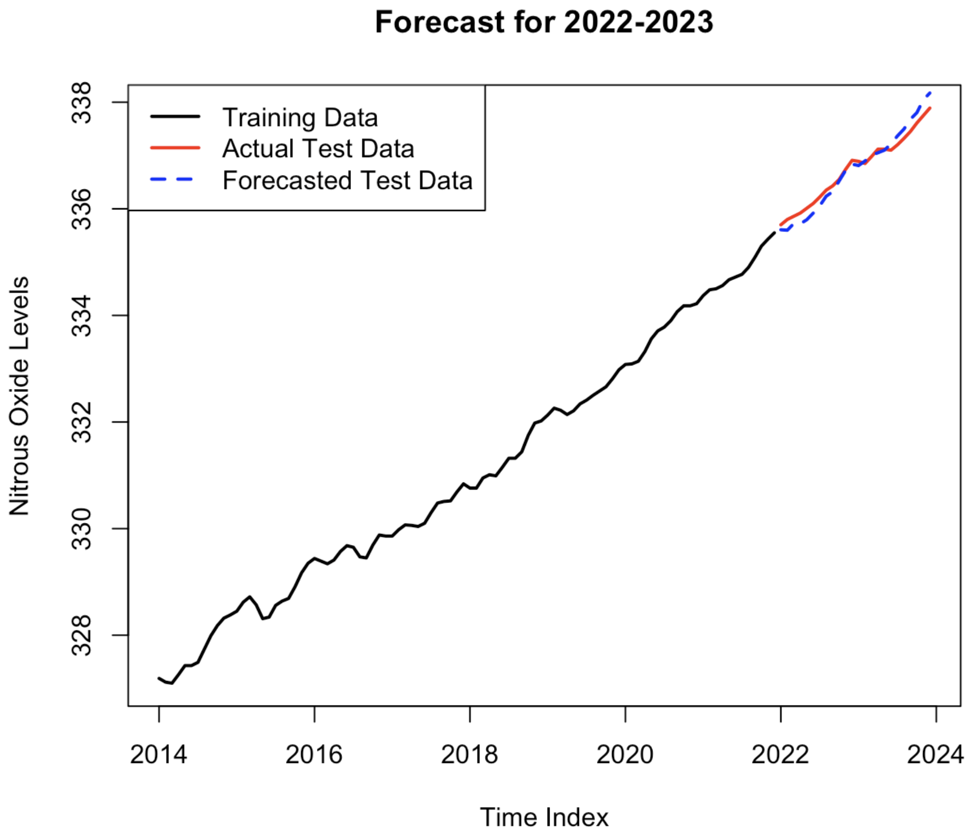


Figure 23: Forecast for 2022-2023

Our model seems to perform very well, and the predicted values for the last two years appear to be very close to the actual test data. The model even covers the small upward hump in the middle, around the beginning of 2023. The next step of course is to check performance metrics such as MSE and MAPE. These values come out to be:

$$\text{MSE} = 0.0246, \quad \text{MAPE} = 0.0400\%$$

The MSE returns a very low value, indicating that the average squared differences between the forecasted and actual values are minimal. A smaller MSE generally reflects better model performance. The very low MAPE is an excellent result, indicating that, on average, the forecasted values deviate from the actual values by only 0.04% relatively to their magnitude. A MAPE below 5% is typically considered very accurate for most forecasting applications.

2.7 Forecasting

Finally, we were ready to forecast future values of Nitrous Oxide in Mauna Loa, Hawaii, which was the main reason this research and analysis was conducted. This final model is used to forecast future nitrous oxide levels for the years 2024-2026, or two years after the final value of our actual data was recorded. To conduct the forecast, two methods were taken:

1. Use R's **forecast()** function from the **forecast** library on our final model
2. Use model we created to manually map values in the future

The former is much simpler to conduct, and the plot with a forecasted future and confidence intervals from our final model as well as the actual values can be seen in Figure 24 and 25:

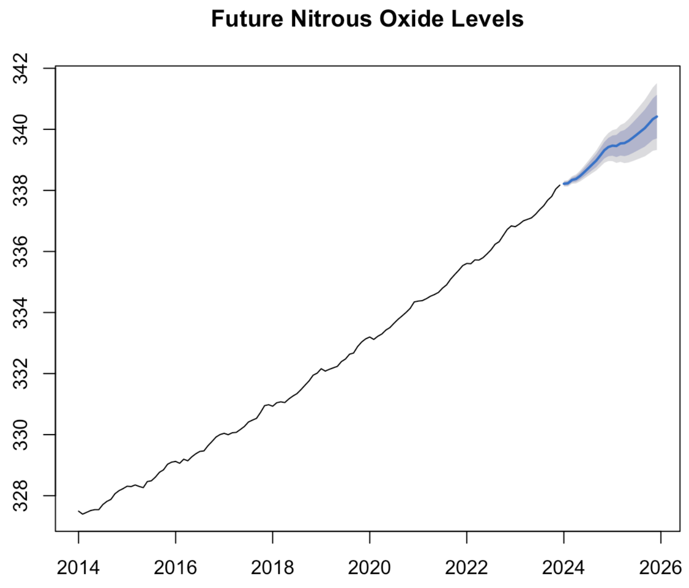
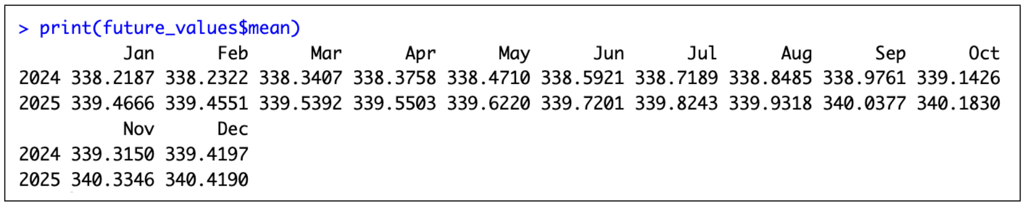


