wavScalogram: An R Package with Wavelet Scalogram Tools for Time Series Analysis

by Vicente J. Bolós and Rafael Benítez,

Abstract In this work we present the **wavScalogram** R package, which contains methods based on wavelet scalograms for time series analysis. These methods are related to two main wavelet tools: the windowed scalogram difference and the scale index. The windowed scalogram difference compares two time series, identifying if their scalograms follow similar patterns at different scales and times, and it is thus a useful complement to other comparison tools such as the squared wavelet coherence. On the other hand, the scale index provides a numerical estimation of the degree of non-periodicity of a time series and it is widely used in many scientific areas.

1 Introduction

Since the works of Mallat (2008) and Daubechies (1992), wavelet analysis has become, in the last few decades, a standard tool in the field of time series analysis. Its ability to simultaneously analyze a signal in frequency space (scales) and in time, allows it to overcome many of the limitations that Fourier analysis presents for non-stationary time series. Furthermore, the algorithms for calculating the different wavelet transforms are characterized by their speed and ease of implementation.

There are currently many software packages that implement functions for wavelet analysis of time series (MATLAB's Wavelet Toolbox, Wavelab, etc.), and in recent years, the exponential growth of the R ecosystem has not been outside the field of wavelet analysis. Within CRAN there are many packages related to wavelet analysis for time series. Specifically, as collected in the *TimeSeries* Task View, the **wavelets** package (Aldrich, 2020) , the **WaveletComp** and **biwavelet** packages (Roesch and Schmidbauer, 2018; Gouhier et al., 2021) , the **mvLSW** package (Taylor et al., 2019) and other packages such as **hwwntest** (Savchev and Nason, 2018), **rwt** (Roebuck and Rice University's DSP group, 2022), **waveslim** (Whitcher, 2020) and **wavethresh** (Nason, 2022).

In this work we will describe in depth the **wavScalogram** package (Bolós and Benítez, 2021) (also mentioned in the *TimeSeries* Task View). In this package, methods based on the wavelet scalogram are introduced as defined in Benítez et al. (2010); Bolós et al. (2017, 2020). These methods are basically related to two main wavelet tools: the *windowed scalogram difference* and the *scale index*. The first one, the windowed scalogram difference, was introduced in Bolós et al. (2017). It allows to compare two time series at different scales and times, determining if their scalograms follow similar patterns. In this sense, it is a complement to other wavelet tools for comparing time series such as the squared wavelet coherence and the phase difference, since there are certain differences in time series that these measurements are not capable of detecting while the windowed scalogram difference can. The second tool is the scale index introduced in Benítez et al. (2010). It focuses on the analysis of the non-periodicity of a signal, giving a numerical measure of its degree of non-periodicity, taking the value 0 if the signal is periodic and a value close to 1 if the signal is totally aperiodic (for example, a purely stochastic signal). The scale index has been used in many scientific areas, being the evaluation of the quality of pseudo-random number generators the area where it has been used the most. In addition, the scale index also has a *"windowed"* version, in which the windowed scalogram is used to calculate the scale index instead, allowing to measure the evolution of the scale index over time, which is useful in the case of non-stationary time series (see Bolós et al. (2020)).

The article is organized as follows: In the next section, we describe the basics of the wavelet analysis and how to use them in the **wavScalogram** package. Then, a description of the wavelet scalogram and its implementation is given. The following sections are devoted to the windowed scalogram difference and the scale index, in its original and windowed versions. Finally we illustrate the use of the package with some examples in applied problems, such as the analysis of time series of sunspots or the use of the windowed scalogram difference in the clustering of time series, particularly the interest rate series of sovereign bonds.

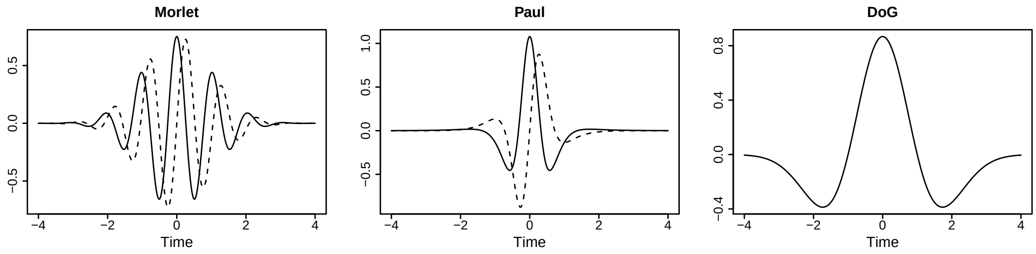

Figure 1: Real part (solid) and imaginary part (dashed) of Morlet, Paul and DoG wavelets for default parameter values, $\omega_0 = 6$ and $m = 4, 2$ respectively. Along with Haar, they are the most used in wavelet analysis.

2 Wavelet introduction

A *wavelet* (or *mother wavelet*) is a function $\psi \in L^2(\mathbb{R})$ with zero average (i.e. $\int_{\mathbb{R}} \psi = 0$), unit energy ($\|\psi\| = 1$, i.e. normalized) and centered in the neighborhood of $t = 0$ (Mallat, 2008). There exists a wide variety of wavelets but in this package we use the following, described in Torrence and Compo (1998) (see Figure 1):

- Morlet:
$$\psi_{\text{Morlet}}(t) = \pi^{-1/4} e^{\mathrm{i}\omega_0 t} e^{-t^2/2}.$$
It is a plane wave modulated by a Gaussian, where the positive parameter ω_0 denotes the central dimensionless frequency. According to Farge (1992), the wavelet function must fulfil an admissibility condition, which for the Morlet wavelet is only accomplished if some correction factors are added. We take as default value $\omega_0 = 6$, for which those correction factors are negligible. Nevertheless, other choices of this parameter can be considered.
- Paul:
$$\psi_{\text{Paul}}(t) = \frac{(2\mathrm{i})^m \, m!}{\sqrt{\pi \, (2m)!}} \, (1 - \mathrm{i}t)^{-(m+1)},$$
where m is a positive integer parameter representing the order. By default, $m = 4$.
- Derivative of a Gaussian (DoG):
$$\psi_{\text{DoG}}(t) = \frac{(-1)^{m+1}}{\sqrt{\Gamma\left(m + \frac{1}{2}\right)}} \frac{\mathrm{d}^m}{\mathrm{d}t^m}\left(e^{-t^2/2}\right),$$
where m is a positive integer parameter representing the derivative. By default, $m = 2$, that coincides with the Marr or Mexican hat wavelet.

Moreover, we have added:

- Haar, centered at 0:
$$\psi_{\text{Haar}}(t) = \begin{cases} 1 & \text{if } -\frac{1}{2} \le t < 0, \\ -1 & \text{if } 0 \le t < \frac{1}{2}, \\ 0 & \text{otherwise.} \end{cases}$$
This is the simplest wavelet, but it is not continuous.

Scaling a wavelet ψ by $s > 0$ and translating it by u, we create a family of unit energy "time-frequency atoms", called *daughter wavelets*, $\psi_{u,s}$, as follows

$$\psi_{u,s}(t) = \frac{1}{\sqrt{s}} \psi\left(\frac{t-u}{s}\right). \tag{1}$$

Remark 2.2.1 (Fourier factor)**.** Usually, the Fourier wavelength of a daughter wavelet does not coincide with its scale s. Nevertheless, they are proportional, and this proportionality factor for converting scales into Fourier periods is called *Fourier factor*. This Fourier factor is taken $4\pi/\left(\omega_0 + \sqrt{2 + \omega_0^2}\right)$, $4\pi/(2m+1)$ and $2\pi/\sqrt{m + 1/2}$ for Morlet, Paul and DoG wavelets respectively (Torrence and Compo, 1998). For the default parameter values, the Fourier factor is approximately 1.033, 1.3963, and 3.9738 respectively. For Haar wavelet, the Fourier factor is 1.

Given a function $f \in L^2(\mathbb{R})$, that we will identify with a *signal* or *time series*, the *continuous wavelet transform* (CWT) of f at time u and scale $s > 0$ is defined as

$$\mathcal{W}f(u,s) = \int_{-\infty}^{+\infty} f(t)\psi^*_{u,s}(t)\,\mathrm{d}t, \tag{2}$$

where * denotes the complex conjugate. The CWT allows us to obtain the frequency components (or *details*) of f corresponding to scale s and time location u.

In practical situations, however, it is common to deal with finite signals. That is, given a time signal x, and a finite time interval $[0,T]$, we shall consider the finite sequence $x_n = x(t_n)$, for $n = 0,\ldots,N$. Here, $t_0,\ldots,t_N$ is a discretization of the interval $[0,T]$, i.e. $t_n = nh$, being $h = T/N$ the time step. According to (1) and (2), the CWT of x at scale $s > 0$ is defined as the sequence

$$\mathcal{W}x_n(s) = h\sum_{i=0}^{N} x_i\psi^*_{n,s}(t_i), \tag{3}$$

where $n = 0,\ldots,N$ and

$$\psi_{n,s}(t) = \frac{1}{\sqrt{s}}\psi\left(\frac{t-t_n}{s}\right). \tag{4}$$

Note that $\psi_{n,s}(t)$ is in fact $\psi_{t_n,s}(t)$, but this abuse of notation between (1) and (4) is assumed for the sake of readability. Using Fourier transform tools, one can calculate (3) for all $n = 0,\ldots N$ simultaneously and efficiently (Torrence and Compo, 1998).

Remark 2.2.2 (Energy density). It is known that the CWT coefficients are biased in favour of large scales (Liu et al., 2007). Nevertheless, if the mother and daughter wavelets are normalized by the L^1-norm (as in the **Rwave** package, by Carmona and Torresani (2021)) instead of the L^2-norm (as in our package), this bias is not produced . Hence, to rectify the bias, the CWT in (2) and (3) can be multiplied by the factor $\frac{1}{\sqrt{s}}$. This rectification will be specially useful in some wavelet tools of our package that quantify the "energy density" of a signal, such as the wavelet power spectrum, the scalograms and the windowed scalogram difference. On the other hand, in the case of the scale index, this correction will not be advisable (see Remark 2.5.1). Usually, the wavelet tools of our package have a logical parameter called `energy_density` that switches this correction.

For computing the CWT of a time series x at a given set of scales, we use `cwt_wst`. For example,

```
# install.packages("wavScalogram")
library(wavScalogram)
h <- 0.1
N <- 1000
time <- seq(from = 0, to = N * h, by = h)
signal <- sin(pi * time)
scales <- seq(from = 0.5, to = 4, by = 0.05)
cwt <- cwt_wst(signal = signal, dt = h,
               scales = scales, powerscales = FALSE,
               wname = "DOG", wparam = 6)
```

computes the CWT of `signal` at `scales` from $s_a = 0.5$ to $s_b = 4$ using DoG wavelet with $m = 6$. The parameter `wname` indicates the wavelet used, and it can be `"MORLET"` (default value), `"PAUL"`, `"DOG"`, `"HAAR"` or `"HAAR2"`. The difference between these two last values is that `"HAAR2"` provides a more accurate but slower algorithm than the one provided by `"HAAR"`. Moreover, we can specify by means of `wparam` the value of the parameters ω_0 or m. As it has been stated before, the default values of these parameters are $\omega_0 = 6$ for Morlet wavelet, $m = 4$ for Paul wavelet and $m = 2$ for DoG wavelet.

