International Journal of Operational Research

ISSN online: 1745-7653 - ISSN print: 1745-7645
https://www.inderscience.com/ijor

MSTL: a seasonal-trend decomposition algorithm for time series with multiple seasonal patterns

Kasun Bandara, Rob J. Hyndman, Christoph Bergmeir

DOI: 10.1504/IJOR.2022.10048281

Article History:

Received:	17 February 2022
Accepted:	08 May 2022
Published online:	16 January 2025

MSTL: a seasonal-trend decomposition algorithm for time series with multiple seasonal patterns

Kasun Bandara*

School of Computing and Information Systems,
Melbourne Centre for Data Science,
University of Melbourne, VIC, Australia
Email: Kasun.Bandara@unimelb.edu.au
*Corresponding author

Rob J. Hyndman

Department of Econometrics and Business Statistics,
Monash University,
VIC, Australia
Email: Rob.Hyndman@monash.edu

Christoph Bergmeir

Department of Data Science and AI,
Monash University,
VIC, Australia
Email: Christoph.Bergmeir@monash.edu

Abstract: The decomposition of time series into components is an important task that helps to understand time series and can enable better forecasting. Nowadays, with high sampling rates leading to high-frequency data (such as daily, hourly, or minutely data), many datasets contain time series data that can exhibit multiple seasonal patterns. Although several methods have been proposed to decompose time series better under these circumstances, they are often computationally inefficient or inaccurate. We propose a procedure to decompose time series with multiple seasonal patterns that is suited to a wide range of high-frequency data. The procedure for multiple seasonal trend decomposition (MSTL) introduced in this paper extends the traditional seasonal-trend decomposition using Loess (STL) algorithm, allowing the decomposition of time series with multiple seasonal patterns. In our evaluation on synthetic and a perturbed real-world time series dataset, compared to other decomposition benchmarks, MSTL demonstrates competitive results with lower computational cost. The implementation of MSTL is available in the R package *forecast*.

Keywords: time series decomposition; multiple seasonality; MSTL; TBATS; STR.

Reference to this paper should be made as follows: Bandara, K., Hyndman, R.J. and Bergmeir, C. (2025) ‘MSTL: a seasonal-trend decomposition algorithm for time series with multiple seasonal patterns’, *Int. J. Operational Research*, Vol. 52, No. 1, pp.79–98.

Biographical notes: Kasun Bandara received his BSc honours degree in Computer Science from the University of Colombo School of Computing, Sri-Lanka, in 2015 and PhD in Computer Science from the Monash University, Australia in 2020. He is currently working as a Postdoctoral Research Fellow at the Melbourne Centre for Data Science at The University of Melbourne, Australia. His research interests include big data, deep neural networks and time series forecasting.

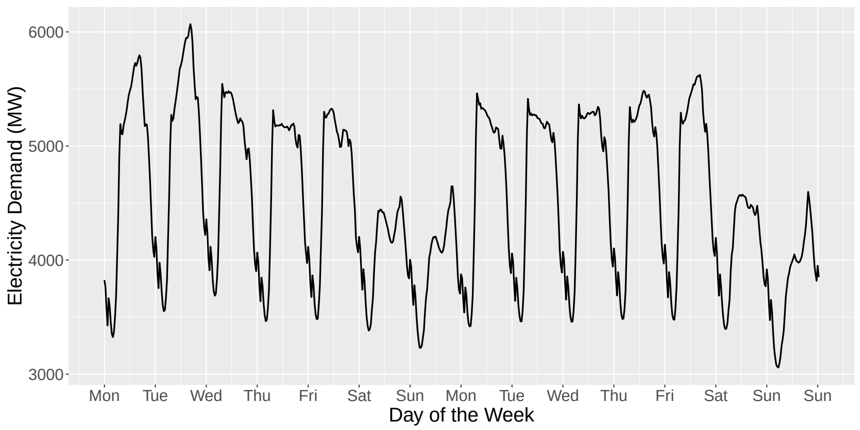
Rob J. Hyndman is Professor of Statistics and the Head of the Department of Econometrics and Business Statistics at Monash University. From 2005 to 2018, he was the Editor-in-Chief of the *International Journal of Forecasting* and the Director of the International Institute of Forecasters. He is an author of over 200 research papers and five books in statistical science. He is an Elected Fellow of both the Australian Academy of Science and the Academy of Social Sciences in Australia.

Christoph Bergmeir is a Senior Lecturer and a 2019 ARC DECRA Fellow in the Department of Data Science and Artificial Intelligence at Monash University. His fellowship is on the development of efficient and effective analytics for real-world time series forecasting. He works as a Data Scientist in a variety of projects with external partners in diverse sectors, e.g., in healthcare or infrastructure maintenance. He holds a PhD in Computer Science from the University of Granada, Spain, and an MSc in Computer Science from the University of Ulm, Germany.

1 Introduction

In time series analysis and forecasting, it is often useful to identify the underlying patterns of time series data, to better understand the contributing phenomena and to enable better forecasting. Time series decomposition techniques have been introduced to decompose a time series into multiple components including trend, seasonality, and remainder. Such methods have important time series applications in seasonal adjustment procedures (Maravall, 2006; Thornton, 2013; Petropoulos et al., 2018), forecasting (Koopman and Ooms, 2006; Zhang and Qi, 2005; Angelopoulos et al., 2019; Hewamalage et al., 2021; Bandara et al., 2021a) and anomaly detection (Wen et al., 2019, 2020). Nowadays, with rapid growth of availability of sensors and data storage capabilities, there is a significant increase of time series with higher sampling rates (sub-hourly, hourly, daily). Compared with traditional time series data, the higher-frequency time series data may exhibit more complex properties, such as multiple seasonal cycles, non-integer seasonality, etc. These time series are commonly found in the utility demand industry (electricity and water usage), mainly due to the intricate usage patterns of humans. For example, Figure 1 illustrates the aggregated half-hourly energy demand time series in the state of Victoria, Australia that has two seasonal periods, a daily seasonality (period $= 48$) and a weekly seasonality (period $= 336$). It can be seen that the average energy consumption in the weekdays is relatively higher to those in the weekends. Furthermore, a longer version of this time series may even show a yearly seasonality (period $= 17{,}532$), with values systematically changing across the seasons such as summer and winter. In this scenario, daily and weekly consumption patterns are useful to estimate the short-term energy requirements; the yearly seasonal usage patterns are beneficial in long-term energy planning. Therefore, the accurate decomposition of time series with multiple seasonal cycles is useful to provide better grounds for decision-making in various circumstances.

