The STL-ARIMA approach for seasonal time series forecast: a preliminary study

Kong Hoong Lem*

Faculty of Science, Universiti Tunku Abdul Rahman, Jalan University, Bandar Barat 31900 Kampar, Perak, Malaysia

Abstract. STL, which stands for Seasonal and Trend decomposition using Loess, is a technique used to decompose a time series into its underlying components: trend, seasonal, and remainder. In this study, STL has been combined with the AutoRegressive Integrated Moving Average, ARIMA model in an effort to improve the forecast performance on seasonal time series. The proposed algorithm used STL decomposition to isolate the trend, seasonal and remainder components within the time series data. ARIMA or SARIMA models were then independently fitted to each component to capture their dynamics. Finally, the component-wise forecasts were aggregated to generate the final overall forecast. Forecast performance was compared with the SARIMA model using metrics such as MAE, RMSE and MAPE. Based on a preliminary case study by using atmospheric carbon dioxide concentration data from Mauna Loa, Hawaii, the findings suggest that the proposed algorithm offers a viable alternative for improving forecast performance in seasonal data.

1 Introduction

The AutoRegressive Integrated Moving Average (ARIMA) model is one of the most classical statistical approach for time series modelling and forecasting. It was proposed by Box and Jenkins in 1970s [1] and subsequently became an important milestone in the area of time series analysis. The ARIMA model is also called the Box Jenkins model, serving to commemorate their contribution in this important time series modelling framework. Over the past few decades, ARIMA models have exerted far-reaching influence on subsequent research in the field. It has been applied to thousands of forecasting problems in various fields.

Seasonal Autoregressive Integrated Moving Average (SARIMA) models are an extension of ARIMA models that specifically account for seasonality. It is one of the most widely-used classical methods for seasonal time series such that it becomes a traditional benchmark for other methods. When a new method fails to outperform SARIMA in forecast accuracy, it becomes less interesting or futile.

SARIMA has been compared with other data-driven approaches. For example, in forecasting the incoming dustcarts of a waste transfer station, Zhou, J. et al. [2] found that

*Corresponding author: lemkh@utar.edu.my

SARIMA model has better prediction results compared with the Long Short-Term Memory (LSTM) model. Liu. X, et. al.'s work [3] also arrived at a similar conclusion. They employed SARIMA model for short-term offshore wind speed forecasting and then compared its performance with deep-learning-based algorithms, namely Gated Recurrent Unit (GRU) and LSTM. They reported that SARIMA outperformed GRU and LSTM.

On the other hand, building on insights from frequency decomposition in spectral analysis, a helpful heuristic in time series forecasting was to see the data as a mix of components with different behaviours. The idea was already mentioned way back in Box and Jenkins' book [1] (section 9.1.1).

The X11 method, evolved from the US Census Bureau and Statistics Canada in 1960's [4], was a widespread method for decomposing quarterly and monthly data. It used iterative process to decompose time series into three components: trend cycle, seasonal, and irregularity. Later, the improved variants X12 and X13 were devised based on X11. Nevertheless, these methods were specifically designed to handle time series data with monthly or quarterly observations. In 1990, Cleveland et. al. developed the STL method [5]. STL is an acronym for Seasonal and Trend decomposition using Loess. According to Hyndman [6], STL exhibits versatility in handling any type of seasonality, overcoming the monthly and quarterly limitations of the fixed-period decomposition methods. STL users can fine-tune seasonal behaviour and trend smoothness. Furthermore, STL shows robust tolerance to outliers.

The logic of decomposition method is to view data as a layered stack of signals with unique dynamics (e.g. high and low frequencies, periodic cycles and temporal trends), extract them and subject them to separated analysis. By incorporating decomposition in time series forecasting, the data is subdivided into smaller, more manageable sub-components based on their characteristics (e.g. trend, seasonality). Hence, the data inherent complexity is effectively reduced. Focused in-depth analysis and forecasting of each component becomes possible. And this lead to improved forecast compared to single-model direct forecasting of the entire series without considering the underlying patterns.

In this preliminary study, we focus on STL decomposition. The aim is to improve the forecast of seasonal time series by hybridizing STL and ARIMA. The idea was implemented onto the atmospheric carbon dioxide concentration data. In the evaluation, we compare the STL-ARIMA with SARIMA. Using forecast horizon $h = 6$, and adopting the rolling window validation approach, we observed the average of the forecasting error of STL-ARIMA was lower than the SARIMA. This preliminary work may provide a groundwork for developing an alternative to seasonal data analysis.

2 Methodology

2.1 ARIMA models

ARIMA models are a widely used class of statistical models for time series forecasting, based on the assumption that the current value of a time series is a function of its past values and past forecast errors. Hence, ARIMA modelling is a regression process that uses the past values of the time series and past forecast errors to predict the current value.