Figure 24: Forecast for 2024-2026



```
> print(future_values$mean)
          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug      Sep      Oct
2024 338.2187 338.2322 338.3407 338.3758 338.4710 338.5921 338.7189 338.8485 338.9761 339.1426
2025 339.4666 339.4551 339.5392 339.5503 339.6220 339.7201 339.8243 339.9318 340.0377 340.1830
          Nov      Dec
2024 339.3150 339.4197
2025 340.3346 340.4190
```

Figure 25: Forecasted Values 2024-2026

The latter deemed to be more challenging, and ultimately less rewarding. We went back to our fits for the polynomial trend, the sinusoidal seasonality, and the AR(17) residuals and simply extended the time index from 120 (ten years) to 140 (twelve years), as this would give us a model that extends to the end of 2025. The forecasted/expected values for this method are plotted below alongside the historical data and continue to show a clear upward trend that reflects the historical growth in nitrous oxide levels.

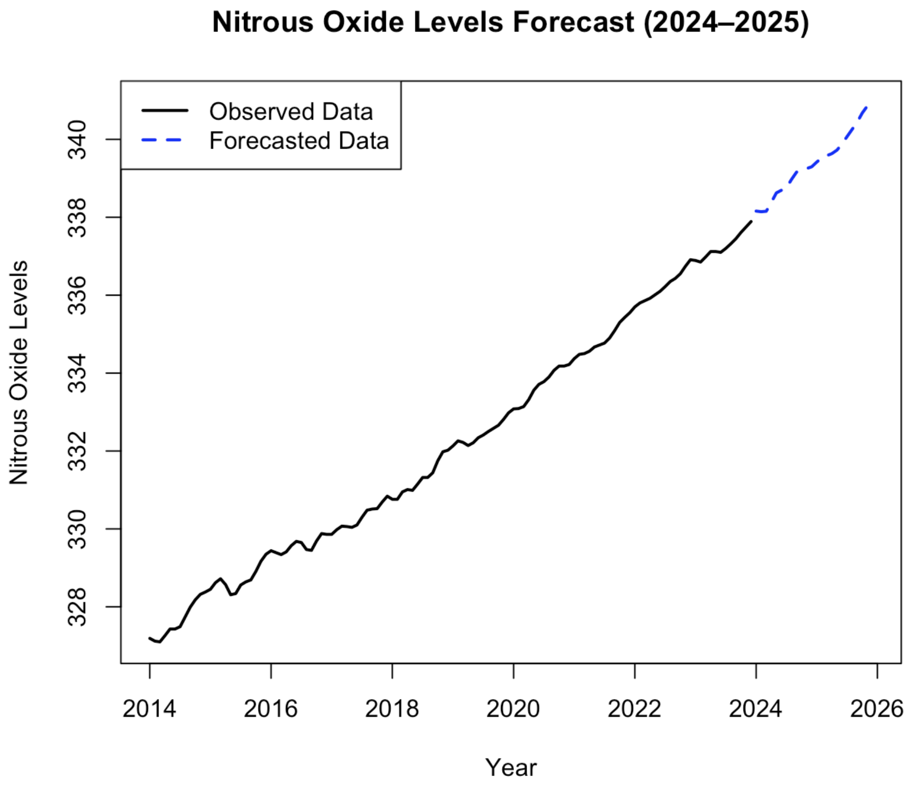


Figure 26: Manual Forecast for 2024-2026

While the values can be extracted, it is more useful to have access to confidence intervals, as there is always room for error in any model, no matter how rigorous. Additionally, the former method allows us to simply change the number of months from h=24 to however many months we'd like to forecast into the future, whilst the latter, manual method would require generating coefficients for each component each time the time is changed. For this reason, the values extracted using the **forecast()** function will be our final predictions.

