Journal of Statistical Software
December 2025, Volume 115, Issue 10. doi: 10.18637/jss.v115.i10

TrendLSW: Trend and Spectral Estimation of Nonstationary Time Series in R

Euan T. McGonigle
University of Southampton

Rebecca Killick
Lancaster University

Matthew A. Nunes
University of Bath

Abstract

The **TrendLSW** R package has been developed to provide users with a suite of wavelet-based techniques to analyze the statistical properties of nonstationary time series. The key components of the package are (a) two approaches for the estimation of the evolutionary wavelet spectrum in the presence of trend; and (b) wavelet-based trend estimation in the presence of locally stationary wavelet errors via both linear and nonlinear wavelet thresholding; and (c) the calculation of associated pointwise confidence intervals. Lastly, the package directly implements boundary handling methods that enable the methods to be performed on data of arbitrary length, not just dyadic length as is common for wavelet-based methods, ensuring no preprocessing of data is necessary. The key functionality of the package is demonstrated through two data examples, arising from biology and activity monitoring.

Keywords: **TrendLSW**, evolutionary wavelet spectrum, trend estimation, locally stationary time series, R.

1. Introduction

Modern time series data can often possess complex characteristics. Given technological advancements in data recording tools, leading to time series being observed over increasingly larger time-scales, it is common for the statistical properties of a time series to vary over time. Examples of so-called 'nonstationary' time series can be found in a wide variety of applications areas, including climatology (Beaulieu and Killick 2018), economics (Roueff and von Sachs 2019), and epidemiology (Jiang, Zhao, and Shao 2023).

Two key quantities of interest, the mean function and dependence structure, are amongst the most commonly studied properties in time series analysis. Modeling how these properties evolve is crucial for making informed inference on the data: incorrectly assuming stationary behavior may lead to drawing misleading conclusions. It is known to be a challenging problem to estimate one of these time-varying properties when the other is stationary, and yet more challenging when both display time-dependent characteristics. Therefore, it is common practice for analysis to be restricted to either the mean or dependence.

For mean function estimation in time series, there is a vast literature dedicated to topics including tackling nonparametric regression (von Sachs and MacGibbon 2000; Vogt 2012), and trend estimation and detection (Wu and Zhao 2007; Zhang and Wu 2011).

For modeling nonstationary dependence structures, a number of approaches have been proposed; for an overview, see Dahlhaus (2012). For example, Priestley (1965) introduces evolutionary processes, Dahlhaus (1997) defines the locally stationary Fourier (LSF) processes, whilst Nason, von Sachs, and Kroisandt (2000) introduce the locally stationary wavelet (LSW) model. Ombao, Raz, von Sachs, and Guo (2002) consider the smoothed localized exponential (SLEX) model, whilst Van Bellegem and Dahlhaus (2006) discuss nonstationary autoregressive models, where the parameters are allowed to vary with time. Zhou and Wu (2009) define locally stationary time series as an "input-output" physical system. Other approaches which can capture both time-varying means and time-varying dynamics are time-varying structural autoregression models (e.g. Kilian and Lütkepohl 2017) and dynamic linear models, see for example, Keele and Kelly (2006).

Existing software in the area of nonstationary time series predominantly focuses on estimation of either the mean or dependence structure. A comprehensive review of all such packages is outside the scope of this article – in particular any R (R Core Team 2025) package implementing a regression approach can be adapted to trend estimation in time series data by using the time index as the covariate. For example, functionality for trend estimation via penalized regression approaches or regression splines (such as in the **genlasso**, Arnold and Tibshirani 2022, and **earth**, Milborrow, Hastie, and Tibshirani 2024, packages respectively) could be adapted to serially dependent noise. For estimation of nonstationary dependence structure, the R package **LSTS** (Olea, Palma, and Rubio 2021) implements the LSF approach. Nason (2024) implements wavelet-based methods for time series in the **wavethresh** package, including the LSW model, as well as wavelet-based trend estimation but for a second-order stationary time series. The **locits** package (Nason 2025) allows for time-varying autocovariance estimation under an LSW model, whilst the **LSWPlib** library (Cardinali and Nason 2022) provides locally stationary wavelet packet representations and associated spectral estimation methods. Taylor, Park, and Eckley (2019) extend the LSW approach to multivariate time series in the package **mvLSW**. Other application-focused packages for nonstationary time series analysis include the **RSEIS** package (Lees and Harris 2024), which has functionality for spectrogram computation and visualization for seismic data. For trend detection, packages include **trend** (Pohlert 2023) and **funtimes** (Lyubchich, Gel, and Vishwakarma 2025). We mention that, for nonparametric regression without a focus on the time series setting, packages include **mgcv** (Wood 2025) and **gam** (Hastie 2025). However, to the best of our knowledge, there are few R packages available specifically designed for estimation of trend in the presence of nonstationary dependence. Examples include software for dynamic linear models in the **dynlm** package (Zeileis 2019), also available in the nonlinear time series setting in **tsDyn** (Di Narzo, Aznarte, and Stigler 2024; Stigler 2019). However, these are not designed for the locally stationary setting.

In this article we present an R implementation of the trend locally stationary wavelet (TLSW) model, proposed in McGonigle, Killick, and Nunes (2022b,a). The approach enables the practitioner to estimate both the time-varying mean and dependence structure of a univariate time series

$$X_t = T_t + \varepsilon_t, \quad 0 \le t \le n-1, \tag{1}$$

where T_t represents a smooth deterministic trend function and ε_t is a mean-zero nonstationary noise term assumed to follow an LSW model, details of which will be specified fully in Section 2.

The **TrendLSW** package implements the work of McGonigle *et al.* (2022a,b), building upon the tools provided in **wavethresh** to include a nonstationary trend component. This enables the estimation of both first- and second-order properties of a nonstationary time series within the same software package. The **TrendLSW** package (McGonigle, Killick, and Nunes 2026) is available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=TrendLSW`.

The remainder of the article is organized as follows. A brief background to wavelets and a description of the TLSW model is given in Section 2. Section 3 describes the estimation of the spectrum and its implementation in R, while Section 4 describes the estimation of the trend and implementation in R. Section 5 discusses an extended worked example, and section 6 describes two real data examples that highlight the main functionality of the **TrendLSW** package. Concluding remarks and discussion are given in Section 7.

2. The Trend-LSW model

This section describes the TLSW model of McGonigle *et al.* (2022a,b) for analysing nonstationary time series with trend components.

2.1. Wavelets

Broadly speaking, wavelets are localized, oscillatory basis functions that possess several useful properties not typically enjoyed by Fourier trigonometric basis functions. In the LSW framework (discrete) wavelets act as building blocks in an analogous fashion to Fourier exponentials in the classical Cramér representation for stationary processes and spline bases for mean representations. Let ψ be a compactly supported wavelet, for example any within the Daubechies family (Daubechies 1992). Denote by $\{h_k, g_k\}$ the low- /high- pass filter pair associated with ψ. The pair $\{h_k, g_k\}$ filter a signal into low and high frequency components respectively. Letting N_h be the number of non-zero values of $\{h_k\}$, the filters are related by the equation $g_k = (-1)^k h_{N_h-k}$ (Nason 2008, Equation (2.52)).

Setting $L_j = (2^j - 1)(N_h - 1) + 1$, the discrete wavelets at a given scale $j \in \mathbb{Z}^+$, as discussed in Nason *et al.* (2000), are defined as the vectors $\psi_j = (\psi_{j,0}, \dots, \psi_{j,L_j-1})$ of length L_j, where

$$\psi_{1,l} = \sum_k g_{l-2k}\delta_{0k} = g_l, \qquad l = 0, \dots, L_1 - 1, \tag{2}$$

$$\psi_{j+1,l} = \sum_k h_{l-2k}\psi_{j,k}, \qquad l = 0, \dots, L_j - 1, \tag{3}$$

where δ_{0k} is the Kronecker delta. The discrete father wavelet is defined similarly using the associated low-pass filter $\{h_k\}$. The simplest example of a wavelet basis is the Haar wavelet, which is given by

$$\psi^H_{j,k} = 2^{-j/2}\mathbb{I}\left(0 \leq k \leq 2^{j-1} - 1\right) - 2^{-j/2}\mathbb{I}\left(2^{j-1} \leq k \leq 2^j - 1\right),$$

where $j = \{1, 2, 3, \ldots\}$ and $k \in \mathbb{Z}$.

For a good introduction of the use of wavelets in statistics, we refer the reader to Antoniadis, Bigot, and Sapatinas (2001) and Nason (2008). Other excellent texts on wavelets include Percival and Walden (2006) and Vidakovic (2009).

2.2. The TLSW process

In this section we define the TLSW process and the key quantities of interest for analysis. A time series $\{X_{t,n}\}_{t=0}^{n-1}$ with $n = 2^J \geq 1$ for $J \in \mathbb{N}$ is said to be a trend locally stationary wavelet (TLSW) process if it admits the representation

$$X_t = T_t + \varepsilon_t = T_t + \sum_{j=1}^{\infty}\sum_{k\in\mathbb{Z}} w_{j,k;n}\psi_{j,k}(t)\xi_{j,k}, \tag{4}$$

where $\{\xi_{j,k}\}$ is a random, uncorrelated, zero-mean orthonormal increment sequence, $\{w_{j,k;n}\}$ is a set of amplitudes (parameters to estimate), and $\{\psi_{j,k}\}_{j,k}$ is a set of discrete non-decimated wavelets defined using the discrete wavelets given in Equations 2 and 3. That is, there is a time-varying trend component and the error component is a locally stationary wavelet (LSW) process. The trend component $T_t := T(t/n)$ in Equation 4 is assumed to be a Hölder continuous function. Furthermore, for each $j \geq 1$, there exists a Lipschitz continuous function $W_j(z)$ for $z \in (0, 1)$ which satisfies the following properties:

1. $\sum_{j=1}^{\infty} |W_j(z)|^2 < \infty$ uniformly in $z \in (0, 1)$.
2. The Lipschitz constants L_j are uniformly bounded in j and $\sum_{j=1}^{\infty} 2^j L_j < \infty$.
3. There exists a sequence of constants C_j such that for each n

$$\sup_k \left| w_{j,k;n} - W_j\left(\frac{k}{n}\right)\right| \leq \frac{C_j}{n},$$

   where for each $j \geq 1$ the supremum is over $k = 0, \ldots, n-1$ and the sequence $\{C_j\}$ satisfies $\sum_{j=1}^{\infty} C_j < \infty$.

The model imposes the standard assumptions on the LSW component in the literature, allowing for locally stationary dependence structure. Model 4 also permits nonstationary first-order behavior by incorporating a smooth mean function T. Whilst the formal definition requires $n = 2^J$, in practice the boundary handling method described in McGonigle *et al.* (2022b) can be used to analyze time series of arbitrary length. Note that there is explicit dependence on n in the model, but this notation is suppressed for clarity of presentation.

As with classical time series theory, the key quantity of interest for characterising dependence structure is the spectrum. The evolutionary wavelet spectrum (EWS) of an LSW process is defined as $S_j(z) = W_j(z)^2$ for rescaled time $z = k/n \in (0, 1)$ and measures the contribution to variance at a particular rescaled time z and scale j. The local autocovariance (LACV) function for a LSW process provides information about the covariance at a rescaled location

$z = k/n \in (0,1)$. The LACV, $c(z,\tau)$, of a LSW process with EWS $\{S_j(z)\}$ is defined as $c(z,\tau) = \sum_{j=1}^{\infty} S_j(z)\Psi_j(\tau)$, for $\tau \in \mathbb{Z}$, $z \in (0,1)$. The LACV is a decomposition of the autocovariance of a process over scales and rescaled time locations, and converges to the process autocovariance $c_n(z,\tau) = \mathbb{E}(\varepsilon_{\lfloor zn \rfloor}\varepsilon_{\lfloor zn \rfloor+\tau})$ with increasing sample size n.