In a condensed presentation, a seasonal ARIMA$(p, d, q)(P, D, Q)_s$ model is given by

$$\phi(B)\Phi(B^s)(1-B)^d(1-B^s)^D y_t = c + \theta(B)\Theta(B^s)\varepsilon_t \quad (1)$$

where

y_t is the data at period t;

$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p$ is a p-th degree polynomial in B;
$\theta(B) = 1 + \theta_1 B + \theta_2 B^2 + \cdots + \theta_q B^q$ is a q-th degree polynomial in B;
$\Phi(B) = 1 - \Phi_1 B^s - \Phi_2 B^{2s} - \cdots - \Phi_P B^{Ps}$ is a (sP)-th order polynomial in B;
$\Theta(B) = 1 + \Theta_1 B^s + \Theta_2 B^{2s} + \cdots + \Theta_Q B^{Qs}$ is a (sQ)-th order polynomial in B;
B is the backshift operator such that $B^i y_t = y_{t-i}$;
ε_t is a normally distributed sequence with mean zero and variance σ^2 (white noise).

Here, p is the autoregressive (AR) order; d is the number of differencing necessary to attain stationarity; q is the moving average (MA) order; P, D and Q are the corresponding seasonal AR, differencing, and MA orders, respectively; s is the number of observations per season (seasonal period), and ϕ_i, θ_i, Φ_i, Θ_i are the parameters of the model. Note that both $\Phi(B)$ and $\Theta(B)$ are polynomial in B in which the powers of B have gaps of size s.

Non-seasonal ARIMA model can be regarded as a special case of equation (1) where all the seasonal orders P, D, Q are zero such that the $\phi(B), \Phi(B), (1 - B^s)^D$ terms reduce to unity. So, an ARIMA(p, d, q) process is described by

$$\phi(B)(1 - B)^d y_t = c + \theta(B)\varepsilon_t. \tag{2}$$

Hyndman and Athanasopoulos [6] have outlined a seven-step general procedure for fitting an ARIMA model to a set of time series data, namely:

(i). Visualize the data through plotting and recognize any abnormal observations.
(ii). Apply data transformations (such as logarithmic or power transformations) to achieve variance stabilization if required.
(iii). If the data exhibits non-stationarity, iteratively take first differences until stationarity is achieved.
(iv). Analyse the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) to determine the appropriate (p, d, q) order in the ARIMA(p, d, q) model.
(v). Select a model and use the corrected Akaike Information Criterion (AICc) to improve the model.
(vi). Assess the residuals of the chosen model by plotting the ACF of the residuals and conducting a portmanteau test. If the residuals deviate from white noise, consider exploring a modified model, i.e. go back to step (iii).
(vii). Once the residuals show characteristics of white noise, proceed to calculate forecasts.

The process of selecting the appropriate order (i.e., the loop comprising steps (iii) to (vi)) is commonly subjective and laborious. In addressing this obstacle, Hyndman and Khandakar [7] have developed an automated algorithm that have been implemented in the forecast package [8] for R. Generally, this algorithm achieves an optimal ARIMA model by conducting a stepwise search to traverse a range of (p, d, q) and their combination using unit root tests, AICc minimization, and maximum likelihood estimation. A detailed description of the algorithm is available in the book by Hyndman and Athanasopoulos [6]. In this study, all the ARIMA and SARIMA models fitting and forecasting were implemented using this Hyndman-Khandakar automated algorithm.

2.2 Seasonal and Trend decomposition using Loess (STL)

STL is a decomposition method based on locally weighted scatterplot smoothing (loess) proposed by Cleveland et al. [5]. It is widely used for decomposing a time series into additive components, such that

$$Y_t = T_t + S_t + R_t \tag{3}$$

where T_t is the trend, S_t is the seasonality, and R_t is the remainder.

The STL algorithm performs decomposition in a nested loop. A sequence of detrending and deseasonalization take place within the inner loop in which the Loess smoother is used repeatedly to refine and improve the seasonal and trend. Meanwhile the outer loop minimizes the influence of outliers on both trend and seasonal components. Seasonal and trend components are then removed to obtain the remainder.

Besides being a robust method that is not sensitive to outliers, STL is non-parametric which make no assumption about the underlying distribution of the data. The STL decomposition was implemented using the 'stl' function available in the inbuilt 'stats' package of R [9].

2.3 The STL-ARIMA approach

To facilitate model training and evaluation, the data was initially divided into a training set and a testing set. STL decomposition was then applied only to the training set, extracting the trend, seasonal, and remainder components. Notably, the original training set $Y_T = \{y_1, y, \dots, y_T\}$ could be perfectly reconstructed by summing up these decomposed components:

$$Y_T = \{y_1, y_2, \dots, y_T\} = \{(t_1 + s_1 + r_1), (t_2 + s_2 + r_2), \dots (t_T + s_T + r_T)\}. \quad (4)$$

Following their separation, individual time series models were constructed for each component. ARIMA or SARIMA models were then independently fit to each component. The model fitting and forecasting procedures were implemented in R-language using the Hyndman-Khandakar automated algorithm [7] under the 'forecast' package. Note that each component was fitted by a separate ARIMA or SARIMA model of appropriate order due to the different characteristics of each component, for example, trend versus seasonality.

