WaveletComp 1.1:
A guided tour through the R package

Angi Rösch[*]/ Harald Schmidbauer[†]

© 2018 Angi Rösch / Harald Schmidbauer

(This version: March 18, 2018)

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 POSIXct 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

[†]Shanxi University of Finance and Economics, Department of Statistics, Taiyuan, China; and BRU-IUL, ISCTE Business Research Unit, ISCTE-IUL, Lisbon, Portugal; e-mail: harald@hs-stat.com

Contents

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 [13]. 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 (version 1.1) 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 (version 1.1). 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:

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

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. [6]. Several research articles provide a comprehensive introduction to continuous wavelet analysis: Torrence and Combo [15], 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. [7]. It builds on a Gaussian-windowed sinusoid, the Gabor transform, which was introduced in 1946 by Gabor [5] 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. [7]), and the feasibility of series reconstruction.

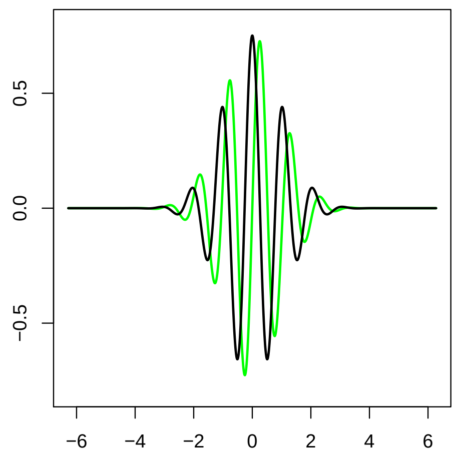


Figure 1: The Morlet mother wavelet — real part (black line) and imaginary part (green line)

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 [15] 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. [9], since the mere modulus produces "biased" wavelet amplitudes in the sense that high-frequency (short-period) phenomena tend to be underestimated:

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

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]):

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

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 easily possible through customizing the axes. 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",

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

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. 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:

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

The reconstruction factor 0.776 is adopted from Torrence and Combo [15] 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. [16]:

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

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 by default (which can be easily changed), 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.

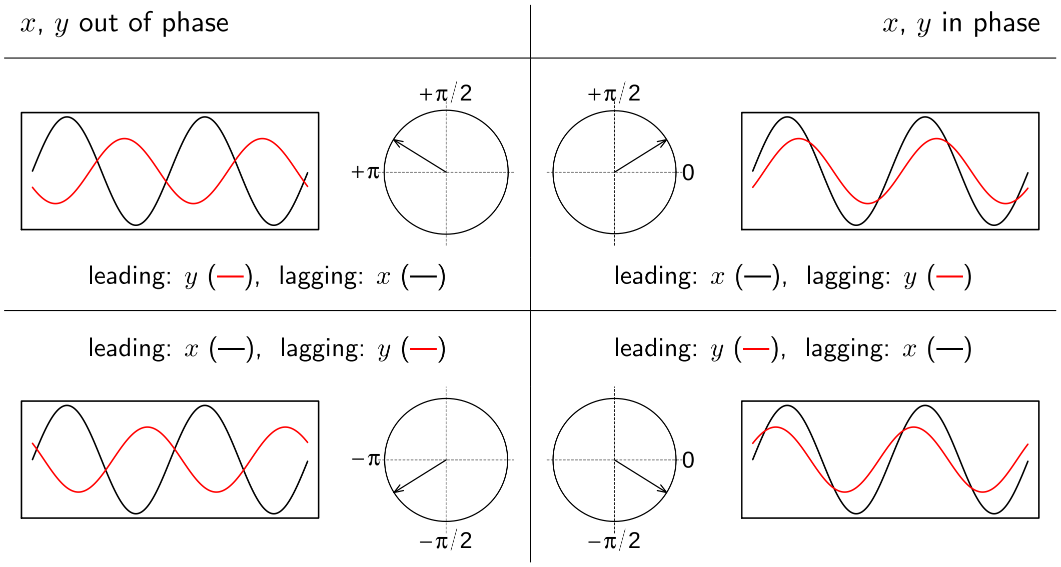

x, y out of phase
x, y in phase
$+\pi/2$
$+\pi$
$+\pi/2$
0
leading: y (—), lagging: x (—)
leading: x (—), lagging: y (—)
leading: x (—), lagging: y (—)
leading: y (—), lagging: x (—)
$-\pi$
0
$-\pi/2$
$-\pi/2$

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 [8]):

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

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 [15], 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. [9] in the univariate, and Veleda et al. [16] 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 [8]), 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 POSIXct 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) and axes 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.

[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.

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 in WaveletComp build on ideas developed by Huidong Tian and Bernard Cazelles (archived R package "WaveletCo"[5]). WaveletComp code for the computation of the cone of influence and for the arrow design to depict phase differences uses ideas 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]).

What's new in Version 1.1?

Tools for displaying and analyzing periodic phenomena across time have been extended. The main innovations are:

- All functions of family `wt.<>` (showing results concerning a single time series) can now also be applied to extract univariate outcomes from cross-wavelet and coherence analysis (objects of class `"analyze.coherency"`).
- It is possible to control the color gradation of time-period spectrum plots, and accentuate the contrast, by raising the wavelet power values to any (positive) exponent before plotting.
- Setting a maximum level for the color bar facilitates the visual comparison of time-period spectrum plots. Maximum and minimum plot levels are options for plots of averages too.
- The time and period axes are now easier to individualize by specifying tick marks and labels. Coordinates on the time axis can be conveniently addressed via an index or a POSIXct object.
- Graphical parameters of global coverage (`cex.axis`, `font.axis`, `cex.lab`, `font.lab`, `mgp` etc., see `par`) as well as parameters of local coverage (within axis specification options) help fine-tune plots.
- Two more real-world data sets have been included in WaveletComp, namely:
  - Data set weather.radiation.Mannheim, containing daily weather and ambient radiation readings from Mannheim (Germany). See Section 6 for details and analysis.
  - Data set USelection2016.Instagram, containing hourly numbers of candidate-related media uploads to Instagram right before the 2016 US presidential election. See Section 7 for details and analysis.

About the present booklet

This booklet shows, step by step, how to use WaveletComp in applications. It is not intended to replace the WaveletComp manual, which contains detailed information about how to use the functions and their parameters, together with many additional examples.

This booklet is optimized for two-sided printout (or dual view on a screen): explanations and WaveletComp code on the left-hand side (that is, on even-numbered pages), and the output in the form of plots on the right-hand side (on odd-numbered pages).

It should be possible to copy and paste ready-to-use code from the booklet to an R script file or into the R terminal, but there may be exceptions. For example, the symbol "^" (as in $x\hat{}2 \equiv x^2$) failed to copy correctly on our platform and had to be inserted manually.

