WaveletComp:
A guided tour through the R-package

Angi Rösch*/ Harald Schmidbauer†

(This version: December 17, 2014)

Abstract

WaveletComp is an R package for continuous wavelet-based analysis of univariate and bivariate time series. Wavelet functions are implemented in WaveletComp such that a wide range of intermediate and final results are easily accessible. The null hypothesis that there is no (joint) periodicity in the series is tested via p-values obtained from simulation, where the model to be simulated can be chosen from a wide variety of options. The reconstruction, and thus filtering, of a given series from its wavelet decomposition, subject to a range of possible constraints, is also possible. WaveletComp provides extended plotting functionality — which objects should be added to a plot (for example, the ridge of wavelet power, contour lines indicating significant periodicity, arrows indicating the leading/lagging series), which kind and degree of smoothing is desired in wavelet coherence plots, which color palette to use, how to define the layout of the time axis (using POSIXt conventions), and others. Technically, we have developed vector- and matrix-based implementations of algorithms to reduce computation time. Easy and intuitive handling was given high priority.

Even though we provide some details concerning the mathematical foundation of the methodology implemented in WaveletComp, the present guide is not intended to give an introduction to wavelet analysis. The goal here is to give a series of constructed as well as real-world examples to illustrate the use and functionality of WaveletComp, with statistical arguments in mind.

Keywords: R project; wavelet analysis; Morlet wavelet; (cross-) wavelet power; wavelet coherence; wavelet reconstruction; wavelet graphics; wavelet filtering

*FOM University of Applied Sciences, Munich, Germany; e-mail: angi@angi-stat.com
†Istanbul Bilgi University, Istanbul, Turkey; e-mail: harald@hs-stat.com

1 Introduction

Wavelet methodology is a reasonable choice to study periodic phenomena in time series, particularly in the presence of potential frequency changes across time. Wavelets provide a workable compromise in the time and frequency resolution dilemma (resulting from the Heisenberg uncertainty principle) arising in this context. Applications in signal and image processing, medicine, geophysics and astronomy have been abounding since the early 1980s, but applications of wavelets in economic investigations are more recent.[1]

Growing interest in the application of wavelet methodology has brought forth several open-source packages for wavelet analysis in R [12]. The present guide is the result of an ongoing development of package WaveletComp for continuous wavelet analysis of univariate and bivariate time series, combining manifold output of intermediate as well as final results with intuitive handling. The present guide is not intended to give an introduction to continuous wavelet methodology, but rather to outline some details of wavelet methodology as implemented, and ready for use, in WaveletComp, and to illustrate the most important capabilities of WaveletComp in a series of constructed and real-world examples.

1.1 Univariate series: theoretical background

WaveletComp analyzes the frequency structure of uni- and bivariate time series using the Morlet wavelet.[2] This continuous, complex-valued wavelet leads to a continuous, complex-valued wavelet transform of the time series at hand, and is therefore information-preserving with any careful selection of time and frequency resolution parameters. The transform can be separated into its real part and its imaginary part, thus providing information on both local amplitude and instantaneous phase of any periodic process across time — a prerequisite for the investigation of coherency between two time series. The "mother" Morlet wavelet, in the version implemented in WaveletComp, is:

$$\psi(t) = \pi^{-1/4}\, \mathrm{e}^{i\omega t}\, \mathrm{e}^{-t^2/2}, \tag{1}$$

see Figure 1. The "angular frequency" ω (or rotation rate in radians per time unit) is set to 6, which is the preferred value in literature since it makes the Morlet wavelet approximately analytic; one revolution is equal to 2π (radians); therefore, the period (or inverse frequency) measured in time units equals $2\pi/6$. — The Morlet wavelet in Equation (1) constitutes the basis of WaveletComp. We'll next throw a glance at what can be done with it, with a view toward its implementation in WaveletComp.

The Morlet wavelet transform of a time series (x_t) is defined as the convolution of the series with a set of "wavelet daughters" generated by the mother wavelet by translation in time by τ and scaling by s:

$$\text{Wave}(\tau, s) = \sum_t x_t \, \frac{1}{\sqrt{s}} \psi^\star \left(\frac{t-\tau}{s} \right) \tag{2}$$

with $\star$ denoting the complex conjugate. The position of the particular daughter wavelet in the time domain is determined by the localizing time parameter τ being shifted by a time increment of $\mathrm{d}t$. The choice of the set of scales s determines the wavelet coverage of the series in the frequency domain. The

[1] There is a variety of wavelet textbooks with applications in natural sciences, e.g. the book by Carmona et al. [3]. Discrete wavelet analysis and applications in the field of finance and economics are illustrated in the textbook by Gencay et al. [8]. Several research articles provide a comprehensive introduction to continuous wavelet analysis: Torrence and Combo [13], Cazelles et al. [4] in the field of geophysics and ecology; Aguiar-Conraria and Soares [1, 2] with a focus on economic time series, to name but a few.

[2] The Morlet wavelet dates back to the early 1980s, when the continuous wavelet transform was identified as such, cf. Morlet et al. [10], [11], Goupillaud et al. [9]. It builds on a Gaussian-windowed sinusoid, the Gabor transform, which was introduced in 1946 by Gabor [7] to decompose a signal into its frequency and phase content as time evolves. Unlike the Gabor transform, the Morlet wavelet keeps its shape through frequency shifts, thus providing a "reasonable" separation of contributions from different frequency bands "without excessive loss" in time resolution (Goupillaud et al. [9]), and the feasibility of series reconstruction.

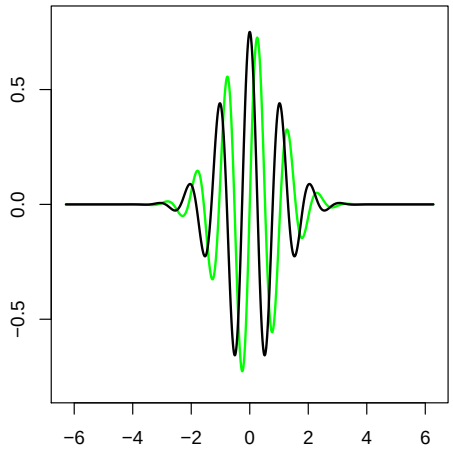

Figure 1: The Morlet mother wavelet — real part (black line) and imaginary part (green line)

scale value is a fractional power of 2, a "voice" in an "octave" with $1/\mathrm{d}j$ determining the number of voices per octave:

$$s_{\min}\, 2^{j \cdot \mathrm{d}j}, \quad j = 0, \ldots, J \tag{3}$$

The minimum (maximum) scale is fixed via the choice of the minimum (maximum) period of interest through the conversion factor $6/(2\pi)$ (which gives consistent results for sinus waves of known frequency).[3] WaveletComp uses Fast Fourier Transform algorithms following the methodology illustrated by Torrence and Compo [13] to evaluate formula (2) efficiently.

The local amplitude of any periodic component of the time series under investigation, and how it evolves with time, can then be retrieved from the modulus of its wavelet transform. WaveletComp adopts the rectified version according to Liu et al. [6], since the mere modulus produces "biased" wavelet amplitudes in the sense that high-frequency (short-period) phenomena tend to be underestimated:

$$\text{Ampl}(\tau, s) = \frac{1}{s^{1/2}} \cdot |\text{Wave}(\tau, s)|. \tag{4}$$

The square of the amplitude has an interpretation as time-frequency (or time-period) wavelet energy density, and is called the wavelet power spectrum (see Carmona et al. [3]):

$$\text{Power}(\tau, s) = \frac{1}{s} \cdot |\text{Wave}(\tau, s)|^2 \tag{5}$$

In case of white noise, its expectation at each time and scale corresponds to the series variance (with proportionality factor $1/s$ in this rectified version of wavelet power). It is therefore conventional to standardize, after detrending, the time series to be analyzed, in order to obtain a measure of the wavelet power which is relative to unit-variance white noise and directly comparable to results of other time series. WaveletComp offers, as default, optional detrending by means of local polynomial regression, and standardization is performed internally. An image plot is the usual way to visualize the wavelet power spectrum; the default in WaveletComp the time/period domain, but conversions to the time/scale or the time/frequency domains are also possible. A procedure to efficiently determine the ridge of power within a band of neighboring periods is performed by default, and the ridge can be retrieved and added to the power spectrum. The same applies to the cone of influence which excludes areas of edge effects. For the purpose of testing the null hypothesis of "no periodicity", significance is assessed with simulation algorithms; a variety of alternatives to test against is available, for which surrogate time series are provided: white noise, shuffling the given time series, time series with a similar spectrum, AR, and ARIMA.

[3] E.g., Aguiar-Conraria and Soares [2] also make use of the Fourier factor $2\pi/6$ to convert scales to periods.

Contour lines added to the wavelet power spectrum delineate areas of high significance. In many applications, a time-averaged wavelet power spectrum is useful to investigate the overall strength of periodic phenomena; this functionality is also provided by WaveletComp.

Displacements of periodic phenomena relative to the localizing origin τ, shifted across the time domain, are given by the instantaneous or local wavelet phase; it can be wrapped to represent an angle in the interval $[-\pi, \pi]$:

$$\text{Phase}(\tau, s) = \text{Arg}(\text{Wave}(\tau, s)) = \tan^{-1}\left(\frac{\text{Im}(\text{Wave}(\tau, s))}{\text{Re}(\text{Wave}(\tau, s))}\right). \tag{6}$$

The investigation of selected phase paths (or phase images) may help to understand the timing of structural breaks (in the sense of phase shifts) in periodic phenomena in certain applications. WaveletComp provides tools to retrieve and plot (average) phase paths for selected periods or period bands and to plot the overall phase image.

Since the Morlet wavelet transform is a bandpass filter and highly redundant, it is possible to smooth and even reconstruct the original time series by summing over a set of reconstruction waves:

$$(x_t) = \frac{\text{d}j \cdot \text{d}t^{1/2}}{0.776 \cdot \psi(0)} \sum_s \frac{\text{Re}(\text{Wave}(., s))}{s^{1/2}} \tag{7}$$

The reconstruction factor 0.776 is adopted from Torrence and Combo [13] as an empirically suggested constant for the case of full reconstruction; nevertheless, and particularly for the case of selective reconstruction, an option is given of whether to recover the (detrended) series' mean and variance or not.

1.2 The bivariate case: theoretical background

The concepts of cross-wavelet analysis provide appropriate tools for (i) comparing the frequency contents of two time series, (ii) drawing conclusions about the series' synchronicity at certain periods and across certain ranges of time. The cross-wavelet transform of two time series (x_t) and (y_t), with respective wavelet transforms Wave.x and Wave.y, decomposes the Fourier co- and quadrature-spectra in the time-frequency (or time-scale) domain. WaveletComp implements the rectified version according to Veleda et al. [14]:

$$\text{Wave.xy}(\tau, s) = \frac{1}{s} \cdot \text{Wave.x}(\tau, s) \cdot \text{Wave.y}^{\star}(\tau, s) \tag{8}$$

Its modulus can be interpreted as cross-wavelet power; it lends itself, with certain limitations, to an assessment of the similarity of the two series' wavelet power in the time-frequency (or time-scale) domain:

$$\text{Power.xy}(\tau, s) = |\text{Wave.xy}(\tau, s)| \tag{9}$$

In a geometric sense, the cross-wavelet transform is the analog of the covariance, and like the latter, it depends on the unit of measurement of the series and may not be ready for interpretation with regard to the degree of association of the two series; wavelet coherency (see below) may remedy this.

WaveletComp provides an image plot of the cross-wavelet power spectrum in the time-period domain, optionally with the cone of influence and with contour lines to indicate significance of joint periodicity or, for checks of consistency, joint significance of periodicity (again assessed with simulation algorithms, with the same variety of alternatives available to test against as in the univariate case). Lines for the ridge of cross-wavelet power can be added to the plot, as well as information about the two series' synchronization in terms of the instantaneous or local phase advance of any periodic component of (x_t) with respect to the correspondent component of (y_t), viz. the so-called phase difference of x over y at each localizing time origin and scale:

$$\text{Angle}(\tau, s) = \text{Arg}(\text{Wave.xy}(\tau, s)) \tag{10}$$

This equals the difference of individual phases, Phase.x − Phase.y, when converted to an angle in the interval $[-\pi, \pi]$. An absolute value less (larger) than $\pi/2$ indicates that the two series move in phase (anti-phase, respectively) referring to the instantaneous time as time origin and at the frequency (or period) in question, while the sign of the phase difference shows which series is the leading one in this relationship. Figure 2 (in the style of a diagram by Aguiar-Conraria and Soares [2]) illustrates the range of possible phase differences and their interpretation. In line with this style, phase differences are displayed as arrows in the image plot of cross-wavelet power.

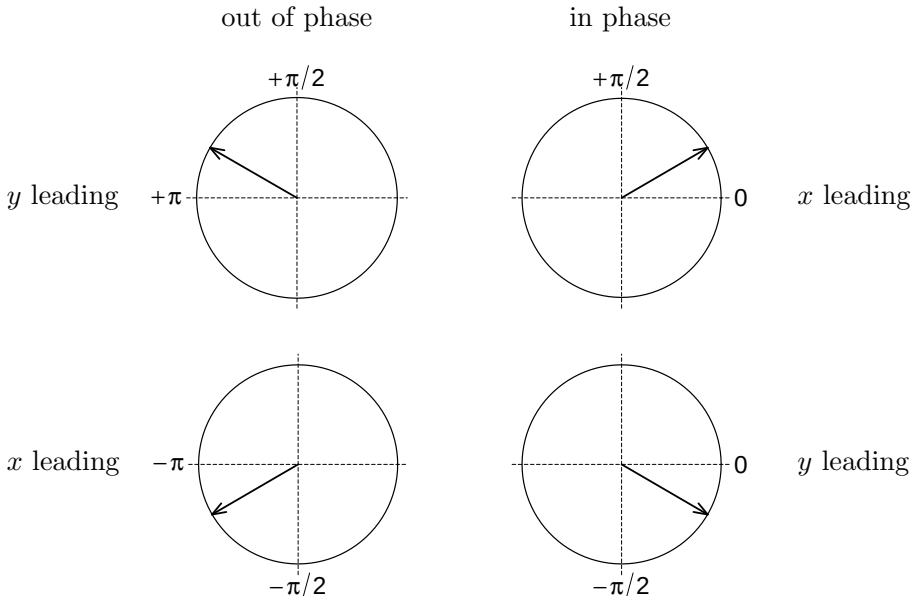

Figure 2: Phase-differences and their interpretation

In order to analyze displacements of periodic phenomena in two time series, a comparative plot of (average) phase paths for selected periods or period bands may be helpful as well. This is also provided by WaveletComp, together with an overall image of phase differences.

The concept of Fourier coherency measures the cross-correlation between two time series as a function of frequency; an analogous concept in wavelet theory is the notion of wavelet coherency, which, however, requires smoothing of both the cross-wavelet spectrum and the normalizing individual wavelet power spectra (without smoothing, its absolute value would be identically 1; see Liu [5]):

$$\text{Coherency} = \frac{\text{sWave.xy}}{(\text{sPower.x} \cdot \text{sPower.y})^{1/2}} \tag{11}$$

The need for smoothing is indicated by the prefix s. Formally (and again: geometrically), coherency is the analog of classical correlation. Consequently, in analogy with the notion of Fourier coherence and the coefficient of determination in statistics, the wavelet coherence is given by the formula:

$$\text{Coherence} = \frac{|\text{sWave.xy}|^2}{\text{sPower.x} \cdot \text{sPower.y}} \tag{12}$$

There is general agreement in literature neither about the direction of smoothing: scale or time or both, nor about the amount of smoothing, to obtain an appropriate measure of coherence without loss of information (cf. Torrence and Combo [13], Cazelles et al. [4]). Therefore, WaveletComp provides all three directional options, and, following Aguiar-Conraria and Soares [2], a variety of filtering windows of constant (over time and scale) but tunable width to choose from. A further notion of phase difference, sAngle, refers to the smoothed cross-wavelet transform and can be used as an alternative to Angle.

The plot of the coherence spectrum can be designed in ways similar to the cross-wavelet power spectrum. Coherence significance can be applied to both plots as well. A time-averaged spectrum of either cross-wavelet power or coherence to investigate the joint overall strength of periodic phenomena is also provided.

1.3 About WaveletComp

Package WaveletComp is written in the spirit that wavelet theory and its applications can be seen as an extension of statistical methodology.

Significance tests of the null hypothesis of no (or no joint) periodicity with a variety of alternatives to test against can be performed with WaveletComp both in the time-period or period-only (and time-averaged) domain. These tests are performed using simulations.[4] Throughout the functionality of WaveletComp, we have tried to use statistical terminology consistently, for example: significance levels are always denoted by "siglvl" (instead of using terms like "tolerance" or "confidence"); matrices or vectors with name ending in ".pval" always contain p-values (and not significance levels). Function calls produce a variety of intermediate and final output results facilitating further statistical analysis.

Another feature of WaveletComp is the implementation of a function for time series reconstruction subject to a variety of potential constraints, for example a focus on the ridge of the spectrum, or on a set of selected periods, or on significance in the time-period domain. Thus, the (partially) reconstructed series is a filtered and detrended version of the original time series. Detrending is accomplished using polynomial regression. The trend is retained in the reconstruction output and easily accessible.