Assuming the decomposition process accurately captures the underlying components of the time series, the sum of the individual component forecasts will recover the overall forecast:

$$\hat{Y}_{T+h} = \hat{T}_{T+h} + \hat{S}_{T+h} + \hat{R}_{T+h}. \quad (5)$$

Here, $h \geq 1$ is an integer representing the forecast horizon and T is the forecast origin.

In essence, the proposed approach does not directly address the original data; rather, it applies seasonal-trend decomposition using STL. Subsequently, the approach fits ARIMA models to each of the extracted components. Therefore, the overall approach remains reliant on ARIMA.

The procedure can be summarized in Figure 1.

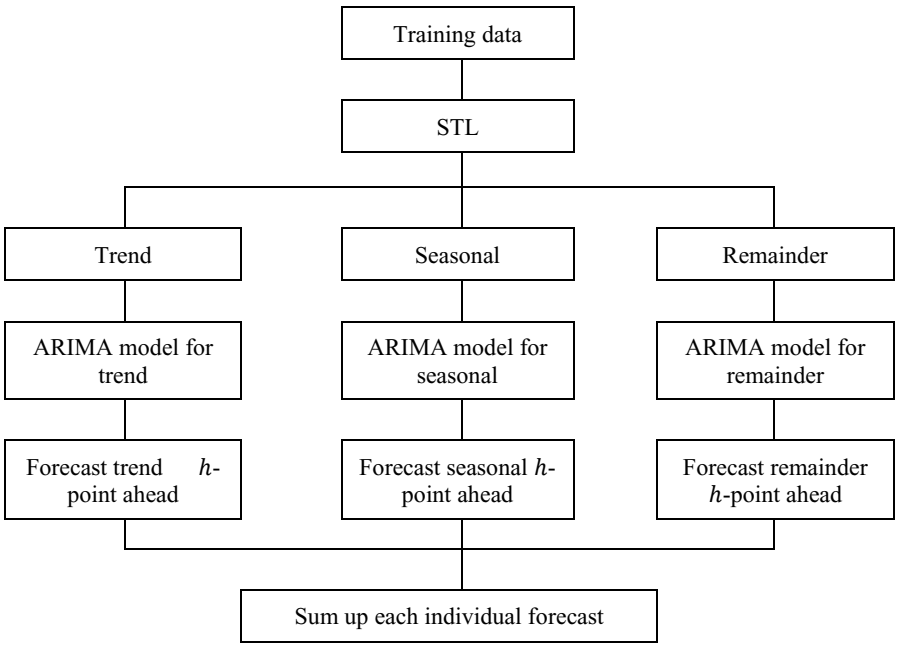


Fig. 1. The STL-ARIMA algorithm: STL decomposition, ARIMA models fitting to the STL components, individual component forecasting, forecasts aggregating to obtain an overall prediction.

STL decomposition allowed the original time series to be broken down into three simpler sub-series so that the underlying patterns and trends could be understood better. This allowed the fitting of separate ARIMA models for each component, which could capture the unique characteristics and dynamics more accurately. By combining these component-level forecasts, we anticipated that this would improve the overall forecasting accuracy.

2.4 Forecast performance of the models

The time series data was partitioned into non-overlapping training and testing subsets. The training set was used to build and fit models, while the testing set was used to evaluate the out-of-sample predictive performance of the fitted models. Based on the difference between the observed value from the testing set and its forecast, a few standard forecast error metrics were calculated, including mean absolute error (MAE), root mean square error (RMSE), mean absolute percentage error (MAPE) and the coefficient of determination, R^2. They can be expressed as

$$\mathrm{MAE} = \frac{1}{n}\sum_{h=1}^{n}|e_{T+h}|, \tag{6}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{h=1}^{n}(e_{T+h})^2}\ , \tag{7}$$

$$\mathrm{MAPE} = \frac{1}{n}\sum_{h=1}^{n}\left|\frac{e_{T+h}}{y_{T+h}}100\right|, \tag{8}$$

$$R^2 = \frac{\sum_{h=1}^{n}(\hat{y}_{T+h} - \bar{y})^2}{\sum_{h=1}^{n}(y_{T+h} - \bar{y})^2}. \quad (9)$$

Here,

$$e_{T+h} = y_{T+h} - \hat{y}_{T+h}, \quad (10)$$

is the forecast error, where $\{y_1, y_2, \dots, y_T\}$ represents the training set, $\{y_{T+1}, y_{T+2}, \dots\}$ represents the testing set, $\hat{y}_{T+h}$ represents the forecasted value and $\bar{y}$ is the mean of the testing set.