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

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

1.4 Citing WaveletComp

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 (2018). WaveletComp: Computational
  Wavelet Analysis. R package version 1.1.
  https://CRAN.R-project.org/package=WaveletComp

A BibTeX entry for LaTeX users is

  @Manual{,
    title = {WaveletComp: Computational Wavelet Analysis},
    author = {Angi Roesch and Harald Schmidbauer},
    year = {2018},
    note = {R package version 1.1},
    url = {https://CRAN.R-project.org/package=WaveletComp},
  }

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

. . . and accordingly refer to WaveletComp in their work, especially in publications.

2 Analysis of a univariate time series

2.1 Example 1: a series with constant period

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(1000)  # 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 = TRUE, 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 (= `1/dt`) observation of x is made per time unit. (This defines the time unit.)
- `lowerPeriod = 16, upperPeriod = 128`: This defines the range of periods, expressed in time units, 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 = TRUE, 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, displayed in Figure 4:

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

The 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 from the color bar to the right margin; 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 = FALSE, lwd = c(1,2),
            legend.coords = "bottomleft", ylim = c(-1.8, 1.8))
```

This produces the plot in Figure 5. 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.

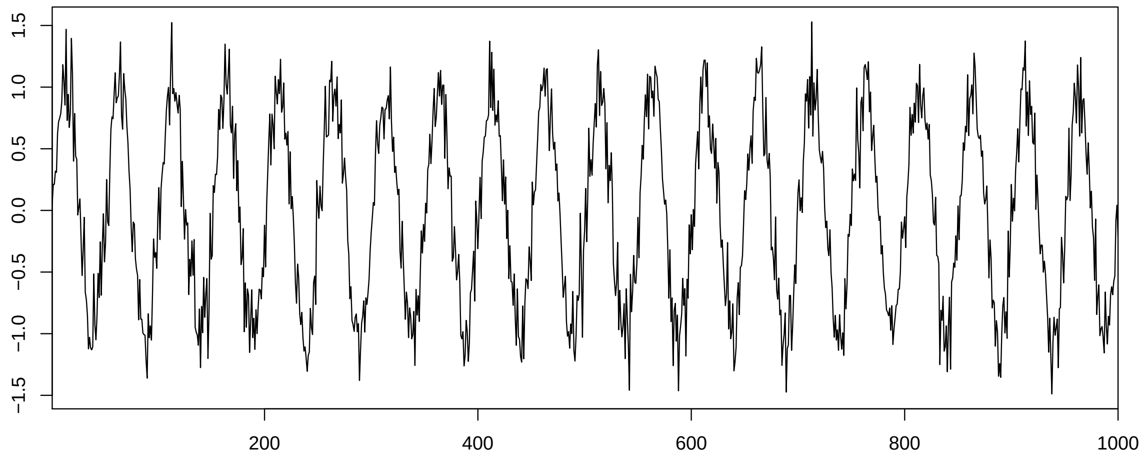


Figure 3: Series with period 50

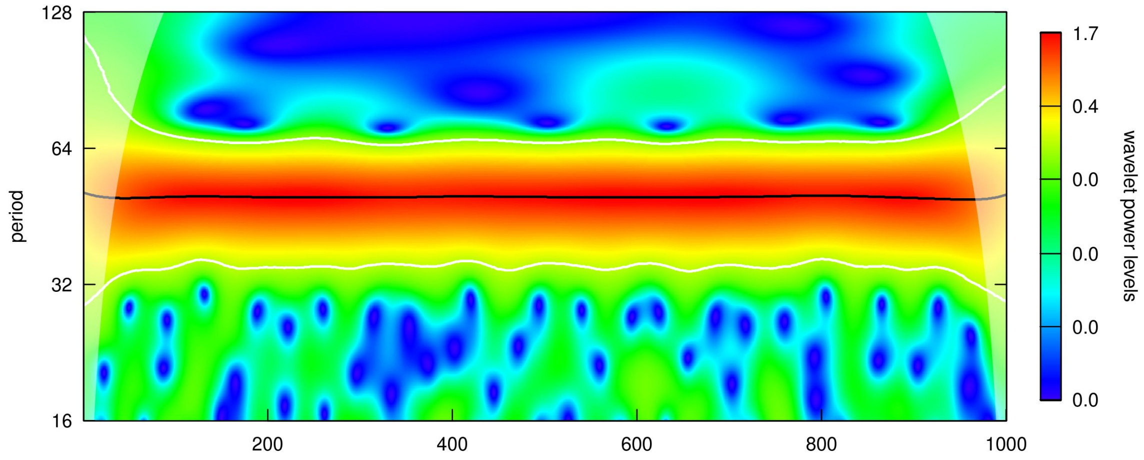


Figure 4: Wavelet power spectrum of the series

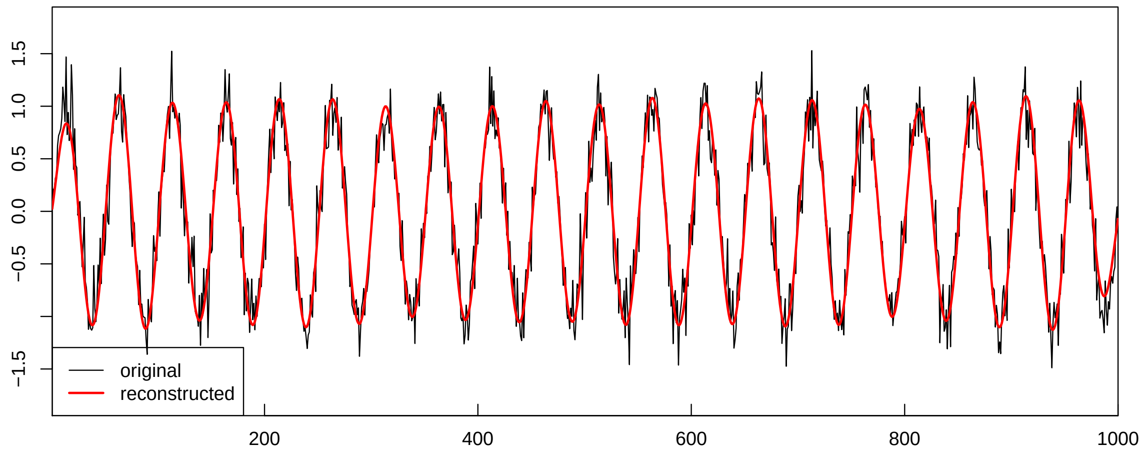


Figure 5: Reconstruction of the series

2.2 Example 2: a series with variable period