WaveletComp rectifies (cross-) wavelet power according to Liu et al. [6] in the univariate, and Veleda et al. [14] in the bivariate case, to avoid "biased" results in the sense that high-frequency (short-period) phenomena tend to be underestimated by conventional approaches. Implemented options of smoothing in time and/or period direction, needed to compute wavelet coherence (see Liu [5]), are manifold.

WaveletComp provides a variety of plotting options allowing for easy fine tuning. In particular, the time domain in spectrum plots is endowed with an appropriate calendar axis if date-time information is included in the data set, either in terms of rownames or as a variable named "date", to be formatted as a "POSIXt" class object (with the usual R conventions). Internal plots use standard graphics and can be tuned and enriched according to one's needs: default plotting elements (e.g. axis labels and titles, colors, color legend, overall title) can be easily redefined; further plotting elements can be added (e.g. horizontal or vertical lines, comments), and there is a choice of what to include in the plot at all (e.g. cone of influence, contour lines of significant time/period domains, ridge). In plots of cross-wavelet power (or coherence), the area of arrows depicting phase differences can be chosen according to level or significance, the latter either with respect to coherence (cross-wavelet power) or individual power spectra. Minimum ridge levels and significance levels can be defined as well.

As far as the technical implementation of algorithms is concerned, we have tried — for the sake of reducing computation time — to refrain from taking recourse to loops whenever possible, and developed vector- and matrix-based formulations instead. Moreover, univariate intermediate results are accessible when analyzing the coherency of a bivariate series, which again saves computation time and provides for consistent results for bivariate series and their univariate components.

Credits are due to Huidong Tian, Bernard Cazelles, Luis Aguiar-Conraria, and Maria Joana Soares. Parts of the functional architecture, the simulation algorithm, and the concept of ridge determination build on ideas developed by Huidong Tian and Bernard Cazelles (archived R package "WaveletCo"[5]). The code for the computation of the cone of influence and for the arrow design to depict phase differences

[4] In order to provide easy-to-replicate examples of the package's overall functionality, the number of simulations is set to 10 (this is the default setting). We used 1000 simulations to produce the results in the present guide.

[5] URL http://cran.r-project.org/src/contrib/Archive/WaveletCo/, archived April 2013; accessed July 26, 2013.

has been adopted from Huidong Tian. The choice of smoothing windows for the computation of wavelet coherence is inspired by Luis Aguiar-Conraria and Maria Joana Soares ("GWPackage"[6]).

If our colleagues find WaveletComp useful for their research, we would like to suggest they call:

```
> citation('WaveletComp')

To cite package WaveletComp in publications use:

  Angi Roesch and Harald Schmidbauer (2014). WaveletComp: Computational
  Wavelet Analysis. R package version 1.0.
  http://www.hs-stat.com/projects/WaveletComp/WaveletComp_guided_tour.pdf

A BibTeX entry for LaTeX users is

  @Manual{,
    title = {WaveletComp: Computational Wavelet Analysis},
    author = {Angi Roesch and Harald Schmidbauer},
    year = {2014},
    note = {R package version 1.0},
    url = {
http://www.hs-stat.com/projects/WaveletComp/WaveletComp_guided_tour.pdf},
  }

ATTENTION: This citation information has been auto-generated from the
package DESCRIPTION file and may need manual editing, see
help("citation") .
```

[6]URL http://sites.google.com/site/aguiarconraria/joanasoares-wavelets; accessed September 4, 2013.

2 Analysis of a univariate time series

2.1 Wavelet transformation and reconstruction, example 1

We begin the practical part of our tour with a simple example:

```
x = periodic.series(start.period = 50, length = 1000)
x = x + 0.2*rnorm(500)  # add some noise
```

This is a series x with a constant period of 50, see Figure 3. The wavelet transform of x is computed as follows:

```
my.data = data.frame(x = x)
my.w = analyze.wavelet(my.data, "x",
                       loess.span = 0,
                       dt = 1, dj = 1/250,
                       lowerPeriod = 16,
                       upperPeriod = 128,
                       make.pval = T, n.sim = 10)
```

The way of entering the data to `analyze.wavelet` (via a data frame) may seem clumsy, but it is very convenient for providing date and time (see the examples in Sections 4 and 5), as well as for entering data to investigate the wavelet coherence of bivariate time series arranged in multi-column data frames (see Section 3). A brief glance at the arguments:

- `loess.span = 0`: There is no need to detrend this series.
- `dt = 1`: one observation of x is made per time unit. (This defines the time unit.)
- `lowerPeriod = 16, upperPeriod = 128`: This defines the range of periods used in the wavelet transformation. Only periods of x within this range will be detected. This range reaches from $16 = 2^4$ to $128 = 2^7$. To define period limits in terms of powers of 2 is convenient (see `dj` below), but no necessity.
- `dj = 1/250`: The period range comprises the three "octaves" $[2^4, 2^5)$, $[2^5, 2^6)$, $[2^6, 2^7)$, together with the value 2^7. Each octave is divided into 250 suboctaves with bounds given in vector `my.w$Period` on a logarithmic scale. In other words: `my.w$Period` has length 751, and all entries of the vector `diff(log(my.w$Period))` are equal. Graphically, the argument `dj` thus determines the resolution along the period axis.
- `make.pval = T, n.sim = 10`: The region of significant periods in x for each t (the area within the white line in Figure 4) is found using 10 simulations.[7]

To plot the wavelet power spectrum:

```
wt.image(my.w, color.key = "quantile", n.levels = 250,
         legend.params = list(lab = "wavelet power levels", mar = 4.7))
```

Default palette is a subrange of `rainbow` and `n.levels` is the number of color levels; its default value is 100. Parameter `mar = 4.7` controls the distance of the color bar from the main plot area; to control this can come in very handy when plotting to a pdf file. — Finally, the wavelet transform can be used to reproduce the original series x, using only those periods whose average power (see also Section 2.4 below) was found significant over the entire time interval:

```
reconstruct(my.w, plot.waves = F, lwd = c(1,2), legend.coords = "bottomleft")
```

This produces the plot in Figure 8. The series x has been "denoised" in the sense that only smooth components are retained.

[7] We used 1000 simulations to produce the images. This will, of course, increase computation time.

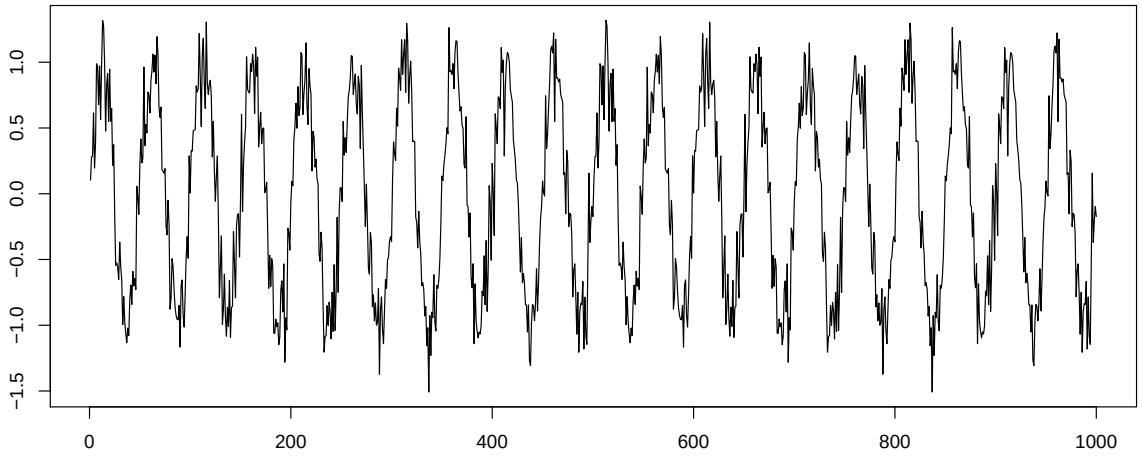

Figure 3: Series with period 50

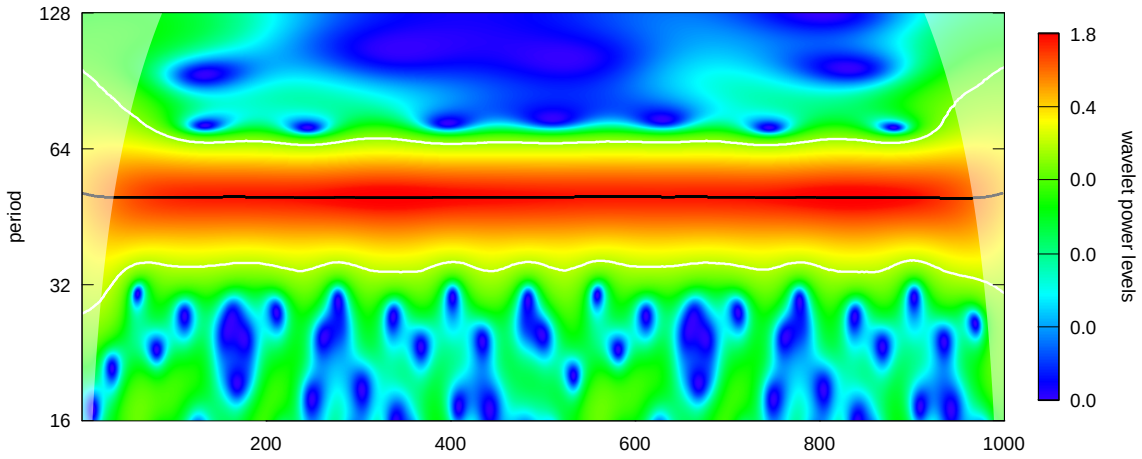

Figure 4: Wavelet power spectrum of the series

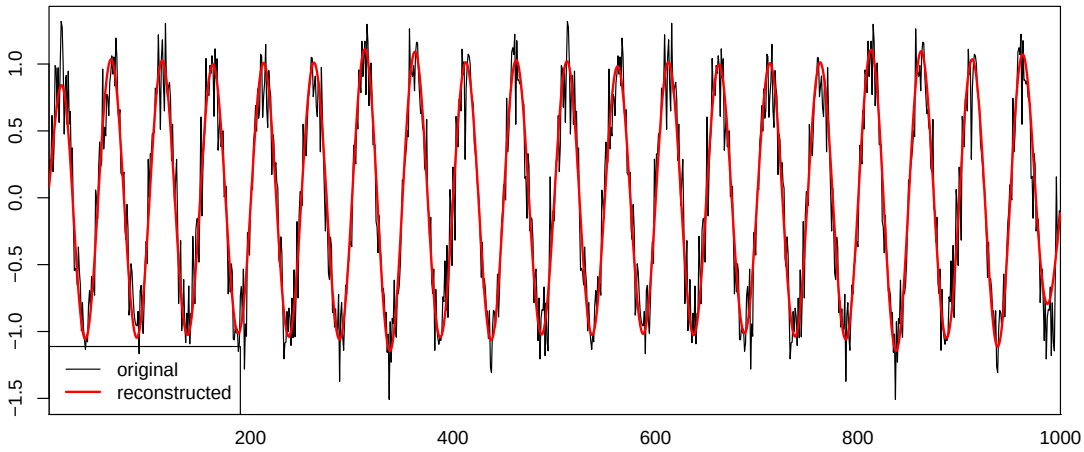

Figure 5: Reconstruction of the series

2.2 Wavelet transformation and reconstruction, example 2

A series with linearly increasing trend (see Figure 6):

```
x = periodic.series(start.period = 20, end.period = 100, length = 1000)
x = x + 0.2*rnorm(1000)
```

The power spectrum (Figure 7) results from:

```
my.data = data.frame(x = x)
my.w = analyze.wavelet(my.data, "x",
                       loess.span = 0,
                       dt = 1, dj = 1/250,
                       lowerPeriod = 16,
                       upperPeriod = 128,
                       make.pval = T, n.sim = 10)
wt.image(my.w, n.levels = 250,
         legend.params = list(lab = "wavelet power levels"))
```

The structure (that is, the time-period grid) is the same as in the example in Section 2.1, and so is the dimension of the objects produced by `analyze.wavelet`. We have already seen the vector `my.w$Period`. Further objects stored in `my.w` are the following matrices, all having 751 rows (corresponding to periods) and 1000 columns (the length of series x):

- `my.w$Power`, which gives the wavelet power for each period and each point in time (the basis of the plot in Figure 7),
- `my.w$Power.pval`, which gives the p-values (computed using simulation) of wavelet power (the basis of the white line in Figure 7), in other words: the p-values of the null hypothesis that there is no period (given by `my.w$Period`) in the series at time t, tested against an alternative specified in `method` (see Section 2.5 below),
- `my.w$Ridge`, a matrix of 0s and 1s where a 1 indicates the presence of a ridge in the "landscape" of power (the basis of the ridge plotted in Figure 7).

The ridge appears nonlinear in Figure 7, in spite of the period in x increasing *linearly*. This is, of course, a consequence of the logarithmic period scale.

If upper and lower period are not set, default values will be assigned as `lowerPeriod = 2*dt`, which is 2 in this case, and `upperPeriod = floor(nrow(my.data)/3)*dt`, which is 333; effectively, the upper period will be $\max_{i \in \mathbb{N}} 2^{8+i/250} \leq 333$, which equals 332.2211 (with $i = 94$), giving `my.w$Period` a length of $7 \times 250 + 1 + 94 = 1845$.

The reconstructed series, based on the wavelet decomposition, is shown in Figure 8. This output is produced automatically by calling function `reconstruct`; it will also produce a (possibly quite cluttered) plot of waves used for reconstruction by adding the argument `plot.waves = TRUE`. Further arguments permit a specific reconstruction by selecting periods to be used; an example will be given in Section 2.3. The reconstructed series is accessible via the following commands:

```
my.rec = reconstruct(my.w)
x.rec = my.rec$series$x.r  # x: name of original series
```

This generally gives the detrended series; here, no trend is produced since `loess.span = 0`.

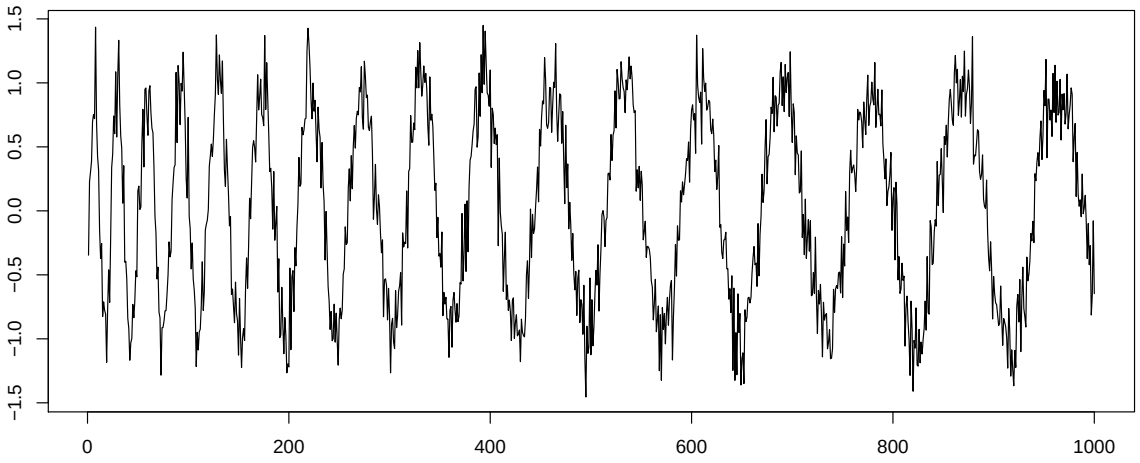

Figure 6: A series wth nonconstant period

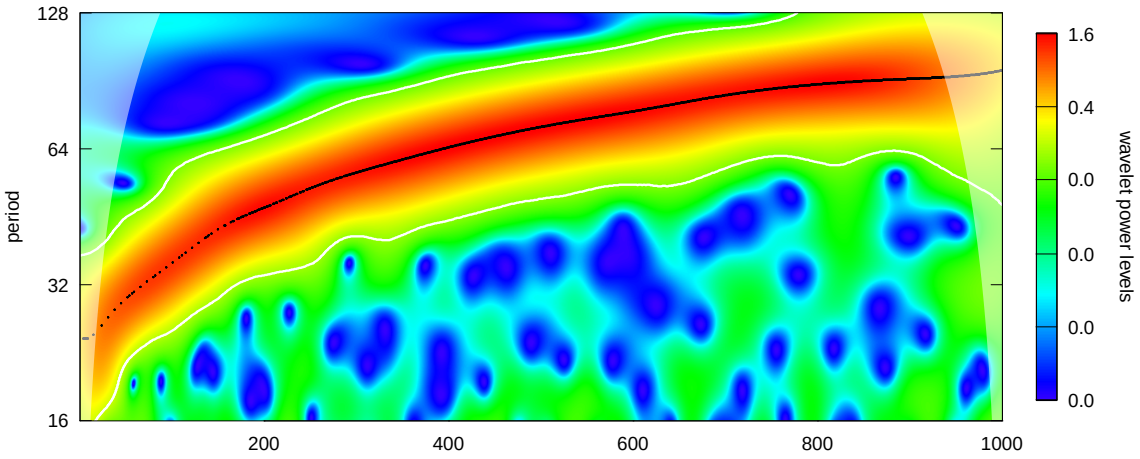

Figure 7: Wavelet power spectrum of the series with nonconstant period

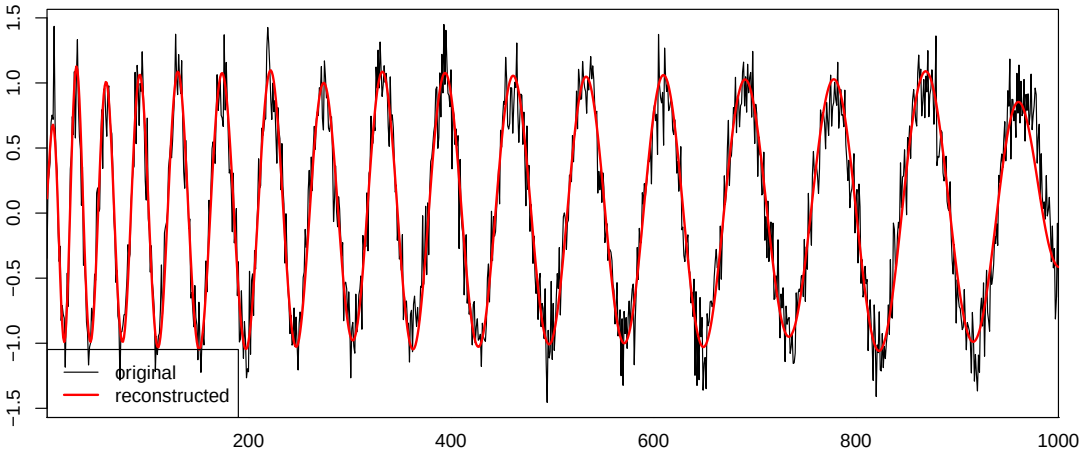

Figure 8: Reconstruction of the series with nonconstant period

2.3 Wavelet transformation and reconstruction, example 3

An important application of wavelets is information reduction, given a time series with distinct periodic structure, retaining only the kind of information considered relevant. As an example, consider series x with two periods superposed (see Figure 9):

```
x1 = periodic.series(start.period = 100, length = 1000)
x2 = periodic.series(start.period = 30, length = 1000)
x = x1 + x2 + 0.2*rnorm(1000)
```