2.5 Comparing forecast performance - multiple-fold rolling forecast origin

Consider a time series denoted as $\{y_1, y_2, \dots, y_T\}$, where y_i represents the value at time index i, and y_T is the most recent known observation. Our objective is to forecast the value y_{T+h}, h steps ahead into the future ($h \geq 1$). Here, h represents the forecast horizon, and T serves as the forecast origin [10]. This study adopts a rolling forecast origin framework with a fixed forecast horizon of six steps.

In essence, this framework simply iteratively employs a series of different training and testing set combinations. This procedure started with a training set of specific length while the subsequent six data points were designated as the testing set. Models were fitted on the training set and their performance were evaluated on the testing set (i.e. the subsequent six data points). The execution of this process was termed a single 'run'. Subsequently, the procedure progressed to another run, where the training set size was diminished by two data points with each iteration. This process was repeated ten times, resulting ten distinct runs, each yielding a separate model and forecast evaluation. The idea was illustrated in the following schematic figure:

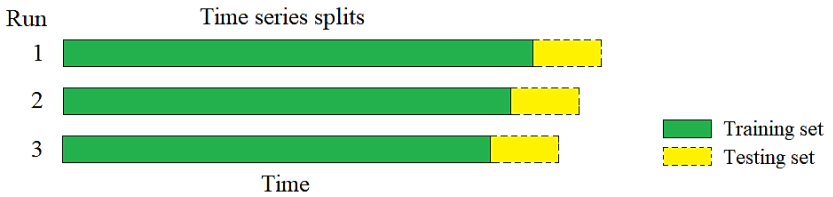


Fig. 2. A schematic diagram to illustrate the rolling forecast origin framework. The first three runs are shown here.

Each run of the rolling forecast framework allowed for an accuracy assessment using metrics like Root Mean Squared Error (RMSE). To evaluate the comparative performance of model-A and model-B, the average accuracy metric (e.g., average RMSE) across the ten runs was calculated. If model-A exhibits a consistently lower average accuracy metric compared to model-B, it provides stronger evidence to suggest its superior performance on the given data.

While relying on a single dataset in this preliminary study, this forecast origin rolling procedure reduced the potential for bias that arises from arbitrarily selecting a single training and testing set.

3 A dataset implementation: result and discussion

3.1 Data source and description

The data used in the study was the monthly average of the atmospheric carbon dioxide concentrations expressed in parts per million (ppm) taken at Mauna Loa, Observatory, Hawaii [11]. The time series comprised 468 observations collected on a monthly basis from January 1959 through December 1997.

A time plot of the data was presented in Figure 3.

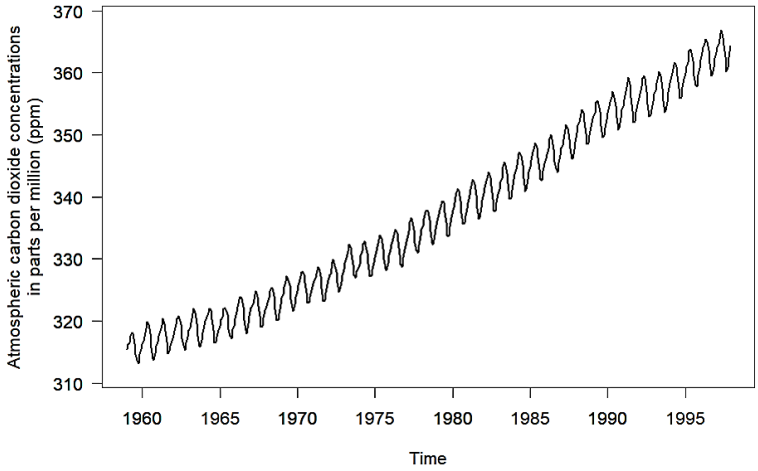


Fig. 3. Atmospheric carbon dioxide concentrations taken at Mauna Loa, Hawaii.

A clear seasonality pattern and a quite linear increasing trend was observed. The amplitude of the cycles appeared to remain relatively stable throughout the observed period.

3.2 STL decompose

The STL decomposition dissects the time series into three components. The plot of each component is shown in Figure 4. The extracted components exhibit simpler patterns relative to the original time series. This reduction in complexity facilitate the application of simpler ARIMA modelling to each component and hence improve the model fitting.

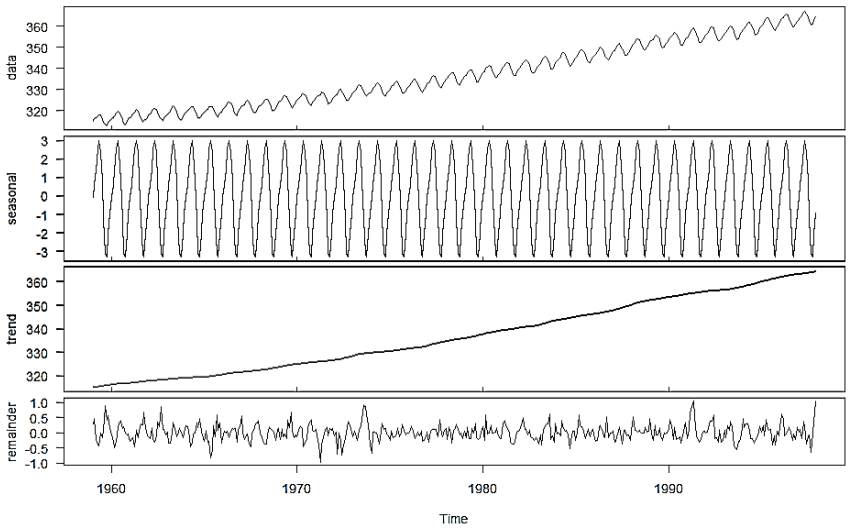


Fig. 4. The components of the time series under the STL decomposition.