2.8 Holt Winter's

For an additional test, we decided to create a forecast using Holt Winter's exponential smoothing. We chose this over a simple and Holt's exponential smoothing as there is clear seasonality and trend in the data. Using the original ten-year data, the **HoltWinters()** function was applied and plotted below. The values of the smoothing parameters are as follows: $\alpha = 0.8332089$, $\beta = 0$, and $\gamma = 1$. With a higher alpha, the model is highly responsive to recent changes in the data's level, adapting quickly to new observations. The trend parameter $\beta = 0$ suggests that the data is considered stationary in terms of its overall growth or decline. This makes sense, as our trend is fairly stable over time. Finally, the seasonal parameter $\gamma = 1$ suggests the data is assumed to be perfectly periodic and repetitive, with no need for smoothing or adjustments from past observations. Additionally, the next 24 months, or two years of Nitrous Oxide levels were forecasted.

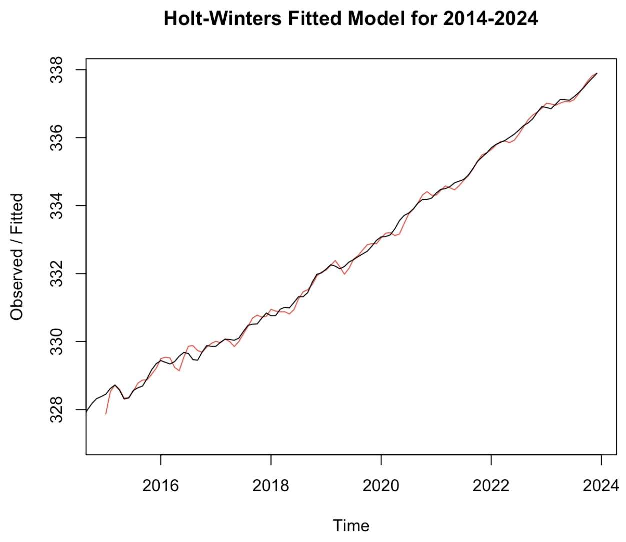


Figure 27: Holt Winter's Model

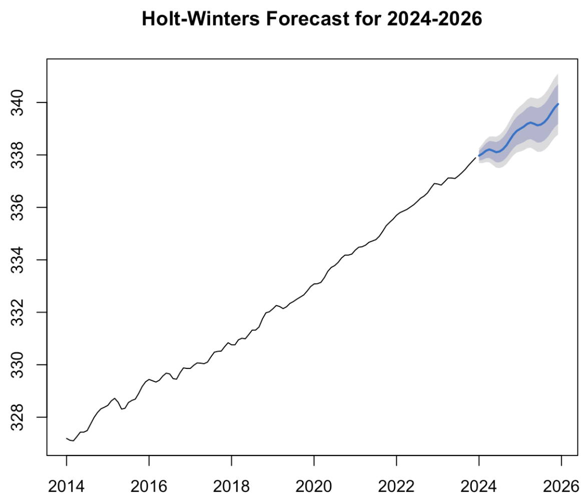
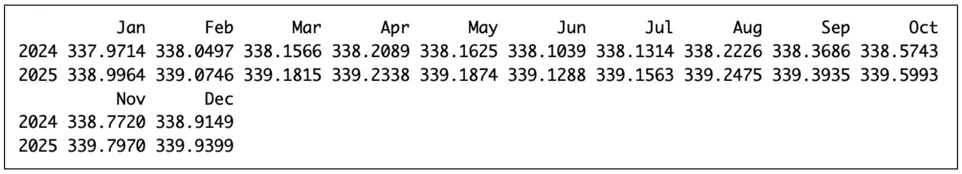


Figure 28: Holt Winter's Forecast



```
           Jan      Feb      Mar      Apr      May      Jun      Jul      Aug      Sep      Oct
2024 337.9714 338.0497 338.1566 338.2089 338.1625 338.1039 338.1314 338.2226 338.3686 338.5743
2025 338.9964 339.0746 339.1815 339.2338 339.1874 339.1288 339.1563 339.2475 339.3935 339.5993
           Nov      Dec
2024 338.7720 338.9149
2025 339.7970 339.9399
```

Figure 29: Holt Winter's Forecasted Values

When calculating performance metrics for the Holt Winter's model, a MSE of 0.020 is given, and a MAPE of 0.031%, both of which come very close to our model's of 0.026 and 0.040%, respectively. The Holt Winter's model consists of slightly less error, which is impressive how close our model came to a computer generated model.

2.9 Annual Spikes

When we plotted the seasonality, we were able to obtain a heavily fluctuating graph. This can be seen in Figure 4. The question lies: Which month of the year has higher or lower Nitrous Oxide levels than others? To perform this test, we created a barplot of the seasonal components averaged over all years. This plot can be seen in Figure 30.