Wavelet transform, plot (see Figure 10):

```
my.data = data.frame(x = x)
my.w = analyze.wavelet(my.data, "x",
                       loess.span = 0,
                       dt = 1, dj = 1/250,
                       lowerPeriod = 16,
                       upperPeriod = 128,
                       make.pval = T, n.sim = 10)
wt.image(my.w, n.levels = 250,
         legend.params = list(lab = "wavelet power levels") )
```

Reconstruction, using only period 100 (see Figure 11):

```
reconstruct(my.w, sel.period = 100, lwd = c(1,2), legend.coords = "bottomleft")
```

Function `reconstruct` actually uses the period in `my.w$Period` closest to 100, which is 100.00997 in this case:

```
> my.w$Period[(my.w$Period > 99) & (my.w$Period < 101)]
[1]  99.18156  99.45693  99.73307 100.00997 100.28764 100.56608 100.84530
```

A selection `sel.period = c(30, 100)`, for example, is also possible. Omitting the optional argument `sel.period = 100` will lead to a "richer" reconstruction, using also other periods. On the other hand, reconstructing a series based on selected periods can be useful for filtering purposes.

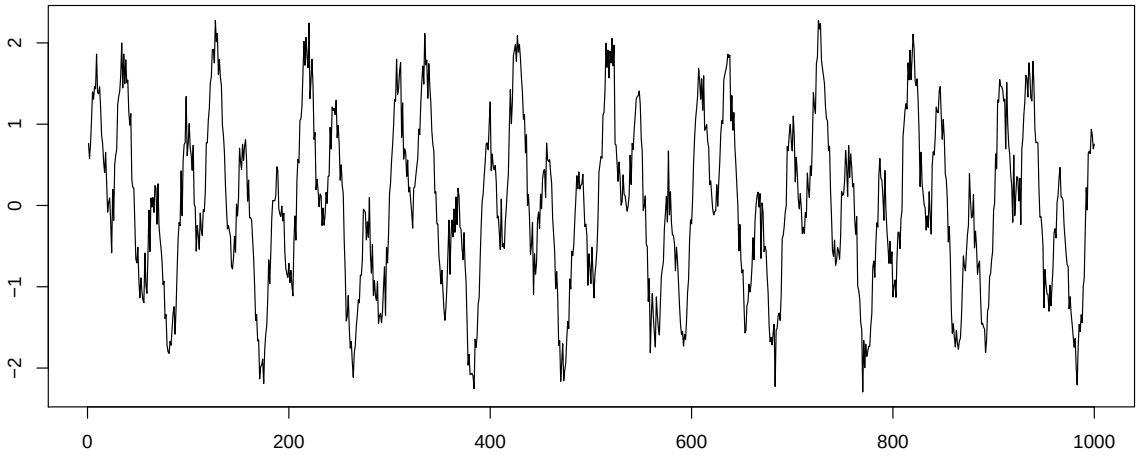

Figure 9: Series, two periods superposed

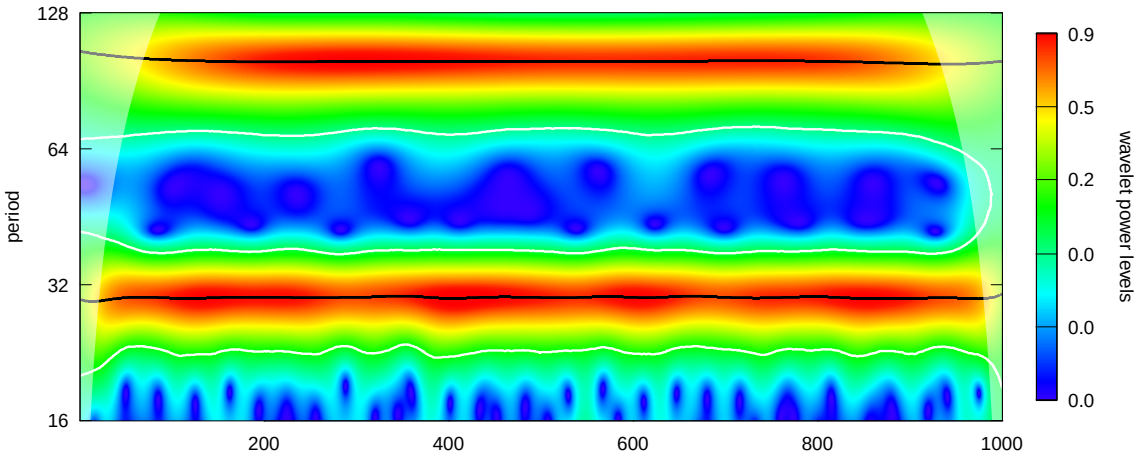

Figure 10: Wavelet power spectrum of the series with superposed periods

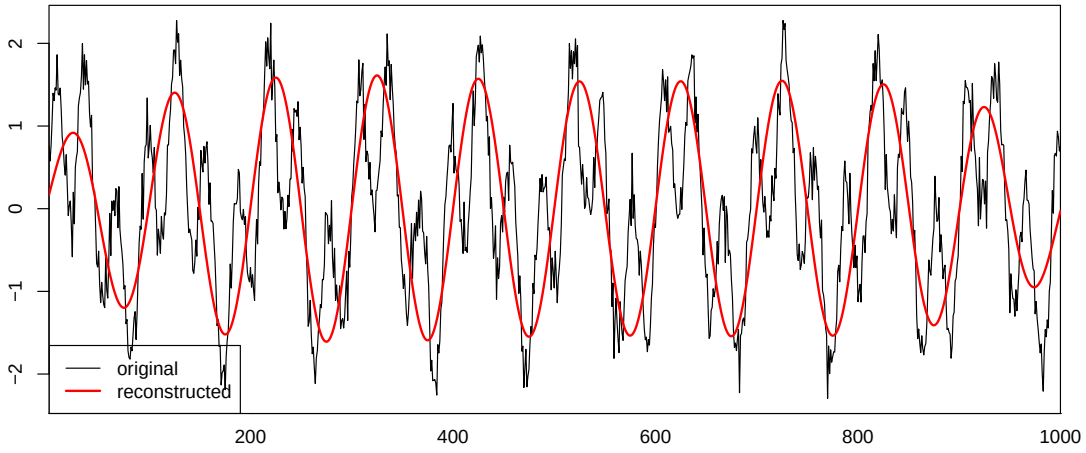

Figure 11: Reconstruction of the series with superposed periods, using only period 100

2.4 Average power

Compare the two series x and y (see Figure 12):

```
x1 = periodic.series(start.period = 100, length = 500)
x2 = 1.2*periodic.series(start.period = 60, length = 500)
x = c(x1, x2) + 0.2*rnorm(1000)
y1 = periodic.series(start.period = 100, length = 1000)
y2 = 1.2*periodic.series(start.period = 60, length = 1000)
y = (y1 + y2)/2  + 0.2*rnorm(1000)
```

What x and y have in common is that they both contain sub-series of periods 60 and 100. Series y carries "denser information" about periodicity than x, since both components are present throughout the entire time interval in y. As before, the wavelet power spectra are computed as:

```
my.data = data.frame(x = x, y = y)
my.wx = analyze.wavelet(my.data, "x",
                        loess.span = 0,
                        dt = 1, dj = 1/20,
                        lowerPeriod = 16,
                        upperPeriod = 256,
                        make.pval = T, n.sim = 10)
my.wy = analyze.wavelet(my.data, "y",
                        loess.span = 0,
                        dt = 1, dj = 1/20,
                        lowerPeriod = 16,
                        upperPeriod = 256,
                        make.pval = T, n.sim = 10)
```

In this case, however, our focal point is a comparison of the *average* power of the series (average taken over time). Average powers can be plotted (see Figures 13 and 14) by calling:

```
wt.avg(my.wx)
wt.avg(my.wy)
```

In this call of `wavelet.analysis`, a low resolution of `dj = 1/20` was used, in order to obtain isolated dots as significance indicators in Figures 13 and 14. The shape of the average power plots of x and y is very similar: The average power plot cannot distinguish between consecutive periods (as in x) and overlapping periods (as in y). Peaks and trough are further apart in the average power plot of y, reflecting the denser information about periodicity in y.

The bimodality of the average power plot will disappear when both periods approach each other. For example, replacing `start.period = 60` with `start.period = 80` will leave a unimodal curve: The discriminatory power of the wavelet transform is no longer enough to distinguish between the longer and the shorter period.

Contrary to statistical intuition, making the time series longer won't increase the wavelet transform's discriminatory power. This is a consequence of the very character of the wavelet transform — it proceeds by transforming pieces of *fixed length* for a given period. A longer series has more such pieces, each with the same characteristics, hence resulting in the same overall average power.

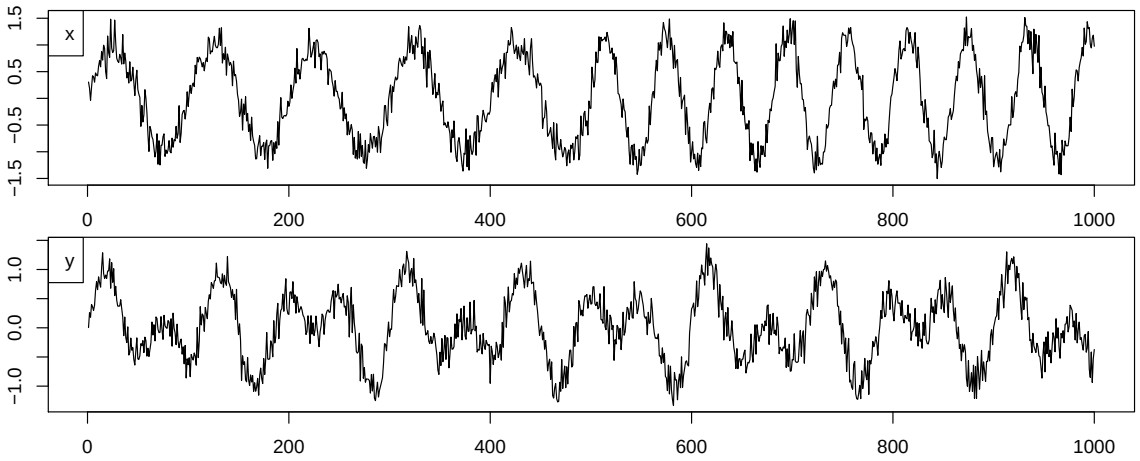

Figure 12: Two series with distinct periods

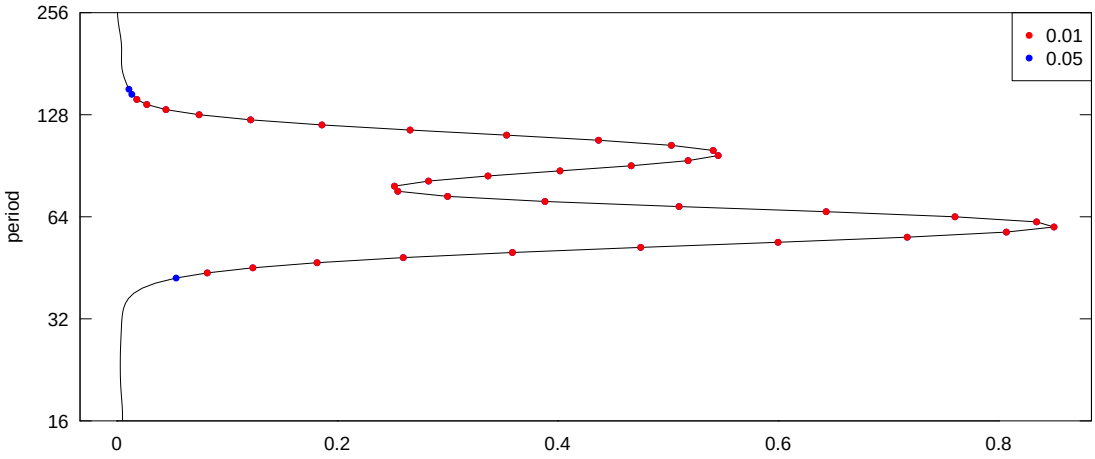

Figure 13: Average power, series x

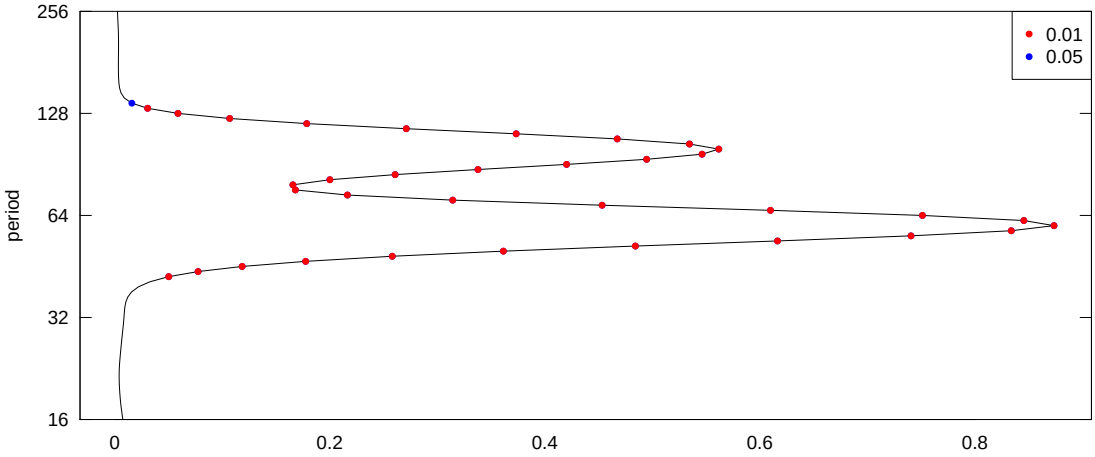

Figure 14: Average power, series y

2.5 Selecting the analysis method

Function `analyze.wavelet` allows to choose `method` from several options. This choice specifies the way in which the null hypothesis that there is no periodicity in the given series (more precisely, in the model that created the given series) is tested. In any case, this test is carried out by WaveletComp using repeated simulations (the number is given as `n.sim`) of a certain model, defined by `method`. The method choice therefore defines the sensitivity of a statistical test: How sensitive is the test with respect to certain deviations from the null hypothesis? This is illustrated in the following example.

The series in Figure 15 has a period of 100, with the exception of the two inner cycles, whose period is 80. These inner cycles also have a slightly higher amplitude:

```
x1 = periodic.series(start.period = 100, length = 500)
x2 = 1.2*periodic.series(start.period = 80, length = 160)
x  = c(x1, x2, x1) + 0.2*rnorm(1160)
```

To produce the power spectrum plot shown in Figure 16:

```
my.data = data.frame(x = x)
my.w = analyze.wavelet(my.data, "x",
                       method = "white.noise",
                       loess.span = 0,
                       dt = 1, dj = 1/250,
                       lowerPeriod = 32, upperPeriod = 256,
                       make.pval = T, n.sim = 10)
wt.image(my.w, color.key = "interval", n.levels = 250,
         legend.params = list(lab = "wavelet power levels"))
```

Technically, p-values of the test are contained in `my.w$Power.pval`, here a matrix of dimension 751×1160, providing the input for the plot of the significant area by `wt.image` (the area outlined by the white line in Figure 16). The significant area (the time-period area for which the null hypothesis is rejected at the default significance level of 10%) ranges over the entire time interval when `method = "white.noise"` is chosen: Compared to white noise, x contains significant periods around 100 all the time. Selecting a significance level of less than 10% (for example, adding option `siglvl = 0.05` in `wt.image`) will make the significant area thinner, but it will still stretch over the entire time interval.