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 = TRUE, 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, in other words: the p-values of the null hypothesis that there is no period (given by `my.w$Period`) in the series at time t, tested against an alternative specified in `method` (see Section 2.5 below). The area with p-values below 0.1 (the default) is marked with a white line in the wavelet power plot (in Figure 7, the area of high power within the white lines).
- `my.w$Ridge`, a matrix of 0s and 1s where a 1 indicates the presence of a ridge in the "landscape" of power. The rigde is represented by a black line in the wavelet power spectrum.

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`. See Section 5.1 for an example where detrending is advisable.

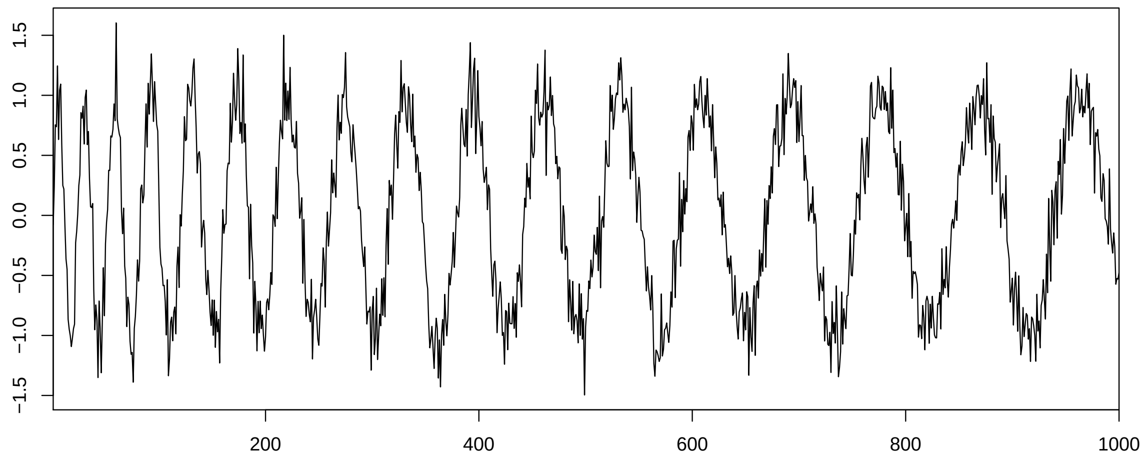


Figure 6: A series wth variable period

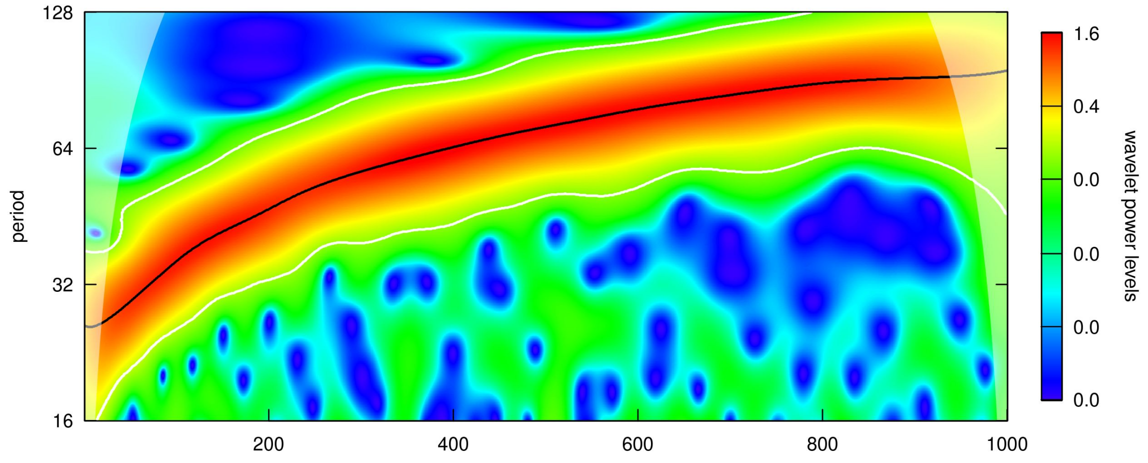


Figure 7: Wavelet power spectrum of the series with variable period

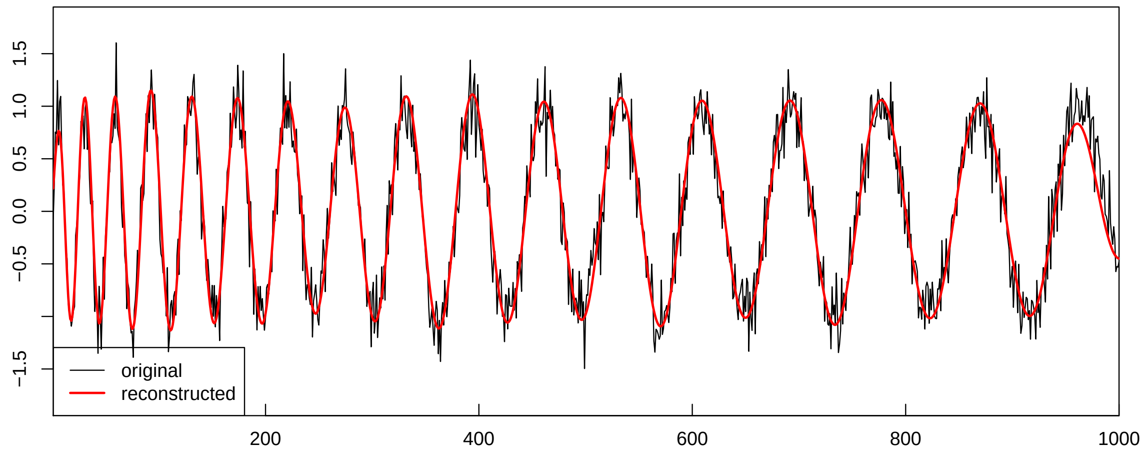


Figure 8: Reconstruction of the series with variable period

2.3 Example 3: a series with two periods

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 = 80, 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 = TRUE, n.sim = 10)
wt.image(my.w, n.levels = 250,
         legend.params = list(lab = "wavelet power levels") )
```

Reconstruction, using only period 80 (see Figure 11):

```
reconstruct(my.w, sel.period = 80, plot.waves = TRUE, lwd = c(1,2),
            legend.coords = "bottomleft")
```

With `sel.period = 80`, the option `plot.waves = TRUE` will produce a wave plot limited to the period chosen — not cluttered, as would have been the case in the earlier examples. Function `reconstruct` actually uses the period in `my.w$Period` closest to 80, which is 80.11497 in this case:

```
> my.w$Period[(my.w$Period > 79) & (my.w$Period < 81)]
[1] 79.01201 79.23138 79.45136 79.67195 79.89316 80.11497 80.33741 80.56046
[9] 80.78413
```

A selection `sel.period = c(30, 80)`, for example, is also possible. Omitting the optional argument `sel.period = 80` 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.