Figure 1 Half-hourly energy demand in the state of Victoria, Australia over a two weeks period of time, extracted from the *vic_elec* dataset (O'Hara-Wild et al., 2021b) displaying daily and weekly seasonal patterns



There is a large body of literature available on time series decomposition techniques. Widely used traditional time series decomposition methods are seasonal-trend decomposition using Loess (STL) (Cleveland et al., 1990), X-13-ARIMA-SEATS (Bell and Hillmer, 1984), X-12-ARIMA (Findley et al., 1998), regularised singular value decomposition (RSVD) (Lin et al., 2020), regularised singular value decomposition (RSVD) (Lin et al., 2020), polynomial roots of ARMA models (McElroy, 2021). Although these methods have been heavily used in many applications due to their robustness and efficiency, these techniques can handle only time series with a single seasonality. More recently, methods to decompose time series with multiple seasonal patterns have been introduced (Dokumentov and Hyndman, 2021; Wen et al., 2020). For example, Dokumentov and Hyndman (2021) introduced seasonal-trend decomposition by regression (STR), a regression-based, additive decomposition technique, which is also capable of modelling the influence of external factors towards the seasonal patterns in a time series. The daily calendar and seasonal adjustment method (DSA) (Daniel, 2021) extends the STL algorithm to retrieve weekly, monthly, yearly, and holiday effects of daily data. However, the DSA algorithm is only suitable for decomposing daily time series. Also, Wen et al. (2020) recently developed Fast-RobustSTL, a decomposition technique that accounts for multiple seasonal patterns and noise in time series. Several studies also support the use of forecasting models to extract seasonal patterns from time series (Bandara et al., 2020, 2021a, 2021b). The idea is to fit a forecasting model to the time series that is capable of handling time series with multiple seasonal patterns, such as TBATS (trigonometric exponential smoothing state space model with Box-Cox transformation, ARMA errors, trend and seasonal components) (De Livera et al., 2011), and Prophet (Taylor and Letham, 2021); thereafter, to extract the fitted time series components, i.e., trend, multiple seasonal components, from the fitted forecast model. Nonetheless, the objective function of these forecast models is to minimise the prediction error and therefore these methods have to limit themselves to only past data for decomposition, whereas dedicated decomposition methods can use both past and future information. Therefore, the use of prediction-based approaches for time series decomposition can be inaccurate and may give not the best possible decomposition of a time series.

In this paper, we introduce multiple STL (MSTL) decomposition, a fully automated, additive time series decomposition algorithm to handle time series with multiple seasonal cycles. The proposed MSTL algorithm is an extended version of the STL decomposition algorithm, where the STL procedure is applied iteratively to estimate the multiple seasonal components in a time series. This allows MSTL to control the smoothness of the change of seasonal components for each seasonal cycle extracted from the time series, and seamlessly separate their seasonal variations (e.g., deterministic and stochastic seasonalities). For non-seasonal time series, MSTL determines only the trend and remainder components of the time series.

Specifically, the MSTL algorithm initially determines the number of distinct seasonal patterns available in the time series. Often times, the multiple seasonal patterns are structurally unnested and interlaced together. As a result, during decomposition, the seasonal components relevant to a lower seasonal cycle can be excessively absorbed by a higher seasonal cycle. To minimise such seasonal confounding, as the second step, MSTL arranges the identified seasonal cycles in an ascending order. Then, if the time series is seasonal, MSTL applies the STL algorithm iteratively to each of the identified seasonal frequencies. Next, the trend component of the time series is computed using the last iteration of STL. On the other hand, if the time series is non-seasonal, MSTL uses Friedman's super smoother (Friedman, 1984), available in R (R Core Team, 2022) as the `supsmu` function, to directly estimate the trend of the time series. Finally, to calculate the remainder part of seasonal time series, the trend component is subtracted from the seasonally adjusted time series. Whereas, for non-seasonal time series, the trend component is subtracted from the original time series to derive the remainder.

MSTL is a robust, accurate seasonal-trend decomposition algorithm that is designed to capture multiple seasonal patterns in a time series. Most importantly, compared with other decomposition alternatives, MSTL is an extremely fast, computationally efficient algorithm, which is scalable to increasing volumes of time series data. In R, the proposed MSTL algorithm is implemented in the `mstl` function from the *forecast* package (Hyndman and Khandakar, 2008; Hyndman et al., 2022). The MSTL algorithm is also implemented in the *feasts* package (O'Hara-Wild et al., 2021a) via the `STL` function, where users can pass time series data as a `tsibble` data frame along with a time or date index (Wang et al., 2020). We describe the implementation from the *forecast* package below.

2 Model overview

Similar to the STL algorithm, MSTL gives an additive decomposition of the time series. Given X_t is the observation at time t, the additive decomposition can be defined as follows:

$$X_t = \hat{S}_t + \hat{T}_t + \hat{R}_t, \tag{1}$$

where $\hat{S}_t$, $\hat{T}_t$, $\hat{R}_t$ denote the seasonal, trend, and remainder components of the observation, respectively. MSTL extends equation (1) to include multiple seasonal patterns in a time series as follows:

$$X_t = \hat{S}_t^1 + \hat{S}_t^2 + \cdots + \hat{S}_t^n + \hat{T}_t + \hat{R}_t, \tag{2}$$

where n represents the number of seasonal cycles present in X_t.

To apply MSTL for time series with multiplicative seasonalities and trends, a transformation strategy such as log-transformation or Box-Cox transformation can be employed as a preprocessing step to ensure the additivity of time series components.

To summarise, a scheme of the MSTL procedure is given in Algorithm 1. Here X represents a `ts` or `msts` time series object from the *forecast* package, where the seasonal frequencies of time series are defined. At first, the frequencies of the seasonal patterns in the time series are identified and sorted in an ascending order. Here, the frequencies which are smaller than half of the length of the series are ignored, as those frequencies cannot exhibit any seasonal patterns. After identifying the seasonality, the missing values of the time series are imputed using the `na.interp` function available from the *forecast* package (Hyndman et al., 2022). Next, if $\lambda \in [0, 1]$ is given, a Box-Cox transformation is applied to the time series accordingly, using the `BoxCox` function available from the *forecast* package (Hyndman et al., 2022). Thereafter, to every selected seasonal cycle the STL decomposition is fitted. Here, the inner-loop repeats the STL procedure to extract the seasonal components from the time series. The STL model is implemented using the `stl` function provided by the *stats* package in R (R Core Team, 2022). In STL, the rate of seasonal variation is controlled by the `s.window` parameter. Here, a smaller value of `s.window` is set if the seasonal pattern evolves quickly, whereas a higher value is used if the seasonal pattern is constant over time. For example, adjusting the s.window parameter to 'periodic' limits the change in the seasonal components to zero, which can extract the deterministic seasonality from a time series. In MSTL, a vector of `s.window` can be provided to control the variation of the seasonal components of each seasonal cycle. We note that the default `s.window` values of the MSTL algorithm were determined through a simulation study (refer to the Appendix). The outer-loop iterates the STL procedure multiple times to refine the extracted seasonal components. After executing the outer-loop, MSTL calculates the trend component of the time series using the final iteration of STL. In situations where a time series fails the seasonality test, MSTL applies the `supsmu` function to ascertain the trend of the time series. Here, the `supsmu` function uses a running line smoother with linear interpolation to determine the trend component of the time series. Finally, the remainder component is retrieved by deducting the trend component from the seasonally-adjusted time series. Also, since MSTL extends the STL algorithm, the other parameters of STL (e.g., `t.window`, `l.window`) are inherited by MSTL, and can be used for decomposition. We note that the seasonal components cannot be aggregated if the multiple seasonal patterns are non-nested, for example when a time series is affected by both the Islamic and Gregorian calendars. On the other hand, seasonal components can be aggregated if the seasonal patterns are nested, though the interpretability of the decomposed components may be lost.