Simulating realizations of a TLSW process

We can simulate an example realization of a TLSW process with a specified trend and given EWS using `TLSWsim()`.

The input arguments for `TLSWsim()` are:

- `trend`: The trend component of the time series, which can be either a numeric vector of length n, or a function of a single argument on rescaled time $(0,1)$.
- `spec`: The spectrum that defines the dependence of the time series. This can be either a numeric $J \times n$ matrix, or a list of length J with each element given by a function of a single argument on rescaled time $(0,1)$ (scales where $S_j(z) = 0$ for all z can be represented by the `NULL` value). For readers familiar with the **wavethresh** package, this argument can also be given as an object of class `wd`.
- `filter.number`: This specifies the number of vanishing moments of the wavelet in the TLSW model 4 i.e., selects the smoothness of wavelet that you want to use in the decomposition. By default this is 4.
- `family`: Specifies the family of wavelets that you want to use in the TLSW model. The options are `"DaubExPhase"` (default) and `"DaubLeAsymm"`.
- `innov.func`: The function used to generate the innovation sequence $\{\xi_{j,k}\}$. By default, this is `rnorm()`.

We note that it is also possible to simulate TLSW processes of non-dyadic length n, provided that the `trend` and `spec` arguments are provided as a numeric vector of length n and numeric matrix of dimensions $\lfloor \log_2(n) \rfloor \times n$ respectively. The recommended choices for the `family` argument are `"DaubExPhase"` or `"DaubLeAsymm"`, corresponding to the Daubechies Extremal Phase and Daubechies Least Asymmetric wavelet families respectively. For these families, the `filter.number` argument can be an integer between $1-10$ and $4-10$ respectively, representing the number of vanishing moments of the wavelet (the Haar wavelet is the Extremal Phase wavelet with 1 vanishing moment).

The example below generates a time series of length $n = 512$ where we impose a cubic trend $T_t = 3(32(t/n)^3 - 48(t/n)^2 + 22(t/n) - 3)$ and an EWS with a quadratically increasing and decreasing power at level $j = 2$, as well as constant power at level $j = 4$ i.e., $S_2(t/n) = 2 + 12(t/n) - 12(t/n)^2$ and $S_4(t/n) = 2$. The specification of the trend is given as a function, and the EWS is defined using a list of functions:

```
R> n1 <- 512
R> trend1 <- function(z) { 3 * (32 * z^3 - 48 * z^2 + 22 * z - 3) }
R> spec1 <- vector(mode = "list", length = log2(n1))
R> spec1[[2]] <- function(z) { 2 + 12 * z - 12 * z^2 }
R> spec1[[4]] <- function(z) { 2 }
```

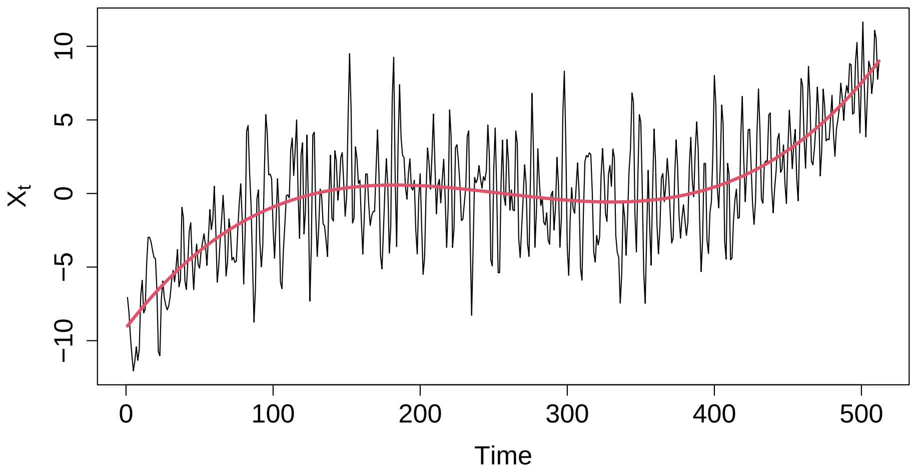

Figure 1: Example TLSW process generated with cubic trend function and spectrum with power at scales 2 and 4. Solid red line shows the true underlying trend function.

```
R> set.seed(123)
R> x1 <- TLSWsim(trend = trend1, spec = spec1, filter.number = 4)
```

Figure 1 shows the resulting simulated time series from the TLSW model, where time-varying behavior in the trend and variability can be seen.

Estimation

Given an observed time series which we model using (4), we are interested in estimating the following three quantities:

1. The spectrum (EWS) $\{S_j(t/n)\}_{j=1}^{J_0}$ for all time locations $t = 0, \ldots, n-1$, with maximum scale of interest $J_0 \leq J$,

2. The autocovariance (LACV) $c(t/n, \tau)$ for some τ of interest and $t = 0, \ldots, n-1$ (and associated local autocorrelation $c(t/n, \tau)/c(t/n, 0)$),

3. The trend T_t for $t = 0, \ldots, n-1$.

We consider the estimation of these quantities in the following sections.

The main function within the package is `TLSW()`, which allows the user to simultaneously estimate both the trend and spectrum of a TLSW process. The package provides two separate, but related, approaches to estimate the trend component of the process. Similarly, there are two approaches implemented for estimating the EWS, each aimed to be used with the corresponding trend estimation.

In Section 3, we describe the two ways in which EWS estimation can be performed, and in Section 4 we describe the methods for trend estimation. In Section 5 we provide worked examples highlighting how these approaches can be applied together using `TLSW()`.

3. EWS estimation

In what follows, we discuss two approaches for estimating the EWS in the presence of trend. Analogously to classical Fourier methods, we base our estimation procedure on the wavelet periodogram, with some modifications to account for the presence of the trend function. Depending on the properties of the trend function, we give two related but different strategies for estimating the EWS. The general approach for both estimation methods consists of three main steps:

1. Calculating the wavelet periodogram using a nondecimated wavelet transform.
2. Smoothing the wavelet periodogram.
3. Obtaining the spectral estimate by bias correction of the wavelet periodogram.

Below, for each of the two methods, we describe these three steps in more concrete detail. The key distinction between the two approaches is that the first directly estimates the EWS accounting for the trend within the estimation. The second approach differences the data and estimates the EWS within the differenced data, then corrects for the effect of the differencing. We first describe the direct estimation approach before detailing the differencing approach.

3.1. Direct estimation

To construct the wavelet periodogram, we calculate the nondecimated wavelet coefficients of the time series $\{X_t\}_{t=0}^{n-1}$. For a given location k and scale j, the wavelet coefficient $d_{j,k}$ is defined as $d_{j,k} = \sum_t X_t \psi_{j,k}(t)$. The periodogram is then constructed as $I_{j,k} = d_{j,k}^2$.

If the trend $T(t/n)$ is smooth and does not display irregularities such as cusps or discontinuities, then the wavelet transform coefficients $d_{j,k}$ will be largely free from the effects of the trend. This is due to fact that wavelets naturally act as differencing operators over different time-scales. As mentioned in Section 2.2, wavelets within both the Daubechies Extremal Phase and Least Asymmetric families are constructed to have a given number of vanishing moments. For a wavelet with m vanishing moments, all wavelet coefficients of a polynomial trend function with degree at most $m-1$ will be zero. Therefore, provided the trend function is well-approximated by a polynomial, this useful property of wavelets ensures that the wavelet coefficients, and therefore the wavelet periodogram, are not contaminated by the trend function.

Let the operator $A = (A_{jl})_{j,l>0}$ be given by $A_{jl} = \sum_\tau \Psi_j(\tau)\Psi_l(\tau)$, where the autocorrelation wavelets $\Psi_j(\cdot)$ are defined by $\Psi_j(\tau) := \sum_{k\in\mathbb{Z}} \psi_{j,k}(0)\psi_{j,k}(\tau)$, for $j > 0, \tau \in \mathbb{Z}$. Assuming we use a wavelet with m vanishing moments and the trend function is polynomial of degree at most $m-1$, then, mirroring a result from Nason *et al.* (2000), McGonigle *et al.* (2022b) show that the expectation and variance of the wavelet periodogram are given by

$$\mathbb{E}\left(I_{j,k}\right) \approx \sum_l A_{jl} S_l(k/n), \tag{5}$$

$$\mathrm{Var}(I_{j,k}) \approx 2\left(\sum_l A_{jl} S_l(k/n)\right)^2, \tag{6}$$

where '≈' denotes asymptotic equivalence with increasing sample size n. This expression also holds for Hölder continuous trend functions but with a larger finite sample bias than polynomial trends. Equations 5 and 6 show that the wavelet periodogram is a biased, inconsistent estimator of the EWS. Therefore, the periodogram is first smoothed, then bias corrected, in order to yield a consistent estimator, which motivates steps 2 and 3 above.

The standard approach to smoothing (step 2) is to use some form of linear kernel smoothing. Here we explicitly describe smoothing using a running mean, but note that the **TrendLSW** package also implements median smoothing and Epanechnikov kernel smoothing. For bin width size $2N+1$, the smoothed wavelet periodogram is given by

$$\widehat{I}_{j,k} = \frac{1}{2N+1} \sum_{m=-N}^{N} I_{j,k+m}. \tag{7}$$

Step 3 then bias corrects the smoothed periodogram. The resulting estimator is given by

$$\widehat{S}_j(k/n) = \sum_{l=1}^{J} A_{lj}^{-1} \widehat{I}_{l,k}, \tag{8}$$

and the LACV is estimated using $\widehat{S}_j(k/n)$ with the equation $\hat{c}(k/n,\tau) = \sum_{j=-J}^{-1} \widehat{S}_j(k/n)\Psi_j(\tau)$. The standard approach works well enough in zero-mean cases. However, the caveat of the above discussion is the occurrence of well-known boundary effects; near the end-points of the time series, where the wavelet filter is of large enough size, wavelet coefficients are computed using reflections of the data, resulting in artefacts. The coarser the scale j, the larger the wavelet filter, and therefore the more pronounced these boundary effects are. These are more problematic in datasets containing non-zero trends. In a practical setting, if the trend is not exactly polynomial, which is often the case in reality, the wavelet coefficients of the trend will not be exactly zero and will be increasingly contaminated at larger scales due to the larger filter length. To mitigate this problem we propose two simple solutions.

The first step involves changing the way that the wavelet transform is performed to improve performance at the boundaries (step 1). Briefly, we use a modification of reflective boundary handling that projects the trend smoothly at the boundaries. This helps to minimize boundary problems and can also be applied to time series of arbitrary length, removing the restrictive assumption in the **wavethresh** package that the time series length must be dyadic. For full details, see Section 4.4 of McGonigle *et al.* (2022b).

The second mitigation strategy involves discarding some of the coarsest wavelet scales used in the periodogram bias correction (step 3 above). Instead of using all J available scales as is customary in LSW modeling (see e.g., Nason *et al.* 2000), we instead use $J_0 < J$ scales. This can be thought of as a type of tapered estimator. Using $J_0 < J$ scales improves practical performance of the estimators in the presence of trends, and reduces the variability of the resulting estimator by discarding wavelet coefficients at coarser scales, which become increasingly autocorrelated. McGonigle *et al.* (2022b) show the consistency of these estimators and demonstrate the superiority over the standard estimators of Nason *et al.* (2000) in the presence of trends.

Thus the final TLSW estimator is given by

$$\widehat{S}_j(k/n) = \sum_{l=1}^{J_0} A_{lj}^{-1} \widehat{I}_{l,k}, \tag{9}$$

and the LACV is estimated using $\widehat{S}_j(k/n)$ with the equation $\hat{c}(k/n,\tau) = \sum_{j=1}^{J_0} \widehat{S}_j(k/n)\Psi_j(\tau)$.