If the set of scales is a base 2 power scales set (Torrence and Compo, 1998), the parameter `scales` can be a vector of three elements with the lowest scale s_a, the highest scale s_b and the number of suboctaves per octave. This vector is internally passed to function `pow2scales` that returns the constructed base 2 power scales set. For example,

```
scales <- c(0.5, 4, 16)
cwt <- cwt_wst(signal = signal, dt = h, scales = scales, powerscales = TRUE)
```

computes the CWT of `signal` at scales from $s_a = 0.5$ to $s_b = 4$, with 16 suboctaves per octave. Since parameter `powerscales` is `TRUE` by default, it is not necessary to specify it in the function call. If `scales = NULL` (default value), then the function constructs the scales set automatically: s_a is chosen so that its equivalent Fourier period is $2h$ (Torrence and Compo, 1998), and $s_b = Nh/2r_w$, where r_w is the corresponding *wavelet radius*. Note that in this case s_b maximizes the length of the scales interval

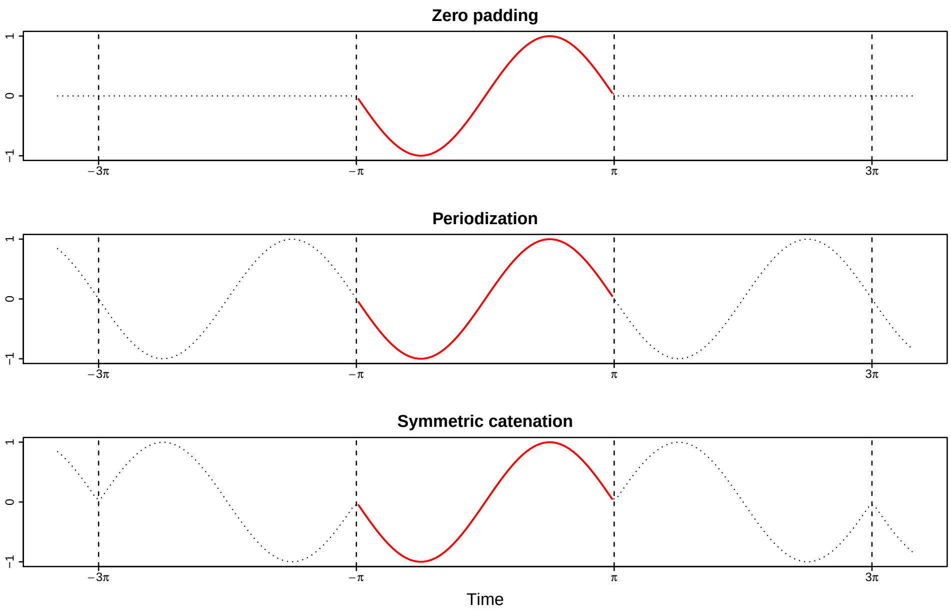

Figure 2: Different constructions of an infinite signal from a finite length signal $\sin(t)$ with $t \in [-\pi, \pi]$ (in red): padding time series with zeroes, using a periodization of the original time series, and using a symmetric catenation of the original time series. They determine the border effects.

taking into account the *cone of influence*. The wavelet radius and the cone of influence are defined and discussed in Remark 2.2.4.

The output `cwt` is a list containing the following fields:

- `coefs` is an $(N+1)\times$`length(scales)` array (either real or complex depending on the wavelet used) containing the corresponding CWT coefficients, i.e. `cwt$coefs[i,j]` is the CWT coefficient at the i-th time and j-th scale.
- `scales` is the vector of scales used, either provided by the user or constructed by the function itself.
- `fourier_factor` is the scalar used to transform scales into Fourier periods (see Remark 2.2.1).
- `coi_maxscale` is a numeric vector of size $N+1$ that defines the *cone of influence* (see Remark 2.2.4).

Remark 2.2.3 (Border effects)**.** In (3) (or (2) for finite length signals) there appear *border effects* (or *edge effects*) when the support of the daughter wavelets is not entirely contained in the time domain $[t_0, t_N]$. In order to try to mitigate border effects, we can construct from the original time series x an infinite time series $\bar{x}$ on $t_i = t_0 + ih$ for $i \in \mathbb{Z}$ and then we define

$$\bar{\mathcal{W}}x_n(s) = \mathcal{W}\bar{x}_n(s) = h \sum_{i\in\mathbb{Z}} \bar{x}_i \psi^*_{n,s}(t_i), \tag{5}$$

where $n = 0, \ldots, N$. The most usual ways to construct $\bar{x}$ are the following:

- Padding time series with zeroes: $\bar{x}_i = x_i$ if $i \in \{0, \ldots, N\}$, and $\bar{x}_i = 0$ otherwise. In this case, (3) and (5) are equivalent, having $\bar{\mathcal{W}}x_n(s) = \mathcal{W}x_n(s)$.
- Using a periodization of the original time series: $\bar{x}_i = x_{i \bmod (N+1)}$.
- Using a symmetric catenation of the original time series: $\bar{x}_i = x_{i \bmod (N+1)}$ if $\lfloor \frac{i}{N+1} \rfloor$ is even, and $\bar{x}_i = x_{(N-i) \bmod (N+1)}$ if $\lfloor \frac{i}{N+1} \rfloor$ is odd.

Depending on the nature of x, it may be preferable to use one construction or another for minimizing the undesirable border effects (see Figure 2). For example, a periodization is advised for stationary short time series, and symmetric catenation for non-stationary short time series. On the other hand, for long time series, border effects are less important and then we can just pad with zeroes, i.e. use the original CWT given in (3).

How the border effects are treated by function `cwt_wst` is determined via the `border_effects` parameter. Possible values for this parameter are `"BE"` (raw border effects, padding with zeroes), "PER" (periodization) and "SYM" (symmetric catenation), corresponding to the three options described above.

Remark 2.2.4 (Cone of influence)**.** The *cone of influence* (CoI) is defined by the scales for which border effects become important at each time. The field coi_maxscale of the output in cwt_wst function contains, for each time, the maximum scale at which border effects are negligible, and consequently determines the CoI.

In order to compute the CoI, we have to set a criterion to distinguish between relevant and negligible border effects. In Torrence and Compo (1998), the CoI is defined by the e-folding time for the autocorrelation of wavelet power spectrum (see the next section) at each scale s, and this e-folding time is chosen so that the wavelet power spectrum for a discontinuity at the edge drops by a factor e^{-2}. For Morlet and DoG wavelets, this e-folding time is $\sqrt{2}s$, and for Paul wavelets is $s/\sqrt{2}$.

For wavelets with symmetric modulus such as Morlet, Paul and DoG, the e-folding time at $s = 1$ is interpreted as a *wavelet radius* r_w that defines an *effective support* $[-r_w, r_w]$ for the mother wavelet. Therefore, the CoI is given by the scales from which the corresponding effective supports of the daughter wavelets $[u - sr_w, u + sr_w]$ are not entirely contained in the time domain.

The wavelet radius r_w determines the CoI in the different functions of this package by means of the parameter waverad. If it is NULL (default value) we consider $r_w = \sqrt{2}$ for Morlet and DoG wavelets, and $r_w = 1/\sqrt{2}$ for Paul wavelets, following Torrence and Compo (1998). On the other hand, we take $r_w = 0.5$ for Haar wavelet, i.e. we assume that its effective support is in fact its support. Nevertheless we can introduce a custom r_w for any wavelet, allowing us in this way to adjust the importance of border effects in the construction of the CoI. For example,

```
cwt <- cwt_wst(signal = signal, dt = h,
               wname = "DOG", wparam = 6, waverad = 2)
```

computes the CWT coefficients of signal for DoG wavelet with $m = 6$. Here, cwt$coi_maxscale is obtained assuming that the wavelet radius is $r_w = 2$. Note that the value of waverad does not affect the computation of the CWT coefficients.

3 Wavelet scalograms

The *wavelet power spectrum* of a signal $f \in L^2(\mathbb{R})$ at time u and scale $s > 0$ is defined as

$$\mathcal{WPS}f(u,s) = |\mathcal{W}f(u,s)|^2. \tag{6}$$

Analogously to (6), the wavelet power spectrum of a time series x at scale $s > 0$ is given by the sequence

$$\mathcal{WPS}x_n(s) = |\mathcal{W}x_n(s)|^2, \tag{7}$$

where $n = 0, \ldots, N$.

We can plot the wavelet power spectrum of a time series x at a given set of scales through function cwt_wst if the parameter makefigure is TRUE (default value). There are other parameters regarding this plot:

- time_values is a vector that provides customized values in the time axis.
- energy_density is a logical parameter. If it is TRUE, it is plotted the wavelet power spectrum divided by the scales, according to Remark 2.2.2. By default, it is FALSE.
- figureperiod is a logical parameter that indicates if they are represented periods or scales in the y-axis (see Remark 2.2.1). By default, it is TRUE.
- xlab,ylab,main,zlim are parameters to customize the figure.

For example,

```
h <- 1 / 12
time1 <- seq(from = 1920, to = 1970 - h, by = h)
time2 <- seq(from = 1970, to = 2020, by = h)
signal <- c(sin(pi * time1), sin(pi * time2 / 2))
cwt_a <- cwt_wst(signal = signal, dt = h, time_values = c(time1, time2))
cwt_b <- cwt_wst(signal = signal, dt = h, time_values = c(time1, time2),
                 energy_density = TRUE)
```

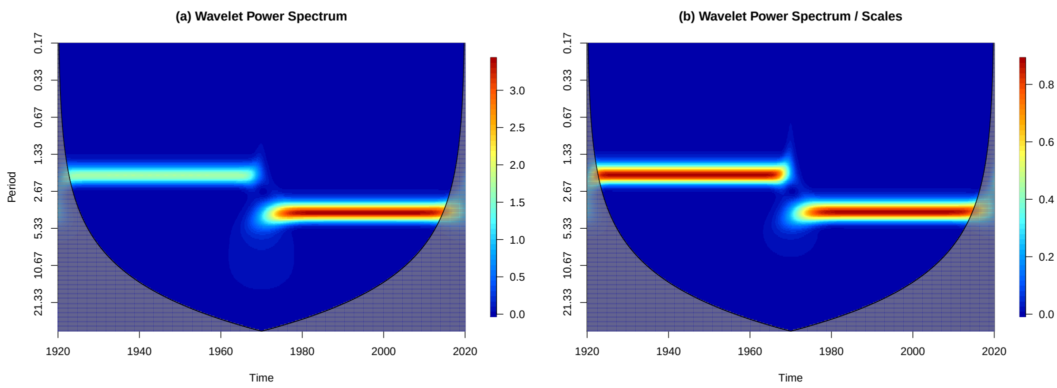

Figure 3: Wavelet power spectra of `signal`, non corrected (a) and corrected (b) via parameter `energy_density`. The CoI is the shadowed region. This signal is the concatenation of two pure sinusoidal time series with the same amplitude and different periods. Note that even though both time series have the same amplitude, when the coefficients are not corrected, the magnitude of the wavelet power spectrum is biased in favour of large scales, while in the corrected version, this bias is not present.

plots Figure 3 (a) and (b) respectively. In this figure it is shown how the wavelet power spectrum is biased in favour of large scales, as it is pointed out in Remark 2.2.2. The parameter `energy_density` corrects it and so, values for different scales become comparable. Note that `energy_density` only affects the plot and hence, `cwt_a` is identical to `cwt_b`.