Algorithm 1 MSTL decomposition

```
 Parameters
 X – ts or msts time series object
 iterate – number of STL iterations
 $\lambda$ – Box-Cox transformation parameter
 s.window[] – s.window values

procedure MSTL(X, iterate, $\lambda$, s.win[], ...)
    if X is multi-seasonal then
        seas.ids[] ← attributes(X)
        k ← len(X)
        seas.ids ← sea.ids[seas.ids < k/ 2]
        seas.ids ← sort(seas.ids, dec = F)
    else if X is single-seasonal then
        seas.ids[] ← frequency(X)
        iterate ← 1
    end if
    X ← na.interp(X, $\lambda$)
    X ← BoxCox(X, $\lambda$)
    if seas.ids[1] > 1 then
        seasonality ← list(rep(0, len(seas.ids)))
        deseas ← X
        for j in 1 to iterate do
            for i in 1 to len(seas.ids) do
                deseas ← deseas + seasonality[[i]]
                fit ← STL(ts(deseas, frequency = seas.ids[i]), s.window = s.win[i], ...)
                seasonality[[i]] ← msts(seasonal(fit))
                deseas ← deseas - seasonality[[i]]
            end for
        end for
        trend ← msts(trendcycle(fit))
    else
        seas.ids← NULL
        deseas ← X
        trend ← ts(SUPSMU(X))
    end if
    remainder ← deseas - trend
    return [trend, remainder, seasonality]
end procedure
```

3 Experimental setup

We evaluate the proposed MSTL decomposition algorithm on both simulated and a perturbed real-world time series dataset, where we know the true composition, i.e., trend, seasonality, and remainder, of time series.

3.1 Benchmarks

We compare MSTL against a collection of current state-of-the-art techniques in decomposing time series with multiple seasonal cycles. This includes STR (Dokumentov and Hyndman, 2021) as a pure decomposition technique, TBATS (De Livera et al., 2011) and Prophet (Taylor and Letham, 2021) as forecasting techniques, where we use the decomposition of the time series.

- *STR:* A regression-based, additive decomposition technique. In R, the STR algorithm is available through the `STR` function from the *stR* package (Dokumentov and Hyndman, 2018).
- *TBATS:* A state-of-the-art technique to forecast time series with multiple seasonal cycles. TBATS uses trigonometric expression terms to model complex seasonal terms in a time series. In our experiments, we use the R implementation of the TBATS algorithm, `tbats`, from the *forecast* package (Hyndman et al., 2022).
- *PROPHET:* An automated forecasting framework, developed by Facebook, that can handle multiple seasonal patterns. Similar to STR and MSTL, Prophet is an additive decomposition technique. In our evaluation, we apply the Prophet algorithm available through the *prophet* package in R (Taylor and Letham, 2021).
- *DSA:* A decomposition algorithm specifically developed for daily time series data. In our experiments, for the simulated and real daily time series, we use the R implementation of the DSA algorithm from the *dsa* package (Ollech, 2021).

3.2 Evaluation metric

The accuracy of the decomposition methods are evaluated using the root mean square error (RMSE). The RMSE is defined as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_t - \hat{X}_t)^2} \tag{3}$$

Here, X_t represents the actual decomposition value at time t, and $\hat{X}_t$ is the estimated decomposition value. Also, n is the number of observations in the time series.

3.3 Simulated data

We simulate daily and hourly data using four time series components. The additive decomposition of the daily time series (X_t^D) and the hourly time series (X_t^H) can be formulated as follows:

$$X_t^D = T_t^D + \alpha S_t^W + \beta S_t^Y + \gamma R_t^D, \quad t = 1, \ldots, n, \tag{4}$$

$$X_t^H = T_t^H + \alpha S_t^D + \beta S_t^W + \gamma R_t^H, \quad t = 1, \ldots, m, \tag{5}$$

In equation (4), T_t^D is the trend, S_t^W is the weekly seasonal component, S_t^Y is the yearly seasonal component, and R_t^D is the remainder of X_t^D. Whereas in equation (5), T_t^H, S_t^D, S_t^W, and R_t^H corresponds to the trend, daily seasonal component, weekly seasonal component, and remainder component of X_t^H respectively. Here, α, β, and γ are parameters which control the contribution of the components to X_t^D and X_t^H. Also, n denotes the length of the daily time series and m is the length of the hourly time series.

We simulate the time series data using two data generating processes (DGPs), the 'deterministic DGP' and the 'stochastic DGP'. Here, the deterministic DGP generates time series that has deterministic components that are invariant to time, whereas the Stochastic DGP gives time series with time-varying components.

In our experiments, the deterministic trend components T_t^d are generated using a quadratic trend function with random coefficients, $T_t^d = N_1(t + n/2(N_2 - 1))^2$ where N_1 and N_2 are independent N(0, 1) random variables. The deterministic seasonal components S_t^d are composed of five pairs of Fourier terms with random N(0, 1) coefficients. Both T_t^d and S_t^d are normalised to give mean zero and unit variance.

To generate stochastic trend components T_t^s, we use an ARIMA(0,2,0) model with standard normal errors. With respect to the stochastic seasonal components S_t^s, we introduce an additional error term N(0, σ^2) to S_t^d to change the coefficients for the Fourier terms from one seasonal cycle to another. Here, the stochastic strength of the seasonality is controlled by the σ^2 parameter value. In other words, the $\sigma^2 = 0$ scenario of S_t^s represents the S_t^d. In our experiments, we change the values of the σ^2 variable in small values to avoid drastic changes to the seasonal components of the simulated time series, i.e., to avoid possible concept drifts introduced to time series data.

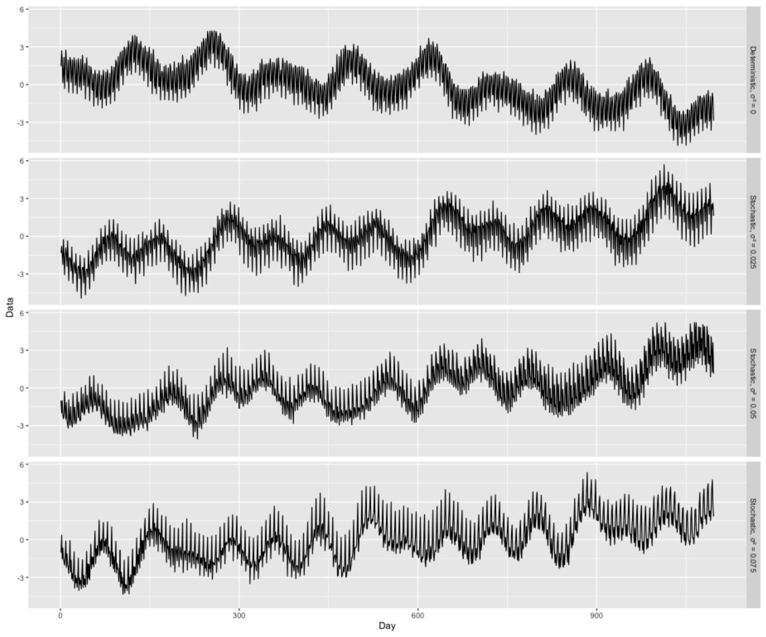
Figure 2 illustrates the examples of simulated daily and hourly time series generated by the two DGPs. Here, we set the $\alpha = \beta = 1$ and $\gamma = 0.2$ for both the DGPs. The lengths of the daily and hourly time series are equivalent to 1096 days and 505 days respectively (one observation more than three seasonal cycles of the highest available seasonality, i.e., 365 * 3 + 1 for the daily data and 168 * 3 + 1 for the hourly data).