3.2. First-differenced estimation with modified correction factor

The first approach described above works well when the trend function does not display cusps. However, in the presence of such irregularities, the wavelet transform may fail to remove the trend function, causing the EWS estimator to be biased. In particular, wavelet coefficients at coarser scales will suffer from increased bias due to the increased filter length. Furthermore, since the periodogram must be corrected across scales, the bias at coarser scales can appear at finer scales.

To solve this problem, we can use the commonly used practice of differencing. Before applying the wavelet transform, we first difference the time series to yield $\{\Delta X_t = X_{t+1} - X_t\}_{t=1}^{n-1}$. Differencing allows us to remove the trend immediately, ensuring that the wavelet transform does not accumulate large bias across increasing scales. Then, the wavelet coefficients $\tilde{d}_{j,k} = \sum_t \Delta X_t \psi_{j,k}(t)$ are computed on the differenced time series, from which the wavelet periodogram $\tilde{I}_{j,k} = \tilde{d}^2_{j,k}$ is constructed. It is possible to use higher order differencing, e.g. second differencing, however we advise against this due to the inherent loss of information and poorer empirical performance as demonstrated in McGonigle *et al.* (2022a).

Estimation then proceeds as before, with one key distinction. Due to the differencing transform, the dependence structure of the time series is changed. Therefore, the bias in the wavelet periodogram is no longer characterized by the matrix $A = (A_{jl})_{j,l>0}$, but instead a related matrix $D^1 = (D^1_{jl})_{j,l>0}$. The entries of the modified bias matrix are given by $D^1_{jl} = A_{jl} - A^1_{jl}$ where $A^1_{jl} = \sum_\tau \Psi_j(\tau)\Psi_l(\tau - 1)$, and we have that

$$\mathbb{E}(\tilde{I}^j_k) \approx \sum_l D^1_{jl} S_l \left(\frac{k}{n}\right).$$

Therefore, we can proceed similarly to before, first smoothing the periodogram by for example using a running mean:

$$\widehat{\tilde{I}}_{l,k} = \frac{1}{2N+1} \sum_{m=-N}^{N} \tilde{I}_{j,k+m},$$

and then correcting using the matrix $(D^1)^{-1}$, giving the estimator

$$\widehat{S}_j(k/n) = \sum_{l=1}^{J_0} (D^1_{lj})^{-1} \widehat{\tilde{I}}_{l,k},$$

where, as seen with the direct estimation approach, we opt to use a tapered estimator with maximum scale of interest $J_0 < J$. Analogously, for a more complex trend component, second differences can also be used. If the time series is known to have a seasonal trend of period p, then the spectrum can be instead estimated using the wavelet periodogram of the lag p differenced time series:

$$\widehat{S}_j(k/n) = \sum_{l=1}^{J_0} (D^p_{lj})^{-1} \widehat{\tilde{I}}_{l,k}, \tag{10}$$

where $D^p_{jl} = A_{jl} - A^p_{jl}$ with $A^p_{jl} = \sum_\tau \Psi_j(\tau)\Psi_l(\tau - p)$. For more details, we refer the reader to McGonigle *et al.* (2022a). The local autocovariance can be estimated using this spectrum estimate in the same manner as the direct estimator, with $\hat{c}(k/n, \tau) = \sum_{j=1}^{J_0} \widehat{S}_j(k/n)\Psi_j(\tau)$.

3.3. EWS estimation using TLSW

In the **TrendLSW** package, the EWS estimation is performed as part of `TLSW()`, which will, by default, directly estimate the trend function and spectrum simultaneously. Here, we describe how the spectral estimation component of `TLSW()` is carried out.

To indicate to `TLSW()` that the user wishes to compute the spectral estimate, the argument `do.spec.est` should be set to `TRUE` (default). Then, the essential arguments to perform spectral estimation in `TLSW()` are:

- `x`: The input time series.
- `S.filter.number`: The filter number of the wavelet used to estimate the EWS. By default this is equal to the `T.filter.number` argument used in trend estimation (see Section 4), which by default is 4. Larger values are smoother, `"DaubExPhase"` can take values 1-10 and `"DaubLeAsymm"` values 4-10.
- `S.family`: The family of the wavelet used to estimate the EWS. By default this is equal to the `T.family` argument used in trend estimation, which by default is `"DaubExPhase"`; `"DaubLeAsymm"` is an alternative.
- `S.smooth`: Logical indicating whether to smooth, default `TRUE`.
- `S.smooth.type`: The type of smoothing to be used on the wavelet periodogram. By default this is `"mean"` for running mean smoothing, but it can also be `"median"` for running median, or `"epan"` for Epanechnikov kernel smoothing.
- `S.binwidth`: The bin width parameter in the wavelet periodogram smoother; this was $2N + 1$ in Equation 7. By default this is $\lfloor 6\sqrt{n} \rfloor$.
- `S.max.scale`: The maximum wavelet scale to be analyzed, J_0 in Equation 9. By default this is $\lfloor 0.7 \log_2 n \rfloor$.
- `S.boundary.handle`: A logical variable, with default value `TRUE`, indicating whether the boundary handling procedure is used.
- `S.inv.mat`: The user can precalculate and supply a correction matrix to correct the raw wavelet periodogram. If left blank, then the correction matrix is calculated when performing spectral estimation.
- `S.do.diff`: A logical variable, to indicate whether the time series should be differenced first before the wavelet periodogram is calculated. If `TRUE`, then the differencing approach is carried out from Section 3.2, otherwise the direct estimation approach is used from Section 3.1. By default this is `FALSE`.
- `S.lag`: If `S.do.diff = TRUE`, the lag of the differencing operator to apply. The default choice is 1, whilst larger values could be used if there is seasonality at a given lag (e.g., lag 12 could be used in monthly data).
- `S.diff.number`: The number of times differencing is applied to the time series if the argument `S.do.diff = TRUE`. The strongly recommended default is 1 to perform a first difference, but can be set to 2 to perform second differencing as opposed to lag 2 differencing.

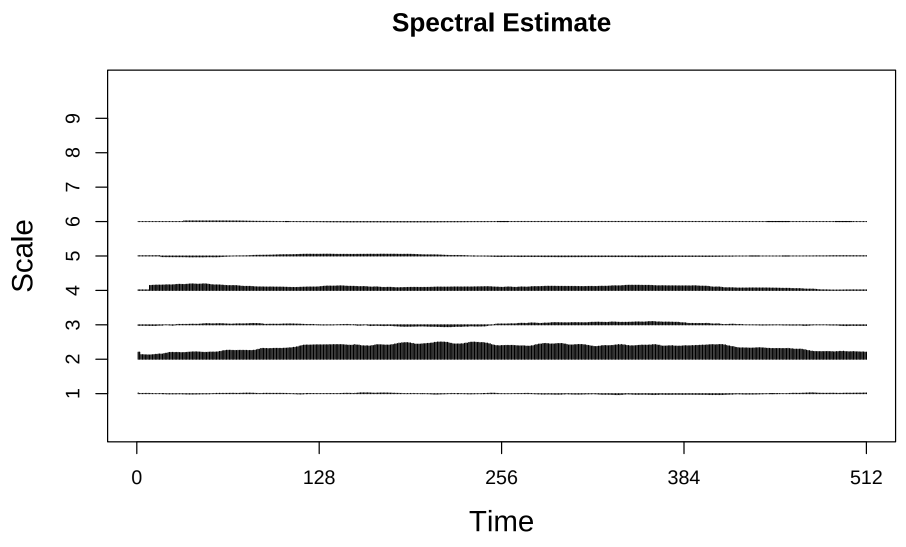

Figure 2: Spectrum estimate for the TLSW process `x1` given by the (default) direct estimator using `TLSW()`.

All arguments related to spectral estimation begin with the prefix `S`. By default, no differencing will be performed, so that the direct spectral estimation method from Section 3.1 is performed. Default values for wavelet choice, maximum scale J_0, and bin width, have been set to align with numerical studies carried out in McGonigle *et al.* (2022a), McGonigle *et al.* (2022b), and widely accepted in the literature. All other arguments necessary for estimating the spectrum have been given default values based on extensive numerical studies. Therefore, basic estimation of the spectrum of the time series `x1` without manually adjusting tuning parameters can be performed by supplying `x1` to `TLSW()` as follows:

```
R> x1.TLSW <- TLSW(x1)
```

`TLSW()` returns an object with S3 class `TLSW`; a list which contains several of the quantities also returned by `ewspec3()` from the `locits` R package, as well as additional input parameters. Outputs associated to the spectrum estimation are stored in the `spec.est` element of the `TLSW` object, itself a list. In this list, the corrected, smoothed spectral estimate is stored in the `S` component, whilst the unsmoothed and smoothed wavelet periodograms are stored in the `WavPer` and `SmoothWavPer` components respectively. As in the **wavethresh** package, these three components are of class `wd` and can be plotted using the plotting functionality therein, as well as being amenable to `print()` or `summary()`.

A `TLSW` object can be plotted using `plot()` as standard: this can plot spectrum and/or trend estimates with a number of options available to the user. For now, we will focus on the produced spectrum plot, and return to more details of plot functionality in a worked example in Section 5. Using `plot()`, we supply the `TLSW` object, and for now we specify the argument `plot.type = "spec"` to only plot the estimated spectrum:

```
R> plot(x1.TLSW, plot.type = "spec")
```

The resulting spectrum estimate is shown in Figure 2. The general features of the spectrum are well-represented in the estimate: scales with no power are generally estimated close to zero,

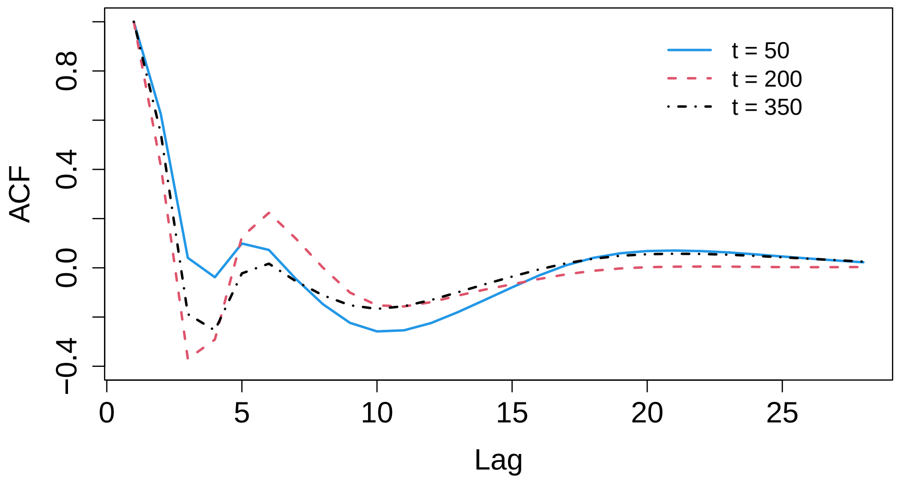

Figure 3: Local autocorrelation estimate for the TLSW process `x1` at three time points $t = 50$ (blue solid line), $t = 200$ (red dashed line), and $t = 350$ (black dash-dotted line).

whilst scale 4 appears to have (near) constant power over time. The increasing and decreasing behavior at scale 2 is also quite clear. Note that nonstationary spectrum estimation is an extremely challenging problem, due to factors including strong autocorrelation and low signal-to-noise ratio. Therefore, spectral estimates are not necessarily expected to be as accurate (as compared to the estimate of the trend function in Section 4).