For the plot in Figure 17, `method = "Fourier.rand"` was used. The difference between the two methods is that the area of significance is now much smaller, and essentially confined to the middle of the time interval: Because Fourier randomization assumes that there is *constant* periodicity over time, it only detects deviations from average periodicity in the middle. That the significant area reaches into higher and lower periods in Figure 17 is a consequence of the higher amplitude in the middle, co-occurring with some earlier and later spikes.

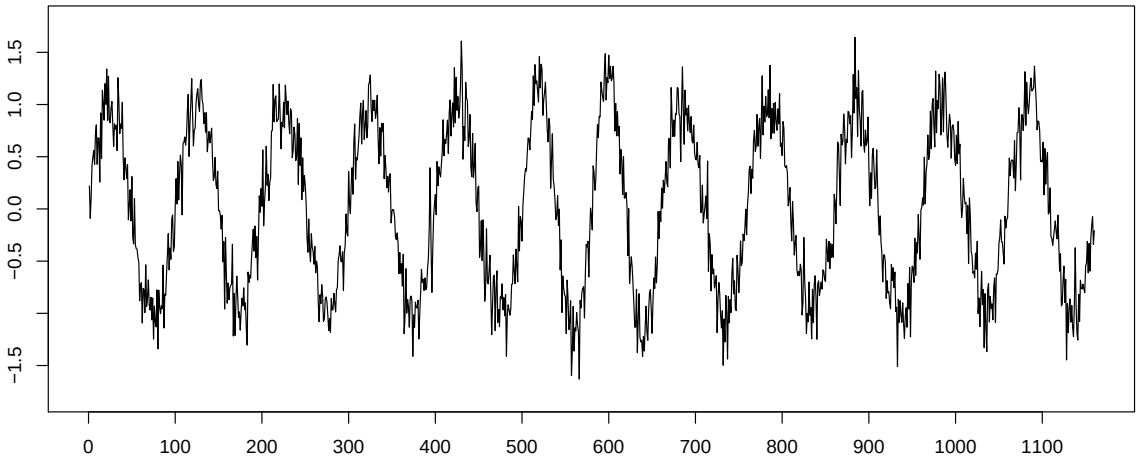

Figure 15: Time series with periods 80 (in the middle) and 100

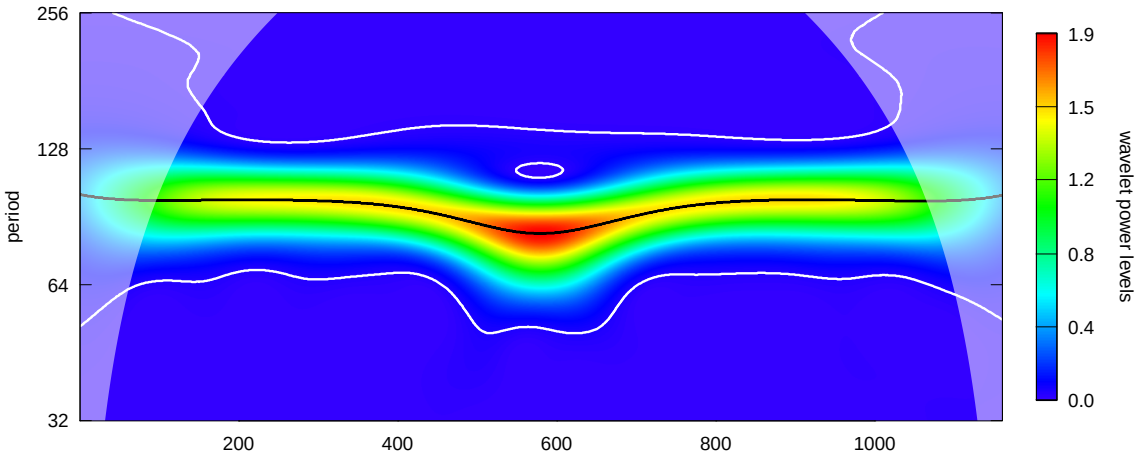

Figure 16: Power spectrum, `method = "white.noise"`

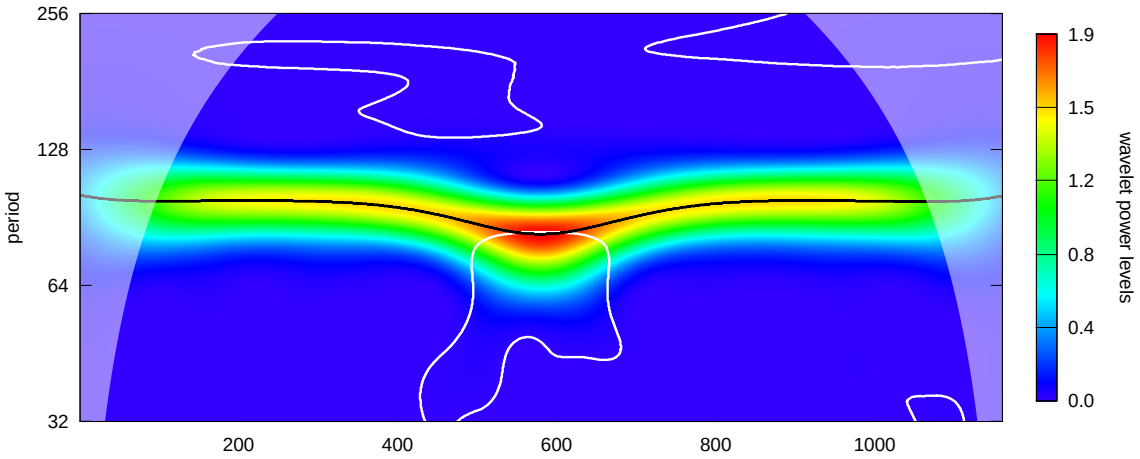

Figure 17: Power spectrum, `method = "Fourier.rand"`

2.6 Plotting the power spectrum

Using the example from Section 2.5 (with `method = "white.noise"`), we will now illustrate some of the plotting options of function `wt.image`.

The power spectrum plot in Figure 18 was made setting `color.key = "quantile"` (the default setting) in `wt.image`:

```
wt.image(my.w, color.key = "quantile", n.levels = 250,
         legend.params = list(lab = "wavelet power levels", label.digits = 2))
```

The color bar reveals the steep power gradient in this example. Roughly speaking, option `color.key = "quantile"` splits power levels into quintiles and thus produces images with colors distributed more evenly, but at the expense of possibly exaggerating artefacts. We have also set `label.digits = 2` (the default setting is 1) to display two digits after the decimal point in the color bar labels.

Plotting parameters in WaveletComp should have default settings representing a reasonable compromise between computation time, image quality, and file size. The following is an example of unfortunate settings, at least when plotting to a pdf file:

```
wt.image(my.w, n.levels = 250,
         legend.params = list(lab = "wavelet power levels", label.digits = 2),
         useRaster = F, max.contour.segments = 1000)
```

The resulting image (see Figure 19) has two problems: there are stripes (due to `useRaster = F`), and the white significance line is interrupted (due to `max.contour.segments = 1000`). The present picture is rendered in the jpeg file format; pdf has the same deficiencies with a much larger file size. Default settings in WaveletComp are `useRaster = T` and a lavish `max.contour.segments = 250000` (the latter was not found to have an adverse effect on file sizes).
A grayscale can be convenient for black-and-white printout; it is easy to realize in WaveletComp (see Figure 20):

```
wt.image(my.w, n.levels = 250,
         legend.params = list(lab = "wavelet power levels", label.digits = 2),
         color.palette = "gray((1:n.levels)/n.levels)",
         col.ridge = "yellow")
```

If computation time of the wavelet transform (function `analyze.wavelet`) is long, it can be stored by `save(my.w, file = <...>)` to be processed (for example, plotted) in a new R session by calling `load(<...>)`. Plotting is usually faster than computing the wavelet transform and may require several trials until the desired output is obtained.

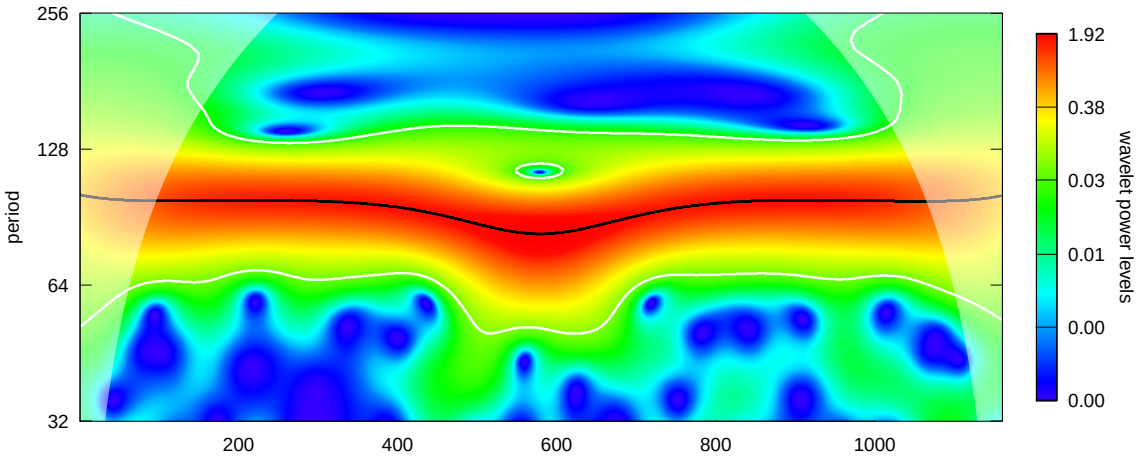

Figure 18: Power spectrum, `color.key = "quantile"`

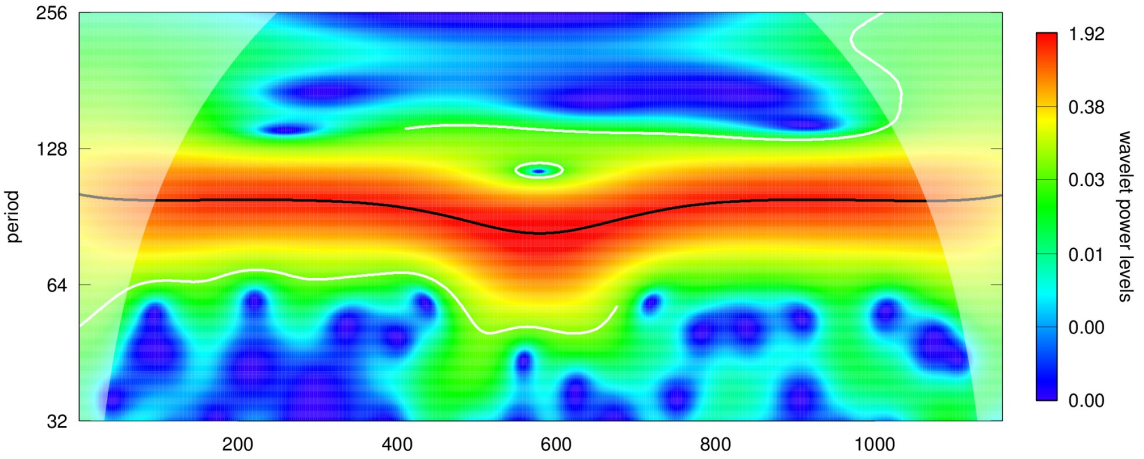

Figure 19: Power spectrum, plot gone wrong

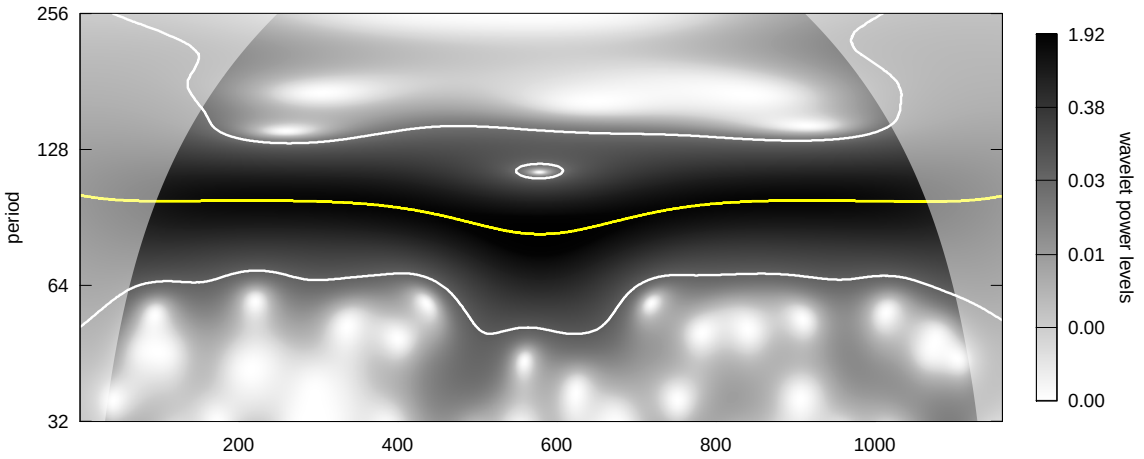

Figure 20: Power spectrum, grayscale, yellow ridge

3 Analysis of a bivariate time series

3.1 Cross-wavelet transformation, example 1

The following is a simplified version of an example discussed by Veleda et al. [14]:

```
x1 = periodic.series(start.period = 1*24, length = 24*96)
x2 = periodic.series(start.period = 2*24, length = 24*96)
x3 = periodic.series(start.period = 4*24, length = 24*96)
x4 = periodic.series(start.period = 8*24, length = 24*96)
x5 = periodic.series(start.period = 16*24, length = 24*96)
x = x1 + x2 + 3*x3 + x4 + x5 + 0.5*rnorm(24*96)
y = x1 + x2 + 3*x3 + x4 + 3*x5 + 0.5*rnorm(24*96)
```

The idea here is that x and y (see Figure 21) represent hourly observations from a 96-day interval. Series x and y have joint constant periods 1, 2, 4, 8, 16, but different amplitudes at period 16. The cross-wavelet transform of x and y is computed as follows:

```
my.data = data.frame(x = x, y = y)
my.wc = analyze.coherency(my.data, my.pair = c("x","y"),
                          loess.span = 0,
                          dt = 1/24, dj = 1/100,
                          lowerPeriod = 1/2,
                          make.pval = T, n.sim = 10)
```

Again, we set `loess.span = 0` because there is no need to detrend the series; `dt = 1/24` means we have 24 observations per time unit (1 day, this actually defines the time unit); `lowerPeriod = 1/2` defines the lowest period to be 12 hours. — To plot the cross-wavelet power spectrum:

```
wc.image(my.wc, n.levels = 250,
         legend.params = list(lab = "cross-wavelet power levels"),
         timelab = "time (days)", periodlab = "period (days)")
```

This produces the plot in Figure 22. Arrows pointing exactly to the right indicate that the two series are in phase at the respective period with vanishing phase differences. They are plotted only within white contour lines indicating significance (with respect to white noise processes) at the 10% level.

Adding the ridge of the cross-wavelet power spectrum, via the argument `plot.ridge = T`, would produce a cluttered plot in this example. Finally, a plot of the time-averaged cross-wavelet power is produced by the following command:

```
wc.avg(my.wc, sigpch = 20)
```

By default, the color of plotting characters (here: a solid circle since `sigpch=20`, which is the default setting) indicates significance levels of 10% (blue) and 5% (red). Figure 23 confirms that the rectified version of cross-wavelet power gives sound results for all periods. (Omitting rectification would severely underestimate the lower periods' power.) The average power decreases for periods 1-2-8; this is due to border effects (the cone of influence moves in to shade the plot at higher periods, hereby reducing power).

Now that function `analyze.coherency` has been called to produce `my.wc`, individual wavelet power spectra can be plotted by simply calling function `wt.image`:

```
wt.image(my.wc, my.series = "x")
wt.image(my.wc, my.series = "y")
```

Functions `wt.avg` and `reconstruct` work in a similar way.