Table 1 summarises the parameter sets (α, β, γ, σ^2) used in our experiments. We create 150 time series for each DGP, generating 300 time series of daily and hourly data. As we know the true values of the seasonal, trend, and remainder components of the simulated time series, we calculate the root mean square error (RMSE) for each component by averaging across all datasets and dates. The presence of statistical significance of differences within multiple decomposition methods is assessed using a linear model applied to the squared errors.

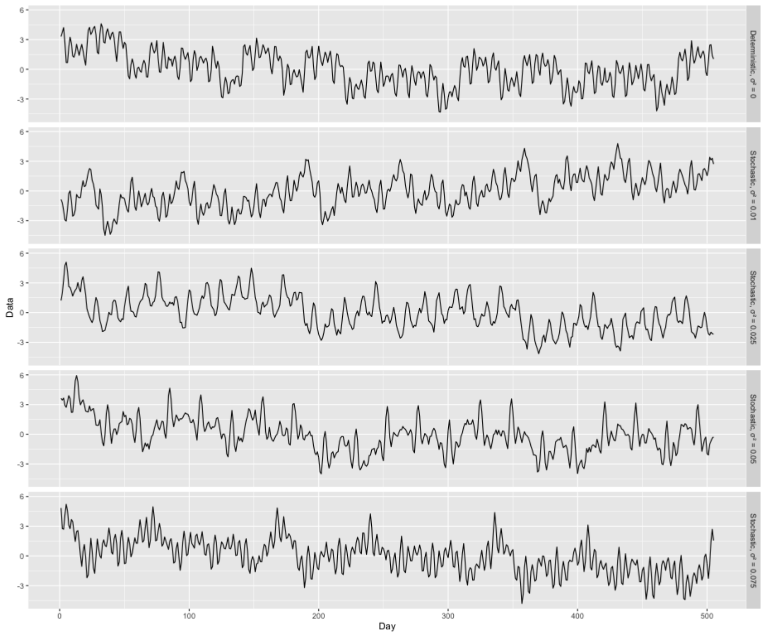
Moreover, for the deterministic DGPs we set the `s.window` values of MSTL to *periodic*, whereas for the stochastic DGPs, we use the default `s.window` values of the MSTL, which were obtained through a simulation study (refer to the Appendix).

Table 2 shows the evaluation summary for the daily simulated time series. With respect to deterministic DGPs, it can be seen that on the trend, weekly and yearly seasonal components, MSTL outperforms the TBATS method in the majority of cases. With respect to Weekly RMSE values for stochastic DGPs, we observe that MSTL outperforms the majority of the benchmarks, except for DSA. Also, for stochastic DGPs, MSTL gives better results compared to TBATS on the Trend, Weekly and Yearly seasonal components.

Figure 2 Examples of daily and hourly time series generated by the deterministic and stochastic DGPs, (a) the simulated daily time series with different values of σ^2 that control the seasonal stochastic components (b) the simulated hourly time series with different values of σ^2 that control the seasonal stochastic components



(a)



(b)

Table 1 The parameter values used by each DGP to simulate time series

DGP	α	β	γ	σ^2
Deterministic	1	1	0.2	0
Deterministic	1	1	0.4	0
Deterministic	1	1	0.6	0
Stochastic	1	1	0.2	0.025
Stochastic	1	1	0.4	0.050
Stochastic	1	1	0.6	0.075

Table 2 The RMSE over 150 simulations for each DGP on daily simulated time series data

γ	σ^2	*Method*	*Trend RMSE*	*Weekly RMSE*	*Yearly RMSE*	*Remainder RMSE*
			Deterministic DGP			
0.2	0	STR	**0.0174**	**0.0151**	*0.0477*	**0.0514**
0.2	0	TBATS	*0.1896*	*0.0171*	*0.1822*	*0.0569*
0.2	0	PROPHET	*0.0328*	*0.0465*	**0.0423**	*0.0571*
0.2	0	MSTL	0.0623	0.0166	0.1471	0.1429
0.2	0	DSA	*0.0675*	*0.0314*	*0.1242*	*0.1193*
0.4	0	STR	**0.0314**	**0.0291**	*0.0959*	*0.1023*
0.4	0	TBATS	*0.3752*	*0.0324*	*0.3678*	*0.1081*
0.4	0	PROPHET	*0.0524*	*0.0514*	**0.0686**	**0.0805**
0.4	0	MSTL	0.0786	0.0342	0.2471	0.2497
0.4	0	DSA	*0.0772*	*0.0608*	*0.2034*	*0.2196*
0.6	0	STR	**0.0467**	***0.0463***	*0.1438*	*0.1554*
0.6	0	TBATS	*0.5371*	*0.0526*	*0.5228*	*0.1640*
0.6	0	PROPHET	*0.0637*	*0.0631*	***0.0918***	**0.1170**
0.6	0	MSTL	0.0787	0.0556	0.3597	0.3628
0.6	0	DSA	*0.1060*	*0.0917*	*0.2975*	*0.3252*
			Stochastic GDP			
0.2	0.025	STR	*0.0586*	*0.1492*	*0.0795*	*0.1550*
0.2	0.025	TBATS	*0.2097*	*0.1292*	*0.2004*	*0.1430*
0.2	0.025	PROPHET	*0.0708*	*0.1542*	**0.0666**	*0.1633*
0.2	0.025	MSTL	0.1700	0.0669	0.1638	0.1936
0.2	0.025	DSA	**0.0461**	**0.0641**	*0.1082*	**0.1203**
0.4	0.05	STR	**0.0703**	*0.2623*	*0.1354*	*0.2756*
0.4	0.05	TBATS	*0.3677*	*0.2318*	*0.3506*	*0.2592*
0.4	0.05	PROPHET	*0.0800*	*0.2629*	**0.0837**	*0.2764*
0.4	0.05	MSTL	0.1389	0.1315	0.2439	0.2918
0.4	0.05	DSA	*0.0866*	***0.1255***	*0.2062*	**0.2351**
0.6	0.075	STR	**0.0715**	*0.3811*	**0.1740**	*0.3942*
0.6	0.075	TBATS	*0.5211*	*0.3439*	*0.4934*	*0.3777*
0.6	0.075	PROPHET	*0.0837*	*0.3810*	*0.1069*	*0.3997*
0.6	0.075	MSTL	0.1010	0.1982	0.3482	0.4046
0.6	0.075	DSA	*0.1212*	**0.1521**	*0.3075*	**0.3420**

Note: Italic values indicate results that are significantly different from the MSTL values and the best performing method(s) for each time series component is highlighted in bold.