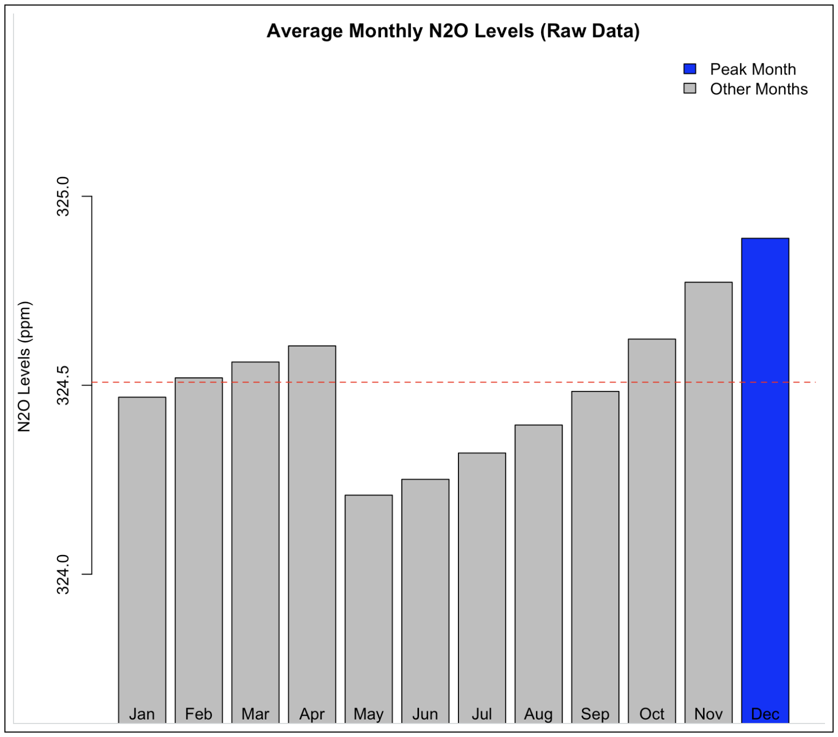


Figure 30: Bar Plot of Monthly Average N_2O Levels

It seams as if the levels at the end of the year spike, only to dip down in May. This may be caused by seasonal climate that affects Nitrous Oxide in the air or increased yearly volcanic activity.

3 Conclusion

This analysis successfully explored the trends, seasonality, and residual components of Nitrous Oxide (N_2O) levels recorded at Mauna Loa, Hawaii, over the last decade. Through careful decomposition, model fitting, and evaluation, we developed a robust time series model capable of accurately forecasting future N_2O levels. The final model—combining a third-degree polynomial trend, sinusoidal seasonality, and AR(17) residuals—demonstrated strong predictive accuracy with minimal error, as indicated by its MSE (0.025) and MAPE (0.039%).
The analysis revealed key insights into the data:

- A clear upward trend in N_2O levels, highlighting long-term growth likely driven by human activities and climate effects.
- Strong seasonality, with higher levels typically occurring in late autumn and early winter, potentially influenced by seasonal climatic or biological factors.
- Residuals showing no evidence of autocorrelation after incorporating AR(17), validating the adequacy of the model components.

When comparing our custom model to the Holt-Winters forecast method, it is evident that both approaches provide highly accurate forecasts, with the Holt-Winters forecast achieving slightly lower error metrics. However, the custom model offers greater interpretability, enabling deeper insights into the distinct contributions of trend, seasonality, and residuals.
The forecast for 2024–2026 projects a continued rise in N_2O levels, emphasizing the need for ongoing monitoring and mitigation strategies. Additionally, the discovery of seasonal peaks suggests further investigation into how environmental or anthropogenic factors influence Nitrous Oxide emissions throughout the year.

Future work could involve:

- Expanding the dataset to include recent years for improved forecasting.
- Incorporating external factors (e.g., temperature, vegetation cycles) to enhance the model's explanatory power.
- Exploring other advanced modeling techniques like machine learning to refine accuracy.

This analysis underscores the power of time series modeling for environmental monitoring and provides a strong foundation for further research into greenhouse gas dynamics at Mauna Loa and beyond.

References

[1] Coghlan, A. (2024). "Welcome to a Little Book of R for Time Series." *Welcome to a Little Book of R for Time Series! - Time Series 0.2 Documentation.* Retrieved December 4, 2024, from *https://a-little-book-of-r-for-time-series.readthedocs.io/en/latest/.*

[2] m Laboratory, NOAA Global Monitoring. "Dataset MLO N2O - NOAA Global Monitoring Laboratory." *GML*, 30 July 2024, `https://gml.noaa.gov/data/dataset.php?item=mlo-n2o-flask-month`.

[3] Wickham, H. (2016). *ggplot2: Elegant graphics for data analysis* (2nd ed.). Springer