The *scalogram* of f at scale s is defined as

$$\mathcal{S}f(s) = \left(\int_{-\infty}^{+\infty} |\mathcal{W}f\,(u,s)\,|^2\,\mathrm{d}u \right)^{1/2}. \tag{8}$$

It gives the contribution of each scale to the total "energy" of the signal and so, the notion of scalogram here is analogous to the spectrum of the Fourier transform. It is important to note that the term "scalogram" is often used to refer the wavelet power spectrum, but in this package, we call "scalogram" to (8).

If f is a finite length signal with time domain $I = [a, b]$, it is usual to consider a normalized version of the scalogram for comparison purposes, given by

$$\mathcal{S}f(s) = \left(\frac{1}{b-a} \int_{a}^{b} |\mathcal{W}f\,(u,s)\,|^2\,\mathrm{d}u \right)^{1/2}. \tag{9}$$

Hence, according to (9), the *(normalized) scalogram* of x at scale s is given by

$$\mathcal{S}x(s) = \left(\frac{1}{N+1} \sum_{i=0}^{N} |\mathcal{W}x_i(s)|^2 \right)^{1/2}. \tag{10}$$

The normalization coefficient $1/(N+1)$ in (10) allows us to compare scalograms of time series with different lengths.

We can compute the (normalized) scalograms of a time series x at a given set of scales by means of function `scalogram`. This function follows the same rules as `cwt_wst` regarding the data entry, construction of scales, choice of the wavelet, border effects and application of Fourier factor. So, parameters `signal`, `dt`, `scales`, `powerscales`, `wname`, `wparam`, `waverad`, `border_effects`, `makefigure`, `figureperiod`, `xlab`, `ylab` and `main` are analogous to those in function `cwt_wst` with same default values. On the other hand, parameter `energy_density` is `TRUE` by default. For example,

```
sc_a <- scalogram(signal = signal, dt = h,
                  energy_density = FALSE)
```

computes the (normalized) scalogram of `signal` given by (10) at a base 2 power scales set constructed automatically and plots Figure 4 (a). The output `sc_a` is a list with the following fields:

- `scalog` is a vector of length `length(scales)` with the values of the (normalized) scalogram at each scale.
- `energy` is the total energy of `scalog` (i.e. its L^2-norm) if parameter `energy_density` is `TRUE`.

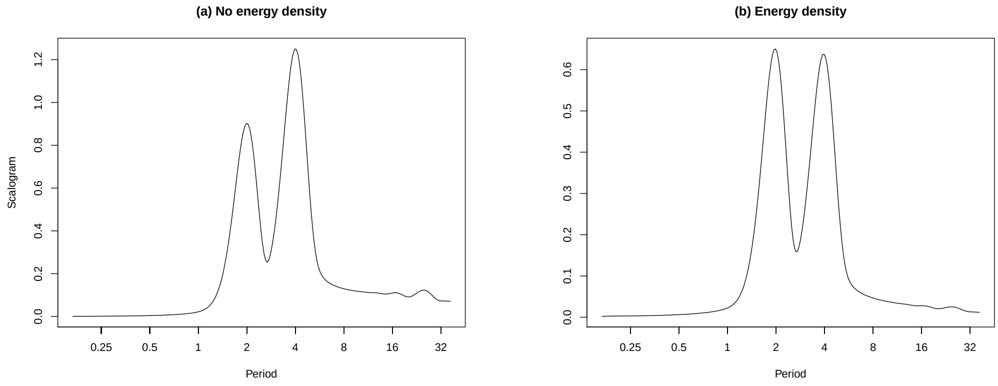

Figure 4: Original scalogram (a) and corrected scalogram representing an energy density measure (b), both relative to `signal`. As in Figure 3, it can be seen how the scalogram is biased in favour of large scales when the parameter `energy_density` is `FALSE` (plot (a)).

- `scales` and `fourier_factor` are analogous to those in the output of function `cwt_wst`.

If parameter `energy_density` is set to `TRUE` (default value), then the scalogram is divided by the square root of the scales, converting it into an energy density measure (see Remark 2.2.2). For example, if we write

```
sc_b <- scalogram(signal = signal, dt = h)
```

it plots Figure 4 (b), and then `sc_b$scalog` is in fact `sc_a$scalog / sqrt(scales)`.

Inner Scalogram

Given a compactly supported wavelet ψ and f a finite length signal with time domain $I = [a, b]$, the *(normalized) inner scalogram* of f at scale s is defined as

$$\mathcal{S}^{\text{inner}} f\,(s) = \left(\frac{1}{d(s) - c(s)} \int_{c(s)}^{d(s)} |\mathcal{W} f\,(u,s)\,|^2 \mathrm{d}u \right)^{1/2}, \tag{11}$$

where $[c(s), d(s)]$ is the maximal subinterval in I for which the support of $\psi_{u,s}$ is included in I for all $u \in [c(s), d(s)]$. Hence, according to (11), the *(normalized) inner scalogram* of x at scale s is given by

$$\mathcal{S}^{\text{inner}} x(s) = \left(\frac{1}{n_2(s) - n_1(s) + 1} \sum_{i=n_1(s)}^{n_2(s)} |\mathcal{W} x_i(s)|^2 \right)^{1/2},$$

where $\{n_1(s), \ldots, n_2(s)\}$ is the maximal subset of time indices for which the support of $\psi_{i,s}$ is included in $[t_0, t_N]$ for all $i \in \{n_1(s), \ldots, n_2(s)\}$.

This concept of inner scalogram can be extended to wavelets that do not have compact support, considering the effective support (see Remark 2.2.4) instead of the support. But we have to take into account that in this case, some theoretical results exposed in Benítez et al. (2010) may not hold.

We can compute the (normalized) inner scalograms of a time series x at a given set of scales by means of function `scalogram` setting the parameter `border_effects` equal to `"INNER"`. Since Morlet, Paul and DoG wavelets are not compactly supported, it is considered the effective support given by the wavelet radius r_w.

Windowed Scalogram

The *(normalized) windowed scalogram* of f centered at time t with time radius $\tau > 0$ at scale s is defined as

$$\mathcal{WS}_\tau f(t,s) = \left(\frac{1}{2\tau} \int_{t-\tau}^{t+\tau} |\mathcal{W} f\,(u,s)\,|^2 \,\mathrm{d}u \right)^{1/2}. \tag{12}$$

It was introduced in Bolós et al. (2017) and it allows to determine the relative importance of the different scales around a given time point. According to (12), the *(normalized) windowed scalogram*

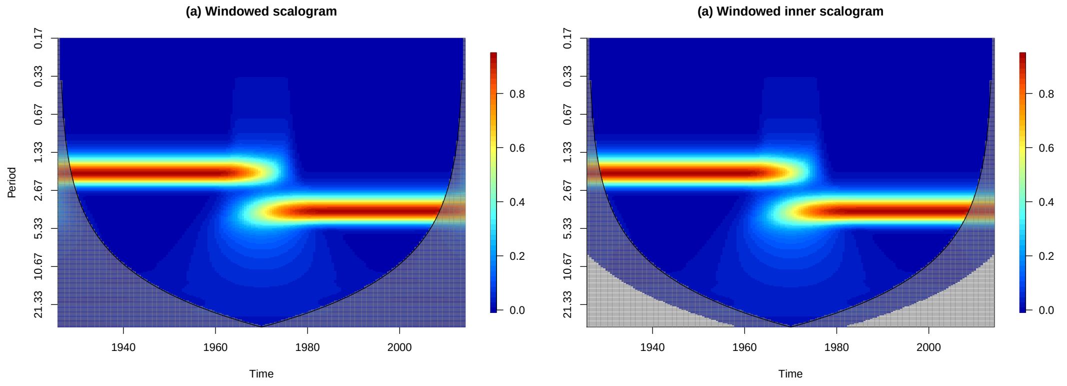

Figure 5: Windowed scalogram (a) and windowed inner scalogram (b) of `signal`. The CoI is the shadowed region and, for the inner scalogram, the region where the scalogram cannot be computed is coloured in gray.

of x with time index radius $\tau \in \mathbb{N}$ at scale s is given by the sequence

$$\mathcal{WS}_\tau x_n(s) = \left(\frac{1}{2\tau+1} \sum_{i=n-\tau}^{n+\tau} |\mathcal{W}x_i(s)|^2 \right)^{1/2}, \tag{13}$$

where $n = \tau, \ldots, N-\tau$. In the particular case of $\tau = 0$, (13) coincides with $|\mathcal{W}x_n(s)|$.

We can compute the (normalized) windowed scalograms of a time series x at a given set of scales by means of function `windowed_scalogram`. Parameters `signal`, `dt`, `scales`, `powerscales`, `wname`, `wparam`, `waverad`, `border_effects`, `energy_density`, `makefigure`, `figureperiod`, `xlab`, `ylab` and `main` are analogous to those in function `scalogram`, and parameters `time_values` and `zlim` are analogous to those in function `cwt_wst`. For example,

```
wsc <- windowed_scalogram(signal = signal, dt = h,
                          windowrad = 72, delta_t = 6,
                          time_values = c(time1, time2))
```

computes the (normalized) windowed scalograms of `signal` with time index radius $\tau =$ `windowrad` at a base 2 power scales set constructed automatically. Moreover, it plots Figure 5 (a). If `windowrad` is `NULL` (default value), then it is set to $\lceil (N+1)/20 \rceil$. The parameter `delta_t` is the index increment for the computation of the windowed scalograms, i.e. (13) is computed only for n from τ to $N-\tau$ by `delta_t`. If `delta_t` is `NULL` (default value) then it is taken $\lceil (N+1)/256 \rceil$.

The output `wsc` is a list with the following fields:

- `tcentral` is the vector of times at which the windows are centered, i.e. the times of the form t_n where n goes from τ to $N-\tau$ by `delta_t`.
- `wsc` is a matrix of size `length(tcentral)`×`length(scales)` containing the values of the windowed scalograms at each scale and at each central time.
- `windowrad` is the time index radius τ used.
- `scales`, `fourier_factor` and `coi_maxscale` are analogous to those in the output of function `cwt_wst`.