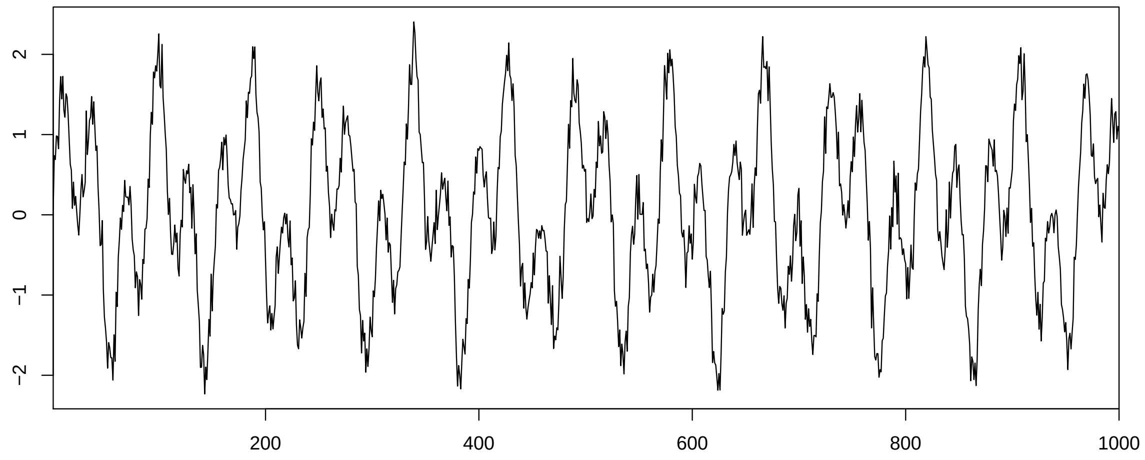


Figure 9: Series, two periods superposed

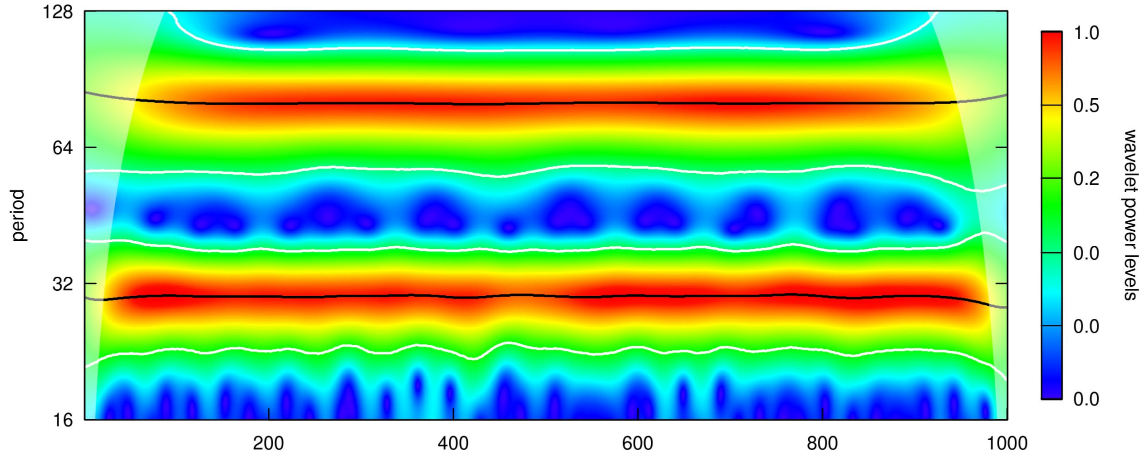


Figure 10: Wavelet power spectrum of the series with superposed periods

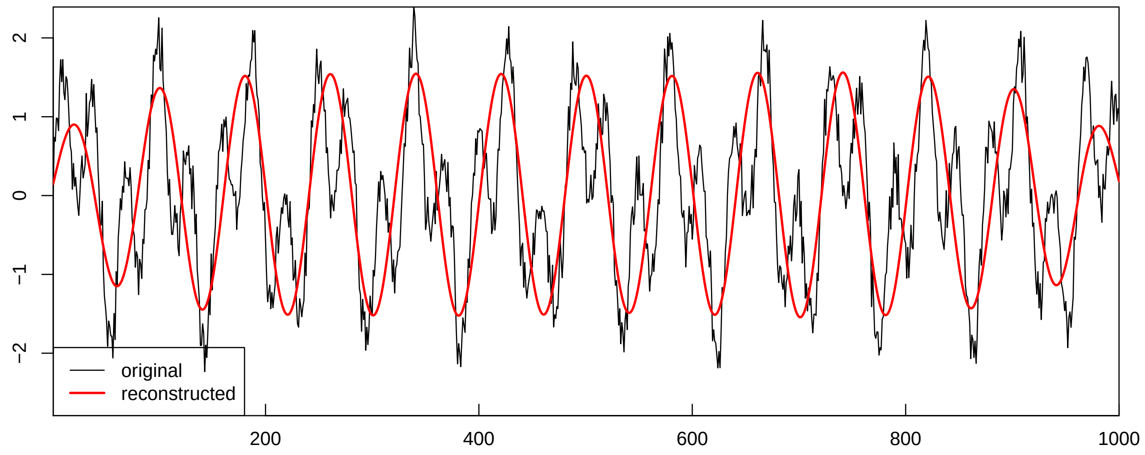


Figure 11: Reconstruction of the series with superposed periods, using only period 80

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) + 0.3*rnorm(1000)
y1 <- periodic.series(start.period = 100, length = 1000)
y2 <- 1.2*periodic.series(start.period = 60, length = 1000)
y <- (y1 + y2)/2  + 0.3*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 = TRUE, n.sim = 10)
my.wy <- analyze.wavelet(my.data, "y", loess.span = 0,
                         dt = 1, dj = 1/20,
                         lowerPeriod = 16, upperPeriod = 256,
                         make.pval = TRUE, 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:

```
maximum.level = 1.001*max(my.wx$Power.avg, my.wy$Power.avg)
wt.avg(my.wx, maximum.level = maximum.level)
wt.avg(my.wy, maximum.level = maximum.level)
```

The maximum average power is made slightly larger by multiplication with 1.001 because the *reported* maximum value may not match R's *internal* accuracy; omitting this step may lead to an error. In the call of `analyze.wavelet` above, 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.

Further experiments can be conducted to get a feel for wavelet analysis, for example:

- Redistribute the weight of `x1` and `x2` in the definition of `x`. For example, replace 1.2 with 1.0. The relative average power of period 100 will increase, but still be less than the average power of period 60: only five cycles of period 100, but more than eight cycles of period 60, fit into length 500, so that the series provides more information about period 60. The relative difference of the respective average of the powers will disappear when the series is made much longer (2×5000 observations, say). This is due to decreasing border effects at a given period as the series becomes longer; in other words: the cone of influence becomes relatively larger.