The local autocovariance and autocorrelation of the times series can be computed from the spectral estimate using `TLSWlacf()`. The user can set a maximum lag for calculating the autocovariance in the same way as in `stats::acf()`, by specifying a value for the `lag.max` argument. `TLSWlacf()` takes as input a `TLSW` object produced from `TLSW()`, and outputs a list object of class `lacf`, which is identical to the output of `lacf()` in the `locits` package. The key outputs are `lacv` and `lacf`, which correspond to the estimated local autocovariance and autocorrelation respectively. These are given in matrix form with rows representing time and columns representing lags. We plot the autocorrelation estimate at time points $t = 50, 200, 350$ as an illustration in Figure 3.

```
R> x1.lacf <- TLSWlacf(x1.TLSW)
R> plot(x1.lacf$lacr[50, ], ylim = c(-0.4, 1), type = "l", xlab = "Lag",
+    ylab = "ACF", lwd = 2, col = 4)
R> lines(x1.lacf$lacr[200, ], col = 2, lwd = 2, lty = 2)
R> lines(x1.lacf$lacr[350, ], col = 1, lwd = 2, lty = 4)
R> legend(20, 1, c("t = 10", "t = 230", "t = 450"), lty = c(1, 2, 4),
+    lwd = c(2, 2, 2), col = c(4, 2, 1), bty = "n", cex = 1.2)
```

4. Trend estimation

We now turn to describe two procedures for trend estimation in the presence of time-varying dependence structure. The general approach for trend estimation again consists of three main steps:

1. Calculating the wavelet coefficients via a wavelet transform.
2. Thresholding the wavelet coefficients.
3. Obtaining the trend estimate by inverse transforming the thresholded wavelet coefficients.

Here, the distinction between the two approaches lies in how the wavelet thresholding is performed. Mirroring Section 3, the following sections discuss in detail how the two methods are implemented before exploring the package structure.

4.1. Linear wavelet thresholding estimator

In the case where the the trend function is sufficiently smooth, and the EWS was estimated using the direct approach described in Section 3.1, then we recommend to estimate the trend function using a linear wavelet thresholding estimator in the spirit of e.g., Craigmile, Guttorp, and Percival (2004). Observe that, when the trend is exactly a polynomial, using a wavelet with high enough vanishing moments will result in zero wavelet coefficients apart from at the boundaries; the zero mean assumption for the the LSW increment process in (4) means that the wavelet coefficients have expectation zero. Thus the boundary wavelet coefficients and the scaling coefficients will be the only coefficients to meaningfully contribute to the trend component. Likewise, if the trend is closely approximated by a polynomial, the trend component is well-represented by only the boundary and scaling coefficients.

We hence proceed as follows. First we perform a discrete wavelet transform (DWT) of the time series with a given choice of wavelet ψ and a prespecified coarsest scale J_0, similar to the spectral estimation procedure. Setting the non-boundary coefficients to zero will reflect the above observation on these coefficients being zero mean, in effect performing hard thresholding of the non-boundary coefficients, so that performing the inverse discrete wavelet transform will obtain a trend estimate. McGonigle *et al.* (2022b) show that this procedure results in an unbiased and mean square consistent estimator of the trend when the true underlying trend is polynomial or Hölder continuous. Experiments have shown that in practice, setting a coarsest scale of $J_0 = \lfloor 0.7 \log_2(T) \rfloor$ is robust to a wide range of process scenarios.

To obtain robust estimation of the trend at the boundary of the series, McGonigle *et al.* (2022b) suggest using a procedure akin to classical boundary handling in wavelet methods, in which a long series of length $3n$ is constructed by reflecting the time series data so that the original series is at the centre of the newly constructed data. Performing trend estimation on this longer series mitigates the fact that, particularly at coarser scales, the wavelet filter can be longer than the available data near the boundary. Note that a nondecimated discrete wavelet transform (NDWT) can be performed instead of the DWT to represent more temporally-localized behavior in the series, in which case basis averaging is used to obtain the trend estimate after hard thresholding, see e.g., Nason *et al.* (2000, Chapter 3.12) for more details. The key difference between the DWT and NDWT is that DWT forms an orthogonal basis whereas the NDWT is an overcomplete basis representation. The DWT downsamples the data at each step of the algorithm to obtain a sparse (efficient) representation of the underlying signal. The disadvantage of this is that if the data is shifted by a single place to the left or right, the DWT representation can change dramatically (not just shifted one to the left/right). In contrast, the NWDT is translation-invariant as it considers all potential paths of shifting and downsampling the data at each step of the algorithm and does not suffer from this drawback.

When the DWT is used, pointwise $(1-\alpha)\%$ confidence intervals for the trend can be derived under the assumption of normality, via

$$\widehat{T}_t \pm q_{1-\frac{\alpha}{2}} \sqrt{\mathrm{Var}\left(\widehat{T}_t\right)}, \tag{11}$$

where the variance term can be estimated using the (consistent) estimate of the local autocovariance $\hat{c}(k/n, \tau) = \sum_{j=-J_0}^{-1} \widehat{S}_j(k/n)\Psi_j(\tau)$.

When the NDWT is used, pointwise confidence intervals are computed via the bootstrap, in a similar fashion to the approach of Friedrich, Smeekes, and Urbain (2020). For some chosen total number of bootstrap replications, B, bootstrapped replications $\{X_t^{(b)}\}_{t=0}^{n-1}$ for $b = 1, \dots, B$ can be simulated using the spectral estimate and trend estimate, from which bootstrapped trend estimates can be calculated.

Concretely, using the spectral estimate $\{\widehat{S}_j(k/n)\}_{j=1}^{J_0}$, for $k = 0, \dots, n-1$, and the trend estimate $\widehat{T}_t$, bootstrapped replications

$$X_t^{(b)} = \widehat{T}_t + \sum_j \sum_k \widehat{w}_{j,k}\psi_{j,t-k}\xi_{j,k}^{(b)}$$

are generated, where $\widehat{w}_{j,k} = \widehat{S}_j(k/n)^{1/2}$, and $\xi_{j,k}^{(b)}$ are independent, identically distributed standard Normal random variables. For each of the B bootstrapped replications, a bootstrapped trend estimate $\{\widehat{T}_t^{(b)}\}_{t=0}^{n-1}$ is obtained using the linear wavelet thresholding procedure as described above. Then, for each time point $t = 0, \dots, n-1$, pointwise $(1-\alpha)\%$ confidence intervals can be calculated using either:

1. A normal approximation $\widehat{T}_t \pm q_{1-\frac{\alpha}{2}} \sqrt{\mathrm{Var}(\widehat{T}_t)}$, where the variance term is the sample variance of the B bootstrapped trend estimates at time t.

2. The empirical $\frac{\alpha}{2}\%$ and $(1-\frac{\alpha}{2})\%$ quantiles of the B bootstrapped trend estimates.

4.2. Nonlinear wavelet thresholding estimator

In the case where the EWS is estimated using the differencing approach of McGonigle *et al.* (2022a) described in Section 2.2, then we recommend to estimate the trend function using a nonlinear wavelet thresholding estimator, using the spectral estimate $\widehat{S}_j(k/n)$ from Equation 10. More specifically, we employ classical wavelet thresholding to provide a trend estimate. For a particular wavelet basis used for thresholding, $\{\psi_{j,k}^1\}_{j,k}$, we first compute the corresponding DWT wavelet coefficients of the series X_t, $d_{r,s}^1 = \sum_t X_t \psi_{j,k}^1(t)$, where we use the notation $d_{r,s}^1$ to differentiate from the wavelet coefficients corresponding to the generating wavelet in model 1. The coefficients are then thresholded using a coefficient-specific hard or soft threshold

$$\begin{aligned} \hat{v}_{r,s}^S &= \mathrm{sgn}(d_{r,s}^1)(|d_{r,s}^1| - \lambda_{r,s,n})\mathbb{I}(|d_{r,s}^1| > \lambda_{r,s,n}) && \text{(soft thresholding)} \\ \hat{v}_{r,s}^H &= d_{r,s}^1 \mathbb{I}(|d_{r,s}^1| > \lambda_{r,s,n}), && \text{(hard thresholding)} \end{aligned}$$

where $\lambda(r, s, n) = \hat{\sigma}_{r,s}\sqrt{2\log(n)}$ and $\hat{\sigma}_{r,s}$ is an estimate of the standard deviation of the coefficient $d_{r,s}^1$. This estimate can be obtained from the expression for the variance of the wavelet coefficients

$$\mathrm{Var}(d^1_{r,s}) \approx \sum_l C^{1,0}_{r,l} S_l(s/n), \tag{12}$$

where $C^{1,0}_{r,l} = \sum_\tau \Psi^0_r(\tau)\Psi^1_l(\tau)$ is computed using the autocorrelation wavelets $\Psi^0_r(\tau)$ and $\Psi^1_l(\tau)$ corresponding to the generating wavelet ψ and thresholding wavelet ψ^1 respectively (McGonigle *et al.* 2022a). The estimate and therefore the coefficient-dependent threshold $\lambda(r, s, n)$ is computed by plugging in the spectral estimate $\widehat{S}_j(k/n)$ from Equation 10 into Equation 12; a trend estimate $\widehat{T}_t$ can then be obtained by performing the inverse DWT on the coefficients $\{\hat{v}_{r,s}\}_{r,s}$.

As in the linear wavelet thresholding approach, a prespecified coarsest scale J_0 is set for the wavelet decomposition. Stronger estimation performance is also achieved using so-called translation-invariant (TI) denoising, in which the thresholding procedure described here is performed on the coefficients resulting from an NDWT on the data, followed by basis averaging; see von Sachs and MacGibbon (2000) or McGonigle *et al.* (2022a). Boundary handling can also be performed, in a similar manner to that described for the linear wavelet estimator. Lastly, pointwise $(1 - \alpha)\%$ confidence intervals can be construted via bootstrapping in an analogous way to that described for the linear wavelet estimator.

4.3. Trend estimation using TLSW

Both trend estimation procedures described above can be performed using `TLSW()`, setting the `do.trend.est` argument to `TRUE` (default). By default, the function will use the linear wavelet estimator (paired with direct spectrum estimation), but the nonlinear (translation-invariant) wavelet-based estimation can be chosen by setting the `T.est.type` argument to `"nonlinear"`.

For both linear and nonlinear wavelet trend estimation, the following arguments can be specified by the user:

- `T.filter.number`: The filter number of the wavelet used to estimate the trend. By default this is 4.
- `T.family`: The family of the wavelet used to estimate the trend. By default this is `"DaubExPhase"`.
- `T.transform`: The type of wavelet transform used in the thresholding estimator, either `"dec"` (DWT) or `"nondec"` (NDWT, default).
- `T.boundary.handle`: A logical variable, with default value `TRUE`, indicating whether the reflection-based boundary handling procedure should be performed.
- `T.max.scale`: The maximum wavelet scale to be used in the wavelet transform, J_0. By default this is $\lfloor 0.7 \log_2 n \rfloor$.
- `T.CI`: A logical variable, to indicate whether a pointwise confidence interval for the trend should be computed. By default this is `FALSE`.
- `T.sig.lvl`: A significance level, α for the constructed confidence interval, defaulting to `0.05`.