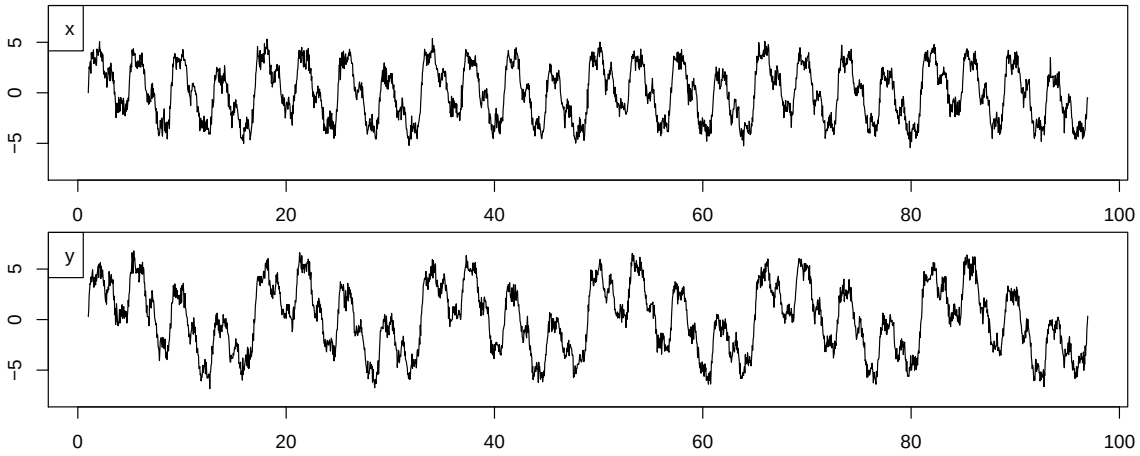

Figure 21: Two series with periods 1, 2, 4, 8, 16 and different amplitudes

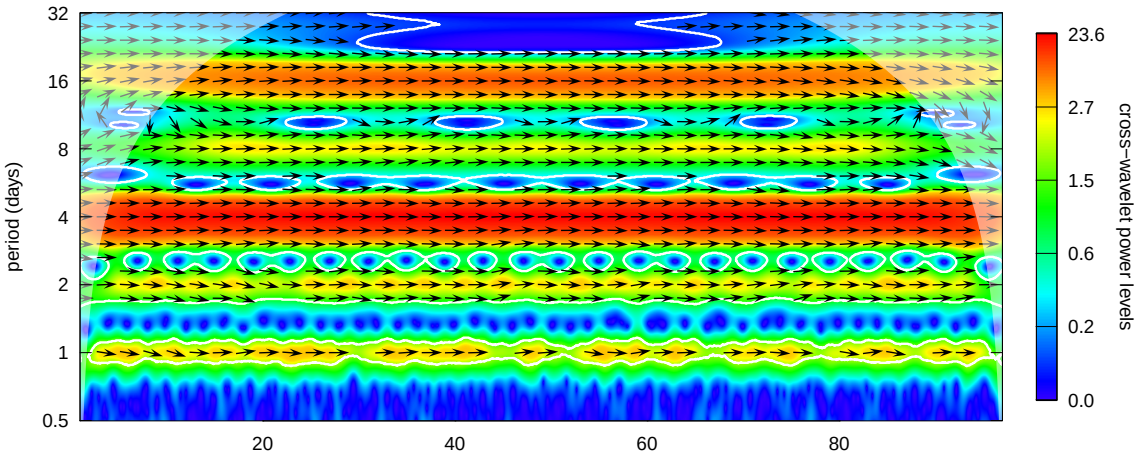

Figure 22: Cross-wavelet power spectrum of x and y

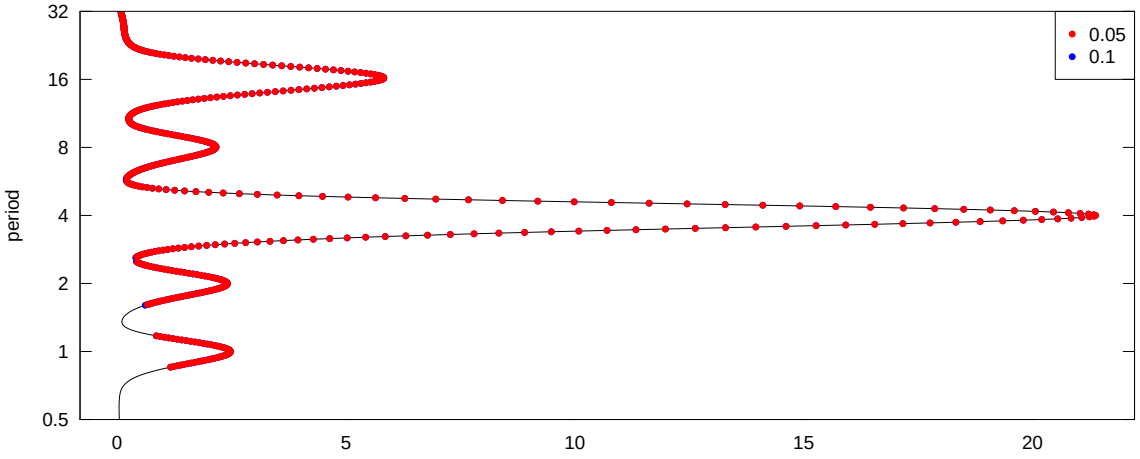

Figure 23: Cross-wavelet average power

3.2 Cross-wavelet transformation, example 2

The time series x shares period 128 with time series y within a certain range of time, but periodicity of y is influenced by a phase shift:

```
xx = periodic.series(start.period = 64, length = 128*3)
xy = periodic.series(start.period = 128, length = 2*128*3)
x = c(xx,xy,xx) + 0.1*rnorm(4*128*3)
y = periodic.series(start.period = 128, phase = -16, length = 4*128*3) +
    0.1*rnorm(4*128*3)
```

See Figure 24. The cross-wavelet transform of x and y is computed as usual:

```
my.data = data.frame(x = x, y = y)
my.wc = analyze.coherency(my.data, my.pair = c("x","y"),
                          loess.span = 0,
                          dt = 1, dj = 1/100,
                          make.pval = T, n.sim = 10)
```

To plot the cross-wavelet power spectrum:

```
wc.image(my.wc, n.levels = 250,
         siglvl.contour = 0.1, siglvl.arrow = 0.05,
         legend.params = list(lab = "cross-wavelet power levels"))
```

The somewhat counterintuitive result is displayed in Figure 25. Period 128 shows significance across the *entire* time interval, while one expects period 128 to be jointly important only in the middle, according to the construction of x and y. The arrows indicate that x and y are in phase in the middle, with x leading (see also Figure 2), and (non-significant) phase differences at this period shifting off the middle. This example illustrates the dilemma of the cross-wavelet power (see also the comment in Section 1.2), which corresponds to the covariance — it can be large even if only one component swings widely. There are two ways to produce a more plausible image, reflecting joint periodic properties of x and y:

1. Limit the area where arrows are drawn to the region where the individual wavelet transforms of x and y show significance, and circumvent the artefacts of the image in Figure 25 resulting from the steep power gradient by choosing `color.key = "interval"`:

   ```
   wc.image(my.wc, n.levels = 250, color.key = "interval",
            siglvl.contour = 0.1, siglvl.arrow = 0.05, which.arrow.sig = "wt",
            legend.params = list(lab = "cross-wavelet power levels"))
   ```

   This produces the image in Figure 26.

2. Plot wavelet coherence, rather than the cross-wavelet power — like the coefficient of determination, it adjusts for individual (one-dimensional) power differences in series x and y. This is shown on the next page.

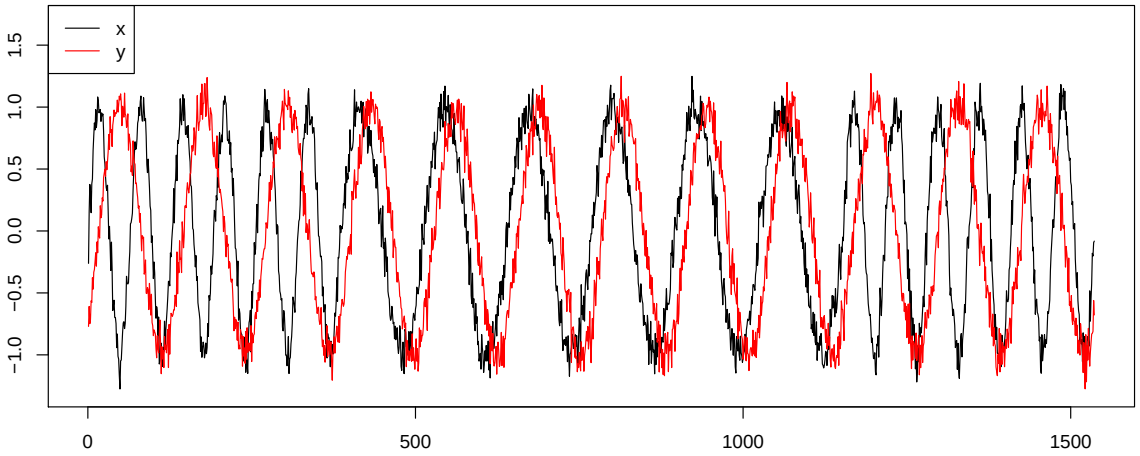

Figure 24: Series with joint period 128 at some time range, but different phase

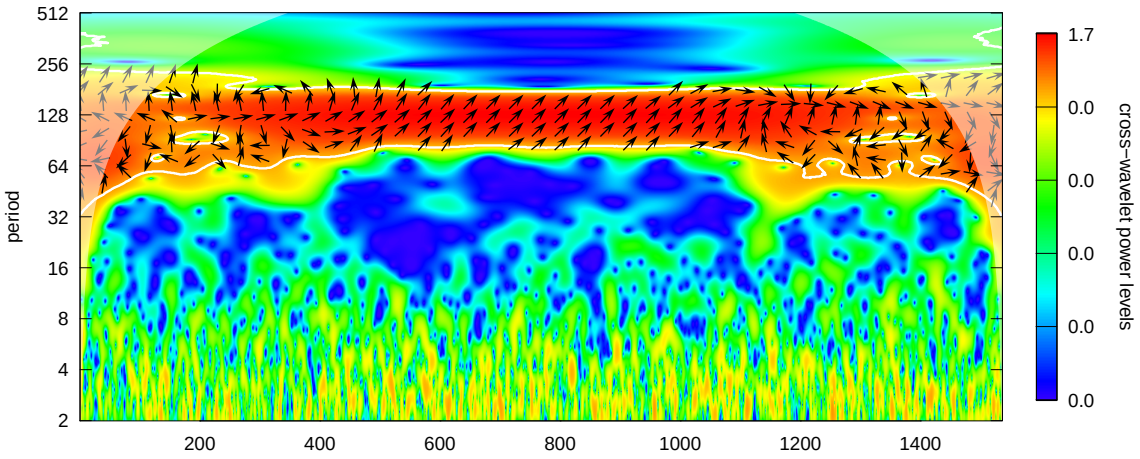

Figure 25: Cross-Wavelet power spectrum of the series

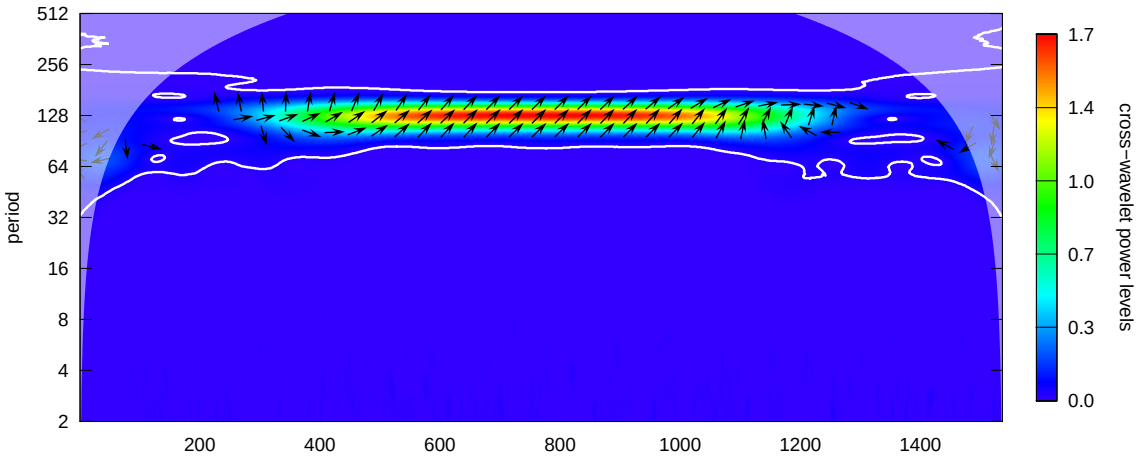

Figure 26: Cross-Wavelet power spectrum of the series with interval color key and restricted arrow area

Plotting the wavelet coherence instead of the cross-wavelet power spectrum is a conceptual circumvention of the need for graphical fine-tuning. A single parameter has to be added in the plotting command to obtain the wavelet coherence:

```
wc.image(my.wc, which.image = "wc", n.levels = 250,
         siglvl.contour = 0.1, siglvl.arrow = 0.05,
         legend.params = list(lab = "wavelet coherence levels"))
```

See Figure 27. The default setting is `which.image = "wp"`, where `p` stands for power. The default smoothing parameter settings have been used in the call of `wc.image` above: Bartlett windows (`window.type.t = 1`, `window.type.s = 1`) of sizes

- `window.size.t = 5` in time units of `1/dt`, which gives length 5 in time direction,
- `window.size.s = 1/4` in time units of `1/dj` to give length 63 in scale (or period) direction.

The relevant paths of phases at period 128 can be traced by means of a phase and phase difference plot created by:

```
wc.sel.phases(my.wc, sel.period = 128, only.sig = T, siglvl = 0.05,
              which.sig = "wc",
              legend.coords = "topright", legend.horiz = F,
              phaselim = c(-pi,+pi), main = "", sub = "")
```

With the default settings `only.sig = T` and `siglvl = 0.05`, but `which.sig = "wc"` (default: `which.sig = "wt"`), the plot is restricted to parts where period 128 is significant with respect to wavelet coherence. Reflecting the construction of x and y, there is a phase difference of 16 in the middle, corresponding to $\pi/4$ when converted to an angle in $[-\pi, +\pi]$, see Figure 28. The "global" image of phase differences in Figure 29 is produced by:

```
wc.phasediff.image(my.wc, which.contour = "wc", n.levels = 250, siglvl = 0.1,
                   legend.params = list(lab = "phase difference levels"))
```

The center area of constant phase differences coincides with the area of coherence significance.

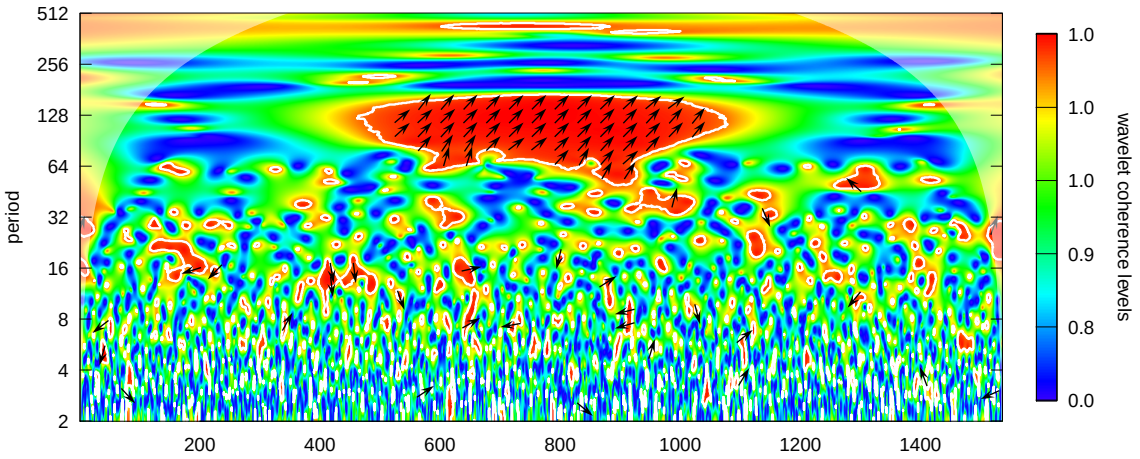

Figure 27: Wavelet coherence of the series

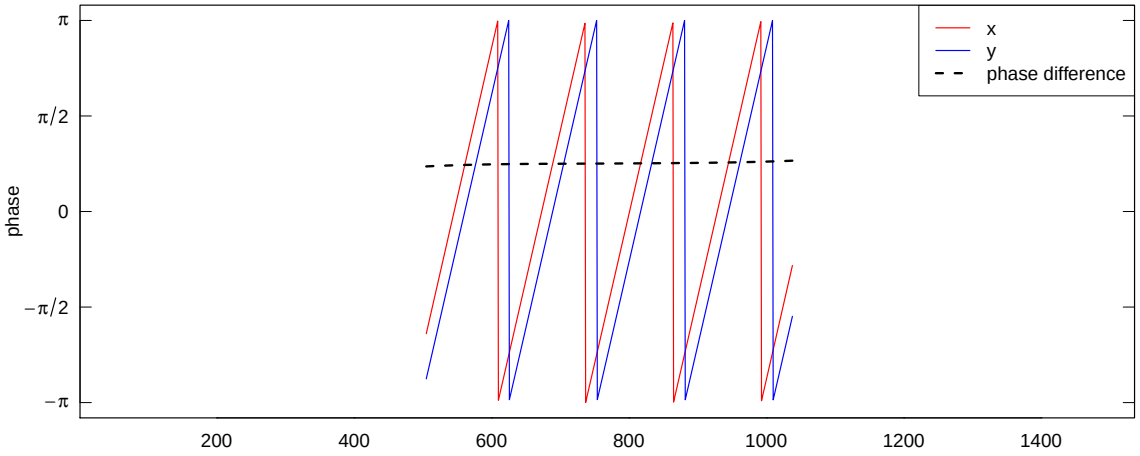

Figure 28: Phases and phase differences at period 128

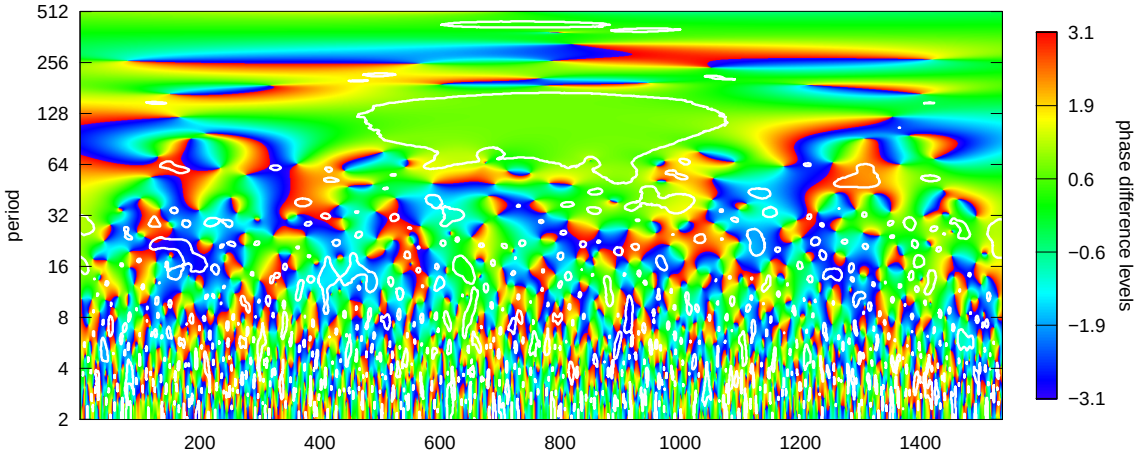

Figure 29: Phase difference image

3.3 Cross-wavelet transformation, example 3

This example picks up the 96-day hourly data example (page 20):

```
x = 2*periodic.series(start.period = 3*24, end.period = 5*24, length = 24*96) +
    0.5*rnorm(24*96)
y = periodic.series(start.period = 4*24, length = 24*96) +
    0.5*rnorm(24*96)
```