Table 3 The RMSE over 150 simulations for each DGP on hourly simulated time series data

γ	σ^2	*Method*	*Trend RMSE*	*Daily RMSE*	*Weekly RMSE*	*Remainder RMSE*
			Deterministic DGP			
0.2	0	STR	***0.0253***	*0.2736*	*0.2827*	*0.0951*
0.2	0	TBATS	*0.1004*	**0.0363**	**0.0864**	**0.0737**
0.2	0	PROPHET	*0.1170*	*0.4686*	*0.6254*	*0.7872*
0.2	0	MSTL	0.0684	0.0487	0.1400	0.1437
0.4	0	STR	**0.0476**	*0.2475*	*0.2697*	*0.1651*
0.4	0	TBATS	*0.1779*	**0.0734**	**0.1580**	**0.1421**
0.4	0	PROPHET	*0.1243*	*0.4377*	*0.6577*	*0.7973*
0.4	0	MSTL	0.0747	0.0885	0.2283	0.2437
0.6	0	STR	**0.0613**	*0.1451*	***0.1992***	*0.2318*
0.6	0	TBATS	*0.2704*	**0.1139**	*0.2436*	**0.2115**
0.6	0	PROPHET	*0.1440*	*0.4606*	*0.6520*	*0.8087*
0.6	0	MSTL	0.0858	0.1289	0.3362	0.3614
			Stochastic GDP			
0.2	0.025	STR	**0.0847**	*0.0905*	*0.1253*	*0.1005*
0.2	0.025	TBATS	*0.1324*	**0.0612**	**0.1227**	**0.0890**
0.2	0.025	PROPHET	*0.2226*	*0.4676*	*0.6483*	*0.8149*
0.2	0.025	MSTL	0.1933	0.0952	0.1803	0.2128
0.4	0.05	STR	**0.0588**	*0.8307*	*0.8351*	*0.1900*
0.4	0.05	TBATS	*0.2215*	**0.1195**	**0.2094**	**0.1716**
0.4	0.05	PROPHET	*0.1201*	*0.4643*	*0.6344*	*0.7906*
0.4	0.05	MSTL	0.0883	0.1779	0.2520	0.2751
0.6	0.075	STR	*0.1741*	*0.2149*	**0.2529**	*0.2720*
0.6	0.075	TBATS	*0.3081*	**0.1820**	*0.2955*	**0.2573**
0.6	0.075	PROPHET	*0.1463*	*0.4415*	*0.6168*	*0.7660*
0.6	0.075	MSTL	**0.1289**	0.2577	0.3699	0.4085

Note: Bold values indicate results that are significantly different from the MSTL values and the best performing method(s) for each time series component is highlighted in italic.

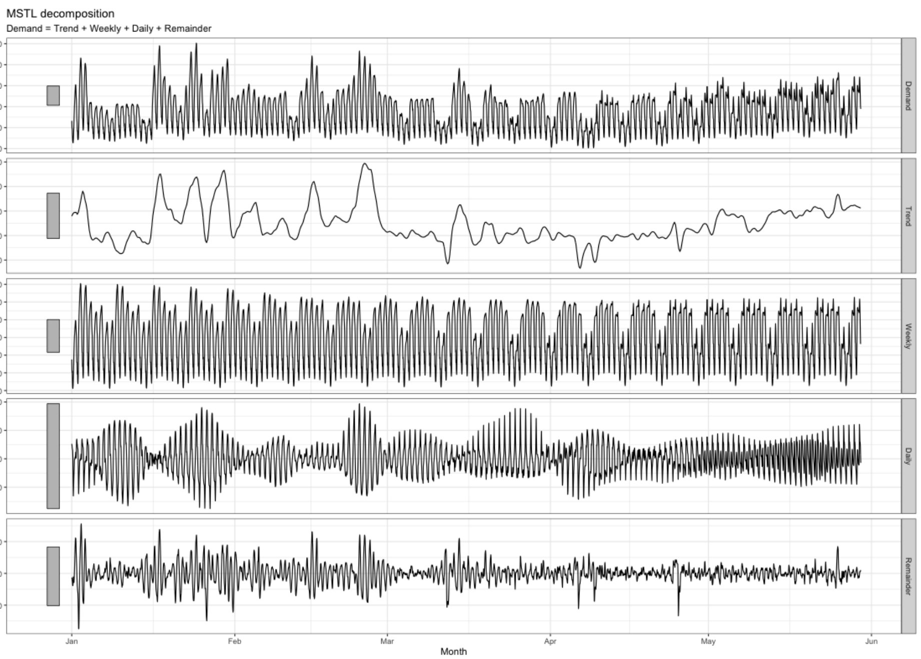
Table 3 shows the results of all the decomposition techniques for the hourly simulated time series. For the deterministic and stochastic DGPs, we see that the proposed MSTL algorithm gives better RMSE on all components, compared to the PROPHET method. Also, for the $(\gamma, \sigma^2) = \{(0.6, 0.075)\}$ scenario on the Trend component, we that the MSTL achieves the best RMSE.

3.4 *Perturbed real-world data*

We also evaluate the performance of MSTL using real time series data. We use the moving block bootstrap (MBB) for perturbing the time series, following the procedure introduced in Bergmeir et al. (2016). Here, we first use MSTL to extract the seasonal, trend and remainder of components of a time series. Assuming these extracted components are the true decomposition of the time series, next we apply the MBB technique to the remainder component of the time series to generate multiple versions of the residual components. Finally, these bootstrapped residual components are added back together with the previously extracted seasonal and trend components to produce new perturbed versions of a time series, where we know the true composition of the series. In our experiments, we use the MBB implementation available through the `MBB` function from the *forecast* package (Hyndman and Khandakar, 2008; Hyndman et al., 2022).

Firstly, we select the half-hourly electricity consumption in the state of Victoria, Australia, extracted from the *vic_elec* dataset (O'Hara-Wild et al., 2021b) to generate multiple versions of the time series. As illustrated in Figure 1, this time series has two seasonal patterns, the daily pattern and the weekly seasonal pattern. In our experiments, we first aggregate the half-hourly data to hourly data, and select 149 days starting from 01 January 2012. Figure 3 shows the application of MSTL to the hourly electricity demand in Victoria (3601 hourly observations).

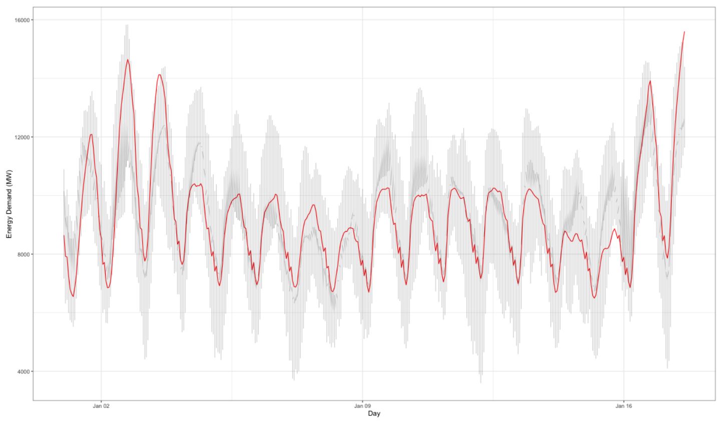
Figure 3 The decomposition of hourly electricity demand in Victoria using MSTL



Note: Each panel represents the original data, the trend, the weekly seasonality, the daily seasonality, and the remainder respectively.

As discussed earlier, we use the MBB technique to generate 100 bootstrapped versions of the hourly electricity demand time series, so we can assess the performance of MSTL on a real-world dataset. Figure 4 illustrates the snippet of original hourly electricity demand time series and the bootstrapped time series.

Figure 4 The generation of multiple versions of hourly electricity demand time series, applying the MBB technique to the remainder component of the original time series (see online version for colours)



Note: The original time series is plotted in red, while the bootstrapped time series are in grey.

Table 4 summarises the overall performance of MSTL and the benchmarks on the perturbed electricity demand time series. According to Table 4, MSTL significantly outperforms STR, TBATS, and PROPHET in estimating all components.