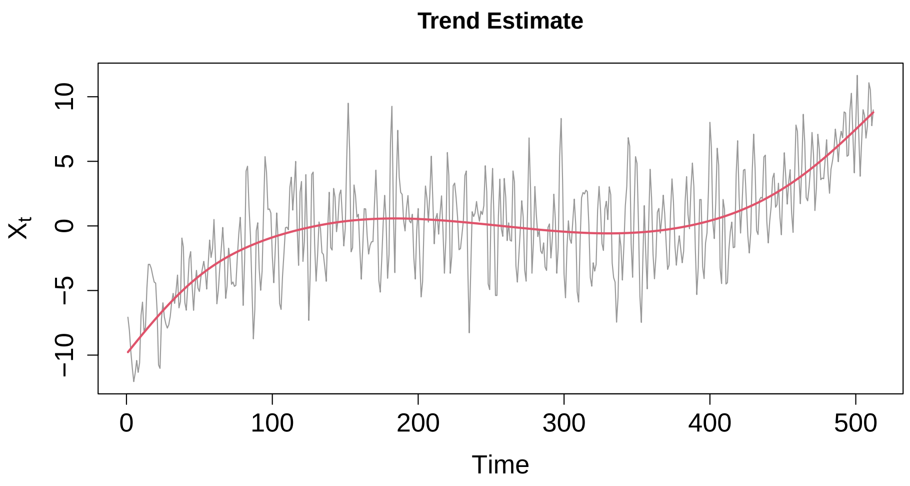

Figure 4: Trend estimate (red) for the TLSW process `x1` given by the linear wavelet estimator described in the text using `TLSW()`. Underlying data given by gray line.

- `T.reps`: The total number of bootstraps, B, used in the construction of pointwise confidence intervals, if `T.CI` is `TRUE`, defaulting to $B = 200$.
- `T.CI.type`: The type of constructed confidence intervals, if `T.CI` is `TRUE`. By default this is `"normal`, which uses the normal approximation quantiles to construct the interval, else `"percentile"` for which empirical quantiles are used in the construction.
- `T.lacf.max.lag`: The maximum lag used for calculating the lacf, default is $\lfloor 10 \log n \rfloor$,

For the nonlinear trend estimation, as well as the arguments above, the user can specify the following additional arguments associated to how the wavelet thresholding is performed:

- `T.thresh.type`: The type of thresholding used in the nonlinear quantiles trend estimation procedure, either `"hard"` (default) or `"soft"`.
- `T.thresh.normal`: A logical variable, indicating whether to use a larger threshold, useful to disable for non-Normally distributed data. Defaults to `TRUE`, in which case $\lambda(r, s, n) = \hat{\sigma}_{r,s}\sqrt{2\log(n)}$ is used, if `FALSE`, then $\lambda(r, s, n) = \hat{\sigma}_{r,s}\log(n)$ is used.

In a similar fashion to that described in Section 3, we can use `plot()` on the `x1.TLSW` object to plot the trend estimate, by supplying the argument `plot.type = "trend"` only the trend is plotted:

```
R> plot(x1.TLSW, plot.type = "trend")
```

The resulting plot is shown in Figure 4, which shows close alignment between the estimated and true trend function.

5. Trend and EWS worked example

Having described the main functionality of the package, in this section we give an extended worked example combining both trend and EWS estimation. We simulate a realization of a TLSW process with trend function given by:

$$T_t = \begin{cases} 5\sin(6\pi t/n) + 10t/300, & 0 \le t \le 300, \\ 5\sin(6\pi t/n) - 4 - 14t/(n-300) - 14n/(300-n), & 301 \le t < n. \end{cases}$$

i.e., a composition of a sinusoidal component, and a piecewise linear trend starting at 0, increasing to 10 at time point 300, then decreasing to -4 at time point n. The spectrum is given by:

$$S_j(t/n) = \begin{cases} 8t/n + 2, & j = 1, \quad 0 \le t < n, \\ 1, & j = 3, \quad 0 \le t \le 200 \\ (5/200)t - 4, & j = 3, \quad 200 \le t \le 400 \\ -(5/200)t + 16, & j = 3, \quad 400 \le t < 600 \\ 1, & j = 3, \quad 600 \le t < n \\ 2 + 4\sin(4\pi t/n)^2, & j = 5, \quad 0 \le t < n. \end{cases}$$

i.e., a linearly increasing power at scale $j = 1$, piecewise linear with peak in power at time point 400 at scale 3, and sinusoidal power at scale $j = 5$. This time, we specify the trend function in R using a numeric vector, and specify the spectrum using a matrix of numeric values, as follows:

```
R> n2 <- 1024
R> index2 <- seq(from = 0, to = 1, length = n2)
R> trend2 <- 5 * sin(pi * 6 * index2) +
+    c(seq(from = 0, to = 10, length = 300),
+      seq(from = 10, to = -4, length = 724))
R> spec2 <- matrix(0, nrow = log2(n2), ncol = n2)
R> spec2[1,] <- seq(from = 2, to = 10, length = n2)
R> spec2[3,] <- c(rep(1, 200), seq(from = 1, to = 6, length = 200),
+    seq(from = 6, to = 1, length = 200), rep(1, 424))
R> spec2[5,] <- 2 + 4 * sin(4 * pi * index2)^2
```

As before, we simulate the TLSW process via `TLSWsim()`:

```
R> set.seed(1234)
R> x2 <- TLSWsim(trend = trend2, spec = spec2)
```

In Figure 5 we plot `x2`, along with the true underlying trend function and spectrum.

For estimating the trend and spectrum of this TLSW process, we make some modifications to the default arguments of `TLSW()` to highlight some potential ways a user can customize the estimation procedure. The following code snippet fits the TLSW model to `x2` shown in Figure 5:

```
R> set.seed(10)
```

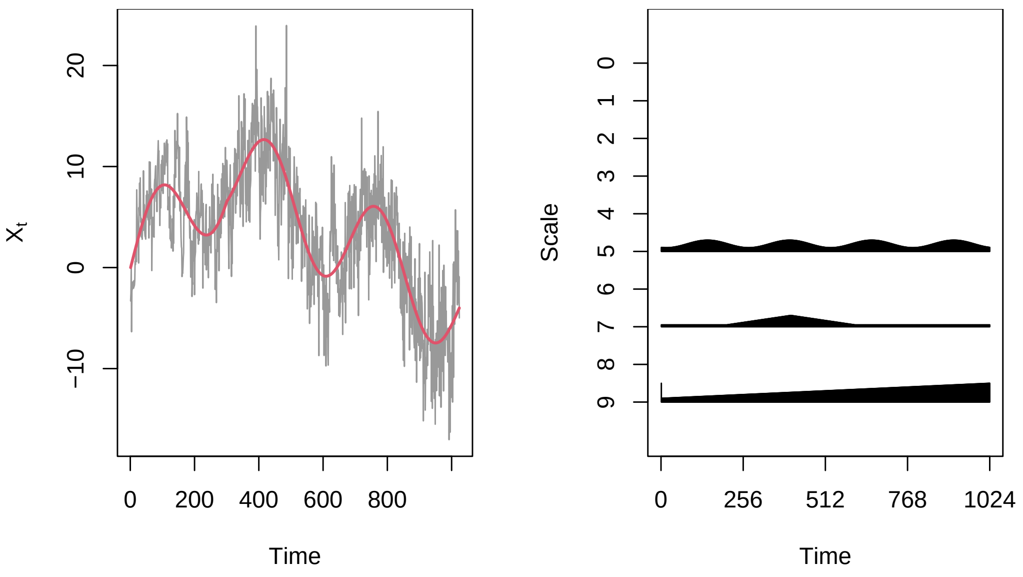

Figure 5: Left: Plot of an example realization of the TLSW process `x2`, with true trend line given in red solid line. Right: True underlying spectrum.

```
R> x2.TLSW <- TLSW(x2, T.filter.number = 6, T.family = "DaubLeAsymm",
+    T.est.type = "nonlinear", T.CI = TRUE,  T.reps = 500,
+    S.filter.number = 4, S.family = "DaubExPhase", S.do.diff = TRUE,
+    S.smooth.type = "median", S.binwidth = 128)
```

In this example, we perform trend estimation using nonlinear thresholding, and use the Daubechies Least Asymmetric wavelet with 6 vanishing moments for trend estimation. The choice of wavelet is an important consideration in wavelet analysis, and is akin to selecting the kernel in nonparametric modeling. It is recommended to check the robustness of any conclusions to the choice of wavelet: the best performing wavelet will depend on the unknown trend function. A pointwise confidence interval is calculated for the trend estimate (which by default is at the 95% significance level) using 500 bootstrap replications.

The resulting estimate is shown in Figure 6, which aligns closely with the true trend function. By default, the calculated confidence interval is plotted, but this can be changed by setting the argument `plot.CI = FALSE` when calling `plot()`. As expected, the confidence intervals tend to be wider at the boundaries due to the boundary handling procedure and boundary effects.

The spectrum estimate is computed with the Daubechies Extremal Phase wavelet with 4 vanishing moments, using the differenced time series, (paired with the nonlinear trend estimate) and smoothed with a running median of bin width size 128. A running median can be used for additional robustness in the estimator; for further information see McGonigle, Killick, and Nunes (2021). A smaller bin width is used due to the quickly evolving spectral structure. The estimate is shown in Figure 7. The estimate correlates well with the general features of the underlying spectrum, such as the slowly increasing power at scale 1, bump at scale 3, and the periodic pattern at scale 5.

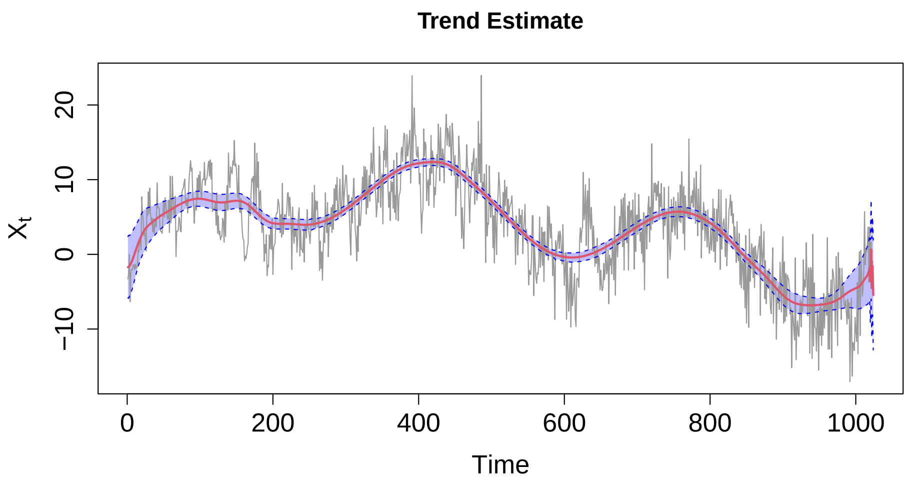

Figure 6: Estimated trend function (red solid line) with 95% pointwise confidence interval given by the blue shaded region and dashed blue lines. Original time series shown in gray solid line.

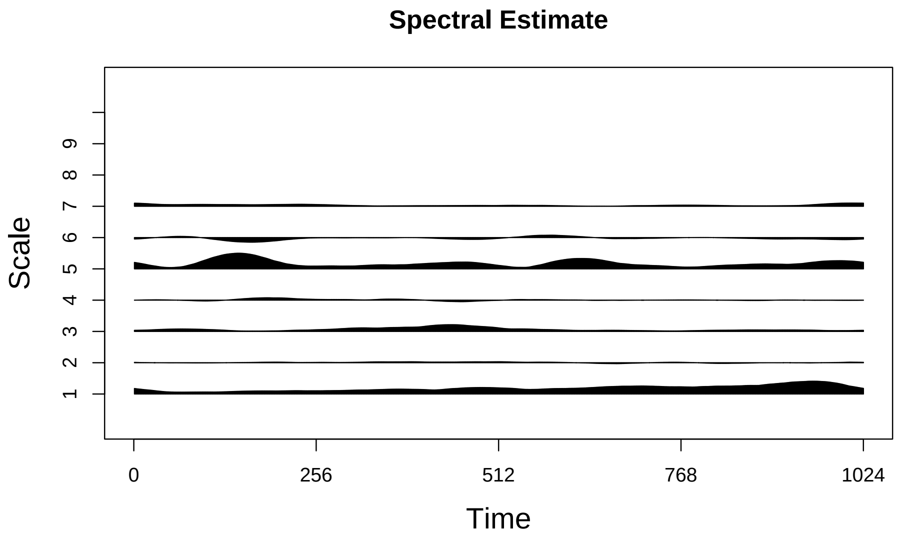

Figure 7: Estimated spectrum. Scaling is applied globally across each scale for direct comparison.