Series x and y have joint period 4 (days) in the middle, with the period of x growing linearly (see Figure 30). The cross-wavelet power spectrum is computed and plotted as follows:

```
my.date = seq(as.POSIXct("2014-11-14 00:00:00","%F %T"),
              by = "hour", length.out = 24*96)
my.data = data.frame(date = my.date, x=x, y=y)
my.wc = analyze.coherency(my.data, my.pair = c("x","y"),
                          loess.span = 0,
                          dt = 1/24, dj = 1/100,
                          lowerPeriod = 1/2,
                          make.pval = T, n.sim = 10)
wc.image(my.wc, n.levels = 250,
         legend.params = list(lab = "cross-wavelet power levels"),
         timelab = "time (days)", periodlab = "period (days)")
```

See Figure 31. The area of significance (where arrows are plotted) shows periods increasing with time, which reflects the structure of series x. This image reveals again the shortcomings of the cross-wavelet power spectrum as a tool to detect *joint* periods of two series (see also the previous example, page 22), and the wavelet coherence image will be helpful again (see the next two facing pages).

The phase and phase difference image, shown in Figure 32, can be obtained via:

```
wc.sel.phases(my.wc, sel.period = 4, only.sig = T, which.sig = "wt",
              timelab = "", main = "", sub = "", show.legend = T)
```

It is important to remember that phases and phase differences in this plot are *instantaneous* (or local), as outlined in Section 1; this is why a nonconstant phase difference can result in the case of a constant period. The concave part of the dashed line in Figure 32 is asymmetric: it is steeper on the left side. This is a consequence of the increasing period in series x, which slows down the change in phase difference. However plausible this result may be, care should be taken when interpreting phase differences at period 4 — after all, this period is actually relevant for x only during a short time interval. Indeed, plotting the phase difference image with parameter `which.sig = "wc"` instead of `"wt"` (the default setting) will restrict the plot to a short interval in the middle.

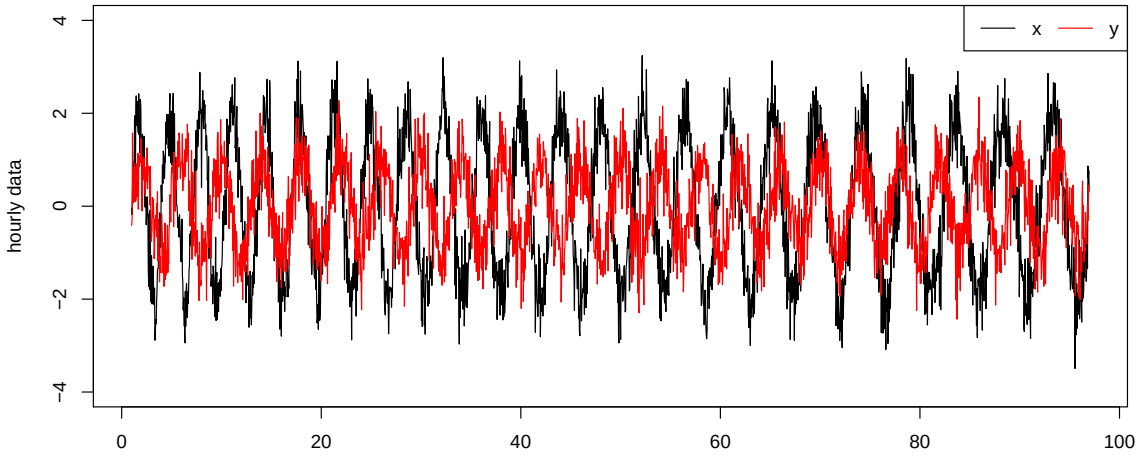

Figure 30: Two series, overlapping periods

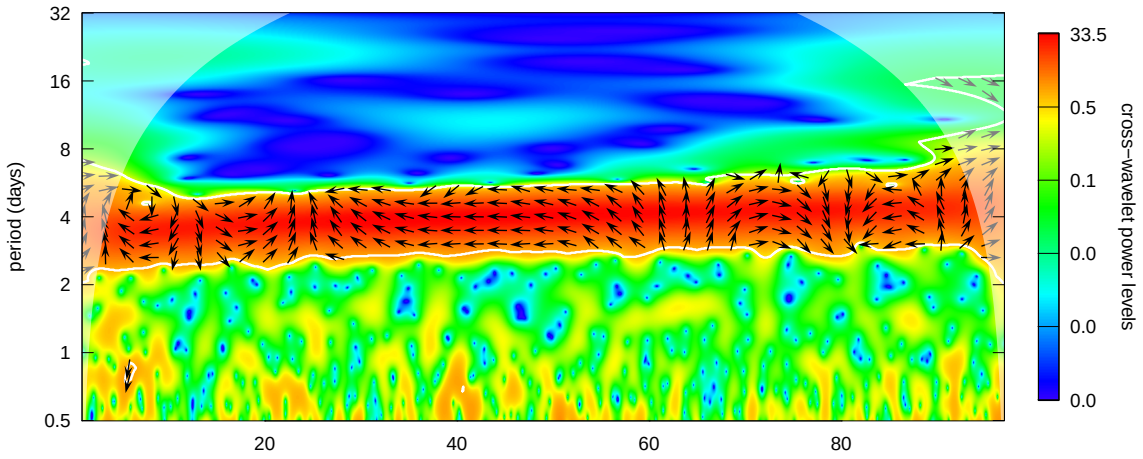

Figure 31: Cross-wavelet power spectrum: significant periods are increasing

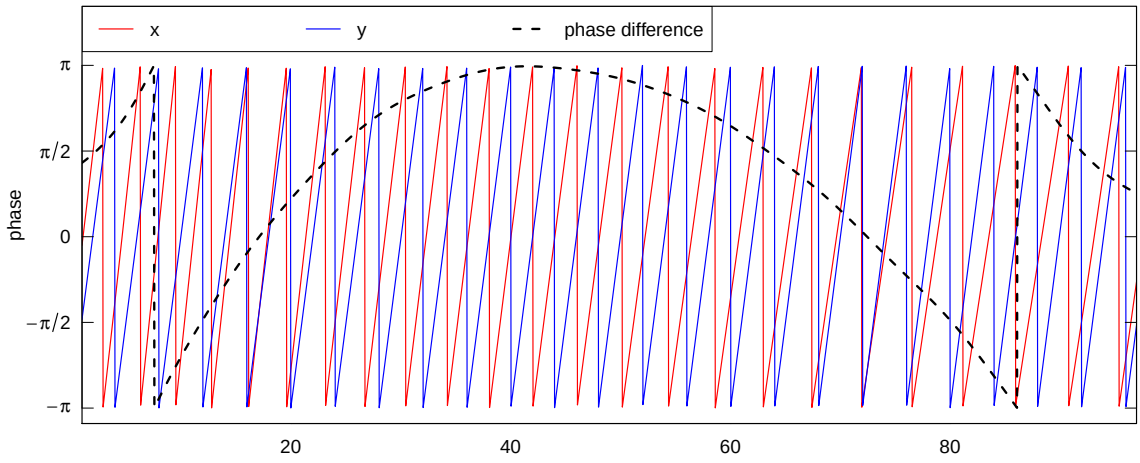

Figure 32: Phases and phase differences: asymmetry due to increasing period

Finally, we'll explore the influence of different smoothing settings on the plot of wavelet coherence. The parameters controlling smoothing are: `window.type.t` and `window.size.t` (`window.type.s` and `window.size.s`), specifying smoothing window type and size in time (period/scale, respectively) direction; see also page 24 and the help text of function `analyze.coherency` for details. Figure 33 is the result of plotting the wavelet coherence with default settings (Bartlett windows):

```
my.wc = analyze.coherency(my.data, my.pair = c("x","y"),
                          loess.span = 0,
                          dt = 1/24, dj = 1/100,
                          lowerPeriod = 1/2,
                          window.type.t = 1, window.type.s = 1,
                          window.size.t = 5, window.size.s = 1/4,
                          make.pval = T, n.sim = 10)
wc.image(my.wc, which.image = "wc", n.levels = 250, color.key = "interval",
         siglvl.contour = 0.1, siglvl.arrow = 0.05,
         legend.params = list(lab = "wavelet coherence levels"))
```

As in Figure 31, the increasing trend of significant periods becomes visible — and as in the previous example (Section 3.2), arrows are only plotted in the area of joint significance. We set `color.key = "interval"` in the call of `wc.image` because it facilitates the comparison of the images on the facing page (different smoothing settings will also affect the distribution of coherence levels, and hence color quantiles).

When the settings of window type and size are changed, `analyze.coherency` has to be run again before calling `wc.image`; however, we use the same simulation of the time series x and y in the following illustration. Figure 34 results from using window type Boxcar: `window.type.t = 3`, `window.type.s = 3`. Due to repeated simulations (carried out by `analyze.coherency`), the white lines indicating significance are now slightly different, and so will be the selection of arrows to be plotted (however, arrows appearing at the same position in Figure 33 and in Figure 31 will have the same direction since `use.sAngle = F` by default); differences other than white lines and arrows are due to different smoothing settings. No smoothing in time direction was applied to produce the wavelet coherence image in Figure 35: `window.type.t = 0`, `window.type.s = 1`. The absence of smoothing in time direction may exaggerate the impression of periodic association of the two series.

As mentioned in Section 1.2, a wavelet coherence of 1 (or, numerically, very close to 1) will result if `window.type.t = 0`, `window.type.s = 0`: if no smoothing is applied, `wc.image` will produce an entirely red image if `color.key = "interval"` and a scratchy image if `color.key = "quantile"`.

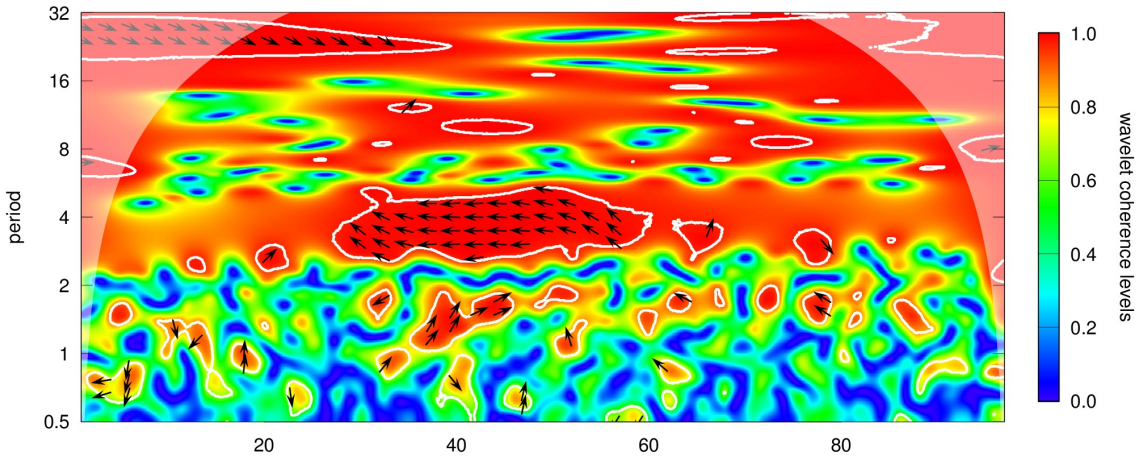

Figure 33: Wavelet coherence, window type: Bartlett

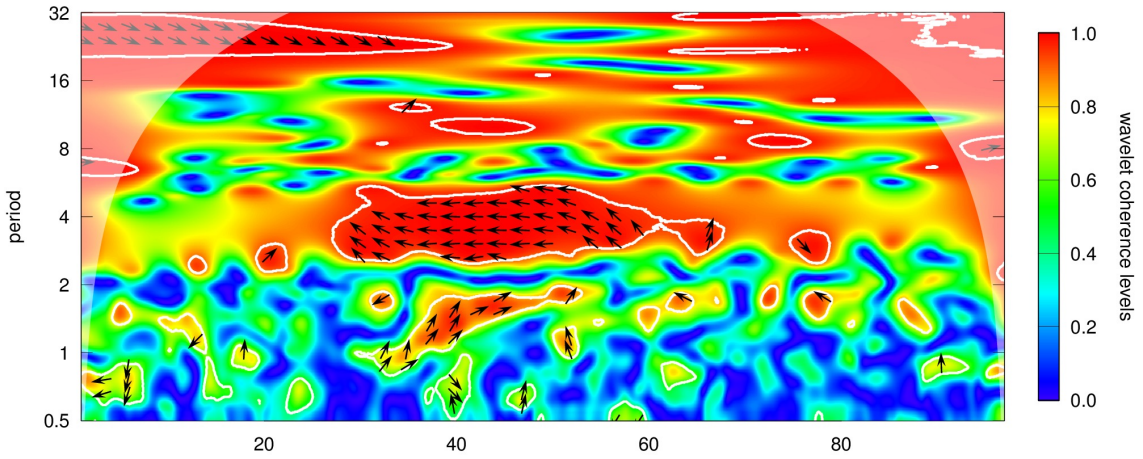

Figure 34: Wavelet coherence, window type: Boxcar

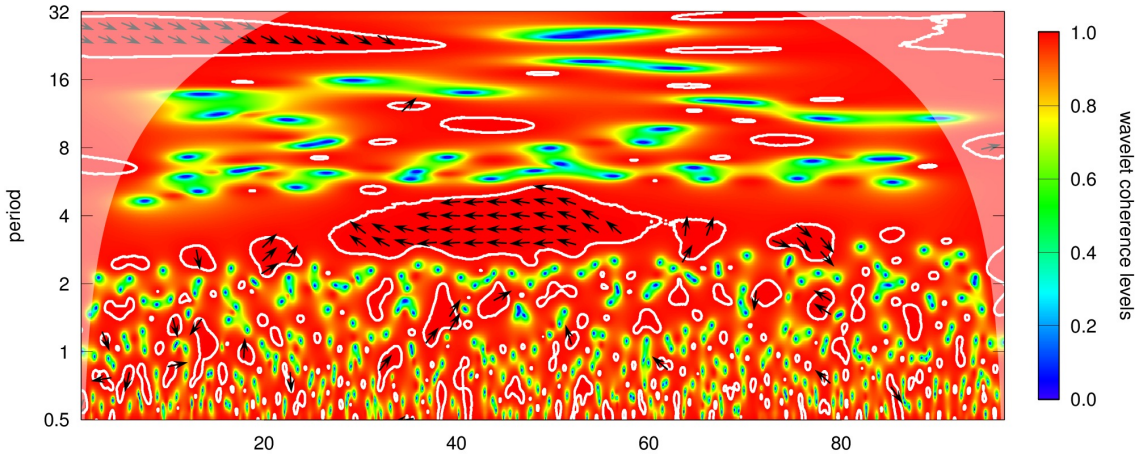

Figure 35: Wavelet coherence, no smoothing along time axis

4 Example: Transactions in the foreign exchange market

4.1 The series of transactions

WaveletComp includes the data set FXtrade.transactions. It gives the worldwide number of USD/euro FX (foreign exchange) transactions recorded in time intervals of five minutes in July 2010.[8] The data set contains four full weekly cycles, plus three days at the beginning of July 2010, where a weekly (active) cycle lasts from Sunday, 21:00, to Friday, 20:55. Each cycle has 1440 five-minute time slots. The number of transactions between Friday, 21:00, and Sunday, 20:55, is 0 or close to 0. The data set has a variable called `active` indicating whether a five-minute interval belongs to an active cycle:

```
> data(FXtrade.transactions)
> head(FXtrade.transactions)
                 date transactions active
1 2010-07-01 00:00:00          603   TRUE
2 2010-07-01 00:05:00          529   TRUE
3 2010-07-01 00:10:00          516   TRUE
4 2010-07-01 00:15:00          711   TRUE
5 2010-07-01 00:20:00          571   TRUE
6 2010-07-01 00:25:00          726   TRUE
```

Our goals here are: (i) to understand the periodic behavior of the series, (ii) to estimate the intensity of transactions through an active cycle. The latter can be used, for example, as intensity function of a Poisson process. The intensity estimate is the result of averaging four parts of the reconstructed series.