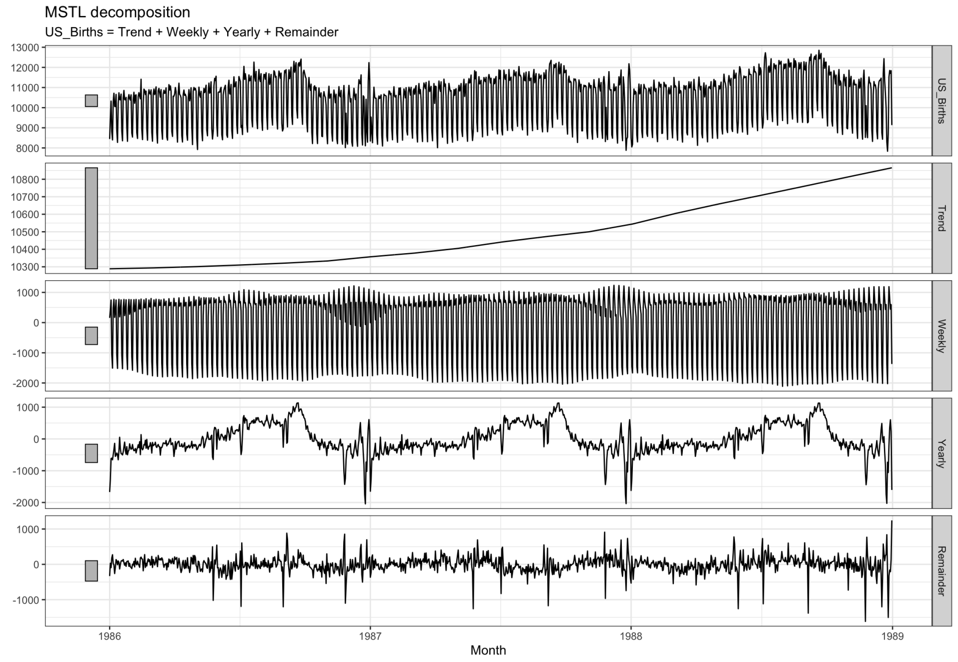
Table 4 The RMSE over 100 bootstrapped versions of the hourly electricity demand time series

Method	*Trend RMSE*	*Daily RMSE*	*Weekly RMSE*	*Remainder RMSE*
STR	*399.4*	*408.0*	*214.8*	*580.9*
TBATS	*742.1*	*348.9*	*383.1*	*614.1*
PROPHET	*243.7*	*371.1*	*403.9*	*605.4*
MSTL	**207.6**	**149.2**	**180.5**	**312.7**

Note: Italic values indicate results that are significantly different from the MSTL values and the best performing method(s) for each time series component is highlighted in bold.

As a second example of a real time series dataset, we use the daily births time series in the United States from 1 January 1969 to 31 December 1988 (Godahewa et al., 2021; Pruim et al., 2021). From Figure 5, it can be seen that this time series has two seasonal patterns, the weekly pattern and the yearly pattern. In our experiments, we select three years worth of daily data from 1 January 1986 to 31 December 1988. Similar to the half-hourly electricity consumption experiments, we generate 100 bootstrapped versions of the daily US births time series to calculate the decomposition error.

Figure 5 The decomposition of daily US births in USA using MSTL



Note: Each panel represents the original data, the trend, the weekly seasonality, the yearly seasonality, and the remainder respectively.

Table 5 summarises the overall performance of MSTL and the benchmarks on the perturbed US births time series. The results are consistent with our previous findings from Table 4, the proposed MSTL algorithm shows the lowest RMSE across all components, outperforming STR, TBATS, PROPHET, and DSA. To understand the variability of decompositions due to the introduction of new data points (the problem of revisions), we extend our experiments on the US births dataset. Here, we hold out the last observation in the dataset (for all 100 bootstrapped time series), and re-run MSTL and benchmarks to evaluate the component change in the last period of time series, i.e., last 7 days. To evaluate the variability, we calculate the mean absolute difference between the values of original components and the component values after removing the last observation. Table 6 summarises the results of the proposed component variance experiment for MSTL and the benchmarks on the perturbed US births time series dataset. According to Table 6, MSTL has the lowest variance for yearly and remainder components. Moreover, MSTL has the second lowest variance for the trend component, outperforming STR, TBATS, and Prophet.

Furthermore, Table 7 provides a summary of the computational cost of MSTL and the benchmarks over the 100 bootstrapped time series. The experiments are run on an Intel(R) i7 processor (1.8 GHz), with 2 threads per core, 8 cores in total, and 16 GB of main memory. As shown in Table 7, MSTL has the lowest execution time compared to other benchmarks, highlighting the scalability of MSTL to the increasing volumes of time series data. When used for sub-daily time series data (e.g., hourly, half-hourly, minutely), the computational efficiency of a decomposition method can be important as those time series are generally longer and contain a higher number of observations.

Table 5 The RMSE over 100 bootstrapped versions of the daily US births time series

Method	*Trend RMSE*	*Daily RMSE*	*Weekly RMSE*	*Remainder RMSE*
STR	31.1	*156.4*	*184.6*	*164.6*
TBATS	*606.2*	*147.8*	*642.6*	*296.3*
PROPHET	22.27	*144.4*	*249.2*	*297.8*
MSTL	**19.16**	**74.9**	**134.8**	**151.1**
DSA	*50.2761*	*1133.1*	*192.6*	*207.8*

Note: Italic values indicate results that are significantly different from the MSTL values and the best performing method(s) for each time series component is highlighted in bold.

Table 6 The mean absolute difference over 100 bootstrapped versions of the daily US births time series

Method	*Trend variance*	*Daily variance*	*Yearly variance*	*Remainder RMSE*
STR	238.0	66.6	532.0	539.0
TBATS	998.0	53.7	967.0	564.0
PROPHET	250.0	**50.7**	716.0	614.0
MSTL	225.0	112.0	**499.0**	**473.0**
DSA	**224.0**	69.2	538.0	502.0

Note: Bold values indicate the method(s) that have the lowest variability for each time series component.

Table 7 The total computational cost of the decomposition methods for the electricity demand dataset, measured in seconds

Method	*Total time*
MSTL	7
STR	612
PROPHET	936
TBATS	2521

4 Implications for decision making

As discussed earlier, decomposition of univariate time series data has many important applications for governments and businesses. Here are a few ways that MSTL can be used for decision-making.

- The seasonally-adjusted data is often used by governments to assess the effect of their policy decisions on the economy, after adjusting for periodic effects. For example, MSTL can be used to determine the seasonally adjusted unemployment rate in a given year to assess the state of the economy.

- Given the availability of high-frequency time series data (e.g., hourly, half-hourly), energy providers wish to measure the effect of time-of-day (daily seasonality), day-of-week (weekly seasonality), and time-of-year (annual seasonality) on the energy demand. In such situations, MSTL is useful to separate the trend and identify the multi-seasonal effects, allowing a greater understanding of peak times and an estimation of the relative effect of working days, weekends, and changes over the year. This information also provides support for decision-making of energy providers when planning short-term, medium-term and long-term generation.
- For applications that monitor the network or systems for malicious activity, MSTL can be used to identify the traffic caused by unusual variations in the network. The regular traffic variations caused by time-of-day, day-of-week, and time-of-year, can be first determined by the MSTL; then analysts can search for abnormal values in the remainder series. Many organisations encounter millions of time series describing web-traffic at high frequency; therefore, it is essential to have computationally efficient, accurate solutions such as MSTL to assist in identifying scenarios that may need attention.