R Code

```
library(tseries)
library(forecast)

mauna_loa <- read.table("mauna_loa.txt", header = TRUE, sep = "", fill = TRUE)
head(mauna_loa)
tail(mauna_loa)
length(mauna_loa)

values <- ts(mauna_loa[,4], start=c(1997,5), frequency=12)
values


plot.ts(values, main='Monthly Nitrous Oxide Levels 1997-2023')


values_last_10 <- window(values, start=c(2014, 1))
values_last_10

plot(values_last_10, ylab='Value', main='Monthly Nitrous Oxide Levels 2014-2023', col = "black")
lines(lowess(values_last_10, f=0.10), lwd=2, lty=1, col="blue")
lines(lowess(values_last_10, f=0.05), lwd=2, lty=2, col="red")

# Decompose time series into seasonality, trend, and remainders
decomp <- stl(values_last_10, "per")
plot(decomp)
names(decomp)
head(decomp$time.series)
```

```
# Save and plot these features
seasonal <- decomp$time.series[,1]
trend <- decomp$time.series[,2]
residuals <- decomp$time.series[,3]

par(mfrow=c(1,1))
plot(seasonal, main='Seasonality')
plot(trend, main='Trend')
plot(residuals, main="Residuals")

# Use acf to see if residuals are white noise
acf(residuals, lag.max = 500, main='Mauna Loa Time Series Residuals')
# We can see that it is not quite white noise, as values are outside blue lines

# Let's try to model the residuals autoregressively to see where this dependency lies
# We consider the function ar.yw to fit the AR(p) model where the order p is
# determined by the information criterion AIC.
fit_ar <- ar.yw(residuals, order.max = NULL)
fit_ar
# This fit suggests the order is 17, AR(17), with Wt ~ WN(0, 0.002344)

# Let's study the residuals from this AR(17) model to check for autocorrelation
names(fit_ar)
model_resid <- ts(fit_ar$resid)
model_resid
# Remove first 17 values as they are NA (No autocorrelation until after lag p=17)
acf(model_resid[-c(1:17)], main="ACF of residuals from AR(17) fit", lag.max = 500)
# This resembles white noise!
checkresiduals(model_resid[-c(1:17)])

# We have a good model for our residuals to be white noise, now let's focus on trend
# and seasonality

plot(trend)

# Extract time information from the time series object
time_index <- seq_along(values_last_10)
time_index

# Linear trend
linear_fit <- lm(as.numeric(values_last_10) ~ time_index)
linear_trend <- ts(predict(linear_fit), start = start(values_last_10), frequency =
frequency(values_last_10))

plot(trend, type = "l", col = "black", lwd = 2,
     main = "Linear Model with Nitrous Oxide Levels Trend",
     ylab = "Nitrous Oxide Levels (ppm)", xlab = "Time")
lines(linear_trend, col = "blue", lwd = 2, lty = 2)
legend("topleft", legend = c("Observed Trend", "Modeled Trend"),
       col = c("black", "blue"), lty = c(1, 2), lwd = 2)

# Polynomial trend (degree 3)
poly_fit <- lm(as.numeric(values_last_10) ~ poly(time_index, 3))
poly_fit
```

```
poly_trend <- ts(predict(poly_fit), start = start(values_last_10), frequency =
frequency(values_last_10))

plot(trend, type = "l", col = "black", lwd = 2,
     main = "Polynomial Model (degree 3) with Nitrous Oxide Levels Trend",
     ylab = "Nitrous Oxide Levels (ppm)", xlab = "Time")
lines(poly_trend, col = "red", lwd = 2, lty = 2)
legend("topleft", legend = c("Observed Trend", "Modeled Trend"),
       col = c("black", "red"), lty = c(1, 2), lwd = 2)

# Exponential trend
a_use <- (max(values_last_10) - min(values_last_10)) / 10
b_use <- 0.001
c_use <- mean(values_last_10)

exp_fit <- nls(as.numeric(values_last_10) ~ a * exp(b * time_index) + c,
               start = list(a = a_use, b = b_use, c = c_use))
exp_trend <- ts(predict(exp_fit), start = start(values_last_10), frequency =
frequency(values_last_10))

plot(trend, type = "l", col = "black", lwd = 2,
     main = "Exponential Model with Nitrous Oxide Levels Trend",
     ylab = "Nitrous Oxide Levels (ppm)", xlab = "Time")
lines(exp_trend, col = "purple", lwd = 2, lty = 2)
legend("topleft", legend = c("Observed Trend", "Modeled Trend"),
       col = c("black", "purple"), lty = c(1, 2), lwd = 2)


# We have an okay linear model, and pretty accurate looking exponential and polynomial models!
# They are saved as linear_trend, poly_trend, and exp_trend, respectively.

# Now we will turn our attention to modeling seasonality

plot(seasonal, main='Seasonality')

time_index <- seq_along(seasonal)

a <- 2 * pi/12
dd <- data.frame(season = seasonal, time = time_index)
head(dd)

fit_ss <- lm(season ~ I(cos(a * time)) + I(sin(a * time)) +
               I(cos(2 * a * time)) + I(sin(2 * a * time)) +
               I(cos(3 * a * time)) + I(sin(3 * a * time)), data = dd)
fit_ss
coef_ss <- summary(fit_ss)$coefficients