4.2 Seasonality in the series of transactions

An obvious, but somewhat naive way to capture the seasonality in the transaction series is

```
my.w = analyze.wavelet(my.data = FXtrade.transactions, "transactions")
wt.image(my.w)
```

The result is not shown here. It turns out that there is no distinct seasonality below ⅛ day = 3 hours. (We could therefore think of collapsing FXtrade.transactions to 30-minute intervals.) To avoid excessive run times and unnecessarily large R objects, we should limit the period range: `lowerPeriod = 1/8`. The result when calling `analyze.wavelet` and `wt.image` is shown in Figure 37: There is 1-day periodicity, interrupted by weekends, and ridges indicating 3-to-4-day and 7-day periodicity.

An idle time interval (`active = FALSE`) lasts exactly 2×24 hours, and the best strategy will be to exclude idle times before conducting wavelet transformation:

```
my.data.a = FXtrade.transactions[FXtrade.transactions$active == T, ]
my.w.a = analyze.wavelet(
           my.data.a, "transactions",
           loess.span = 0.0,  # no detrending required
           dt = 1/(12*24),    # one day has 12*24 5-minute time slots
           dj = 1/250,        # resolution along period axis
           lowerPeriod = 1/8, # lowest period of interest: 3 hours
           make.pval = T,     # draws white lines indicating significance
           n.sim = 10)        # higher number will give smoother white lines

wt.image(my.w.a, n.levels = 250, periodlab = "periods (days)",
         legend.params = list(lab = "wavelet power levels"),
         show.date = T, date.format = "%F %T", timelab = "")
```

The resulting wavelet power transform is shown in Figure 38.

[8] This data set was derived from data provided by Morning Star.

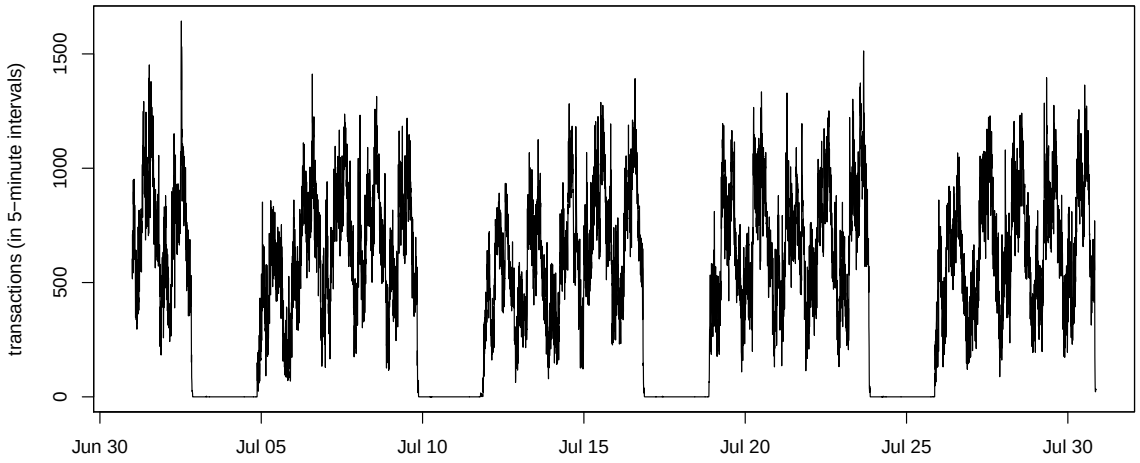

Figure 36: Number of transactions in five-minute intervals

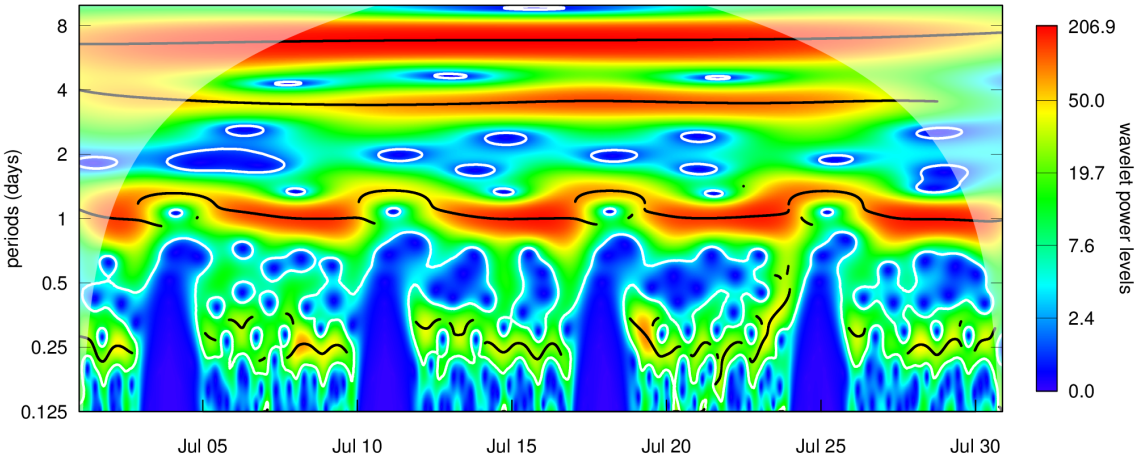

Figure 37: Transactions (original series), the wavelet power spectrum

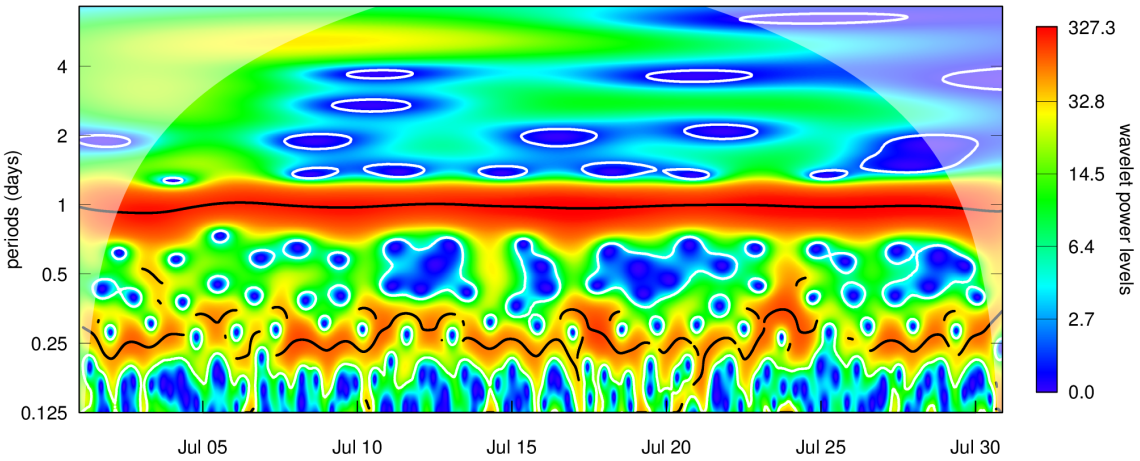

Figure 38: Transactions (active time only), the wavelet power spectrum

4.3 Reconstruction

A reconstructed version of the original series, based on significant periods, is obtained as:

```
my.rec.a = reconstruct(my.w.a, show.date = T, date.format = "%F %T",
                       timelab = "", plot.waves = F)
transactions.rec.a = my.rec.a$series$transactions.r
transactions.rec.a[transactions.rec.a < 0] = 0  # some values are negative
```

A cluttered plot will appear with option `plot.waves = T` — many periods are significant. Of course, a further option can be chosen in `reconstruct`, such as `only.ridge = T`, in order to obtain insight into which frequencies contribute the most. The series `transactions.rec.a` refers to *active* time intervals only; for plotting (see Figure 39) and other purposes, it is useful to fill in zeros for idle times and append the resulting series to data frame `FXtrade.transactions`:

```
transactions.rec = rep(0, nrow(FXtrade.transactions))
transactions.rec[FXtrade.transactions$active == T] = transactions.rec.a
FXtrade.transactions = data.frame(FXtrade.transactions,
                                  transactions.rec = transactions.rec)
```

4.4 Intensity estimation

Transaction intensity can now be estimated by averaging, for each five-minute time slot, the four cycles shown in Figure 39. Series `transactions.rec.a` has length 6300, while a weekly active cycle (5×24 hours) has length 1440 (five-minute intervals). Cutting off $6300 - 4 \cdot 1440 = 540$ values at the beginning (the incomplete cycle):

```
transactions.cycle = matrix(transactions.rec.a[-(1:540)], ncol = 4)
transactions.cycle = rowMeans(transactions.cycle)/5 # transactions per minute
transactions.cycle = data.frame(avg = transactions.cycle,
                                time = substr(my.data.a$date[541:1980], 12, 16))
```

Division by 5 in the second command scales the intensity estimate down to one-minute intervals. The third command provides an easy-to-handle time stamp. The intensity estimate is displayed in Figure 40; the plot is produced as follows:

```
plot(transactions.cycle$avg, type = "l", xaxt = "n", yaxt = "n",
     xlab = "", ylab = "weekly transaction intensity",
     ylim = c(0, max(transactions.cycle$avg)))
index.sel = seq(37, length(transactions.cycle$avg), by = 72)
axis(1, labels = as.character(transactions.cycle$time[index.sel]),
     at = index.sel)
axis(2, labels = seq(0, 200, by = 50),
     at = seq(0, 200, by = 50))
mtext(text = c("Mon", "Tue", "Wed", "Thu", "Fri"), side = 1, line = 2,
      at = seq(181, length(transactions.cycle$avg), by = 4*72))
```

Wavelet analysis assumes that a given series can be decomposed efficiently into periodic components — an assumption which is often quite plausible when investigating recurrent phenomena. In this example, it leads to a neat estimate of transaction intensity. A less sophisticated way to estimate transaction intensity would simply compute, for each five-minute time slot, averages of the *observed* series, rather than bother about wavelet reconstruction. The result is shown in Figure 41 — with four observations at hand, the air is still too thin for the law of large numbers to tidy up efficiently.

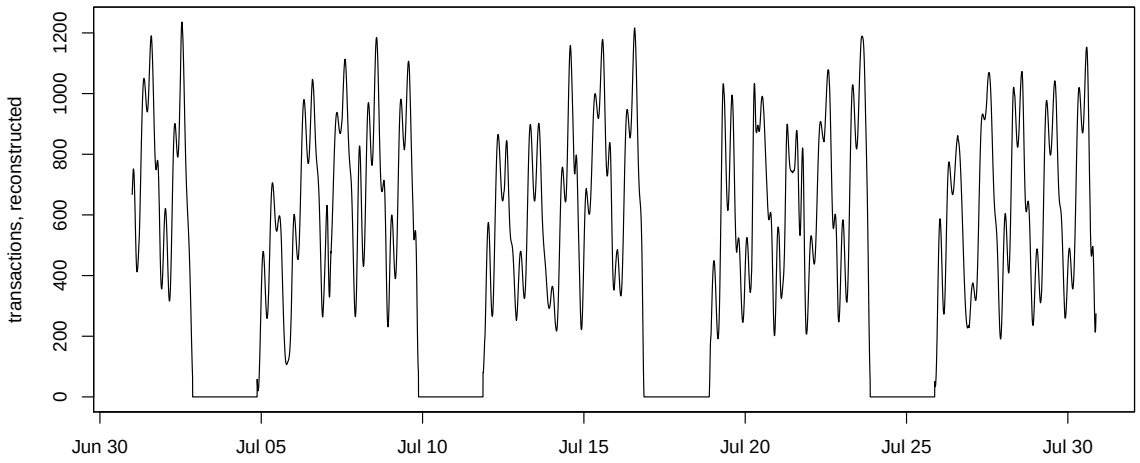

Figure 39: Number of transactions, reconstructed

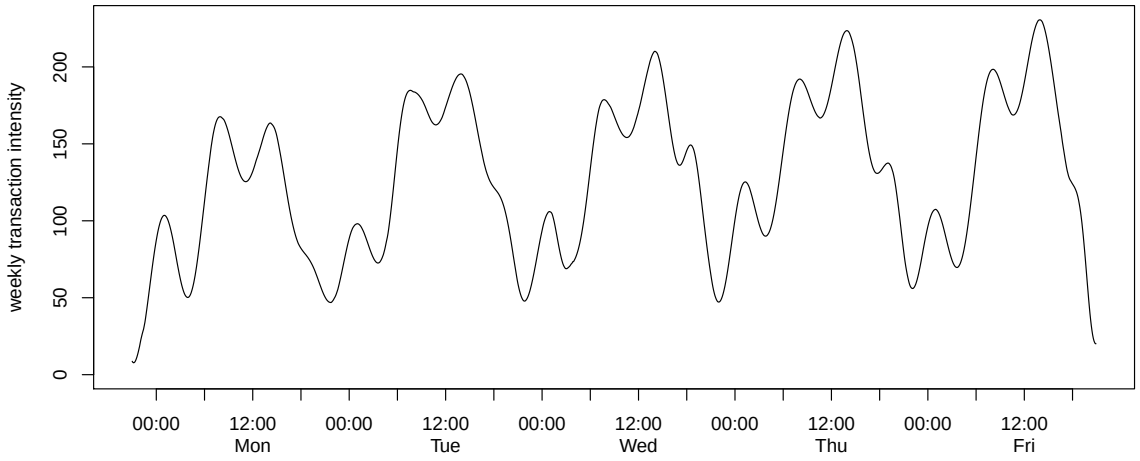

Figure 40: Transaction intensity estimate

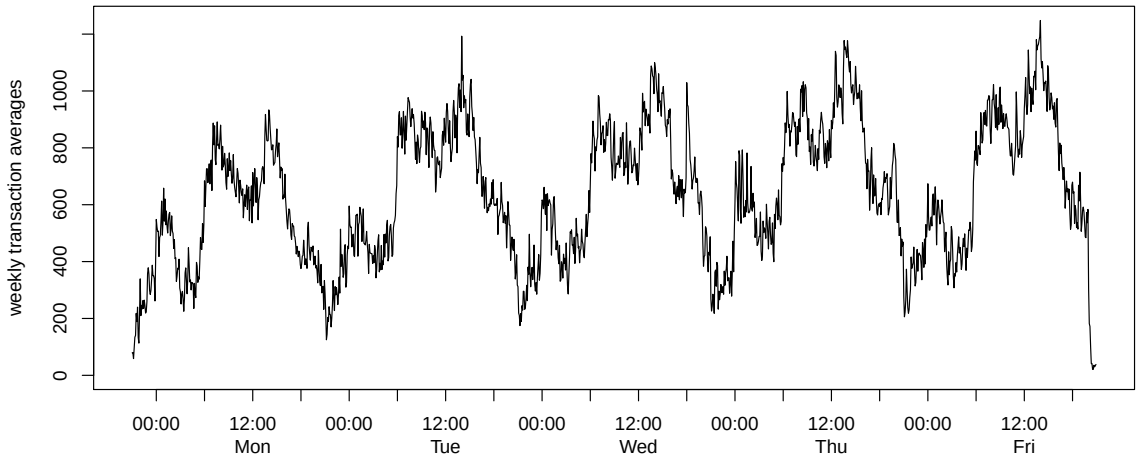

Figure 41: Transactions: weekly averages

5 Example: Marriages in Turkey

5.1 The series of marriages

Data set marriages.Turkey contains the monthly total number of marriages in Turkey from January 1988 through December 2013, together with the number of Sundays for each month:[9]

```
> data(marriages.Turkey)
> head(marriages.Turkey, 3)
        date n.Sun marriages
1 1988-01-31     5     32783
2 1988-02-29     4     32305
3 1988-03-31     4     30596
```

For the analysis below, it is convenient to add columns with "Sunday-adjusted" and logarithmic marriages to the data set:

```
m.adj = marriages.Turkey$marriages*(4/marriages.Turkey$n.Sun)
my.data = data.frame(marriages.Turkey, m.adj = m.adj, log.m.adj = log(m.adj))
```

Series `m.adj` is plotted in Figure 42.

5.2 Seasonality in the series of marriages

The goal here is to understand the structure of seasonality in the time series of marriages. As before, the first step is to compute and plot the wavelet power:

```
my.w = analyze.wavelet(my.data, "log.m.adj",
                       loess.span = 3/26,
                       dt = 1, dj = 1/250,
                       make.pval = T, n.sim = 10)
wt.image(my.w, n.levels = 250,
         legend.params = list(lab = "wavelet power levels"),
         periodlab = "periods (months)", show.date = T, timelab = "")
```

It is adequate to choose `loess.span` as a multiple of 1/26, which corresponds to one year. Setting `loess.span = 3/26` leaves the trend, accessible as `my.w$series$log.marriages.trend`, practically free of seasonality; in other words, setting `loess.span = 3/26` won't interfere with the analysis of seasonality. To see in more detail what is going on around period 12, we restrict the analysis to periods between 8 and 16:

```
my.w = analyze.wavelet(my.data, "log.m.adj",
                       lowerPeriod = 8, upperPeriod = 16,
                       loess.span = 3/26,
                       dt = 1, dj = 1/1000,
                       make.pval = T, n.sim = 10)
wt.image(my.w, n.levels = 250,
         legend.params = list(lab = "wavelet power levels"),
         periodlab = "periods (months)", show.date = T, timelab = "",
         graphics.reset = F)
abline(h = log(12)/log(2))
mtext(text = "12", side = 2, at = log(12)/log(2), las = 1, line = 0.5)
```

The result is shown in Figure 44. In order to plot the horizontal line at 12, we have to set `graphics.reset = F`, which leaves the plot open for further input. Also observe that coordinates on the period axis are given in terms of $\log_2$ values.