5 Conclusions

Time series datasets with multiple seasonalities are now common in applications. To better understand the variations of such time series, it is useful to decompose the time series into their subcomponents, such as trend, seasonality, and remainder. The existing techniques available for decomposing time series with multiple seasonal cycles are mostly based on complex procedures that can be computationally inefficient for long time series.

To this end, we have introduced MSTL, a fast time series decomposition algorithm that is capable of handling time series with multiple seasonal cycles. The proposed MSTL algorithm is an extension of the STL decomposition algorithm, which can only extract a single seasonality from a time series. Experimental results on both simulated data and perturbed real-world data have demonstrated that MSTL provides competitive results with lower computational cost in comparison with other state-of-the-art decomposition algorithms, such as DSA, STR, TBATS, and PROPHET. The current implementation of MSTL is unable to account for exogenous variables such as holiday effects. Furthermore, MSTL is not capable of handling non-integer seasonalities and irregular seasonal patterns (e.g., daily data with monthly seasonality). Therefore, in situations where holiday effects and non-integer seasonalities are present in the time series, it is better to use decomposition algorithms such as STR and DSA (for daily data), which are capable of handling such scenarios.

The MSTL algorithm is implemented in the `mstl` function in the *forecast* package (Hyndman et al., 2022), and in the `STL` function in the *feasts* package (O'Hara-Wild et al., 2021a), both of which are available on the Comprehensive R Archive Network (CRAN).

References

Angelopoulos, D., Siskos, Y. and Psarras, J. (2019) 'Disaggregating time series on multiple criteria for robust forecasting: the case of long-term electricity demand in Greece', *Eur. J. Oper. Res.*, Vol. 275, No. 1, pp.252–265.

Bandara, K., Bergmeir, C., Campbell, S., Scott, D. and Lubman, D. (2020) 'Towards accurate predictions and causal 'what-if' analyses for planning and policy-making: a case study in emergency medical services demand', *2020 International Joint Conference on Neural Networks (IJCNN)*, pp.1–10.

Bandara, K., Bergmeir, C. and Hewamalage, H. (2021a) 'LSTM-MSNet: leveraging forecasts on sets of related time series with multiple seasonal patterns', *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 32, No. 4, pp.1586–1599.

Bandara, K., Hewamalage, H., Liu, Y.-H., Kang, Y. and Bergmeir, C. (2021b) 'Improving the accuracy of global forecasting models using time series data augmentation', *Pattern Recognittion*, Vol. 124, No. 1, p.108148.

Bell, W.R. and Hillmer, S.C. (1984) 'Issues involved with the seasonal adjustment of economic time series', *J. Bus. Econ. Stat.*, Vol. 2, No. 4, pp.291–320.

Bergmeir, C., Hyndman, R.J. and Benítez, J.M. (2016) 'Bagging exponential smoothing methods using STL decomposition and Box-Cox transformation', *International Journal of Forecasting*, Vol. 32, No. 2, pp.303–312.

Cleveland, R.B., Cleveland, W.S., McRae, J.E. and Terpenning, I.J. (1990) 'STL: a seasonal-trend decomposition procedure based on Loess', *Journal of Official Statistics*, Vol. 6, No. 1, pp.3–73.

Daniel, O. (2021) 'Seasonal adjustment of daily time series', *J. Time Ser. Econom.*, Vol. 13, No. 2, pp.235–264.

De Livera, A.M., Hyndman, R.J. and Snyder, R. (2011) 'Forecasting time series with complex seasonal patterns using exponential smoothing', *J. Am. Stat. Assoc.*, Vol. 106, No. 496, pp.1513–1527.

Dokumentov, A. and Hyndman, R.J. (2018) *stR: STR Decomposition*, R package version 0.4.

Dokumentov, A. and Hyndman, R.J. (2021) 'STR: A seasonal-trend decomposition procedure based on regression', *INFORMS J on Data Science*, in press.

Findley, D.F., Monsell, B.C., Bell, W.R., Otto, M.C. and Chen, B-C. (1998) 'New capabilities and methods of the X-12-ARIMA seasonal-adjustment program', *J. Bus. Econ. Stat.*, Vol. 16, No. 2, pp.127–152.

Friedman, J.H. (1984) *A Variable Span Scatterplot Smoother*, Technical Report 5, Laboratory for Computational Statistics, Stanford University.

Godahewa, R., Bergmeir, C., Webb, G.I., Hyndman, R.J. and Montero-Manso, P. (2021) 'Monash time series forecasting archive', *Neural Information Processing Systems Track on Datasets and Benchmarks*.

Hewamalage, H., Bergmeir, C. and Bandara, K. (2021) 'Recurrent neural networks for time series forecasting: Current status and future directions', *International Journal of Forecasting*, Vol. 37, No. 1, pp.388–427.

Hyndman, R.J., Athanasopoulos, G., Bergmeir, C., Caceres, G., Chhay, L., O'Hara-Wild, M., Petropoulos, F., Razbash, S., Wang, E., Yasmeen, F., R Core Team, Ihaka, R., Reid, D., Shaub, D., Tang, Y. and Zhou, Z. (2022) *forecast: Forecasting Functions for Time Series and Linear Models*, R package version 8.16.

Hyndman, R.J. and Khandakar, Y. (2008) 'Automatic time series forecasting: the forecast package for R', *Journal of Statistical Software*, Vol. 26, No. 1, pp.1–22.

Koopman, S.J. and Ooms, M. (2006) 'Forecasting daily time series using periodic unobserved components time series models', *Computational Statistics & Data Analysis*, Vol. 51, No. 2, pp.885–903.

Lin, W., Huang, J.Z. and McElroy, T. (2020) 'Time series seasonal adjustment using regularized singular value decomposition', *J. Bus. Econ. Stat.*, Vol. 38, No. 3, pp.487–501.

Maravall, A. (2006) 'An application of the TRAMO-SEATS automatic procedure; direct versus indirect adjustment', *Computational Statistics & Data Analysis*, Vol. 50, No. 9, pp.2167–2190.

McElroy, T. (2021) 'A diagnostic for seasonality based upon polynomial roots of ARMA models', *J. Off. Stat.*, Vol. 37, No. 2, pp.367–394.

O'Hara-Wild, M., Hyndman, R. and Wang, E. (2021a) *feasts: Feature Extraction and Statistics for Time Series*, R package version 0.2.2.

O'Hara-Wild, M., Hyndman, R.J. and Wang, E. (2021b) *tsibbledata: Diverse Datasets for 'tsibble'*, R package version 0.3.0.

Ollech, D. (2021) *dsa: Seasonal Adjustment of Daily Time Series*, R package version 1.0.12.

Petropoulos, F., Hyndman, R.J. and Bergmeir, C. (2018) 'Exploring the sources of uncertainty: why does bagging for time series forecasting work?', *Eur. J. Oper. Res.*, Vol. 268, No. 2, pp.545–554.

Pruim, R., Kaplan, D. and Horton, N. (2021) *mosaicData: Project MOSAIC Data Sets*, R package version 0.20.2.

R Core Team (2022) *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria.

Taylor, S. and Letham, B. (2021) *prophet: Automatic Forecasting Procedure*, R package version 1.0.

Thornton, M.A. (2013) 'Removing seasonality under a changing regime: filtering new car sales', *Computational Statistics & Data Analysis*, Vol. 58, No. 1, pp.4–14.