Examination of the autocorrelation function (ACF) plot of the remainder, as shown in Figure 5, reveals the presence of autocorrelation. This suggests that the remainder is not likely white noise. To confirm this observation, a Ljung-Box test was conducted on the remainder series. The test assessed the null hypothesis of no autocorrelation at various lags. The resulting significant p-value (a small p-value) led to the rejection of the null hypothesis, further supporting the presence of autocorrelation. Thus, the remainder series was not discarded along the mode fit and forecast process.

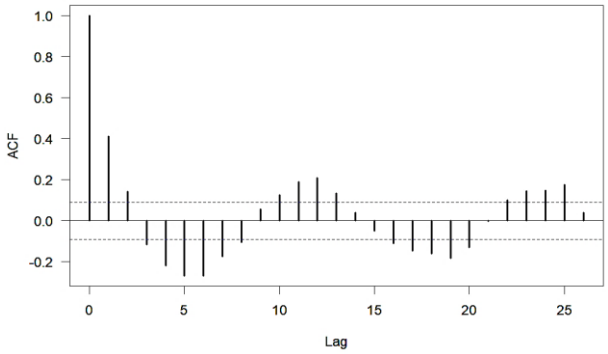


Fig. 5. The autocorrelation function (ACF) plot of the remainder after the STL decomposition.

Real-world time series often contain complex patterns and irregularities that diverge from idealized decomposition frameworks. Perfect separation into precise trend, seasonality and purely stochastic remainder components can be challenging and is commonly unattainable. In many instances, the remainder component may not be purely random white noise but contain some hidden patterns or valuable information.

3.3 Rolling window validation of performance

The model fitting and forecasting using SARIMA and STL-ARIMA models were performed in 10 separate runs, with each run carried out on a different combination of training and

testing set. A rolling forecast origin approach was conducted such that the length of the training set varied while the forecast horizon remained fixed.

The entire dataset consisted of 468 observations. For the rolling forecast framework, a fixed testing set size of six observations was adopted. This testing set consistently comprised the most recent six data points. In each of the ten runs, the training set size was dynamically adjusted. The first run utilized all data points from the 1st up to the 468th observation. Subsequently, in each successive run (Runs 2 to 10), the two most recent data points were excluded. For instance, the second run only used the 1st to the 466th data points, and this iterative reduction in training set size continued throughout the remaining eight runs. This process resulted in training sets of decreasing size while maintaining the constant testing set size. Table 1 presents a detailed breakdown of the training and testing set composition for the first three runs of the rolling forecasting framework.

Table 1. The detailed training/testing set splitting strategy for the first three runs of the rolling forecasting origin approach.

Run number	Training set	Testing set
1	1st to 462th data	463th to 468th data
2	1st to 460th data	461th to 466th data
3	1st to 458th data	459th to 464th data

3.4 Forecast accuracy comparison

The traditional SARIMA model was fitted to the training set using the Hyndman and Khandakar automated algorithm. In contrast, for the proposed STL-ARIMA approach, the same training data underwent STL decomposition and was separated into trend, seasonal, and remainder components. Each component was then individually modeled and forecasted. Finally, the three separated forecasts were combined to produce the overall forecast for the entire testing set. The specific model orders used for each run are detailed in the following table.

Table 2. The orders of the models for every run.