- In `x`, add more noise to *one* part (`x1` or `x2`) only — this will reduce the average power of *both* periods.

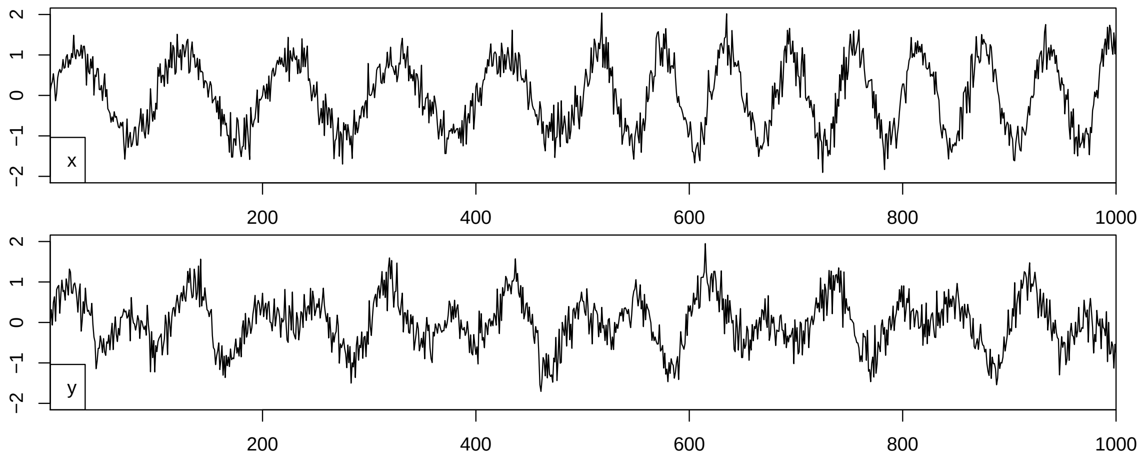


Figure 12: Two series with distinct periods

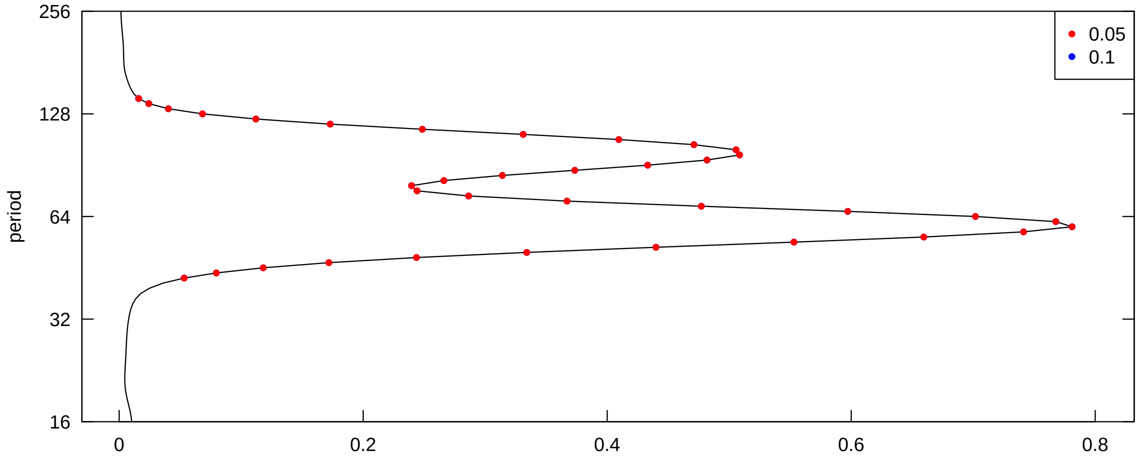


Figure 13: Average power, series x

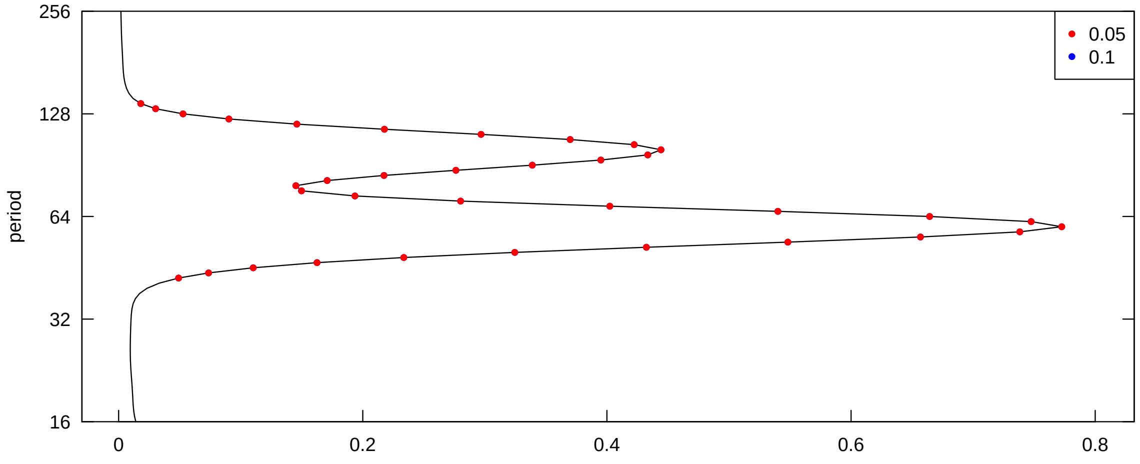


Figure 14: Average power, series y

2.5 Selecting the method of analysis

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 series given (more precisely, in the model that created the series given) is tested. In any case, this test is carried out by WaveletComp using repeated simulations (the number is specified 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 four inner cycles, whose period is 50. These inner cycles also have a slightly higher amplitude:

```
x1 <- periodic.series(start.period = 100, length = 400)
x2 <- 1.2*periodic.series(start.period = 50, length = 200)
x  <- c(x1, x2, x1) + 0.2*rnorm(1000)
```

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 = TRUE, 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 751×1000, 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.

The plot in Figure 17 is the result of using `method = "Fourier.rand"`. 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 where the period is 50: Because Fourier randomization assumes that there is *constant* periodicity over time, it only detects the deviations from the average periodicity in the middle.