[9] The series was obtained from DİE, the Turkish State Institute of Statistics (for the period January 1988 through December 2000), and from its successor TÜİK, the Turkish Statistical Institute (for the period January 2001 through 2013). See http://www.tuik.gov.tr/VeriTabanlari.do?vt_id=21&ust_id=109 (accessed Oct 1, 2014).

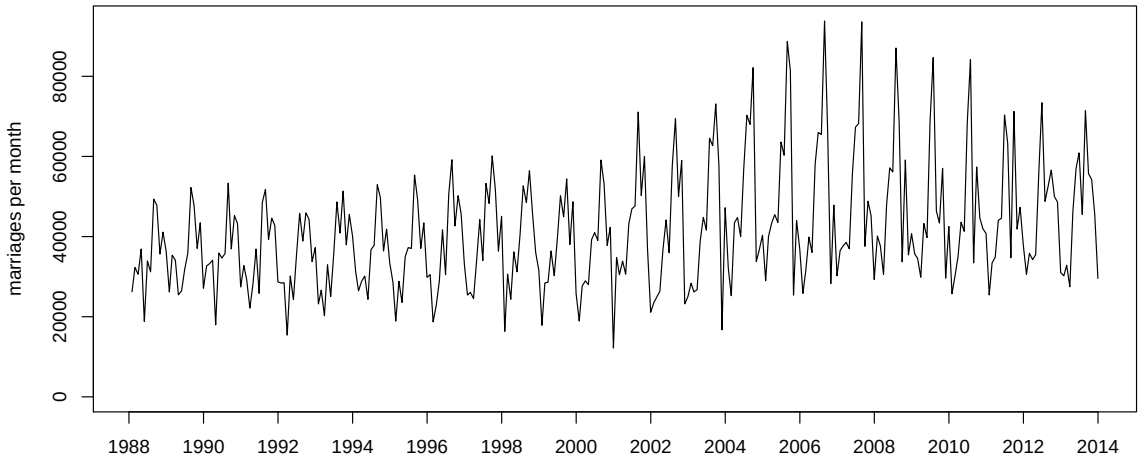

Figure 42: The series of monthly marriages (“Sunday-adjusted”)

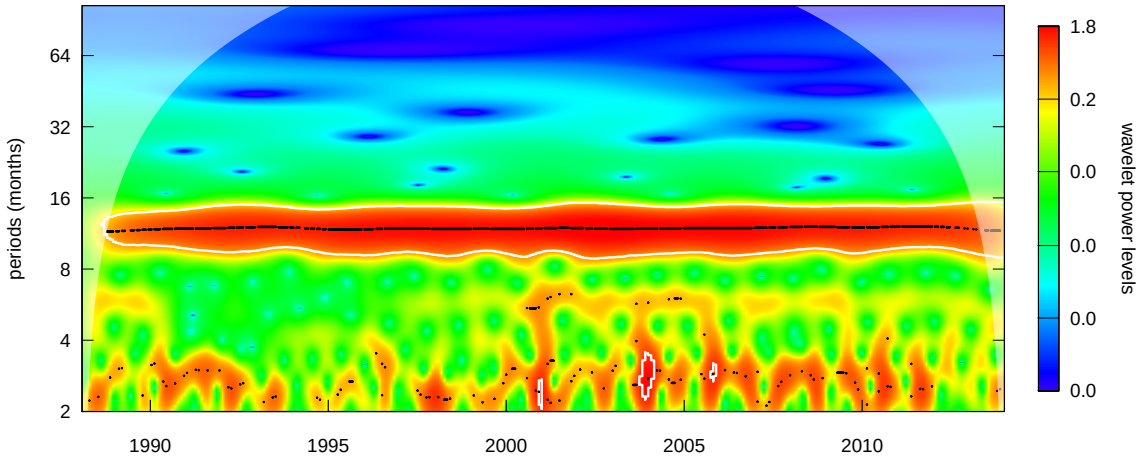

Figure 43: Marriages, the wavelet power spectrum

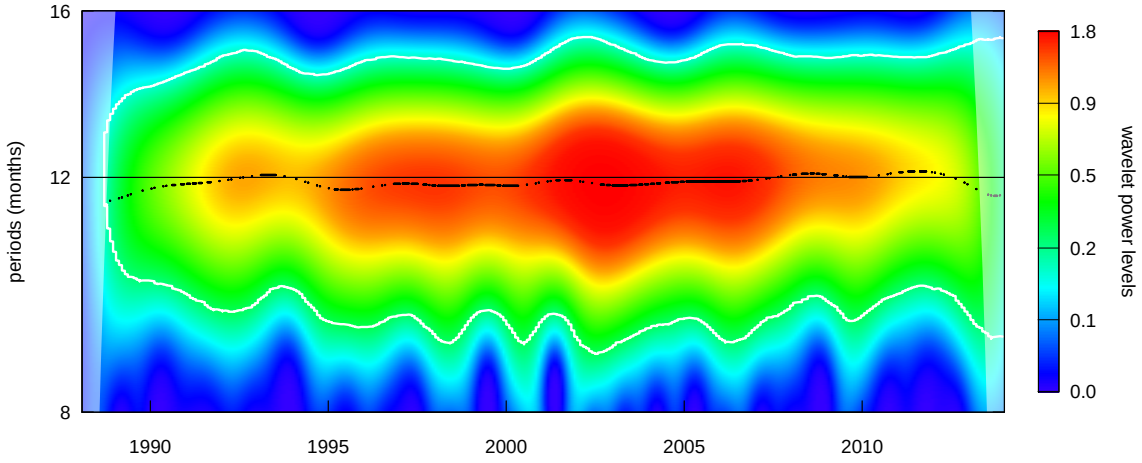

Figure 44: Marriages, the wavelet power spectrum, periods 8 – 16

5.3 Does the Islamic calendar influence seasonality?

The functions of WaveletComp are generally implemented such that intermediate results are easily accessible. In the example at hand, we can extract the ridge of the power spectrum to obtain a function indicating the position of the ridge (in terms of period) for each month. The ridge is given as matrix `my.w$Ridge` of dimension 1001×312; see Section 2.2 above. When converting this matrix into a function, we should make sure each column (referring to a month) contains a single 1 (or none), indicating the presence of a ridge. This can be checked, for example, by entering `colSums(ridge)`, which is indeed a vector of length 312 containing only 0s and 1s. Therefore:

```
ridge = my.w$Period * my.w$Ridge
ridge[ridge == 0] = NA
```

This ridge constitutes the black line in Figure 45. It deviates slightly from the hypothesized 12-month periodicity. These deviations could be explained in terms of a secular change in the timing of marriages, a ridge below 12 (above 12) indicating acceleration (delay, respectively) with respect to the solar calendar. There may be, however, a more straightforward explanation for the shape of the ridge in Figure 45: The timing of marriages in Turkey may be influenced by Islamic festivals. The latter are celebrated according to the Islamic calendar, which is a lunar calendar. As a rule of thumb, it drifts forward 11 or 12 (in the case of a leap year) days each solar year. If Islamic festivals play a role in marriage timing, the lunar calendar may therefore leave traces close to period $12 - 11.25/30 = 11.625$ in the series, and the observed fluctuations might thus result from a superposition of solar and lunar calendar.

5.4 Reproducing the seasonality pattern: a model calculation

Can the simultaneous influence of solar and lunar calendar produce the ridge pattern of Figures 44 and 45? This superposition can be easily simulated as follows:

```
x120 = periodic.series(start.period = 12, length = 312, phase = 7)
x116 = periodic.series(start.period = 11.625, length = 312, phase = 10) * 0.4
y = x120 - x116
my.data.s = data.frame(date = my.data$date, y = y)
```

The first command simulates a series of period 12, attaining its maximum in August (as does the marriage series). The second command simulates what could be the influence of the lunar calendar: maximum influence during Ramadan (in 1988, Ramadan overlapped 12 days with April and 17 days with May), thus leading to the biggest reduction in the number of marriages during Ramadan (this reflects marriage timing in Turkey). Both series are plotted in Figure 46. The wavelet power plot in Figure 47 results from:

```
my.w.s = analyze.wavelet(my.data.s, "y",
                         lowerPeriod = 8, upperPeriod = 16,
                         loess.span = 0,
                         dt = 1, dj = 1/1000,
                         make.pval = F)
wt.image(my.w.s, n.levels = 250,
         legend.params = list(lab = "wavelet power levels"),
         periodlab = "periods (months)",
         show.date = T, timelab = "", graphics.reset = F)
abline(h = log(12)/log(2))
mtext(text = "12", side = 2, at = log(12)/log(2), las = 1, line = 0.5)
```

The ridge can be extracted from `my.w.s`, as shown above, and added to the ridge plot of Figure 45. It turns out that this simple procedure can indeed replicate the overall structure of the ridge in the actual series of marriages. Further analysis (not shown here) reveals that fitting an ARMA model with dummy variables accounting for Ramadan influence to the marriage series produces residuals whose ridge is closer to period 12 than the ridge of the actual series. Wavelet analysis can thus also contribute indirectly to assessing complex seasonality patterns.

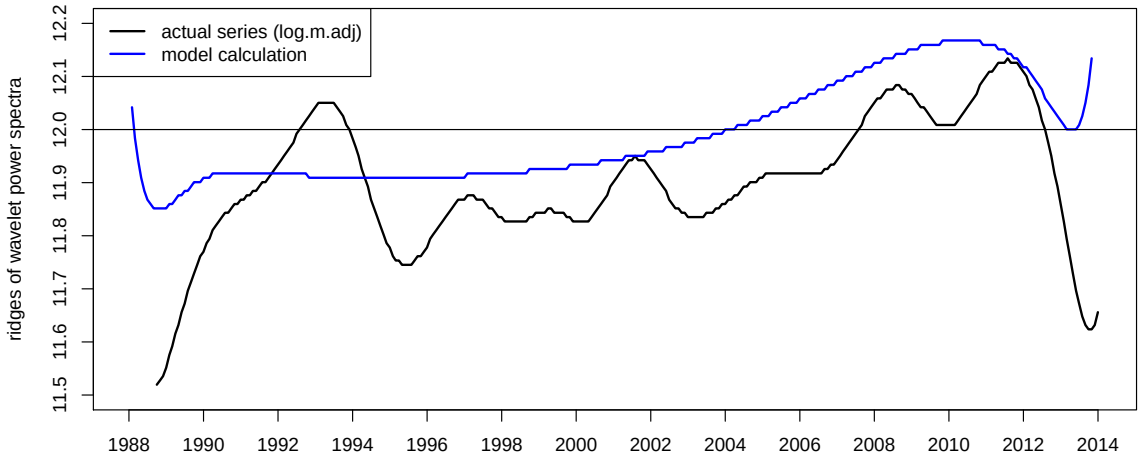

Figure 45: Ridge of power spectra

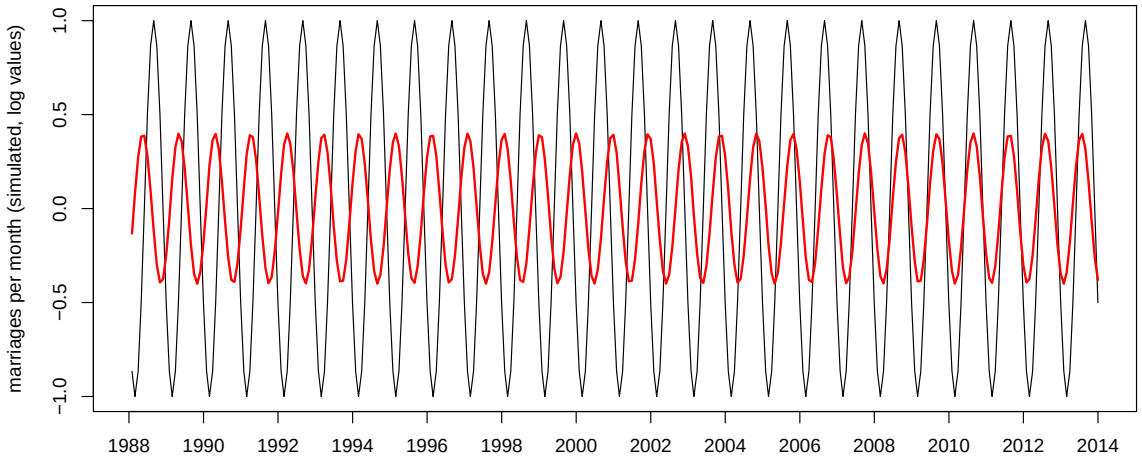

Figure 46: Simulation of marriages, with superposed lunar calendar (red)

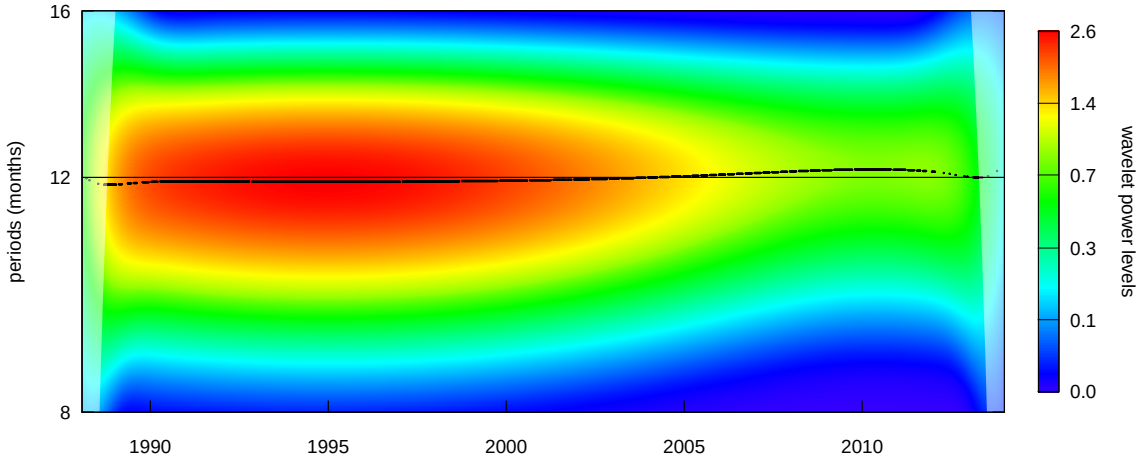

Figure 47: Power spectrum of simulated series

References

[1] Aguiar-Conraria L., and Soares M.J., 2011. Business cycle synchronization and the Euro: A wavelet analysis. *Journal of Macroeconomics* 33 (3), 477–489.

[2] Aguiar-Conraria L., and Soares M.J., 2011. The Continuous Wavelet Transform: A Primer. NIPE Working Paper Series 16/2011.

[3] Carmona R., Hwang W.-L., and Torrésani B., 1998. *Practical Time Frequency Analysis.* Gabor and Wavelet Transforms with an Implementation in S. Academic Press, San Diego.

[4] Cazelles B., Chavez M., Berteaux, D., Ménard F., Vik J.O., Jenouvrier S., and Stenseth N.C., 2008. Wavelet analysis of ecological time series. *Oecologia* 156, 287–304.

[5] Liu P.C., 1994. Wavelet spectrum analysis and ocean wind waves. In: Foufoula-Georgiou E., and Kumar P., (eds.), *Wavelets in Geophysics*, Academic Press, San Diego, 151–166.

[6] Liu Y., Liang X.S., and Weisberg R.H., 2007. Rectification of the Bias in the Wavelet Power Spectrum. *Journal of Atmospheric and Oceanic Technology* 24, 2093–2102.

[7] Gabor D., 1946. Theory of communication. *Journal of the Institution of Electrical Engineers — Part III: Radio and Communication Engineering* 93, 429–441.

[8] Gencay R., Selcuk F., and Whitcher B., 2001. *An Introduction to Wavelets and Other Filtering Methods in Finance and Economics.* Academic Press, San Diego.

[9] Goupillaud P., Grossman A., and Morlet, J., 1984. Cycle-octave and related transforms in seismic signal analysis. *Geoexploration* 23, 85–102.

[10] Morlet J., Arens G., Fourgeau E., and Giard D., 1982. Wave propagation and sampling theory – Part I: complex signal and scattering in multilayered media. *Geophysics* 47, 203–221.

[11] Morlet J., Arens G., Fourgeau E., and Giard D., 1982. Wave propagation and sampling theory – Part II: sampling theory and complex waves. *Geophysics* 47, 222–236.

[12] R Core Team, 2014. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

[13] Torrence C., and Compo G.P., 1998. A practical guide to wavelet analysis. *Bulletin of the American Meteorological Society* 79 (1), 61–78.

[14] Veleda D., Montagne R., and Araujo M., 2012. Cross-Wavelet Bias Corrected by Normalizing Scales. *Journal of Atmospheric and Oceanic Technology 29, 1401–1408.*