5.1. Making modifications to spectrum and trend estimation plots

In Sections 3–5 above, we used default settings to produce plots of the spectrum and trend estimates with the **TrendLSW** package. However, the underlying method `plot.TLSW()` has flexible functionality to allow practitioners to create user-controlled plots of the estimated quantities of interest from the TLSW model. More specifically, the style of the plots is controlled by the following arguments:

- `trend.plot.args`: A list object, giving options to modify the plotting of the TLSW trend estimate, with structure specified as follows:

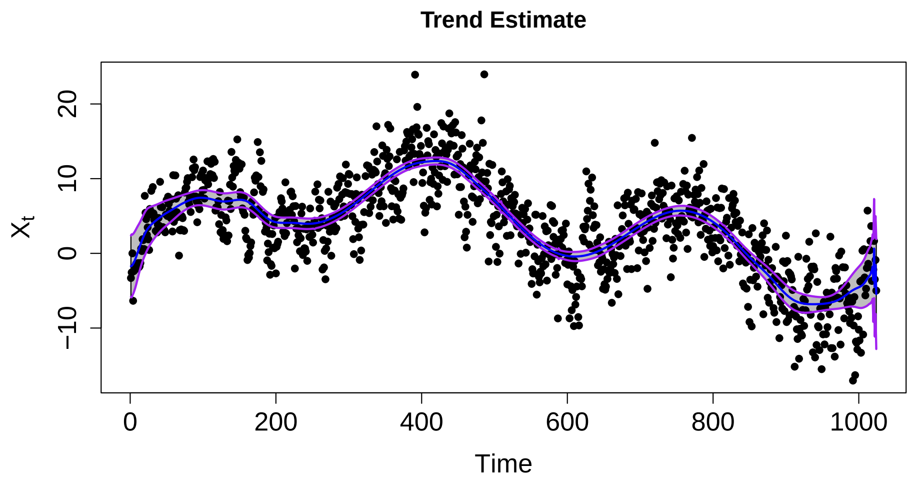

Figure 8: Plot of the trend estimate for the TLSW process `x2` given by the code snippet in the text.

- Graphical parameters related to the display of the overall plot are inherited from `plot.default` and are specified in the usual way: for example, to change the title of the plot to "Plot", use `main = "Plot"`.
- Parameters affecting the display of the estimated trend line should begin with the prefix "`T.`". For example, to set the color of the trend line to blue, use `T.col = "blue"`.
- Parameters affecting the display of the confidence interval lines should begin with the prefix "`CI.`". For example, to set the line width of the confidence interval to 2, use `CI.lwd = 2`.
- Parameters affecting the display of the polygon drawn by the confidence interval should begin with the prefix "`poly.`". For example, to set the color of the confidence interval region to green, use `poly.col = "green"`.

- `spec.plot.args`: A list object giving options to modify a plot of the TLSW spectrum estimate, for example axes specification, inherited from `plot.wd`.
- `plot.CI`: A logical variable indicating whether the confidence interval of the trend estimate (if calculated) should be displayed.
- `...`: Any additional parameters that will modify both the trend and spectrum plots, e.g., `cex.main = 2`.

We now give an example of this plotting functionality for the `x2` example. The code below plots both the trend and spectral estimate as in the example from Section 3.3, with modifications to the graphical parameters of the plot.

```
R> plot(x2.TLSW, trend.plot.args = list(col = "black", type = "p", pch = 16,
+     T.col = "blue", T.lwd = 2, poly.col = "gray",
+     CI.col = "purple", CI.lwd = 2, CI.lty = 1),
+     spec.plot.args = list(scaling = "by.level", ylab = "Level"))
```

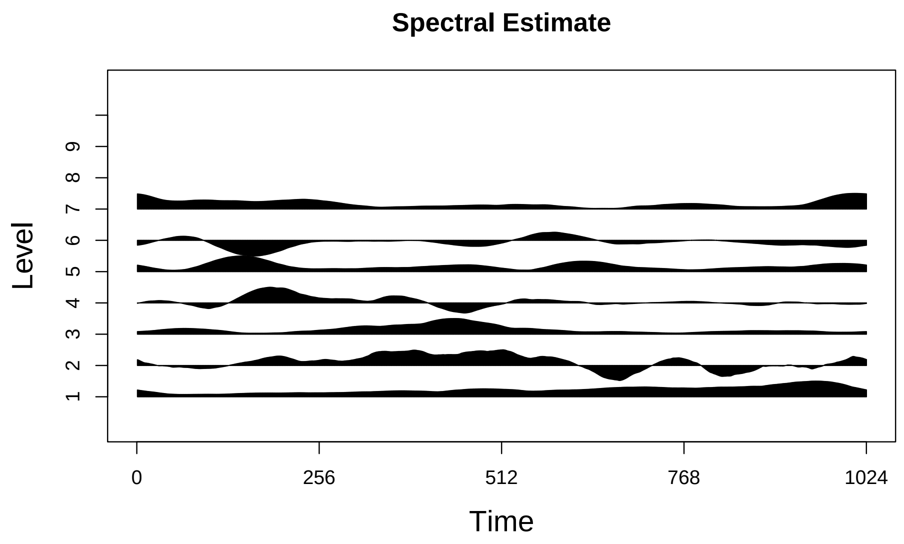

Figure 9: Plot of the spectrum estimate for the TLSW process `x2` given by the code snippet in the text. Each level is scaled individually.

This produces a modified plot of the trend and spectrum for the `x2` series. The features of the trend estimate plot are changed by supplying options to the `trend.plot.args` argument in the form of a list. The underlying data in the trend plot have been changed to be black circles, by setting `col = "black", type = "p", pch = 16` in the `trend.plot.args` argument. The color and line width of the trend estimate are set using `T.col = "blue", T.lwd = 2` inside `trend.plot.args`. The color of the confidence interval's shaded region is set using `poly.col = "gray"`, and the features of the confidence interval lines are changes using `CI.col = "purple", CI.lwd = 2, CI.lty = 1`. The resulting trend plot is shown in Figure 8.

Features of the spectrum estimate plot can be changed by supplying options to the argument`spec.plot.args`: the scaling on each level of the plot (an option that is inherited from `wavethresh::plot.wd()`) is set to be individually scaled by specifying `scaling = "by.level"`, and the y-axis label has been modified using `ylab = "Level"`. The spectrum plot is shown in Figure 9.

6. Data examples

We now describe the usage of the various methods described in Sections 3 and 4 to analyze two real data examples.

6.1. Bioluminescence of Caenorhabditis elegans

Caenorhabditis elegans is an excellent model for high-throughput experimental approaches (O'Reilly, Luke, Perlmutter, Silverman, and Pak 2014). Notably, C. elegans transgenic strains have been developed that express the firefly luciferase enzyme to produce bioluminescence to track the worm metabolic activity over time (Lagido, Pettitt, Flett, and Glover 2008). This has been applied to studying compound toxicity (Baldock *et al.* 2023), ageing (Hahm *et al.*

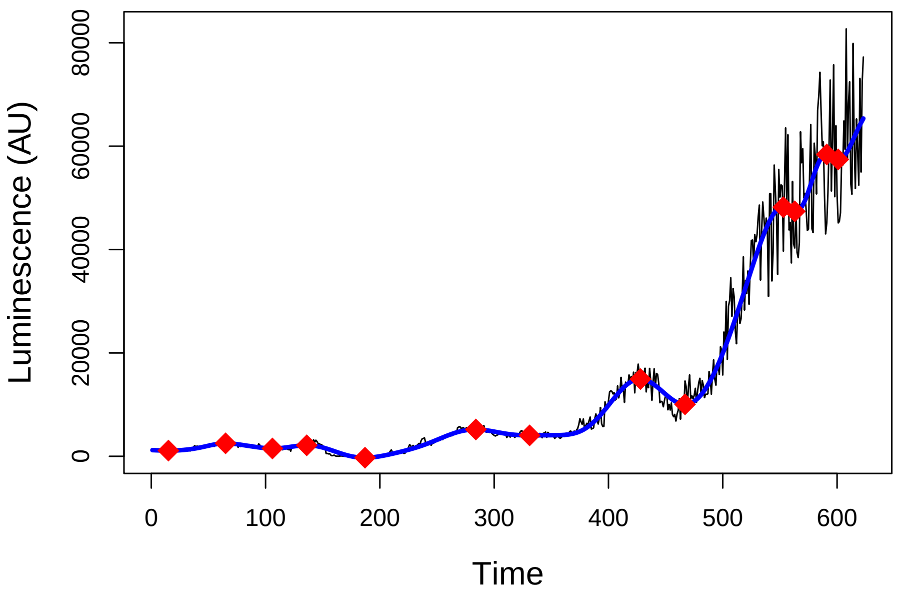

Figure 10: Bioluminescence of a group of C. Elegans peaks and troughs correspond to group feeding and growing phases respectively. Black thin line is the observed data, blue thick line is the TLSW trend fit and the red diamonds are the identified peaks and troughs from the TLSW trend fit.

2020) and development (Olmedo, Geibel, Artal-Sanz, and Merrow 2015). During its life cycle, C. elegans goes through four successive larval stages and molts, during which they temporarily stop feeding and exhibit lower metabolic activity. This can be revealed by bioluminescence readings when the transgenic worms aforementioned are grown in presence of the luciferase substrate luciferin, leading to multimodal bioluminescence readings (Figure 10) that are hard to fit with traditional functions, to automatically extract key parameters (Olmedo *et al.* 2015).

While manual determination of developmental phases and metabolic activity can be achieved easily, such assays are typically carried out in multiple batches of 96 or 384 well arrays, for which manual analyses would take far longer than the experiments themselves. It is challenging for bioscientists to automatically process bioluminescence traces due to the clear non-stationary second-order behavior present in the time series. Transitions between molts and growth phases produce peaks and troughs of varying amplitude and width in these bioluminescence time series. Here we seek to measure these by identifying the peaks and troughs in the estimated trend whilst accounting for the time varying second order properties. The C. elegans data presented in Figure 10 is a time series of bioluminescence readings from 20 L1 worms (strain PE255) taken every six minutes over a 54h period in a M200 Infinite Pro Tecan plate-reader captured by Alexandre Benedetto (Lancaster University). The time series is available as `celegansbio` within the **TrendLSW** package. This is one time series taken from an experiment where hundreds of time series are collected (Baldock *et al.* 2023).