Run	SARIMA model	STL-ARIMA model		
	original data	Trend component	Seasonal component	Remainder component
1	$(1,1,1)(1,1,2)_{12}$	(2,2,2)	$(0,0,0)(0,1,0)_{12}$	$(2,0,4)(1,0,0)_{12}$
2	$(3,1,2)(1,1,2)_{12}$	(2,2,2)	$(0,0,0)(0,1,0)_{12}$	$(2,0,4)(1,0,0)_{12}$
3	$(1,1,1)(1,1,2)_{12}$	(2,2,2)	$(0,0,0)(0,1,0)_{12}$	$(2,0,4)(1,0,0)_{12}$
4	$(1,1,1)(1,1,2)_{12}$	$(0,2,0)(0,0,1)_{12}$	$(0,0,0)(0,1,0)_{12}$	$(2,0,3)(1,0,0)_{12}$
5	$(1,1,1)(1,1,2)_{12}$	(2,2,2)	$(0,0,0)(0,1,0)_{12}$	$(2,0,3)(1,0,0)_{12}$
6	$(1,1,1)(1,1,2)_{12}$	(2,2,2)	$(0,0,0)(0,1,0)_{12}$	$(2,0,3)(1,0,0)_{12}$
7	$(1,1,1)(1,1,2)_{12}$	(2,2,2)	$(0,0,0)(0,1,0)_{12}$	$(2,0,4)(1,0,0)_{12}$
8	$(0,1,1)(1,1,2)_{12}$	$(0,2,0)(0,0,1)_{12}$	$(0,0,0)(0,1,0)_{12}$	$(2,0,4)(1,0,0)_{12}$
9	$(0,1,1)(1,1,2)_{12}$	(2,2,2)	$(0,0,0)(0,1,0)_{12}$	$(2,0,3)(1,0,0)_{12}$
10	$(2,1,2)(1,1,2)_{12}$	(2,2,2)	$(0,0,0)(0,1,0)_{12}$	$(2,0,2)(1,0,0)_{12}$

The time plot of the forecasts by both the SARIMA and STL-ARIMA models as well as the relevant testing set were plotted on a same figure for visual accuracy comparison. The accuracy comparison plot of the first run was shown in Figure 6. The plots for the rest of the nine runs, where the training sets were omitted, were displayed collectively in Figure 7. Throughout all plots in Figure 6 and Figure 7, solid circles represent the (truncated) training data, while hollow circles represent the testing data. The forecasted values by SARIMA

model and by STL-ARIMA model are depicted by the dot-dashed curve and long-dashed curves, respectively.

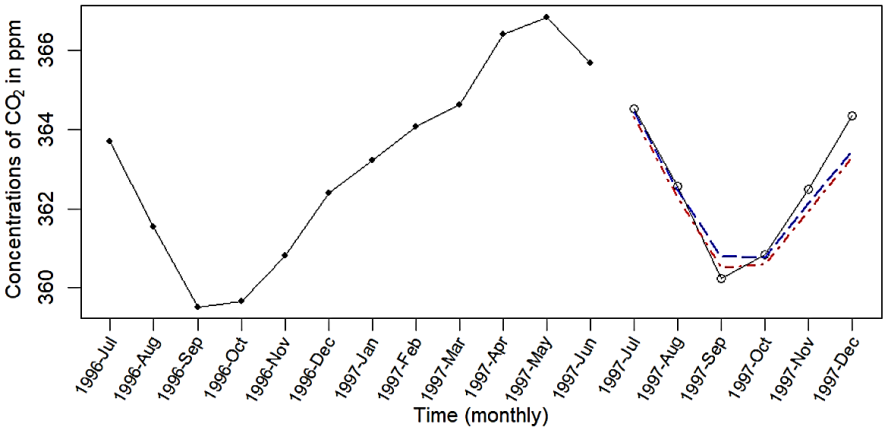
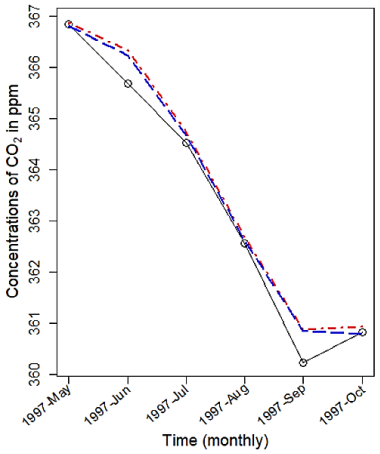
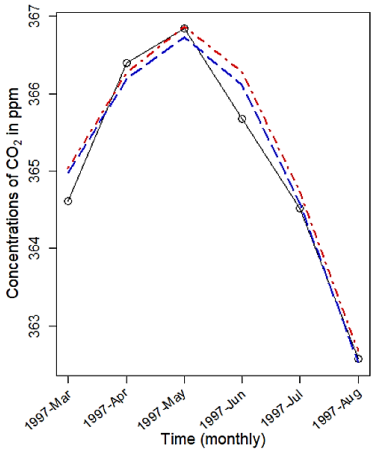


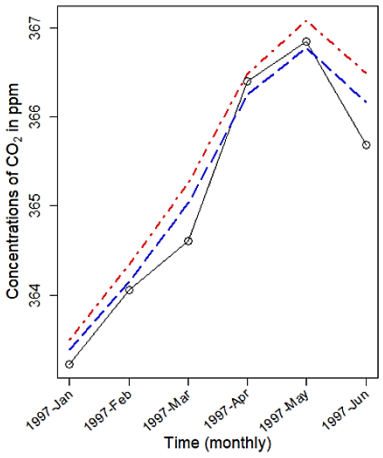
Fig. 6. The forecast accuracy comparison between the SARIMA model and the STL-ARIMA model for the run 1. Solid circles represent the (truncated) training data; hollow circles represent the testing data; dot-dashed curve represent the forecasted values by SARIMA model; long-dashed curve represent the forecasted values by STL-ARIMA model.