# Generate fitted values
fitted_seasonality <- predict(fit_ss)

plot(time_index, seasonal, type = "l", col = "black", lwd = 2,
     main = "Seasonality Fit Using Trig Functions",
     xlab = "Time Index", ylab = "Seasonality")
lines(time_index, fitted_seasonality, col = "blue", lwd = 2, lty = 2)
```

```
summary(fit_ss)


# use dummy variable model for seasonality
M <- factor(rep(1:12, length=length(time_index), levels=1:12))
M
ss_factor_model <- lm(seasonal ~ M)
summary(ss_factor_model)
ss_factor_coeff <- summary(ss_factor_model)$coef
plot(time_index, seasonal, type = "l", col = "black", lwd = 2,
     main = "Seasonality Fit Using Dummy Variables",
     xlab = "Time Index", ylab = "Seasonality")
lines(time_index, ss_factor_model$fitted.values, col="red", lwd=1)


# Now we have a trigonometric trend for the seasonality, which means we have seasonality,
# trend, and white noise residuals. Let's create some models!

# Additive models are denoted by Yt = Trend(t) + Seasonality(t) + Residuals(t)

### MODEL 1 ###

# This will be an additive model with linear trend, seasonality, and white noise

# Ensure they are numeric for adding
class(linear_fit)  # Returns 'lm' - need to convert
class(fitted_seasonality)  # Returns 'numeric' - no need to convert
linear_fit <- as.numeric(linear_trend)
class(linear_fit)

set.seed(12345)
white_noise <- rnorm(length(time_index), mean=0, sd=sd(residuals))
# Create basic white noise for testing purposes

# Make our model
yhat1 <- linear_fit + fitted_seasonality + white_noise

# Plot
plot(time_index, values_last_10, type = "l", col = "black", lwd = 2,
     main = "Linear Trend + Seasonality + White Noise",
     xlab = "Time Index", ylab = "Nitrous Oxide Levels (ppm)")
lines(time_index, yhat1, col = "blue", lwd = 2, lty = 2)
legend("topleft", legend = c("Observed Data", "Modeled Data (yhat1)"),
       col = c("black", "blue"), lty = c(1, 2), lwd = 2)

# Plot Residuals
residuals_yhat1 <- values_last_10 - yhat1
plot(time_index, residuals_yhat1, type = "p", col = "blue", pch = 20,
     main = "Residuals of yhat1 Model",
     xlab = "Time Index", ylab = "Residuals")
abline(h = 0, col = "red", lty = 2)
```

```
### MODEL 2 ###

# This will be an additive model with exponential trend, seasonality, and white noise

# Extract fitted values from exponential model
exp_trend <- predict(exp_fit)

exp_trend <- as.numeric(exp_trend)

yhat2 <- exp_trend + fitted_seasonality + white_noise

plot(time_index, values_last_10, type = 'l', col = 'black', lwd = 2,
     main = "Exponential Trend + Seasonality + White Noise",
     xlab = 'Time Index', ylab = 'Nitrous Oxide Levels (ppm)')
lines(time_index, yhat2, col='red', lwd=2, lty=2)
legend("topleft", legend = c("Observed Data", "Modeled Data (yhat2)"),
       col = c("black", "red"), lty = c(1, 2), lwd = 2)

# Plot Residuals
residuals_yhat2 <- values_last_10 - yhat2
plot(time_index, residuals_yhat2, type = "p", col = "red", pch = 20,
     main = "Residuals of yhat1 Model",
     xlab = "Time Index", ylab = "Residuals")
abline(h = 0, col = "blue", lty = 2)


### MODEL 3 ###

# This will be an additive model with polynomial trend, seasonality, and white noise

class(poly_trend)
poly_trend <- as.numeric(poly_trend)

yhat3 <- poly_trend + fitted_seasonality + white_noise

plot(time_index, values_last_10, type = 'l', col = 'black', lwd = 2,
     main = "Polynomial (Degree 3) Trend + Seasonality + White Noise",
     xlab = 'Time Index', ylab = 'Nitrous Oxide Levels (ppm)')
lines(time_index, yhat3, col='green', lwd=2, lty=2)
legend("topleft", legend = c("Observed Data", "Modeled Data (yhat3)"),
       col = c("black", "green"), lty = c(1, 2), lwd = 2)

# Plot Residuals
residuals_yhat3 <- values_last_10 - yhat3
plot(time_index, residuals_yhat3, type = "p", col = "green", pch = 20,
     main = "Residuals of yhat1 Model",
     xlab = "Time Index", ylab = "Residuals")
abline(h = 0, col = "black", lty = 2)

# It looks like our second and third models are much better than our first - we can throw
# that model out. Let's repeat yhat2 and yhat3 with the residual AR(17) model.


# Let's first fit our AR(17)
```

```
fit.w1 <- arima(residuals, order=c(17,0,0), include.mean=FALSE)
fit.w1
fit.w1$aic # -407.1485 is very low - this is good!

# AR(17) seems to be good, but let's try AR(15) and AR(20) to compare:
test_fit20 <- arima(residuals, order=c(20,0,0), include.mean=FALSE)
test_fit20$aic # -405.9633 - AR(20) is higher than AR(17), worse
test_fit30 <- arima(residuals, order=c(15,0,0), include.mean=FALSE)
test_fit30$aic # -393.767 - AR(15) is higher than AR(17), worse