Windowed Inner Scalogram

The *(normalized) windowed inner scalogram* of f centered at time t with time radius $\tau > 0$ at scale s is defined as

$$\mathcal{WS}_\tau^{\text{inner}} f(t,s) = \left(\frac{1}{d(t,s)-c(t,s)} \int_{c(t,s)}^{d(t,s)} |\mathcal{W}f\,(u,s)\,|^2 \, du \right)^{1/2}, \tag{14}$$

where $[c(t,s), d(t,s)]$ is the maximal subinterval in $[t-\tau, t+\tau]$ for which the effective support of $\psi_{u,s}$ is included in I for all $u \in [c(t,s), d(t,s)]$. Then, the *(normalized) windowed inner scalogram* of x with time index radius $\tau \in \mathbb{N}$ at scale s is given by the sequence

$$\mathcal{WS}_{\tau}^{\text{inner}} x_n(s) = \left(\frac{1}{n_2(n,s) - n_1(n,s) + 1} \sum_{i=n_1(n,s)}^{n_2(n,s)} |\mathcal{W}x_i(s)|^2 \right)^{1/2}, \tag{15}$$

where $\{n_1(n,s), \ldots, n_2(n,s)\}$ is the maximal subset of time indices in $\{n - \tau, \ldots, n + \tau\}$ for which the effective support of $\psi_{i,s}$ is included in $[t_0, t_N]$ for all $i \in \{n_1(n,s), \ldots, n_2(n,s)\}$.

If `border_effects` is set to `"INNER"` in function `windowed_scalogram`, then the (normalized) windowed inner scalograms are computed. For example,

```
wsc <- windowed_scalogram(signal = signal, dt = h,
                          windowrad = 72, delta_t = 6,
                          border_effects = "INNER",
                          time_values = c(time1, time2))
```

computes the (normalized) windowed inner scalogram of `signal` with time index radius τ = `windowrad`, at a base 2 power scales set constructed automatically, and plots Figure 5 (b). Note that in this figure, the "CoI line" is not a real CoI line, because if we consider inner scalograms, border effects are negligible. This line represents, at each time t_n, the maximum scale s such that $n_1(n,s) = n - \tau$ and $n_2(n,s) = n + \tau$, and coincides with the CoI line of the (normalized) windowed scalogram.

4 Windowed scalogram difference

The *windowed scalogram difference* (WSD) is a wavelet tool, introduced in Bolós et al. (2017), whose main objective is to compare time series by means of their respective windowed scalograms.

In order to consider differences between scalograms, it is convenient to use base 2 power scales (Bolós et al., 2017) and hence, we must redefine them by making a change of variable. Thus, for example, from (10), the (normalized) scalogram of a time series x at *log-scale* k should be given by

$$\mathcal{S}x(k) = \left(\frac{1}{N+1} \sum_{i=0}^{N} |\mathcal{W}x_i(2^k)|^2 \right)^{1/2}, \tag{16}$$

where $k \in \mathbb{R}$ is the binary logarithm of the scale. From now on, k will denote log-scales of scales s in the sense that $2^k = s$.

The windowed scalogram difference of two signals $f, g \in L^2(\mathbb{R})$ centered at (t,k) with time radius $\tau > 0$ and log-scale radius $\lambda > 0$ is defined as

$$\mathcal{WSD}_{\tau,\lambda} fg(t,k) = \left(\int_{k-\lambda}^{k+\lambda} \left(\frac{\mathcal{WS}_\tau f(t,\kappa) - \mathcal{WS}_\tau g(t,\kappa)}{\mathcal{WS}_\tau f(t,\kappa)} \right)^2 \mathrm{d}\kappa \right)^{1/2}. \tag{17}$$

The commutative version of (17) is given by

$$\frac{1}{2} \left(\int_{k-\lambda}^{k+\lambda} \left(\frac{\mathcal{WS}_\tau f(t,\kappa)^2 - \mathcal{WS}_\tau g(t,\kappa)^2}{\mathcal{WS}_\tau f(t,\kappa)\, \mathcal{WS}_\tau g(t,\kappa)} \right)^2 \mathrm{d}\kappa \right)^{1/2}. \tag{18}$$

From (17), the *windowed scalogram difference* (WSD) of two time series x, y centered at log-scale k with time index radius $\tau \in \mathbb{N}$ and log-scale radius $\lambda > 0$ is given by the sequence

$$\mathcal{WSD}_{\tau,\lambda} xy_n(k) = \left(\int_{k-\lambda}^{k+\lambda} \left(\frac{\mathcal{WS}_\tau x_n(\kappa) - \mathcal{WS}_\tau y_n(\kappa)}{\mathcal{WS}_\tau x_n(\kappa)} \right)^2 \mathrm{d}\kappa \right)^{1/2}, \tag{19}$$

where $n = \tau, \ldots, N - \tau$. However, in practice, we work with a finite interval of log-scales that is discretized into $k_0, \ldots, k_M$ with constant step. Thus, we can adapt (19) to this situation so that it can be written as

$$\mathcal{WSD}_{\tau,\lambda} xy_n(k_m) = \left(\frac{2\lambda+1}{m_2 - m_1 + 1} \sum_{i=m_1}^{m_2} \left(\frac{\mathcal{WS}_\tau x_n(k_i) - \mathcal{WS}_\tau y_n(k_i)}{\mathcal{WS}_\tau x_n(k_i)} \right)^2 \right)^{1/2}, \tag{20}$$

where $\lambda \in \mathbb{N}$ is the log-scale index radius, $m_1 = \max\{0, m - \lambda\}$ and $m_2 = \min\{M, m + \lambda\}$. The factor $\frac{2\lambda+1}{m_2 - m_1 + 1}$ is added to counteract the "border effects" in the log-scale interval that appear when

$m - \lambda < 0$ or $m + \lambda > M$, because in these cases, the number of addends is less than $2\lambda + 1$. Moreover, according to (18), a commutative version of (20) can be also considered.

We can compute the WSD (20) (or its commutative version) of two time series x, y of the same length and time step by means of function wsd. Parameters dt, windowrad, delta_t, wname, wparam, waverad, border_effects, energy_density, makefigure, time_values, figureperiod, xlab, ylab, main and zlim are analogous to those in function windowed_scalogram. For example,

```
set.seed(12345) # For reproducibility
N <- 1500
time <- 0:N
signal1 <- rnorm(n = N + 1, mean = 0, sd = 0.2) + sin(time / 10)
signal2 <- rnorm(n = N + 1, mean = 0, sd = 0.2) + sin(time / 10)
signal2[500:1000] = signal2[500:1000] + sin((500:1000) / 2)
wsd <- wsd(signal1 = signal1, signal2 = signal2,
           windowrad = 75, rdist = 14)
```

computes the commutative WSD of signal1 and signal2 centered at a log-scales set $\{k_0, \ldots, k_M\}$ constructed automatically, with time index radius $\tau =$ windowrad and log-scale index radius $\lambda =$ rdist. If windowrad is NULL (default value) then it is set to $\lceil (N+1)/20 \rceil$, and if rdist is NULL (default value) then it is set to $\lceil (M+1)/20 \rceil$. The log-scales set can be defined by parameter scaleparam, that must be a vector of three elements with the minimum scale, the maximum scale and the number of suboctaves per octave. Moreover, logical parameter commutative, whose default value is TRUE, determines if it is computed the commutative version of the WSD.

Remark 2.4.1 (Normalization)**.** The WSD compares the patterns of the windowed scalograms of two time series determining if they give similar weights (or energy) to the same scales. Another tool for comparing two time series is the squared wavelet coherence (Torrence and Compo, 1998; Torrence and Webster, 1999), that measures the local linear correlation between them. So, these tools focus on different aspects: while the squared wavelet coherence does not take into account the magnitudes in the signals, for the WSD they are crucial. In fact, the WSD has sense only when the two time series considered are expressed in the same unit of measure or they are dimensionless. Otherwise, it will be necessary to somehow normalize the signals, but depending on the normalization method, some artifices could appear. For example, we can normalize the signals so that their scalograms have the same energy and, in this way, we can compare the relative contributions of each scale to the total energy. Another option is to normalize the signals so that their scalograms attain the same maximum value. Finally, it could be also useful to normalize the signals so that their scalograms reach the same value at a given reference scale.

The normalization method can be chosen through parameter normalize in function wsd. It can be set to "NO" (default value), "ENERGY", "MAX" or "SCALE", according to each normalization method exposed in Remark 2.4.1. In this last option, the reference scale must be given by parameter refscale.

Remark 2.4.2 (Near zero scalogram values)**.** Some problems can arise in the WSD when a scalogram is zero or close to zero for a given log-scale because we are computing relative differences and hence, the WSD can take extremely high values or produce numerical errors. If we consider absolute differences this would not happen but, on the other hand, it would not be appropriate for scalogram values not close to zero. A solution is to establish a threshold for the scalogram values above which a relative difference is computed, and below which a difference proportional to the absolute difference is computed (the proportionality factor would be determined by requiring continuity). This threshold can be interpreted as the relative amplitude of the noise in the scalograms.

Another solution is to substitute the original windowed scalograms $\mathcal{WS}_\tau x_n(s)$, $\mathcal{WS}_\tau y_n(s)$ by

$$C + \left(1 - \frac{C}{\max}\right) \mathcal{WS}_\tau x_n(s), \qquad C + \left(1 - \frac{C}{\max}\right) \mathcal{WS}_\tau y_n(s), \tag{21}$$

where

$$\max = \max_{n,s} \left\{ \mathcal{WS}_\tau x_n(s), \mathcal{WS}_\tau y_n(s) \right\},$$

and $C \geq 0$ is a relatively small value, called *compensation* (see Figure 6).

Parameters wscnoise and compensation of function wsd allow us to deal with the near zero scalogram problem mentioned in Remark 2.4.2. The first one is a value in $[0, 1]$ and establishes the threshold from which a relative difference is computed. As particular cases, if it is 0 then relative differences are always done, and if it is 1 then absolute differences are always done. The default value is set to 0.02. The second one determines the compensation C of (21), which is set to 0 by default.

In practical situations, signals will be usually affected by random noises. Therefore it is necessary to determine whether the results obtained with the WSD are statistically significant or not. In this

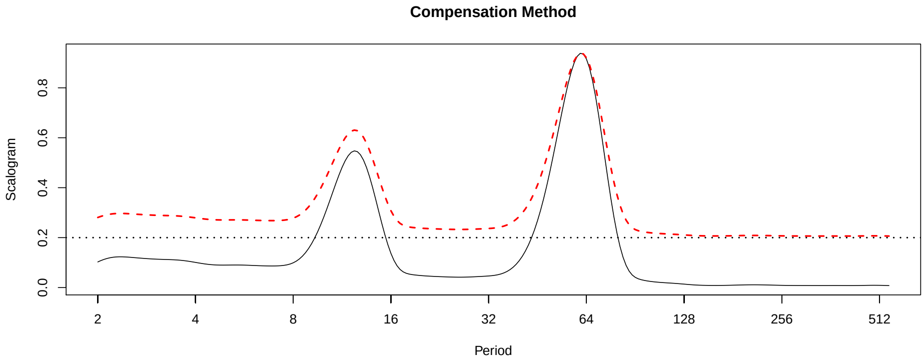

Figure 6: Illustration of the compensation method exposed in Remark 2.4.2 to deal with near zero scalogram values in the computation of the WSD. Original scalogram of `signal2` (solid black line) is transformed into the compensated scalogram (dashed red line) for a compensation parameter $C = 0.2$.

package, we perform Monte Carlo simulations of the WSD (with the same parameter values) of random signals following a normal distribution with the same mean and standard deviation as the original ones. Then, we find the 95% and 5% quantiles to determine significantly high and low values respectively. The number of Monte Carlo simulations is set by parameter `mc_nrand` in function `wsd`, whose default value is 0 (no significant contours are computed). For example,

```
wsd <- wsd(signal1 = signal1, signal2 = signal2,
           mc_nrand = 100, parallel = TRUE)
```

computes the same WSD as before, but determines which values are significant using 100 Monte Carlo simulations, plotting Figure 7. Parameter `parallel` enables parallel computations improving considerably the execution time for high values of `mc_nrand`.