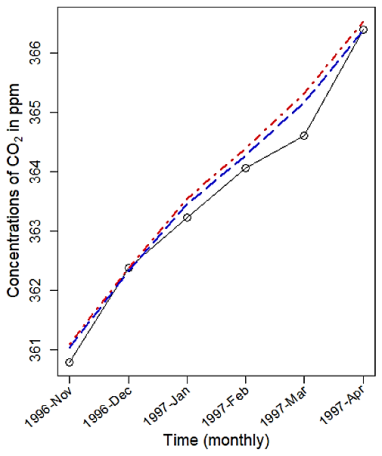
Run 2



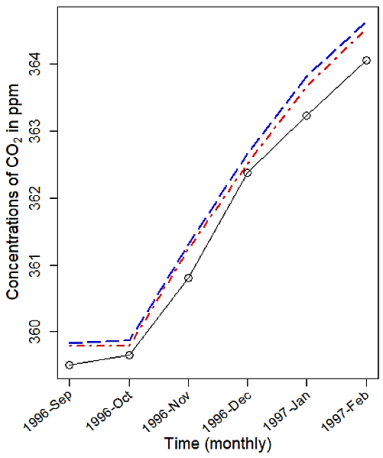
Run 3



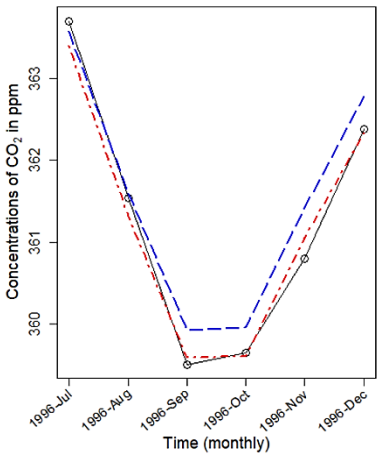
Run 4



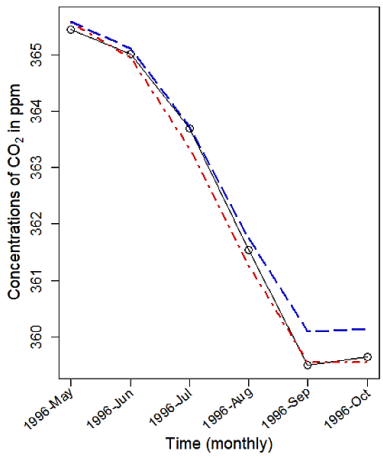
Run 5



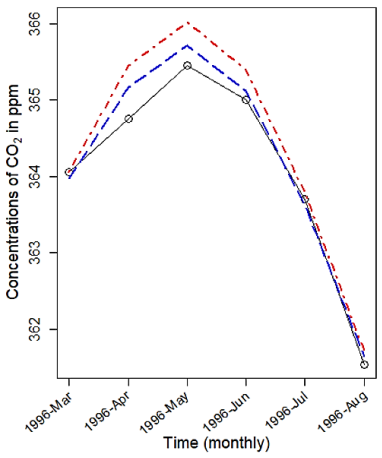
Run 6



Run 7



Run 8



Run 9

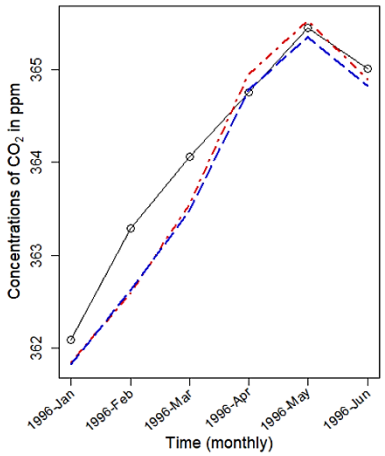


Fig. 7. The forecast accuracy comparison between the SARIMA model and the STL-ARIMA model for run 2 through 10 are presented sequentially, with run 2 displayed in the upper left panel and subsequent runs arranged in a row from right to left, then continuing in a column from top to bottom. Hollow circles represent the testing data; dot-dashed curve represent the forecasted values by SARIMA model; long-dashed curve represent the forecasted values by STL-ARIMA model.

The forecast accuracy of each run for both the SARIMA and STL-ARIMA models, in terms of MAE, RMSE, MAPE, R^2, was listed in Table 3. STL-ARIMA models demonstrate a superiority over SARIMA by showing lower values of MAE and MAPE but higher value of R^2 in 70% of the cases. This dominance is less striking when considering RMSE, where STL-ARIMA leads in only 60% on the cases.

Table 3. The forecast accuracy of each run for both the SARIMA and STL-ARIMA models. Bold items indicate a superior performance of STL-ARIMA over SARIMA.