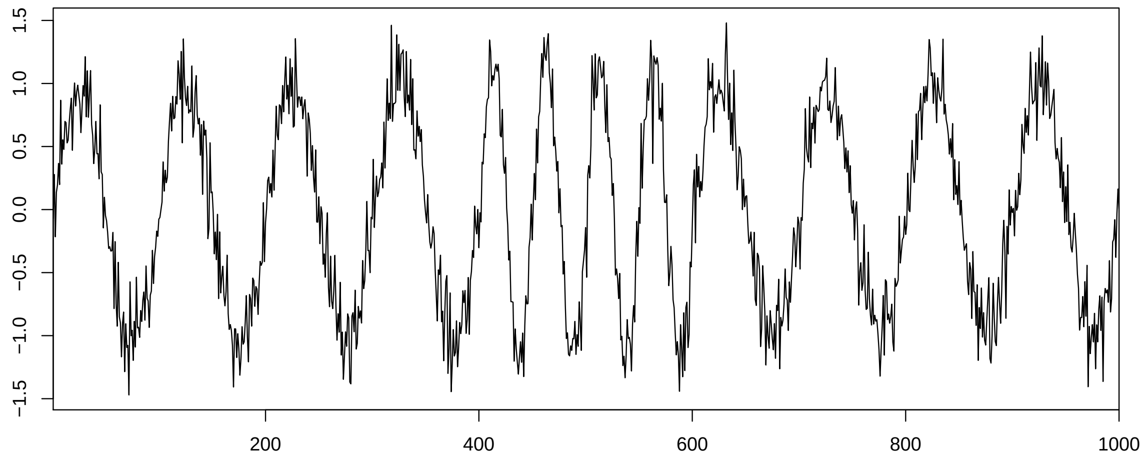


Figure 15: Time series with periods 50 (in the middle) and 100

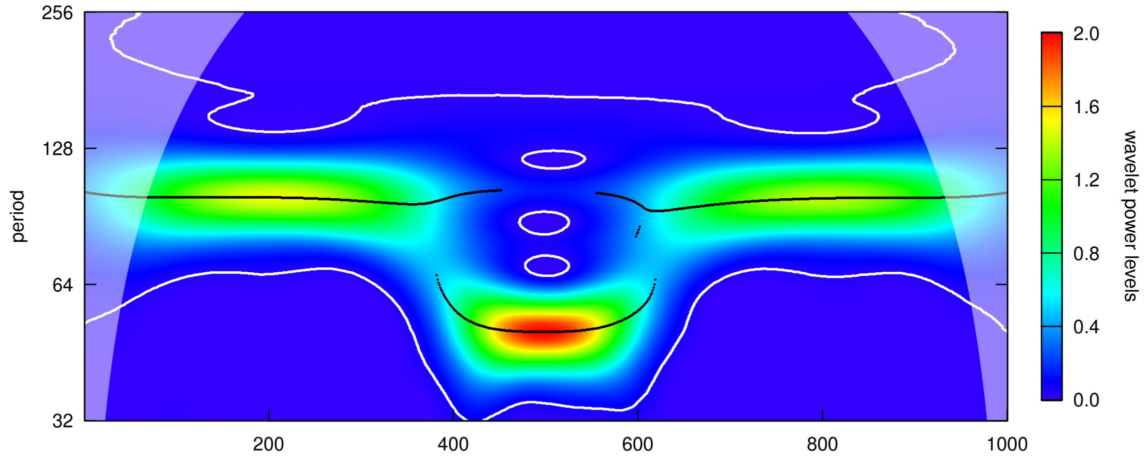


Figure 16: Wavelet power spectrum, `method = "white.noise"`

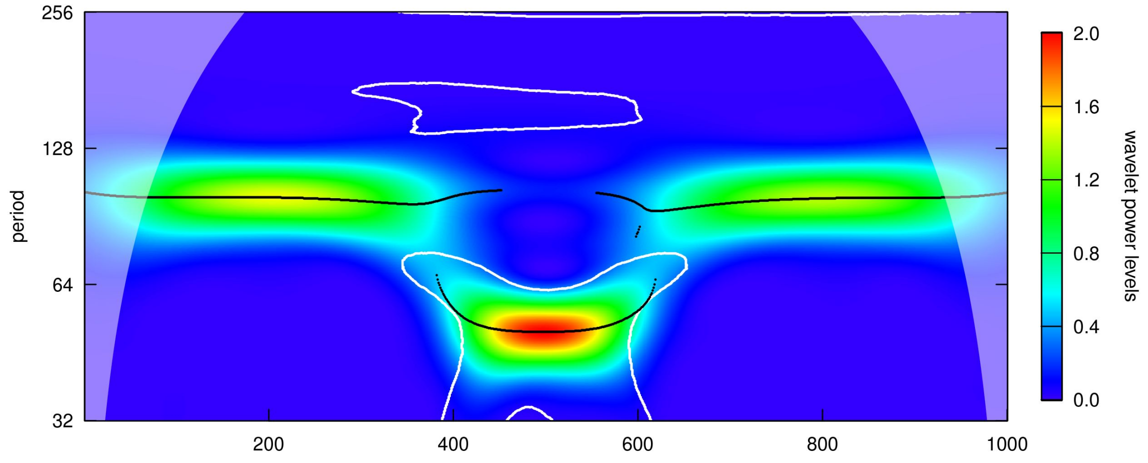


Figure 17: Wavelet power spectrum, `method = "Fourier.rand"`

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 artifacts. 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 = FALSE, max.contour.segments = 500)
```

The resulting image (see Figure 19) has two problems: there are stripes (due to `useRaster = FALSE`), and the white significance line is interrupted (due to `max.contour.segments = 500`). 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 = TRUE` 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((n.levels):1/n.levels)",
         col.ridge = "blue")
```

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.