The following code performs TLSW trend estimation on the C. Elegans time series and identifies the peaks and troughs within the estimated trend to produce Figure 10.

```
R> data("celegansbio")
R> out <- TLSW(celegansbio, T.filter.number = 10, T.family = 'DaubExPhase')
```

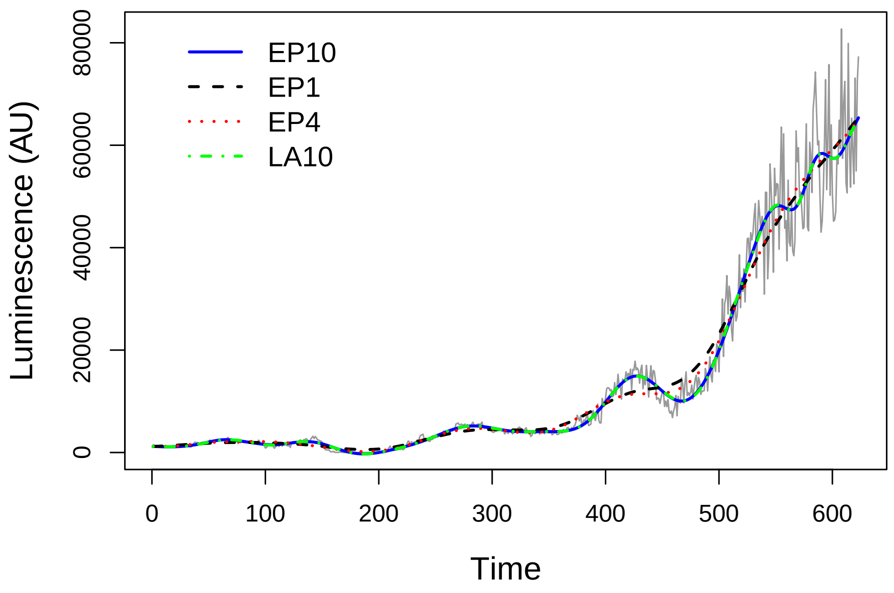

Figure 11: Comparison of TLSW trend fits obtained using: the EP10 wavelet (blue solid line), EP1 wavelet (black dashed line), EP4 wavelet (red dotted line), LA10 wavelet (green dot-dashed line).

```
R> plot(out, plot.type = "trend", trend.plot.args =
+    list(ylab = "Luminescence (AU)", T.col = "blue", T.lwd = 3, main = ""),
+    cex.lab = 1.3, col = "black")
R> delta <- diff(out$trend.est$T)
R> turns <- which(delta[-1] * delta[-length(delta)] < 0) + 1
R> points(turns, out$trend.est$T[turns], col = "red", pch = 18, cex = 2)
```

It can be beneficial to fit a range of wavelets and assess the visual fit to the data in order to choose the most appropriate wavelet for trend estimation. In the code below, we compute the trend estimate for a range of wavelets.

```
R> out.EP1 <- TLSW(celegansbio, T.filter.number = 1,
+    T.family = "DaubExPhase", T.transform = "nondec")
R> out.EP4 <- TLSW(celegansbio, T.filter.number = 4,
+    T.family = "DaubExPhase", T.transform = "nondec")
R> out.LA10 <- TLSW(celegansbio, T.filter.number = 10,
+    T.family = "DaubLeAsymm", T.transform = "nondec")
```

We can then visually compare the fitted trends, as shown in Figure 11.

```
R> plot(out, plot.type = "trend", trend.plot.args =
+    list(ylab = "Luminescence (AU)", T.col = "blue", T.lwd = 2, main = ""),
+    cex.lab = 1.3, col = "gray60")
R> lines(out.EP1$trend.est$T, col = "black", lwd = 2, lty = 2)
R> lines(out.EP4$trend.est$T, col = "red", lty = 3, lwd = 2)
R> lines(out.LA10$trend.est$T, col = "green", lty = 4, lwd = 2)
```

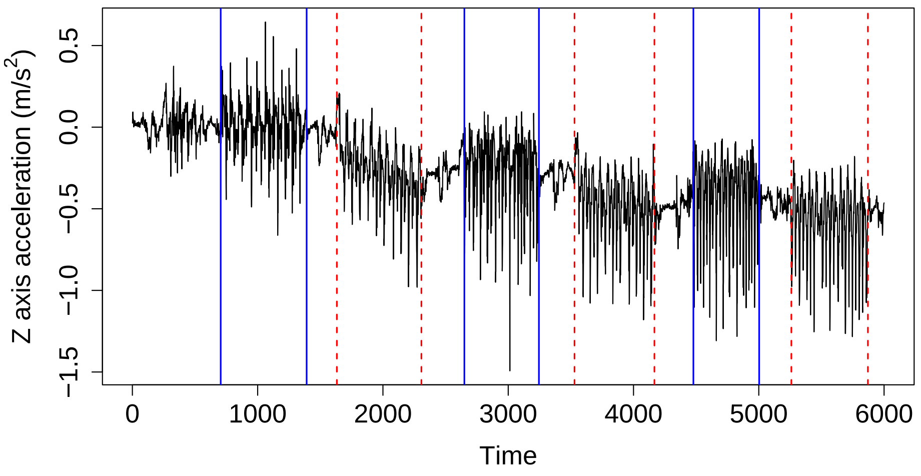

Figure 12: Z-axis acceleration time series. Data between blue solid vertical lines indicates when the participant was walking downstairs, data between red dashed vertical lines indicates walking upstairs.

```
R> legend(10, 85000, c("EP10", "EP1", "EP4", "LA10"),
+    lty = c(1, 2, 3, 4), lwd = c(2, 2, 2, 2),
+    col = c("blue", "black", "red", "green"), bty = "n", cex = 1.2)
```

We compare the fits produced by the Daubechies Extremal Phase (EP) wavelets with 1, 4, and 10 vanishing moments, as well as the Daubechies Least Asymmetric (LA) Wavelet with 10 vanishing moments. The fits for the EP1 and EP4 wavelets appear to be noticeably worse than those for the EP10 and LA10, which show better adaptivity to the data and are almost identical. Heuristically, we can also compute the root mean squared error (RMSE) of the trend estimates, which identify the EP10 and LA10 wavelets as being indistinguishable (RMSEs of 3682.85 and 3682.42 respectively), and superior to the EP1 and EP4 wavelets (RMSEs of 4062.61 and 3990.25 respectively). The code for this is given below.

```
R> n <- length(celegansbio)
R> trends <- list(EP1 = out.EP1$trend.est$T,EP4 = out.EP4$trend.est$T,
+    EP10 = out$trend.est$T, LA10 = out.LA10$trend.est$T)
R> trends.rmse <- lapply(trends, function(y) {
+    sqrt(sum((y - celegansbio) ^ 2) / n) })
```

6.2. Accelerometer data

Accelerometers are used in a variety of fields including engineering, health, and marketing. Here we consider a dataset obtained from the UCI data repository (Kelly, Longjohn, and Nottingham 2024) based on accelerometer readings from a smartphone (Reyes-Ortiz, Oneto, Samà, Parra, and Anguita 2016) during an experiment on daily living activities. Specifically, we analyze a section of the z-axis time series of length $n = 6000$ from experiment 3 of user 2, which we denote `z.acc`. During this time period, the participant performed two activities several times; walking upstairs and walking downstairs. The data are shown in Figure 12, where data between blue solid vertical lines indicate when the participant was

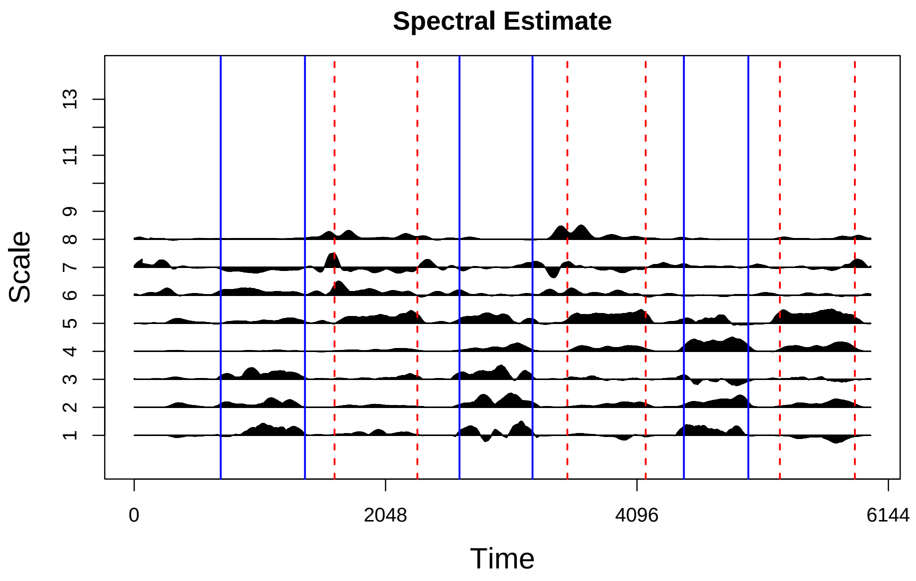

Figure 13: Spectral estimate for the accelerometer data from Figure 12. Each level is scaled individually.

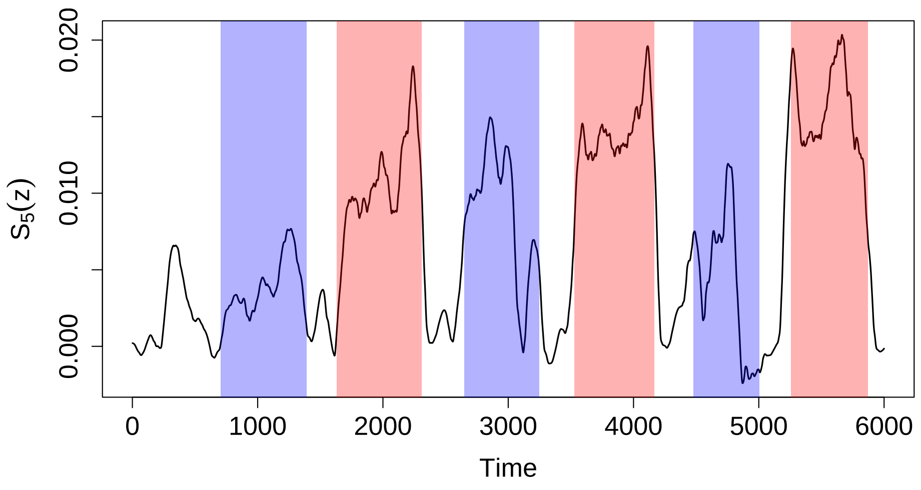

Figure 14: Spectral estimate at scale $j = 5$ for the accelerometer data from Figure 12. Periods of walking upstairs shown in blue, and periods of walking downstairs shown in red.

walking downstairs, and data between red dashed vertical lines indicate walking upstairs. The data show a gradual downward trend, and periods of activity correspond to periods of greater variability. The accelerometer time series is available as `z.acc` within the **TrendLSW** package, and the associated activity labels (along with their start and end times) are available as `z.labels`.

We focus on investigating what insights the spectral estimate reveals about the activity of the participant. The following code performs TLSW estimation on `z.acc` and plots the result, where we have changed the default spectral smoothing to use the Epanechnikov kernel with a bin width of 150, corresponding to a time period of 3 seconds:

```
R> data("z.acc")
R> data("z.labels")
R> z.acc.TLSW <- TLSW(z.acc, S.smooth.type = "epan", S.binwidth = 150)
R> plot(z.acc.TLSW, plot.type = "spec",
+    spec.plot.args = list(scaling = "by.level"))
```

Figure 13 displays the spectral estimate of the time series, which has been scaled individually at each level. The spectral estimate closely correlates with the participant's activities. To highlight this, Figure 14 shows the spectral estimate at scale $j = 5$ only, with periods of walking downstairs and walking upstairs superimposed onto the plot in blue and red respectively. We see that the periods of activity give rise to increased power in $S_5(z)$.

7. Concluding remarks

Time series data with both time-varying mean and dependence structure are increasingly commonplace, and thus there is a growing need for implementations of statistically justified models and inference tools so that practitioners can derive principled insight from scientific data. This article introduces the **TrendLSW** package in R to address this gap in currently available software, implementing recent methods for modeling nonstationary time series. The functionality in the package allows users to simulate time series with first and second order nonstationarity, as well as estimate relevant quantities of interest, such as the trend and wavelet spectrum associated to time series.

Whilst the **TrendLSW** package incorporates implementations of flexible estimation methods for the recent Trend-LSW model for users familiar with locally stationary wavelet processes, the package is designed with a broad end-user base in mind, and hence we provide an automatic way of analysing nonstationary time series data, with few tuning parameters needed to be specified. Trend and dependence time series components are easily estimated and subsequently visualized, together with quantification of estimation uncertainty. We have presented several case studies from different fields to demonstrate the package functionality, and hope the package will provide practical tools for data analysis in a wide range of scientific and industrial settings.