Run	MAE		RMSE	
	STL-ARIMA	SARIMA	STL-ARIMA	SARIMA
1	**0.3424401**	0.4330237	**0.4596398**	0.5243475
2	**0.2364078**	0.2993919	**0.3388073**	0.3918969
3	**0.2038589**	0.2510446	**0.251762**	0.3192535
4	**0.2308134**	0.3879208	**0.2808016**	0.4632452
5	**0.2172974**	0.3014411	**0.2793705**	0.3712752
6	0.4149659	0.3136857	0.4401041	0.343758
7	0.3191683	0.161379	0.3715303	0.1914901
8	0.2623998	0.1586092	0.3359287	0.1965349
9	**0.1730149**	0.3233952	**0.2144588**	0.4107637
10	**0.3023298**	0.3060106	0.3844788	0.3781558

Run	MAPE		R^2	
	STL-ARIMA	SARIMA	STL-ARIMA	SARIMA
1	**0.09440543**	0.1192916	0.948596	0.952981
2	**0.06511778**	0.082459	0.9883607	0.9892563
3	**0.05578802**	0.0687647	**0.9720618**	0.9706367
4	**0.06321946**	0.1062665	**0.9681045**	0.961332
5	**0.05977452**	0.0828703	**0.987191**	0.9857973
6	0.11462376	0.0866552	**0.9967394**	0.9955475
7	0.08844588	0.0446067	0.9793153	0.989135
8	0.07273402	0.0437169	**0.9980849**	0.9952738
9	**0.047464**	0.0886826	**0.9856543**	0.9792952
10	**0.08313169**	0.0841398	**0.9711646**	0.9643983

Table 4 presents the average of all the forecast accuracy measures calculated across all ten runs for both the SARIMA and STL-ARIMA models, allowing for direct comparison. The results indicate that, on average, STL-ARIMA consistently outperforms SARIMA in forecasting performance, as evidenced by lower MAE, RMSE, MAPE and higher R^2 values.

Table 4. SARIMA and STL-ARIMA models: the average forecast accuracy metric across 10 runs for each measure.

	MAE	RMSE	MAPE	R^2
SARIMA	0.2935902	0.3590721	0.08074532	0.9783653
STL-ARIMA	0.2702696	0.3356882	0.07447046	0.9795272

4 Conclusion

Challenges associated with modeling and forecasting the entire time series at once often stem from its inherent complexity. In this study, the proposed STL-ARIMA algorithm achieved superior forecast accuracy by decomposing the data into its trend, seasonal, and remainder components. This decomposition into less complicated elements likely facilitated superior model fitting and forecasting. These findings suggest that the STL-ARIMA approach holds promise as a viable method for enhancing forecast accuracy, particularly for seasonal data.

While the proposed algorithm's outperformance of the SARIMA model on this specific dataset is encouraging, generalizability to other data requires further investigation. Assessing its performance across a broader range of datasets and benchmarking it against alternative methods is crucial for understanding its strengths and limitations. Additionally, exploring alternative decomposition techniques, such as those based on Fourier or wavelet transforms, in conjunction with the STL decomposition, merits further research.

The author gratefully acknowledges the financial support from UTAR for attending this conference. The author expresses sincere gratitude to the anonymous reviewers for their insightful comments, which significantly improved the quality of the paper from its initial version. Special thanks must go to Dr. Wong for providing invaluable assistance with the English editing of this manuscript.

References

1. G.E.P. Box, G.M. Jenkins, Time series analysis forecasting and control (Prentice-Hall, Englewood Cliffs, 1976)
2. J. Zhou, Q. Fang, H. Zhu, H. Song, Y. Wang, *Forecasting of the incoming dustcarts of a waste transfer station based on Sarima Model*, in the 35th Chinese Control and Decision Conference, CCDC, 20-22 May 2023, Yichang, China, 4953 – 4958 (2023)
3. X. Liu, Z. Lin, Z. Feng, Energy, **227**, 120492 (2021)
4. E.B. Dagum, S. Bianconcini, Seasonal adjustment methods and real time trend-cycle estimation, section 1.2.2 (Springer, Switzerland, 2016)
5. R.B. Cleveland, W.S. Cleveland, J.E. McRae, I. Terpenning, J. Off. Stat., **6**(1), 3–73, (1990)
6. R.J. Hyndman, G. Athanasopoulos, Forecasting: principles and practice, 2nd edition (OTexts, Melbourne, 2018)
7. R.J. Hyndman, Y. Khandakar, J. Stat. Softw. **27**(3), 1 – 22 (2008)
8. R.J. Hyndman et. al., *forecast: Forecasting functions for time series and linear models*, R package version 8.21.1 (2023), https://pkg.robjhyndman.com/forecast/ Accessed on 30 Nov 2023
9. https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/stl, Accessed on 30 Nov 2023
10. R.S. Tsay, Analysis of Financial Time Series, 3rd Edition, section 2.4.4 (Wiley, New Jersey, 2010)
11. Atmospheric CO2 Data | Scripps CO2 Program, https://scrippsco2.ucsd.edu/data/atmospheric_co2/ Accessed on 30 Nov 2023