coef_w1 <- coef(fit.w1)
coef_w1

# We plot the ACF for the initial residuals and the residuals from the AR(17)

par(mfrow=c(1,1))
acf(residuals, main="ACF Plot of MLO Data")
res_w1 <- ts(fit.w1$residuals)
acf(res_w1, main="ACF Plot of AR(17) Model") # This one clearly better

# Now let's recreate our two models using AR(17)

### MODEL 2 with AR(17) ###

yhat2.2 <- exp_trend + fitted_seasonality + res_w1

plot(time_index, values_last_10, type = 'l', col = 'black', lwd = 2,
     main = "Exponential Trend + Seasonality + AR(17)",
     xlab = 'Time Index', ylab = 'Nitrous Oxide Levels (ppm)')
lines(time_index, yhat2.2, col='red', lwd=2, lty=2)
legend("topleft", legend = c("Observed Data", "Modeled Data (yhat2.2)"),
       col = c("black", "red"), lty = c(1, 2), lwd = 2)


### MODEL 3 with AR(17) ###

yhat3.2 <- poly_trend + fitted_seasonality + res_w1

plot(time_index, values_last_10, type = 'l', col = 'black', lwd = 2,
     main = "Polynomial (Degree 3) Trend + Seasonality + AR(17)",
     xlab = 'Time Index', ylab = 'Nitrous Oxide Levels (ppm)')
lines(time_index, yhat2.2, col='green', lwd=2, lty=2)
legend("topleft", legend = c("Observed Data", "Modeled Data (yhat2.3)"),
       col = c("black", "green"), lty = c(1, 2), lwd = 2)


#MSE

y <- as.numeric(values_last_10)

sum((y-yhat1)^2) / length(y)     # 0.09144829
sum((y-yhat2)^2) / length(y)     # 0.04082456
sum((y-yhat3)^2) / length(y)     # 0.0399147
sum((y-yhat2.2)^2) / length(y)   # 0.02597242
sum((y-yhat3.2)^2) / length(y)   # 0.02507394

# Compute MAPE for each model
mean(abs((y - yhat1) / y)) * 100     # 0.07777365
mean(abs((y - yhat2) / y)) * 100     # 0.05045046
mean(abs((y - yhat3) / y)) * 100     # 0.04934287
mean(abs((y - yhat2.2) / y)) * 100   # 0.04028163
mean(abs((y - yhat3.2) / y)) * 100   # 0.03879621

# yhat3.2 is the best model, save it as final_model
final_model <- yhat3.2

plot(time_index, values_last_10, type = 'l', col = 'grey', lwd = 2,
     main = "Final Model",
     xlab = 'Time Index', ylab = 'Nitrous Oxide Levels (ppm)')
lines(time_index, final_model, col='blue', lwd=2, lty=2)
legend("topleft", legend = c("Observed Data", "Modeled Data"),
       col = c("grey", "blue"), lty = c(1, 2), lwd = 2)

# Check residuals
acf(res_w1, main='Final Model Residuals', lag.max = 50)
checkresiduals(res_w1)
#No autocorrelation!




#### FORECASTING ####


# Train/Test Split to test model with actual data (2022-23)


# Split data: Training set (2014-2021) and test set (2022-2023)
training_data <- window(values_last_10, end = c(2021, 12))
test_data <- window(values_last_10, start = c(2022, 1))

# Plot the split
x_range <- range(time(training_data), time(test_data))
y_range <- range(training_data, test_data)


final_model <- ts(final_model, start = c(2014,1), frequency = 12)
final_model


final_model_last_years <- window(final_model, start = c(2022,1))

# Plot the forecast
plot(training_data, type = "l", col = "black", lwd = 2,
     main = "Forecast for 2022-2023", xlab = "Time Index", ylab = "Nitrous Oxide Levels",
     xlim = x_range, ylim = y_range)
```

```
lines(test_data, col = "red", lwd = 2)
lines(final_model_last_years, col = "blue", lwd = 2, lty = 2)
legend("topleft", legend = c("Training Data", "Actual Test Data", "Forecasted Test Data"),
       col = c("black", "red", "blue"), lty = c(1, 1, 2), lwd = 2)




# Calculate errors
forecast_errors <- test_data - final_model_last_years
mse <- mean(forecast_errors^2)  # Mean Squared Error
mape <- mean(abs(forecast_errors / test_data)) * 100  # Mean Absolute Percentage Error

# Print evaluation metrics
cat("MSE:", mse, "\n")
cat("MAPE:", mape, "%\n")

# MSE=0.0246: This is a very low value, indicating that the average squared
# differences between the forecasted and actual values are minimal. A smaller
# MSE generally reflects better model performance.



# MAPE=0.04%: This is an excellent result, indicating that, on average,
# the forecasted values deviate from the actual values by only 0.04% relative
# to their magnitude. A MAPE below 5% is typically considered very accurate for
# most forecasting applications.


### FORECASTING (2024-25) with predict ###


plot(final_model)

# Forecast future values, let’s say for 12 future periods
future_values <- forecast(final_model, h=24)

# Plot the forecast
plot(future_values, main=’Future Nitrous Oxide Levels’)

# Print the forecasted values
print(future_values$mean)
```

```
# Forecasting manually with model

# Generate future indices for 2024-2025
future_index_24 <- seq(from = length(values_last_10) + 1, to = length(values_last_10) + 24)