Finally, the output of `wsd` is a list with the following fields:

- `wsd` is a matrix of size `length(tcentral)`$\times$`length(scales)` containing the values of the WSD at each scale and at each central time.
- `rdist` is the log-scale index radius λ used.
- `signif95` and `signif05` are logical matrices of size `length(tcentral)`$\times$`length(scales)` that determine if the corresponding values of the `wsd` matrix are significantly high or low respectively, following the 95% and 5% quantiles method described above.
- `tcentral`, `scales`, `windowrad`, `fourier_factor` and `coi_maxscale` are analogous to those in the output of function `windowed_scalogram`.

With respect to the output image, it is plotted the base 2 logarithm of the inverse of the WSD because in this way high values represent small differences (i.e. high similarity) and low values represent large differences (i.e. low similarity) (Bolós et al., 2017).

5 Scale index and windowed scale index

Periodicity is one of the most basic characteristics to be determined in a time series study. Mathematically, the definition is clear: a time series f is periodic of period T whenever $f(t+T) = f(t)$ for all t, and a time series that fails to be periodic is a non-periodic signal. However, within this definition, there are very different types of non-periodic signals (e.g. stochastic, quasi-periodic, chaotic signals), and an interesting question to analyze is how much non-periodic a time series is. Within this regard, the *scale index* and the *windowed scale index* (Benítez et al., 2010; Bolós et al., 2020) are two wavelet tools that give a satisfactory answer to this question.

The *scale index* of a signal $f \in L^2(\mathbb{R})$ in the scale interval $[s_0, s_1]$ is defined as the quotient

$$i_{\text{scale}} f = \frac{\mathcal{S}f(s_{\min})}{\mathcal{S}f(s_{\max})}, \tag{22}$$

where $s_{\max} \in [s_0, s_1]$ is the smallest scale such that $\mathcal{S}f(s) \le \mathcal{S}f(s_{\max})$ for all $s \in [s_0, s_1]$, and $s_{\min} \in [s_{\max}, 2s_1]$ is the smallest scale such that $\mathcal{S}f(s_{\min}) \le \mathcal{S}f(s)$ for all $s \in [s_{\max}, 2s_1]$ (Benítez et al., 2010;

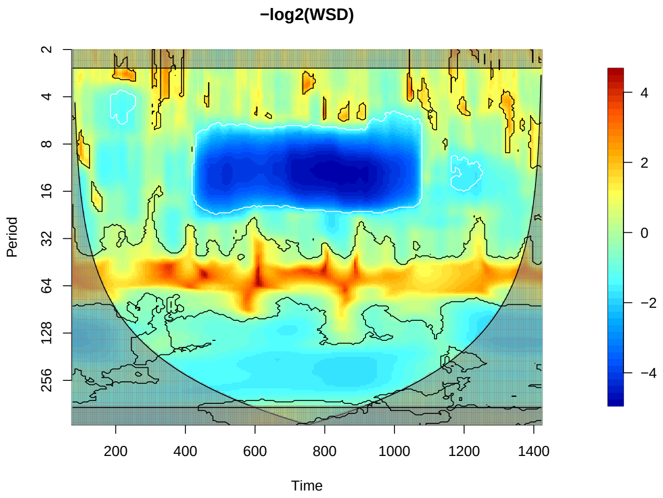

Figure 7: Base 2 logarithm of the inverse of the commutative WSD of `signal1` and `signal2` centered at a log-scales set constructed automatically, with time index radius $\tau = 75$ and log-scale index radius $\lambda = 14$. The significant contours are plotted in black (significantly high) and white (significantly low) lines, using 100 Monte Carlo simulations. Both time series are the same sinusoidal signal of period 20π plus different white noises with the same amplitude. Moreover, `signal2` has been manually modified for $500 \leq t \leq 1000$, with the addition of another pure sin signal of period 4π. The red band around period 20π corresponds to the period both signals have in common while the dark blue region around period 4π corresponds to the period both differ.

Bolós et al., 2020). Hence, according to (22), the *scale index* of a time series x in the scale interval $[s_0, s_1]$ is given by

$$i_{\text{scale}}x = \frac{\mathcal{S}x(s_{\min})}{\mathcal{S}x(s_{\max})}, \tag{23}$$

where $s_{\max}$ and $s_{\min}$ are defined analogously.

The scale index is a quantity in $[0, 1]$ and measures the degree of non-periodicity of a signal in a given scale interval $[s_0, s_1]$: It is close to zero for periodic and quasi-periodic signals, and close to one for highly non-periodic signals. The choice of the scale interval $[s_0, s_1]$ is very important, and it should contain all the relevant scales that we want to study.

Remark 2.5.1 (No energy density)**.** The correction exposed in Remark 2.2.2 should not be carried out because the scalogram of a white noise signal is more or less constant at all scales giving a scale index close to 1 for any scale interval $[s_0, s_1]$, and this is the property that we want to preserve. If, on the other hand, we apply the correction for converting the scalogram into an "energy density" measure, the scale index of a white noise signal would tend to zero as we increase s_1 and this is not desirable (Bolós et al., 2020).

We can compute the scale index of a time series x by means of function `scale_index`. Parameters `signal`, `dt`, `scales`, `powerscales`, `wname`, `wparam`, `waverad`, `border_effects`, `makefigure`, `figureperiod`, `xlab`, `ylab` and `main` are analogous to those in function `scalogram`. Note that, according to Remark 2.5.1, there is no parameter `energy_density` because scalograms must be computed without this correction. For example,

```
set.seed(12345) # For reproducibility
N <- 999
h <- 1 / 8
time <- seq(from = 0, to = N * h, by = h)
signal_si <- sin(pi * time) + rnorm(n = N + 1, mean = 0, sd = 2)
s0 <- 1
s1 <- 4
si <- scale_index(signal = signal_si, dt = h,
                  scales = c(s0, 2 * s1, 24), s1 = s1,
                  border_effects = "INNER", makefigure = FALSE)
```

computes the scale index of `signal_si` in the scale interval $[s_0, s_1]$ where $s_0 = 1$ and $s_1 = 4$. The parameter `scales` determines the scales set at which the scalograms are computed: In this case, it is

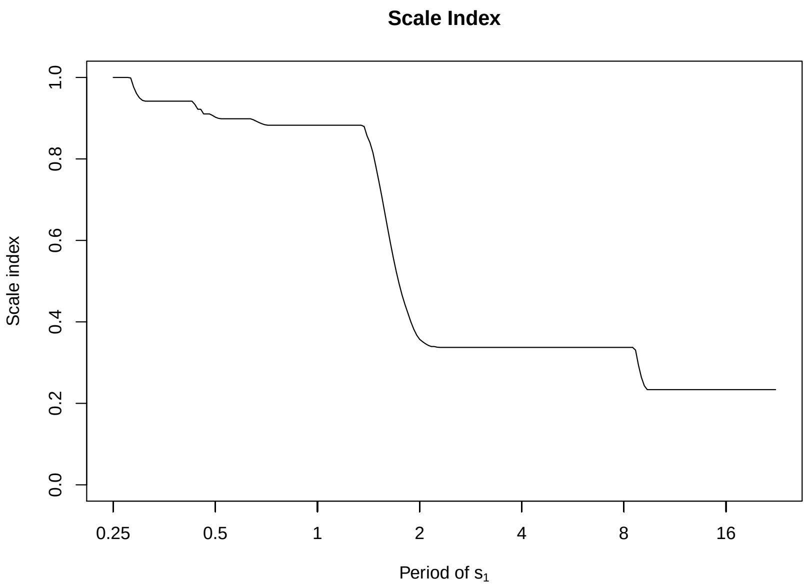

Figure 8: Scale indices of `signal_si` in scale intervals $[s_0, s_1]$ where s_0 is the scale whose equivalent Fourier period is 0.25 and s_1 varies. The signal is a pure sin of period 2 plus a noise term. Thus, for values of s_1 lower than 2, the scale index is very high because the scalogram still has not considered this period. At $s_1 = 2$, there is a sudden drop in the scale index and from that point onwards, as s_1 increases, the scale index decreases until it reaches a stable plateau (at 0.2 approximately) for large values of s_1.

a base 2 power scales set from $s_a = s_0$ to $s_b = 2s_1$ with 24 suboctaves per octave. Note that we take $s_b = 2s_1$ according to the definition of the scale index, because s_b can not be lesser than $2s_1$ and there is no need for s_b to be greater than $2s_1$. Moreover, function `scale_index` takes s_0 equal to the lowest scale s_a always. If `scales = NULL` (default value), then the scalograms are computed at an automatically constructed set of scales with s_a equal to the scale whose equivalent Fourier period is $2h$, and $s_b = 2s_1$.

We can also compute the scale indices of a signal in scale intervals $[s_0, s_1]$ for different values of s_1 assigning a vector of scales to the parameter `s1`. Thus,

```
maxs1 <- 4
si <- scale_index(signal = signal_si, dt = h, scales = c(s0, 2 * maxs1, 24),
                  s1 = pow2scales(c(s0, maxs1, 24)),
                  border_effects = "INNER")
```

computes the scale indices of `signal_si` in scale intervals $[s_0, s_1]$ where $s_0 = 1$ and s_1 varies in a base 2 power scales set from 1 to 4 with 24 suboctaves per octave. Moreover, if `s1 = NULL` (default value), then `s1` is automatically computed as a base 2 power scales set from s_0 to $s_b/2$. If `scales` is also `NULL`, then $s_b = Nh/2r_w$ as usual, where r_w is the corresponding wavelet radius (see Remark 2.2.4). Hence,

```
si <- scale_index(signal = signal_si, dt = h, border_effects = "INNER")
```

computes the scale indices of `signal_si` in scale intervals $[s_0, s_1]$ where s_0 is the scale whose equivalent Fourier period is $2h$ and s_1 varies in a base 2 power scales set from s_0 to Nh/r_w. Moreover, it returns a plot like Figure 8. It is important to remark that if `s1` are not base 2 power scales then `powerscales` must be `FALSE`. For example,

```
si <- scale_index(signal = signal_si, dt = h,
                  s1 = seq(from = s0, to = maxs1, by = 0.1),
                  powerscales = FALSE, border_effects = "INNER")
```

computes the scale indices of `signal_si` in scale intervals $[s_0, s_1]$ where $s_0 = 1$ and s_1 varies linearly from 1 to 4 with step 0.1. In this case, since `scales` are not given, they are constructed automatically in a linear form, since `powerscales` must be `FALSE`.