Wang, E., Cook, D. and Hyndman, R.J. (2020) 'A new tidy data structure to support exploration and modeling of temporal data', *J Computational & Graphical Statistics*, Vol. 29, No. 3, pp.466–478.

Wen, Q., Gao, J., Song, X., Sun, L., Xu, H. and Zhu, S. (2019) 'RobustSTL: A robust Seasonal-Trend decomposition algorithm for long time series', *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33, No. 1, pp.5409–5416.

Wen, Q., Zhang, Z., Li, Y. and Sun, L. (2020) 'Fast RobustSTL: efficient and robust Seasonal-Trend decomposition for time series with complex patterns', *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '20*, Association for Computing Machinery, New York, NY, USA, pp.2203–2213.

Zhang, G.P. and Qi, M. (2005) 'Neural network forecasting for seasonal and trend time series', *Eur. J. Oper. Res.*, Vol. 160, No. 2, pp.501–514.

Appendix

Results of seasonal window simulations

To determine the default parameter values for the `s.window` parameter values of MSTL, we conduct a series of experiments using a simulation setup similar to Subsection 3.3. Table A1 summarises the parameter sets (α, β, γ, σ^2) used in our experiments. We only use stochastic DGPs to generate time series for this simulation study as real-world time series often contain stochastic time series components.

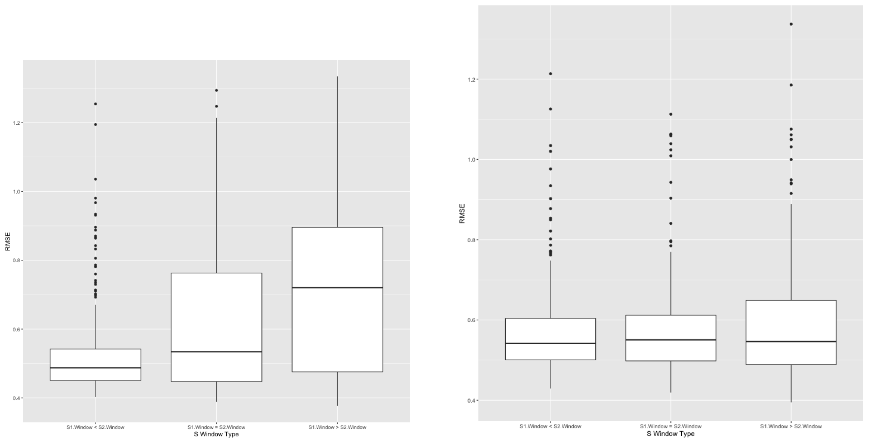
As we use daily and hourly simulated data, we define the first `s.window` value for the lowest seasonal cycle as S1.Window and the second `s.window` value for the next highest seasonal cycle as S2.Window. For example, S1.Window and S2.Window are the `s.window` values for the weekly and yearly seasonal cycles of daily time series and the `s.window` values for the daily and weekly seasonal cycles of hourly time series. Figure A1 shows the RMSE error distribution boxplots for the different types of

S1.Window and S2.Window combinations on the weekly and hourly datasets. Here, the S1.Window and S2.Window pairs are chosen from the vector $S = (7, 15, 23, 9999)$.

Table A1 The parameter values used for the seasonal window simulations

DGP	α	β	γ	σ^2
Stochastic	1	1	0.2	0.025
Stochastic	1	1	0.4	0.050
Stochastic	1	1	0.6	0.075

Figure A1 The RMSE error distribution of MSTL across different categories of S1.Window and S2.Window combinations, (a) the performance of MSTL across different categories of S1.Window and S2.Window combinations on the weekly dataset (b) the performance of MSTL across different categories of S1.Window and S2.Window combinations on the hourly dataset



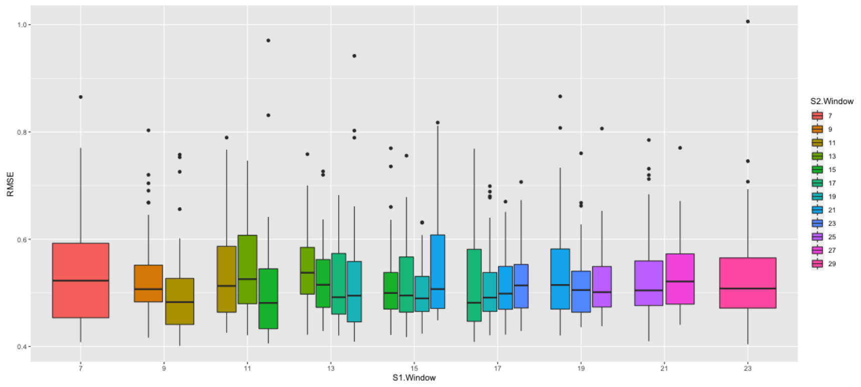
(a) (b)

To identify default parameter values for the `s.window` parameter, we extend this experiments to include more combinations of `s.window` values that satisfy the S1.Window < S2.Window premise. Figure A2 shows RMSE error distribution boxplots for the different combinations of S1.Window and S2.Window values on the weekly and hourly datasets. The S1.Window and S2.Window pairs are generated from $S = (C + K * i, C + K * i + 1)$, where $C = (7, 9, 11, 13, 15)$ and $K = (0, 1, 2, 3, 4, 5, 6, 7)$. Here i represents the seasonal cycle number, i.e., for the first and second seasonality $i = 1$ and $i = 2$ respectively. Also, we select the smallest odd value from S, when choosing a `s.window` value.

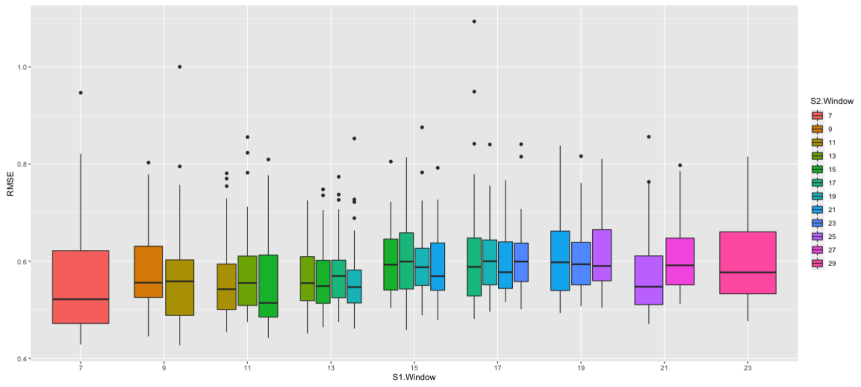
Overall, it can be seen that the S1.Window value of 11 and the S2.Window value of 15 gives the best median RMSE for weekly and hourly time series, which is the $C = 7$ and $K = 4$ instance of the above formula. Based on these results, we set the default `s.window` parameter of MSTL using the $S = (C + K * i, C + K * i + 1)$ formula, where C and K values are set to 7 and 4 respectively.

According to Figure A1, the `s.window` combinations that meet the S1.Window < S2.Window condition give the best median RMSE.

Figure A2 The RMSE error distribution of MSTL across different combinations of S1.Window and S2.Window values, (a) the performance of MSTL across different combinations of S1.Window and S2.Window values on the weekly dataset (b) the performance of MSTL across different combinations of S1.Window and S2.Window values on the hourly dataset (see online version for colours)



(a)



(b)