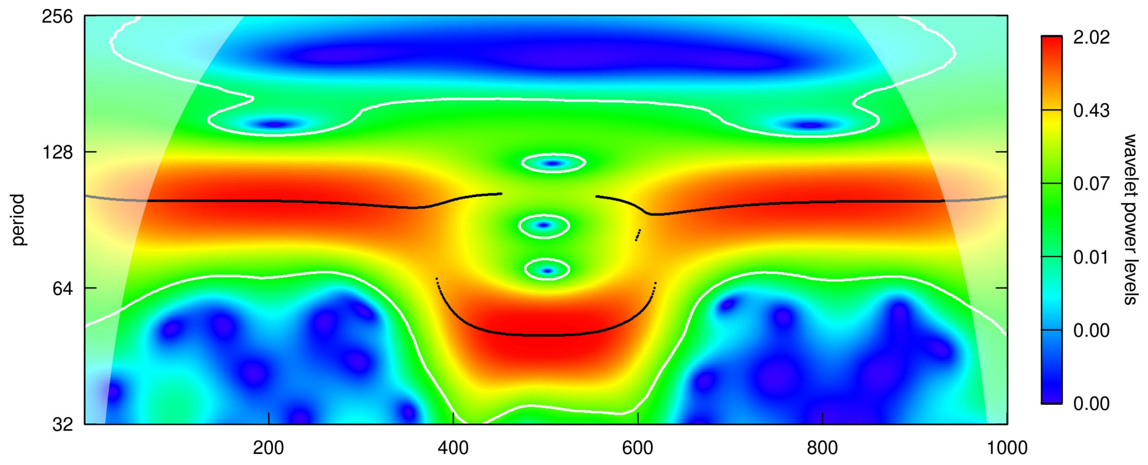


Figure 18: Wavelet power spectrum, `color.key = "quantile"`

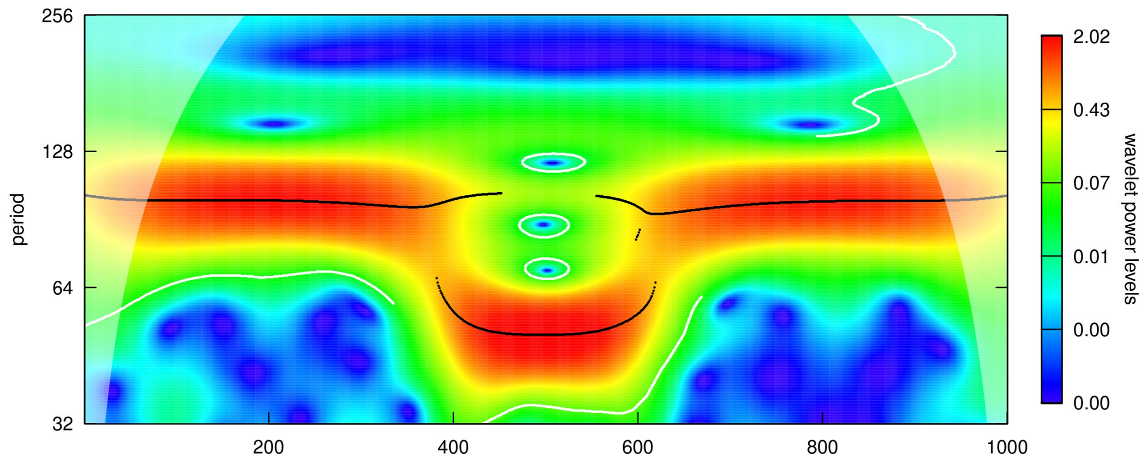


Figure 19: Wavelet power spectrum, plot gone wrong

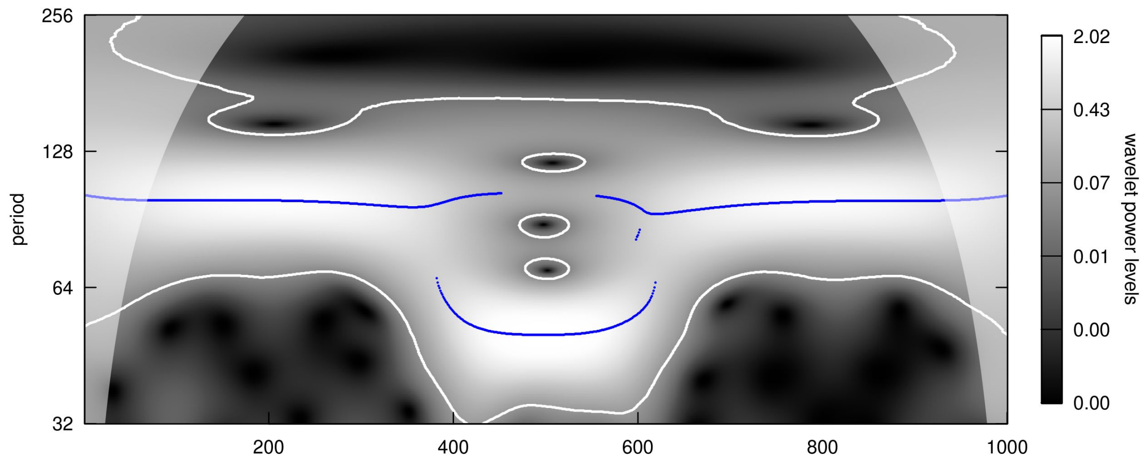


Figure 20: Wavelet power spectrum, grayscale, blue ridge

2.7 Time axis styles

We simulate an hourly time series, with a 24-hour period, over the first 30 days of 2018:

```
epoch.seq <- seq(from = as.POSIXct("2018-01-01 00:00:00"),
                 to = as.POSIXct("2018-01-30 23:00:00"), by = 3600)
x <- periodic.series(start.period = 24, length = 720)
x <- x + rnorm(720)
my.data <- data.frame(date = epoch.seq, x = x)
```

(This series is not plotted here.) The wavelet power spectrum, resulting from the following analysis, is shown in Figure 21:

```
my.w <- analyze.wavelet(my.data, "x", loess.span = 0, dt = 1/24, dj = 1/500,
                        lowerPeriod = 1/4, upperPeriod = 2, make.pval = FALSE)
```

The purpose of the following is to show how to produce different time axis styles when plotting the power spectrum. WaveletComp allows for two ways to address the coordinates of the time axis:

- Enumerating the observations; this is the case with `show.date = FALSE` (the default setting). In the example above, this means the points on the time axis are assigned coordinates $1, \ldots, 720$.
- Overlaying the time axis with a calendar ranging from the first to the last observation epoch; this is the case with `show.date = TRUE`. This option, which internally uses as.POSIXct to process the dates given in `my.data`, will yield correct results only if time in `my.data` proceeds linearly (see Section 2.8 below). In the example above, this means any object of class POSIXct representing a date and time between 2018-01-01, 0:00, and 2018-01-30, 23:00, corresponds to a point on the time axis.

The base code for the plots in Figure 21 is:

```
wt.image(my.w, periodlab = "periods (days)",
         legend.params = list(lab = "wavelet power levels"),
         label.time.axis = TRUE,  # default setting
         ...)
```

Examples of time axis styles are:

- **Version 1** — the default axis, obtained with the base code.
- **Version 2** — time elapsed, measured in days: add the following to the commands above:

  ```
  spec.time.axis = list(at = seq(1, 720, by = 48),
                        labels = seq(0, 28, by = 2))
  ```
- **Version 3** — time elapsed, measured in hours: add the following to the commands above:

  ```
  spec.time.axis = list(at = seq(1, 720, by = 50),
                        labels = seq(0, 719, by = 50))
  ```
- **Version 4** — automatic calendar: add the following to the commands above:

  ```
  show.date = TRUE, date.format = "%F %T"
  ```
- **Version 5** — user-defined calendar: *before* calling `wt.image`, define:

  ```
  ticks <- seq(as.POSIXct("2018-01-01 00:00:00", format = "%F %T"),
               as.POSIXct("2018-01-31 23:00:00", format = "%F %T"), by = "week")
  labels <- seq(as.Date("2018-01-01"), as.Date("2018-01-29"), by = "week")
  labels <- paste("Mon, ", labels)
  ```

  Then, add the following to the commands above:

  ```
  show.date = TRUE, date.format = "%F %T",
  spec.time.axis = list(at = ticks, labels = labels, las = 2)
  ```

For Versions 2 and 3, `timelab` also has to be specified. The same commands also work with all other WaveletComp plot commands involving a time axis. Ticks can be modified too; see Section 6 below.