Alternatively, we can compute the scale indices directly from a scalogram instead of giving the original signal by means of parameter `scalog`. In this case, we must give the scales at which the scalogram has been computed. Thus, we can compute the scale indices of an artificially constructed

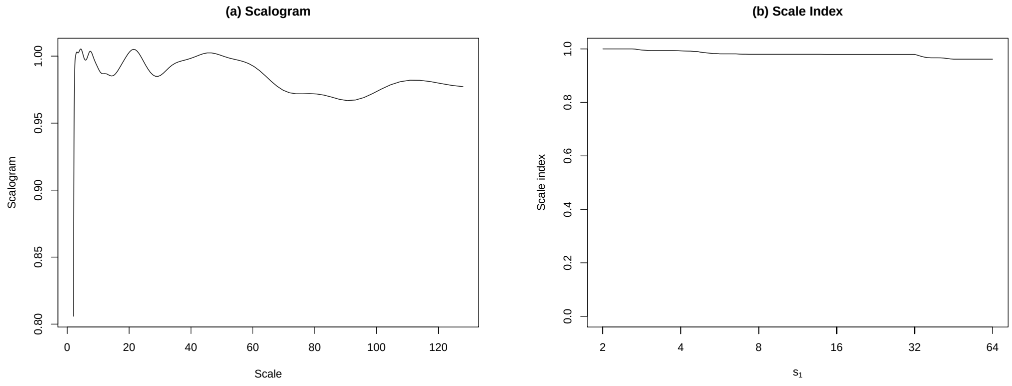

Figure 9: (a) Average of 100 scalograms of noise signals. (b) Scale indices computed from this averaged scalogram. The scale indices are very close to 1, indicating a high degree of non-periodicity.

scalogram which does not necessarily have to correspond to any signal. For example, we can compute the scale indices corresponding to the average of 100 scalograms of noise signals:

```
set.seed(12345) # For reproducibility
N <- 1000
nrand <- 100
X <- matrix(rnorm(N * nrand), nrow = N, ncol = nrand)
scales = pow2scales(c(2, 128, 24))
ns = length(scales)
sc_list <- apply(X, 2, scalogram, scales = scales, border_effects = "INNER",
                 energy_density = FALSE, makefigure = FALSE)
sc_matrix <- matrix(unlist(lapply(sc_list, "[[", "scalog")),
                    nrow = ns, ncol = nrand)
sc_mean <- apply(sc_matrix, 1, mean)
s1 = pow2scales(c(2, 64, 24))
si_mean <- scale_index(scalog = sc_mean, scales = scales, s1 = s1,
                       figureperiod = FALSE, plot_scalog = TRUE)
```

This code also returns figures like those in Figure 9. The logical parameter `plot_scalog` is used for plotting the scalogram from which the scale indices are computed.

The output of `scale_index` is a list with the following fields:

- `si` is a vector with the scale indices, for each value of s_1.
- `s0` is the scale s_0.
- `s1` is a vector with the scales s_1.
- `smax` and `smin` are vectors with the scales $s_{\max}$ and $s_{\min}$ respectively, for each value of s_1.
- `scalog` is the the scalogram $\mathcal{S}x$ from which the scale indices are computed.
- `scalog_smax` and `scalog_smin` are vectors with the scalogram values $\mathcal{S}x(s_{\max})$ and $\mathcal{S}x(s_{\min})$ respectively, for each value of s_1.
- `fourierfactor` is the scalar used to transform scales into Fourier periods (see Remark 2.2.1).

Windowed Scale Index

As was mentioned in the introduction, wavelet analysis is a very useful tool for non-stationary time series. If we are interested in analyzing the non-periodicity of a non-stationary time series, we should be aware that the scale index is going to give us a single number between 0 and 1 which represents the degree of non-periodicity of the signal in the overall time interval of interest. However we may be interested in how this degree of non-periodicity is changing along this interval. To this aim, Bolós et al. (2020) introduced the *windowed scale index*, which uses the windowed scalogram in order to obtain scale indices for different time and scale intervals.

In particular, the *windowed scale index* of f in the scale interval $[s_0, s_1]$ centered at time t with time radius $\tau > 0$ is defined as

$$wi_{\text{scale},\tau} f(t) = \frac{\mathcal{WS}_\tau f\,(t, s_{\min})}{\mathcal{WS}_\tau f\,(t, s_{\max})}, \tag{24}$$

where, analogously to (22), $s_{\max}$ is the smallest scale such that $\mathcal{WS}_\tau f(t,s) \leq \mathcal{WS}_\tau f(t, s_{\max})$ for all $s \in [s_0, s_1]$, and $s_{\min}$ is the smallest scale such that $\mathcal{WS}_\tau f(t, s_{\min}) \leq \mathcal{WS}_\tau f(t,s)$ for all $s \in [s_{\max}, 2s_1]$ (Bolós et al., 2020). Finally, according to (24), the *windowed scale index* of a time series x in the scale interval $[s_0, s_1]$ with time index radius $\tau \in \mathbb{N}$ is given by the sequence

$$wi_{\text{scale},\tau} x_n = \frac{\mathcal{WS}_\tau x_n\,(s_{\min})}{\mathcal{WS}_\tau x_n\,(s_{\max})},$$

where $n = \tau, \ldots, N - \tau$, and $s_{\max}, s_{\min}$ are defined analogously to (24).

Remark 2.5.2 (Inner scalograms)**.** Although in the computation of the scale index it is recommended the use of (normalized) inner scalograms in order to fulfil some theoretical results and avoid border effects, this recommendation is less important in the case of the windowed scale index, because for long time series and relatively small time radii there would be no relevant border effects in most of the windowed scalograms.

By means of function `windowed_scale_index`, we can compute the windowed scale index of a time series x. As usual, parameters `signal`, `dt`, `scales`, `powerscales`, `windowrad`, `delta_t`, `wname`, `wparam`, `waverad`, `border_effects`, `makefigure`, `time_values`, `figureperiod`, `xlab`, `ylab`, `main` and `zlim` are analogous to those in function `windowed_scalogram`. Moreover, parameter `s1` is analogous to that in function `scale_index`. For example,

```
set.seed(12345) # For reproducibility
s0 <- 1
s1 <- 4
signal1_wsi <- sin(pi * time[1:500]) + rnorm(n = 500, mean = 0, sd = 2)
signal2_wsi <- sin(pi * time[501:1000] / 2) + rnorm(n = 500, mean = 0, sd = 0.5)
signal_wsi <- c(signal1_wsi, signal2_wsi)
wsi <- windowed_scale_index(signal = signal_wsi, dt = h,
                            scales = c(s0, 2 * s1, 24), s1 = s1,
                            windowrad = 50,
                            time_values = time)
```

computes the windowed scale index of `signal_wsi` in a scale interval $[s_0, s_1]$ where $s_0 = 1$ and $s_1 = 4$. The time index radius τ is given by the parameter `windowrad`. If it is `NULL` (default value), then it is set to $\lceil (N+1)/20 \rceil$ that, in this case, coincides with the value of `windowrad`. Moreover, it returns a plot like Figure 10.

We can compute the windowed scale indices for different values of s_1 assigning a vector of scales to the parameter `s1`. It is important to remark that if `s1` are not base 2 power scales, then `powerscales` must be `FALSE`. If `s1 = NULL` and/or `scales = NULL` (default values), then they are automatically computed in the same way as it is done in function `scale_index`. So,

```
wsi <- windowed_scale_index(signal = signal_wsi, dt = h,
                            time_values = time)
```

computes the windowed scale indices of `signal_wsi` in scale intervals $[s_0, s_1]$ where s_0 is the scale whose equivalent Fourier period is $2h$ and s_1 varies in a base 2 power scales set from s_0 to Nh/r_w. The time index radius τ is taken automatically as $\lceil (N+1)/20 \rceil = 50$. It also returns a plot, like Figure 11.

Alternatively, we can compute the windowed scale indices directly from a windowed scalogram instead of giving the original signal by means of parameter `wsc`. This parameter must be equal to a matrix of size (number of central times)×(number of scales), as it is returned by the `windowed_scalogram` function. In this case, we must give the scales at which the windowed scalogram `wsc` has been computed and, in addition, we can give the cone of influence by means of parameter `wsc_coi`, that must be a vector containing the values of the maximum scale at each central time from which there are border effects in `wsc`. Thus, we can compute the windowed scale indices of an artificially constructed windowed scalogram as it was shown in the case of the scale index. Taking the same example, we can compute the windowed scale indices corresponding to the average of 100 windowed scalograms of noise signals:

```
set.seed(12345) # For reproducibility
N <- 1000
nrand <- 100
X <- matrix(rnorm(N * nrand), nrow = N, ncol = nrand)
scales = pow2scales(c(2, 128, 24))
ns = length(scales)
wsc_list <- apply(X, 2, windowed_scalogram, scales = scales,
                  energy_density = FALSE, makefigure = FALSE)
```

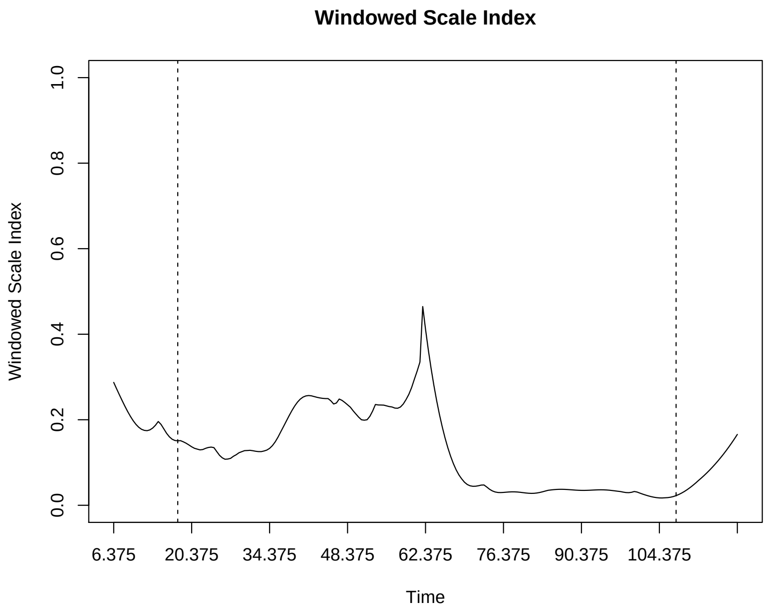

Figure 10: Windowed scale index of `signal_wsi` in a scale interval $[1, 4]$ and with time index radius $\tau = 50$. The dashed vertical lines represent the CoI limits. This time series is the concatenation of two sinusoidal signals of periods 2 and 4, modified with two white noises of different variance. In the first part, where the noise has a higher standard deviation, the windowed scale index is also higher. Moreover, it can be seen how the windowed scale index captures the moment of change in the noise.