Future development of **TrendLSW** will include adding further functionality including a predict method for `TLSW` objects.

Acknowledgments

Thanks to Alexandre Benedetto for collecting the data from the C. Elegans experiment and allowing us to provide it as open-source data within the package.

References

Antoniadis A, Bigot J, Sapatinas T (2001). "Wavelet Estimators in Nonparametric Regression: A Comparative Simulation Study." *Journal of Statistical Software*, **6**, 1–83. `doi:10.18637/jss.v006.i06`.

Arnold TB, Tibshirani RJ (2022). ***genlasso**: Path Algorithm for Generalized Lasso Problems.* doi:10.32614/CRAN.package.genlasso. R package version 1.6.1.

Baldock SJ, Kevin P, Harper GR, Griffin R, Genedy HH, Fong MJ, Zhao Z, Zhang Z, Shen Y, Lin H, Au C, Martin JR, Ashton MD, Haskew MJ, Stewart B, Efremova O, Esfahani RN, Emsley HCA, Appleby JB, Cheneler D, Cummings DM, Benedetto A, Hardy JG (2023). "Creating 3D Objects with Integrated Electronics via Multiphoton Fabrication In Vitro and In Vivo." *Advanced Materials Technologies*, **8**(11), 2201274. doi:10.1002/admt.202201274.

Beaulieu C, Killick R (2018). "Distinguishing Trends and Shifts from Memory in Climate Data." *Journal of Climate*, **31**(23), 9519–9543. doi:10.1175/jcli-d-17-0863.1.

Cardinali A, Nason G (2022). ***LSWPlib**: Simulation and Spectral Estimation of Locally Stationary Wavelet Packet Processes.* doi:10.32614/CRAN.package.LSWPlib. R package version 0.1.0.

Craigmile PF, Guttorp P, Percival DB (2004). "Trend Assessment in a Long Memory Dependence Model Using the Discrete Wavelet Transform." *Environmetrics*, **15**(4), 313–335. doi:10.1002/env.642.

Dahlhaus R (1997). "Fitting Time Series Models to Nonstationary Processes." *The Annals of Statistics*, **25**(1), 1–37. doi:10.1214/aos/1034276620.

Dahlhaus R (2012). "Locally Stationary Processes." In *Handbook of Statistics*, volume 30, pp. 351–413. Elsevier.

Daubechies I (1992). *Ten Lectures on Wavelets.* SIAM.

Di Narzo AF, Aznarte JL, Stigler M (2024). ***tsDyn**: Time Series Analysis Based on Dynamical Systems Theory.* doi:10.32614/CRAN.package.tsDyn. R package version 11.0.5.2.

Friedrich M, Smeekes S, Urbain JP (2020). "Autoregressive Wild Bootstrap Inference for Nonparametric Trends." *Journal of Econometrics*, **214**(1), 81–109. doi:10.1016/j.jeconom.2019.05.006.

Hahm JH, Jeong C, Lee W, Koo HJ, Kim S, Hwang D, Nam HG (2020). "A Cellular Surveillance and Defense System That Delays Aging Phenotypes in *C. Elegans*." *Aging*, **12**(9), 8202–8220. doi:10.18632/aging.103134.

Hastie T (2025). ***gam**: Generalized Additive Models.* doi:10.32614/CRAN.package.gam. R package version 1.22-7.

Jiang F, Zhao Z, Shao X (2023). "Time Series Analysis of COVID-19 Infection Curve: A Change-Point Perspective." *Journal of Econometrics*, **232**(1), 1–17. doi:10.1016/j.jeconom.2020.07.039.

Keele L, Kelly NJ (2006). "Dynamic Models for Dynamic Theories: The Ins and Outs of Lagged Dependent Variables." *Political Analysis*, **14**(2), 186–205. doi:10.1093/pan/mpj006.

Kelly M, Longjohn R, Nottingham K (2024). "The UCI Machine Learning Repository." URL https://archive.ics.uci.edu/.

Kilian L, Lütkepohl H (2017). *Structural Vector Autoregressive Analysis*. Cambridge University Press.

Lagido C, Pettitt J, Flett A, Glover LA (2008). "Bridging the Phenotypic Gap: Real-Time Assessment of Mitochondrial Function and Metabolism of the Nematode Caenorhabditis Elegant." *BMC Physiology*, **8**(7). doi:10.1186/1472-6793-8-7.

Lees JM, Harris D (2024). ***RSEIS**: Seismic Time Series Analysis Tools*. doi:10.32614/CRAN.package.RSEIS. R package version 4.2-4.

Lyubchich V, Gel YR, Vishwakarma S (2025). ***funtimes**: Functions for Time Series Analysis*. doi:10.32614/CRAN.package.funtimes. R package version 10.0.

McGonigle ET, Killick R, Nunes MA (2021). "Detecting Changes in Mean in the Presence of Time-Varying Autocovariance." *Stat*, **10**(1), e351. doi:10.1002/sta4.351.

McGonigle ET, Killick R, Nunes MA (2022a). "Modelling Time-Varying First and Second-Order Structure of Time Series via Wavelets and Differencing." *Electronic Journal of Statistics*, **16**(2), 4398–4448. doi:10.1214/22-ejs2044.

McGonigle ET, Killick R, Nunes MA (2022b). "Trend Locally Stationary Wavelet Processes." *Journal of Time Series Analysis*, **43**(6), 895–917. doi:10.1111/jtsa.12643.

McGonigle ET, Killick R, Nunes MA (2026). ***TrendLSW**: Wavelet Methods for Analysing Locally Stationary Time Series*. doi:10.32614/CRAN.package.TrendLSW. R package version 1.0.6.

Milborrow S, Hastie T, Tibshirani R (2024). ***earth**: Multivariate Adaptive Regression Splines*. doi:10.32614/CRAN.package.earth. R package version 5.3.4.

Nason G (2008). *Wavelet Methods in Statistics with R*. Springer-Verlag. doi:10.1007/978-0-387-75961-6.

Nason G (2024). ***wavethresh**: Wavelets Statistics and Transforms*. doi:10.32614/CRAN.package.wavethresh. R package version 4.7-3.

Nason G (2025). ***locits**: Tests of Stationarity and Localized Autocovariance*. doi:10.32614/CRAN.package.locits. R package version 1.7-8.

Nason GP, von Sachs R, Kroisandt G (2000). "Wavelet Processes and Adaptive Estimation of the Evolutionary Wavelet Spectrum." *Journal of the Royal Statistical Society B*, **62**(2), 271–292. doi:10.1111/1467-9868.00231.

Olea R, Palma W, Rubio P (2021). ***LSTS**: Locally Stationary Time Series*. doi:10.32614/CRAN.package.LSTS. R package version 2.1.

Olmedo M, Geibel M, Artal-Sanz M, Merrow M (2015). "A High-Throughput Method for the Analysis of Larval Developmental Phenotypes in Caenorhabditis Elegans." *Genetics*, **201**(2), 443–448. doi:10.1534/genetics.115.179242.

Ombao H, Raz J, von Sachs R, Guo W (2002). "The SLEX Model of a Non-Stationary Random Process." *Annals of the Institute of Statistical Mathematics*, **54**(1), 171–200. doi:10.1023/a:1016130108440.

O'Reilly LP, Luke CJ, Perlmutter DH, Silverman GA, Pak SC (2014). "C. Elegans in High-Throughput Drug Discovery." *Advanced Drug Delivery Reviews*, **69-70**, 247?253. doi:10.1016/j.addr.2013.12.001.

Percival DB, Walden AT (2006). *Wavelet Methods for Time Series Analysis*, volume 4. Cambridge University Press.

Pohlert T (2023). ***trend**: Non-Parametric Trend Tests and Change-Point Detection.* doi:10.32614/CRAN.package.Trend. R package version 1.1-6.

Priestley MB (1965). "Evolutionary Spectra and Non-Stationary Processes." *Journal of the Royal Statistical Society B*, pp. 204–237. doi:10.1111/j.2517-6161.1965.tb01488.x.

R Core Team (2025). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria. doi:10.32614/R.manuals. URL https://www.R-project.org/.

Reyes-Ortiz JL, Oneto L, Samà A, Parra X, Anguita D (2016). "Transition-Aware Human Activity Recognition Using Smartphones." *Neurocomputing*, **171**, 754–767. doi:10.1016/j.neucom.2015.07.085.

Roueff F, von Sachs R (2019). "Time-Frequency Analysis of Locally Stationary Hawkes Processes." *Bernoulli*, **25**(2), 1355–1385. doi:10.3150/18-bej1023.

Stigler M (2019). "Nonlinear Time Series in R: Threshold Cointegration with tsDyn." In HD Vinod, CR Rao (eds.), *Handbook of Statistics, Volume 41*, volume 42, pp. 229–264. Elsevier. doi:10.1016/bs.host.2019.01.008.

Taylor SAC, Park T, Eckley IA (2019). "Multivariate Locally Stationary Wavelet Analysis With the **mvLSW** R Package." *Journal of Statistical Software*, **90**(11), 1–19. doi:10.18637/jss.v090.i11.

Van Bellegem S, Dahlhaus R (2006). "Semiparametric Estimation by Model Selection for Locally Stationary Processes." *Journal of the Royal Statistical Society B*, **68**(5), 721–746. doi:10.1111/j.1467-9868.2006.00564.x.

Vidakovic B (2009). *Statistical Modeling by Wavelets.* John Wiley & Sons.

Vogt M (2012). "Nonparametric Regression for Locally Stationary Time Series." *The Annals of Statistics*, **40**(5), 2601–2633. doi:10.1214/12-aos1043.

von Sachs R, MacGibbon B (2000). "Non-Parametric Curve Estimation by Wavelet Thresholding with Locally Stationary Errors." *Scandinavian Journal of Statistics*, **27**(3), 475–499. doi:10.1111/1467-9469.00202.

Wood S (2025). ***mgcv**: Mixed GAM Computation Vehicle with Automatic Smoothness Estimation.* doi:10.32614/CRAN.package.mgcv. R package version 1.9-4.

Wu WB, Zhao Z (2007). "Inference of Trends in Time Series." *Journal of the Royal Statistical Society B*, **69**(3), 391–410. doi:10.1111/j.1467-9868.2007.00594.x.

Zeileis A (2019). *dynlm: Dynamic Linear Regression*. doi:10.32614/CRAN.package.dynlm. R package version 0.3-6.

Zhang T, Wu WB (2011). "Testing Parametric Assumptions of Trends of a Nonstationary Time Series." *Biometrika*, **98**(3), 599–614. doi:10.1093/biomet/asr017.

Zhou Z, Wu WB (2009). "Local Linear Quantile Estimation for Nonstationary Time Series." *The Annals of Statistics*, **37**(5B), 2696–2729. doi:10.1214/08-aos636.

Affiliation:

Euan T. McGonigle
School of Mathematical Sciences
University of Southampton
Southampton, United Kingdom
E-mail: e.t.mcgonigle@soton.ac.uk

Rebecca Killick
Department of Mathematics and Statistics
Lancaster University
Lancaster, United Kingdom
E-mail: r.killick@lancaster.ac.uk

Matthew A. Nunes
Department of Mathematical Sciences
University of Bath
Bath, United Kingdom
E-mail: m.a.nunes@bath.ac.uk

Journal of Statistical Software https://www.jstatsoft.org/
published by the Foundation for Open Access Statistics https://www.foastat.org/

December 2025, Volume 115, Issue 10 *Submitted:* 2024-05-08
doi:10.18637/jss.v115.i10 *Accepted:* 2024-11-01