# Extend the polynomial trend for 2024-2025
future_trend_24 <- predict(poly_fit, newdata = data.frame(time_index = future_index_24))

# Extend the seasonality for 2024-2025
future_seasonality_24 <- predict(fit_ss, newdata = data.frame(time = future_index_24))

# Generate residuals for 2024-2025 using the AR(17) model
future_residuals_24 <- arima.sim(model = list(ar = coef(fit.w1)), n = 24, sd = sqrt(fit.w1$sigma2))

# Combine the components to generate the forecast
forecast_24 <- future_trend_24 + future_seasonality_24 + future_residuals_24

# Generate x-axis values matching the forecasted period (2024-2025)
future_time <- seq(from = 2024 + (1 - 1) / 12, to = 2025 + 11 / 12, by = 1 / 12)

# Check lengths for debugging
length(future_time)  # Should be 24
length(forecast_24)  # Should also be 24

# Determine x-axis range (observed + forecasted data)
x_range <- range(time(values_last_10), future_time)

# Determine y-axis range (observed + forecasted data)
y_range <- range(values_last_10, forecast_24)

# Plot the observed data
plot(values_last_10, type = "l", col = "black", lwd = 2,
     main = "Nitrous Oxide Levels Forecast (2024{2025)",
     xlab = "Year", ylab = "Nitrous Oxide Levels",
     xlim = x_range, ylim = y_range)

# Add the forecasted data
lines(future_time, forecast_24, col = "blue", lwd = 2, lty = 2)

# Add a legend
legend("topleft", legend = c("Observed Data", "Forecasted Data"),
       col = c("black", "blue"), lty = c(1, 2), lwd = 2)

forecast_24


# Holt-Winters

values_last_10

# Apply Holt-Winters exponential smoothing
hw_model <- HoltWinters(values_last_10)
```

```
# Display the model summary
print(hw_model)

# Plot the fitted model
plot(hw_model, main = "Holt-Winters Fitted Model for 2014-2024")

# Forecast the next 24 months (2 years)
hw_forecast <- forecast(hw_model, h = 24)

# Plot the forecast
plot(hw_forecast, main = "Holt-Winters Forecast for 2024-2026")

# Print forecasted values
print(hw_forecast$mean)

# Evaluate residuals of the Holt-Winters model
hw_residuals <- residuals(hw_model)
acf(hw_residuals, main = "ACF of Holt-Winters Residuals")

# Perform Ljung-Box test on residuals to check for autocorrelation
Box.test(hw_residuals, lag = 20, type = "Ljung-Box")



# Predictions from Holt-Winters for the training period
hw_predictions <- fitted(hw_model)[, "xhat"]

# Predictions from your custom model
custom_predictions <- final_model

# Compute MSE for both models
mse_custom <- mean((values_last_10 - custom_predictions)^2)
mse_hw <- mean((values_last_10 - hw_predictions)^2)

cat("Custom Model MSE:", mse_custom, "\n")
cat("Holt-Winters Model MSE:", mse_hw, "\n")

# Compute MAPE for both models
mape_custom <- mean(abs((values_last_10 - custom_predictions) / values_last_10)) * 100
mape_hw <- mean(abs((values_last_10 - hw_predictions) / values_last_10)) * 100

cat("Custom Model MAPE:", mape_custom, "%\n")
cat("Holt-Winters Model MAPE:", mape_hw, "%\n")




### ADDITIONAL TESTS ###

# Test to see which month results in highest N2O values

# Identify the month with the highest average N2O levels
library(dplyr)
```

```
# Convert the time series data to a data frame for easier manipulation
values_df <- data.frame(
  Year = floor(time(values_last_10)),
  Month = cycle(values_last_10),
  Value = as.numeric(values_last_10)
)

# Calculate the average value per month
monthly_avg <- values_df %>%
  group_by(Month) %>%
  summarise(Average = mean(Value, na.rm = TRUE)) %>%
  arrange(desc(Average))

print(monthly_avg)

# Highlight the month with the highest N2O levels
highest_month <- monthly_avg[1, ]
cat("Month with highest N2O levels:", highest_month$Month, "\n")


boxplot(Value ~ Month, data = values_df,
        main = "Monthly Distribution of N2O Levels",
        xlab = "Month", ylab = "Nitrous Oxide Levels (ppm)",
        col = "lightblue")


# Create monthly time index
months <- 1:12





par(mfrow = c(1,1))
plot(seasonal)
seasonal


# Extract raw NO values for each month
monthly_avg <- tapply(values, cycle(values), mean)

# Create a bar chart of raw NO averages
barplot(monthly_avg, names.arg = month.abb, col = ifelse(monthly_avg == max(monthly_avg),
"blue", "gray"),
        main = "Average Monthly N2O Levels (Raw Data)",
        ylab = "N2O Levels (ppm)", xlab = "Month",
        ylim = c(min(monthly_avg) - 0.5, max(monthly_avg) + 0.5))
abline(h = mean(monthly_avg), col = "red", lty = 2)  # Add average line

# Highlight the peak month
legend("topright", legend = c("Peak Month", "Other Months"),
       fill = c("blue", "gray"), bty = "n")
```