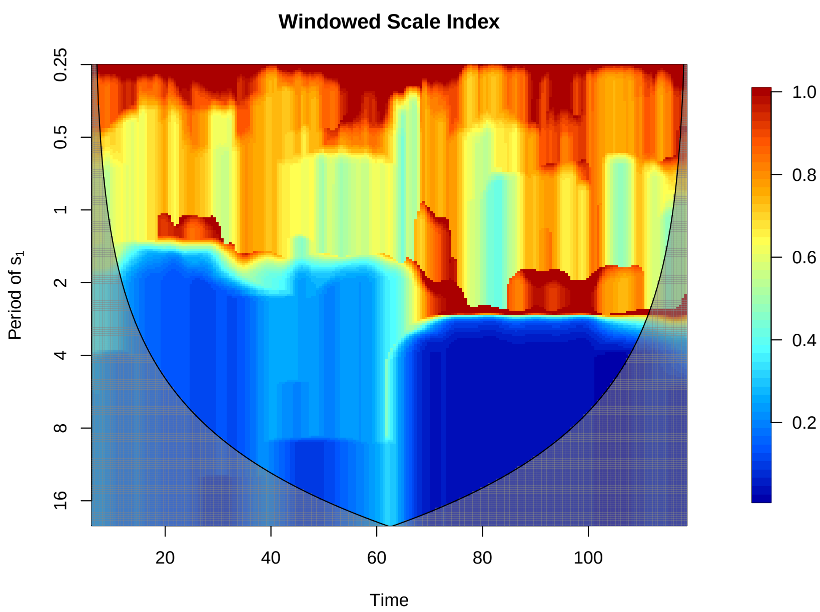

Figure 11: Windowed scale indices of `signal_wsi` in scale intervals $[s_0, s_1]$ where s_0 is the scale whose equivalent Fourier period is 0.25 and s_1 varies, with time index radius $\tau = 50$. This plot also shows that s_1 should be at least 4 for the scale indices to capture all relevant periods.

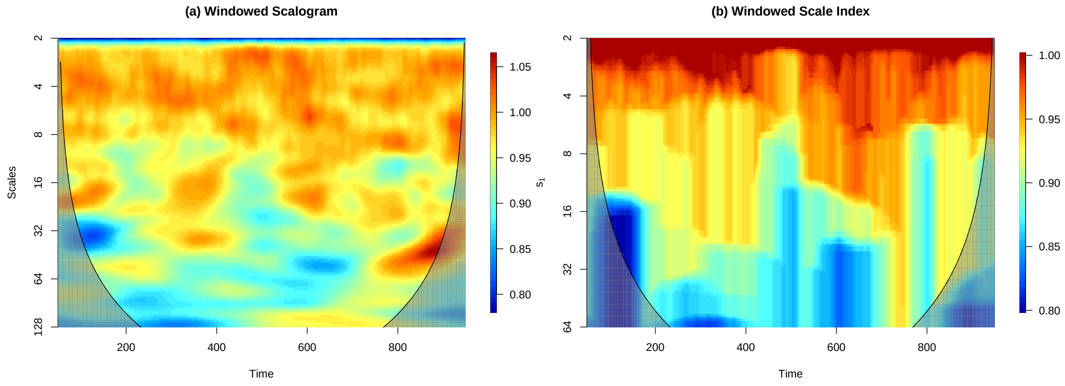

Figure 12: (a) Average of 100 windowed scalograms of noise signals. (b) Windowed scale indices computed from this averaged windowed scalogram. The windowed scale indices are always close to 1, indicating a high degree of non-periodicity.

```
tcentral <- wsc_list[[1]]$tcentral
ntc <- length(tcentral)
wsc_matrix <- array(unlist(lapply(wsc_list, "[[", "wsc")), c(ntc, ns, nrand))
wsc_mean <- apply(wsc_matrix, 1:2, mean)
wsc_coi <- wsc_list[[1]]$coi_maxscale
wsi_mean <- windowed_scale_index(wsc = wsc_mean, wsc_coi = wsc_coi,
                                 scales = scales, time_values = tcentral,
                                 figureperiod = FALSE, plot_wsc = TRUE)
```

This code also returns figures like those in Figure 12. The logical parameter `plot_wsc` is used for plotting the windowed scalogram from which the windowed scale indices are computed.

The output of `windowed_scale_index` is a list with the following fields:

- `wsi` is a matrix of size `length(tcentral)`×`length(s1)` with the windowed scale indices at each s_1 and at each central time.
- `wsc` is a matrix of size `length(tcentral)`×`length(scales)` with the windowed scalograms from which the windowed scale indices are computed. Note that scales greater than `2*max(s1)` are not necessary and they are internally removed from `scales`.
- `s0`, `s1`, `smax`, `smin`, `scalog_smax` and `scalog_smin` are analogous to those in the output of function `scale_index`.
- `tcentral`, `windowrad`, `fourierfactor` and `coi_maxscale` are analogous to those in the output of function `windowed_scalogram`.

6 Examples and applications

Windowed scalogram difference and clustering

As an application, we are going to show an example of how to define a dissimilarity measure from the windowed scalogram difference (WSD), which can then be applied to perform time series clustering. We are going to use the `interest.rates` time series from package **TSclust** (Montero and Vilar, 2014), which consists on 215 observations of the monthly long-term interest rates (10-year bonds) from January 1995 to November 2012 of several countries.

First, we define the *returns* time series for each country and then we compute the corresponding WSD of any pair of countries (see Figure 13). Next, we define the dissimilarity measure as the binary logarithm of the WSD mean plus 1 (in order to avoid negative distances). Finally, we plot the hierarchical clusters according to this dissimilarity measure (see Figure 14).

When defining the dissimilarity measure, we can restrict the WSD to only some areas instead of considering it entirely. For example, if we want to study the relationships between the different countries from the beginning of the century to the 2008 crisis at long-term scales, then we could only take into account the WSD area between 2001 and 2007, considering exclusively scales greater than 2 years. On the other hand, if border effects are relevant, only the WSD zone outside the cone of

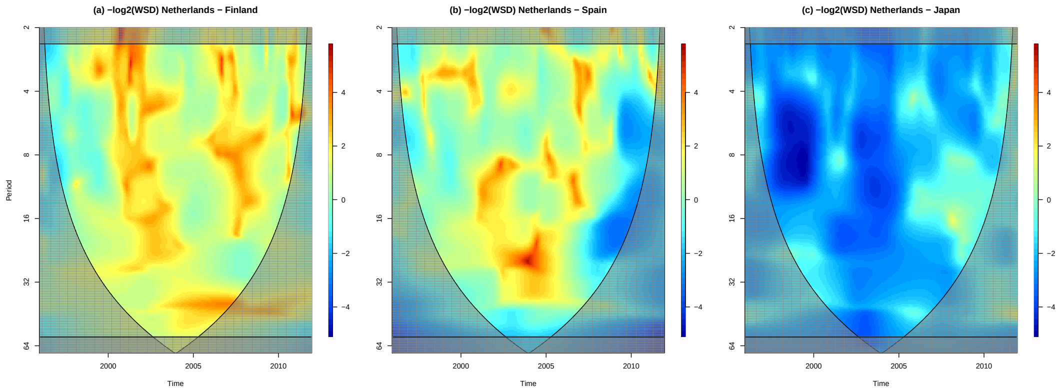

Figure 13: Plots of base 2 logarithms of the inverse of the commutative WSD of returns of Netherlands and (a) Finland, (b) Spain and (c) Japan. The corresponding dissimilarity measures of these pairs are 0.7395, 1.6279 and 2.819 respectively. Red zones indicate time-scale regions where the two signals are more similar, while blue zones correspond to less similarity between the signals. Note that Netherlands and Finland are two countries whose economies are similar in both time and scale (plot (a)) but, on the other hand, Netherlands has a very different economic behaviour than Japan (plot (c)). The big blue spot in plot (b) corresponds to the 2008 financial crisis, which hit Spain harder than the Netherlands.

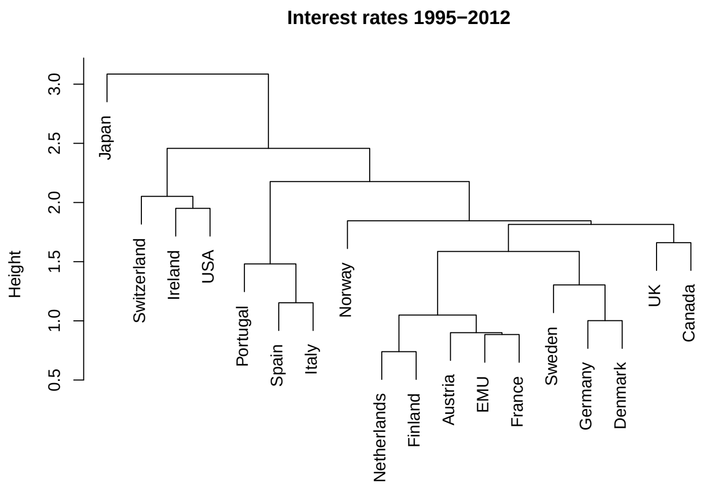

Figure 14: Hierarchical clustering of several countries according to their interest rates from 1995 to 2012. Similar countries are close together in the diagram.

influence could be considered. However, in our example, border effects do not substantially alter the clustering result.

```
library(wavScalogram)
library(TSclust)
data("interest.rates")
returns <- apply(interest.rates, MARGIN = 2, function(x) diff(log(x)))
Nsignals <- ncol(returns)
countries <- colnames(returns)
M <- Nsignals * (Nsignals - 1) / 2 # Number of pairings
auxpair <- vector(mode = "list", M)
k <- 1
for (i in 1:(Nsignals - 1)) {
  for (j in (i + 1):Nsignals) {
    auxpair[[k]] <- c(i, j)
    k <- k + 1
  }
}
fwsd <- function(x) wsd(signal1 = returns[, x[1]],
                        signal2 = returns[, x[2]],
                        makefigure = FALSE)
Allwsd <- lapply(auxpair, FUN = fwsd)
ntimes <- length(Allwsd[[1]]$tcentral)
nscales <- length(Allwsd[[1]]$scales)
area <- ntimes * nscales
meanwsd <- rep(0, M)
for (i in 1:M) {
  meanwsd[i] <- sum(Allwsd[[i]]$wsd) / area
}
d1 <- matrix(0, Nsignals, Nsignals)
d1[lower.tri(d1, diag = FALSE)] <- log2(meanwsd + 1)
dm1 <- as.dist(t(d1) + d1)
names(dm1) <- countries
plot(hclust(dm1), main = "Interest rates 1995-2012", xlab = "", sub = "")
```

Sunspots

In the next example we are going to illustrate the different tools of **wavScalogram** on the most famous *sunspot number time series* and how to use them in order to find the sunspots period, which is estimated to be around 11 years.

Let us consider the sunspot.month R dataset consisting on monthly numbers of sunspots from 1749 to present. Firstly, we can estimate the sunspots period by means of the scale at which the scalogram reaches its maximum. Using this criterion, we obtain that the sunspots period is 10.3254 approximately (see Figure 15 (b)). For this method, it is recommended that energy_density = TRUE since otherwise, larger scales would be over-estimated. Note that the wavelet power spectrum and the windowed scalograms present, as expected, horizontal bands of high values precisely around the scale 10.3254 (see Figure 15 (a) and (c)). Hence, they can be used to estimate a sunspot period that depends on time.

On the other hand, we can also estimate the sunspots period by means of the scale at which the scale index reaches its minimum. Contrary to the previous case and according to Remark 2.5.1, it is recommended that energy_density = FALSE for computing scale indices (see Figure 16). Using this criterion, we obtain that the sunspots period is 11.1215, approximately (see Figure 17 (a)). Therefore, the windowed scale index can also be used, analogously to the scalogram and the windowed scalogram, to estimate sunspots periods depending on time (see Figure 17 (b)).

Bibliography

E. Aldrich. *wavelets: Functions for Computing Wavelet Filters, Wavelet Transforms and Multiresolution Analyses*, 2020. URL https://CRAN.R-project.org/package=wavelets. R package version 0.3-0.2. [p164]

R. Benítez, V. J. Bolós, and M. E. Ramírez. A wavelet-based tool for studying non-periodicity. *Computers*

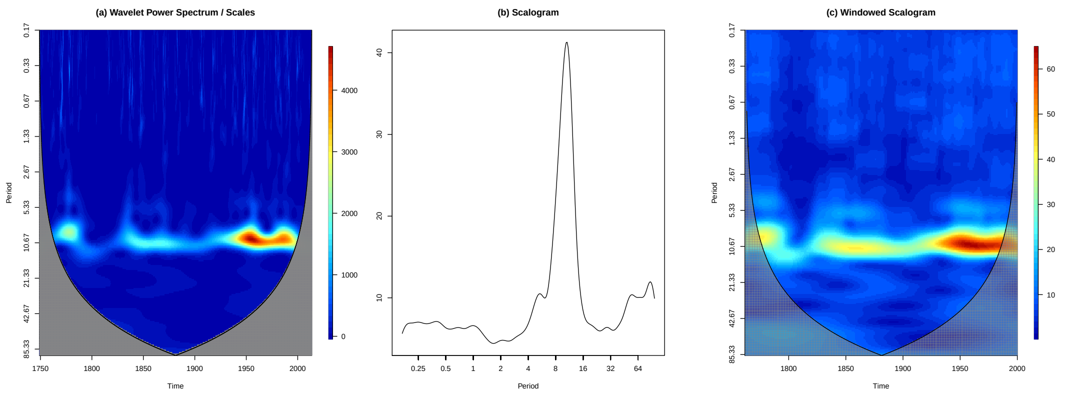

Figure 15: (a) Wavelet power spectrum divided by scales, (b) scalogram, and (c) windowed scalogram of the sunspots time series, with `energy_density = TRUE`. These plots show how the scalogram can be used for determining the sunspots period. In plots (a) and (c) the yellow-red band should be centered in the sunspots period, while in plot (b), this period should be given by the peak in the scalogram.

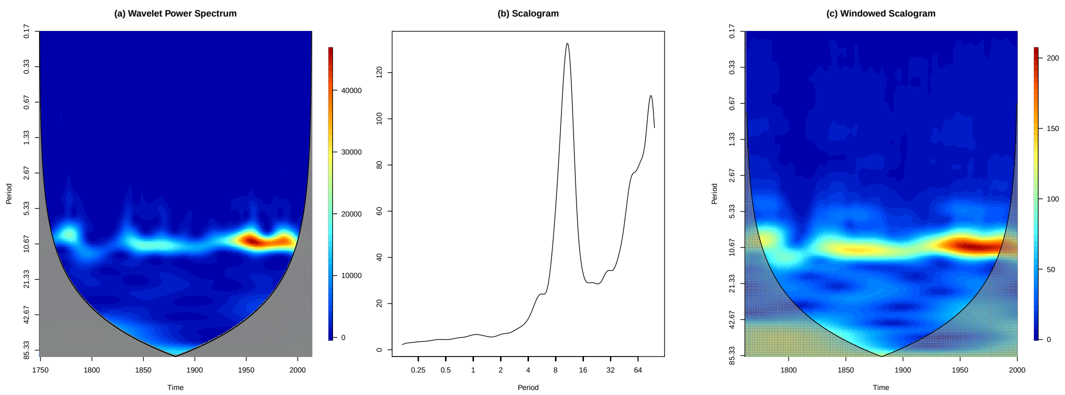

Figure 16: (a) Wavelet power spectrum, (b) scalogram, and (c) windowed scalogram of the sunspots time series, with `energy_density = FALSE`. These plots depict the same as Figure 15, but the bias in favour of large scales is present. Nevertheless, this is recommended for computing the corresponding scale indices.

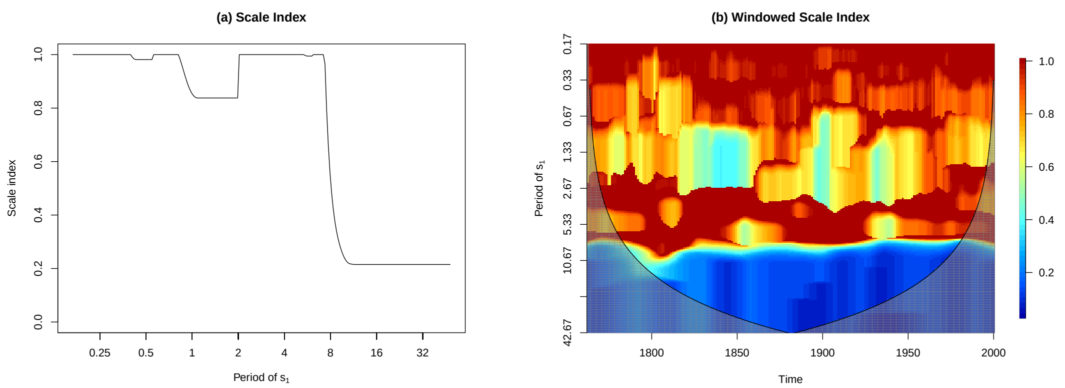

Figure 17: (a) Scale indices and (b) windowed scale indices of the sunspots time series. In these plots, the use of the scale indices to determine the sunspots period is depicted. The period is estimated by the minimum scale s_1 for which the scale indices are stabilized around lower values, presenting the transition from non-periodicity to a far more periodic signal. The windowed scale indices in plot (b) are specially useful for non-stationary time series because they can detect changes in the sunspots period over time.

& *Mathematics with Applications*, 60(3):634–641, 2010. URL https://doi.org/10.1016/j.camwa.2010.05.010. [p164, 170, 174]

V. J. Bolós and R. Benítez. *wavScalogram: Wavelet Scalogram Tools for Time Series Analysis*, 2021. URL https://CRAN.R-project.org/package=wavScalogram. R package version 1.1.1. [p164]

V. J. Bolós, R. Benítez, R. Ferrer, and R. Jammazi. The windowed scalogram difference: a novel wavelet tool for comparing time series. *Applied Mathematics and Computation*, 312:49–65, 2017. URL https://doi.org/10.1016/j.amc.2017.05.046. [p164, 170, 172, 174]

V. J. Bolós, R. Benítez, and R. Ferrer. A new wavelet tool to quantify non-periodicity of non-stationary economic time series. *Mathematics*, 8:844–859, 2020. URL https://doi.org/10.3390/math8050844. [p164, 174, 175, 177, 178]

R. Carmona and B. Torresani. *Rwave: Time-Frequency Analysis of 1-D Signals*, 2021. URL https://CRAN.R-project.org/package=Rwave. R package version 2.6-0. [p166]

I. Daubechies. *Ten Lectures on Wavelets*. CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics, 1992. ISBN 9780898712742. [p164]

M. Farge. Wavelet transforms and their applications to turbulence. *Annual Review of Fluid Mechanics*, 24:395–458, 1992. URL https://doi.org/10.1146/annurev.fl.24.010192.002143. [p165]

T. C. Gouhier, A. Grinsted, and V. Simko. *R package biwavelet: Conduct Univariate and Bivariate Wavelet Analyses*, 2021. URL https://github.com/tgouhier/biwavelet. (Version 0.20.21). [p164]

Y. Liu, X. San Liang, and R. H. Weisberg. Rectification of the bias in the wavelet power spectrum. *Journal of Atmospheric and Oceanic Technology*, 24(12):2093–2102, 2007. URL https://doi.org/10.1175/2007JTECHO511.1. [p166]

S. Mallat. *A Wavelet Tour of Signal Processing: The Sparse Way*. Academic Press, 2008. ISBN 978-0-08-092202-7. [p164, 165]

P. Montero and J. Vilar. TSclust: an R package for time series clustering. *Journal of Statistical Software*, 62(1):1–43, 2014. URL https://doi.org/10.18637/jss.v062.i01. [p180]

G. Nason. *wavethresh: Wavelets Statistics and Transforms*, 2022. URL https://CRAN.R-project.org/package=wavethresh. R package version 4.6.9. [p164]

P. Roebuck and Rice University's DSP group. *rwt: 'Rice Wavelet Toolbox' Wrapper*, 2022. URL https://CRAN.R-project.org/package=rwt. R package version 1.0.2. [p164]

A. Roesch and H. Schmidbauer. *WaveletComp: Computational Wavelet Analysis*, 2018. URL https://CRAN.R-project.org/package=WaveletComp. R package version 1.1. [p164]

D. Savchev and G. Nason. *hwwntest: Tests of White Noise using Wavelets*, 2018. URL https://CRAN.R-project.org/package=hwwntest. R package version 1.3.1. [p164]

S. A. C. Taylor, T. Park, and I. A. Eckley. Multivariate locally stationary wavelet analysis with the mvLSW R package. *Journal of Statistical Software*, 90(11):1–19, 2019. URL https://doi.org/10.18637/jss.v090.i11. [p164]

C. Torrence and G. Compo. A practical guide to wavelet analysis. *Bulletin of the American Meteorological Society*, 79(1):61–78, 1998. URL https://doi.org/10.1175/1520-0477(1998)079<0061:APGTWA>2.0.CO;2. [p165, 166, 168, 173]

C. Torrence and P. J. Webster. Interdecadal changes in the ENSO–monsoon system. *Journal of Climate*, 12(8):2679–2690, 1999. URL https://doi.org/10.1175/1520-0442(1999)012<2679:ICITEM>2.0.CO;2. [p173]

B. Whitcher. *waveslim: Basic Wavelet Routines for One-, Two-, and Three-Dimensional Signal Processing*, 2020. URL https://CRAN.R-project.org/package=waveslim. R package version 1.8.2. [p164]

Vicente J. Bolós
Department of Bussiness Mathematics. University of Valencia
Avda. Tarongers s/n. 46022. Valencia
Spain
vicente.bolos@uv.es

Rafael Benítez
Department of Bussiness Mathematics. University of Valencia
Avda. Tarongers s/n. 46022. Valencia
Spain
rabesua@uv.es