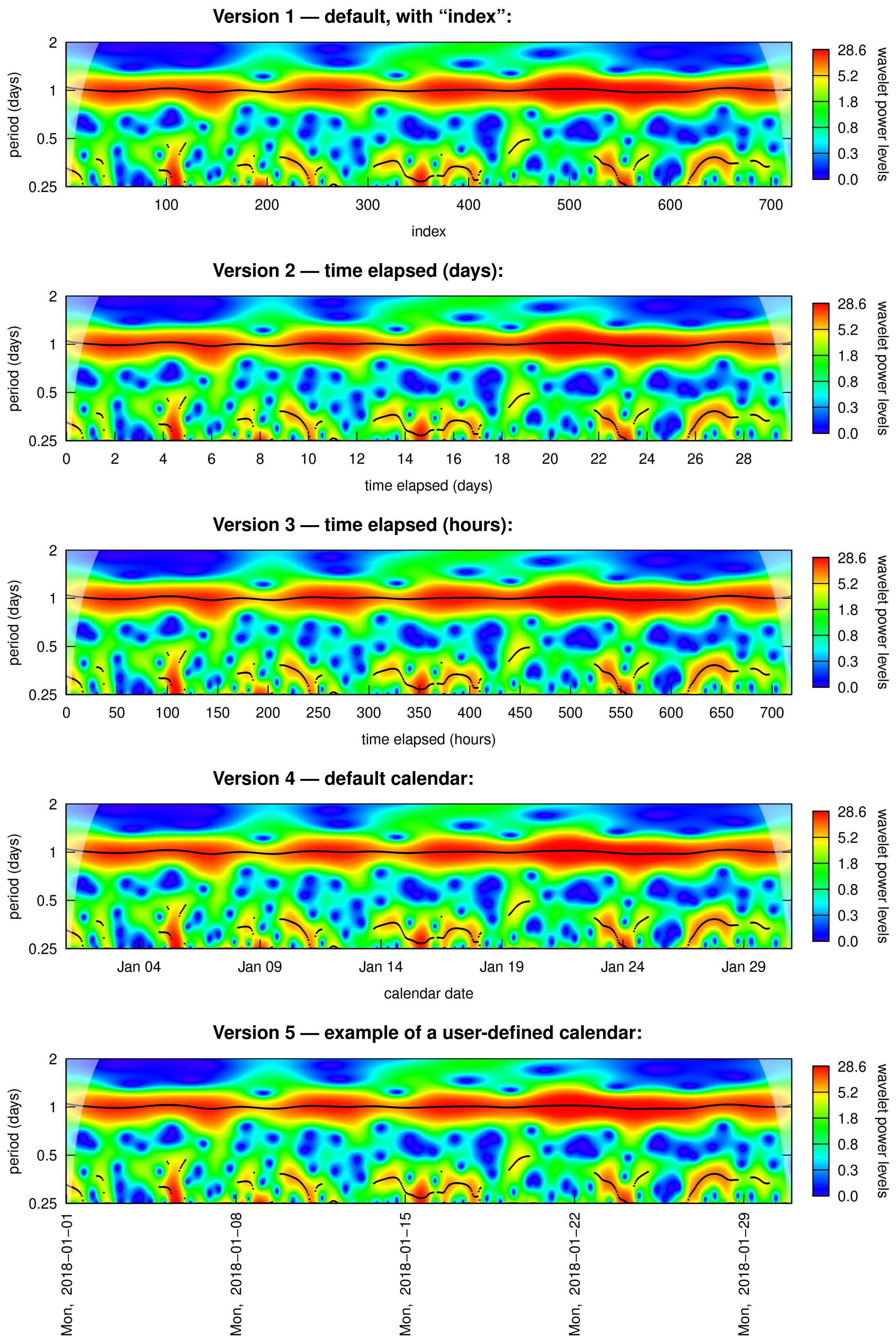


Figure 21: Wavelet power spectrum, different time axis styles

2.8 Some reflections on time — and an example gone wrong

A method to reveal the periodic properties of a time series, such as wavelet transformation, will only yield valid results if observations are made at equidistant points in time. WaveletComp does not test whether input data satisfy this condition, as it is probably not possible to formulate general criteria precisely — what can be accepted as "equidistant" depends on the particular application. For example, daily observations of a stock index are usually considered as being made at equidistant epochs, even though no observation is available at weekends when the stock exchange is closed. However, the situation is often more intricate, which may entail serious consequences for wavelet analysis. This is illustrated in the example below.

The following code constructs a data frame with all weekdays of 2018 and a time series with weekly (low on Mondays, high on Fridays) and monthly (for simplicity, we let a month have 22 workdays) seasonality:

```
days <- seq(from = as.Date("2018-01-01"),
            to = as.Date("2018-12-31"), by = "day")
days <- days[!is.element(weekdays(days), c("Saturday", "Sunday"))]
x <- rep(c(-1, -0.5, 0, 0.5, 1), 53)[1:length(days)] # 53 weeks of 5-day period
x <- x + periodic.series(start.period = 22, length = length(days))
x <- x + 0.5 * rnorm(length(days))                     # noise
my.data <- data.frame(date = days, x = x)
```

A typical realization is shown in Figure 22. One could imagine finance or business data of a company. Analyzing this series:

```
my.w <- analyze.wavelet(my.data, "x", loess.span = 0, dt = 1, dj = 1/200,
                        lowerPeriod = 3, upperPeriod = 50, make.pval = FALSE)
wt.image(my.w, color.key = "interval", n.levels = 250,
             legend.params = list(lab = "wavelet power levels"),
             periodlab = "periods (days)",
             show.date = TRUE, date.format = "%Y-%m-%d", timelab = "")
```

The result is shown in Figure 23. So far, so good. Now suppose the company is closed for vacation from the beginning of July to mid-August, so that the time series is interrupted, retaining only what is outside of the shaded interval in Figure 22:

```
my.data.part <-
   my.data[ (my.data$date < "2018-07-01") | (my.data$date > "2018-08-15"), ]
```

The code above, with `my.data.part` substituted for `my.data`, then leads to the result shown in Figure 24. There are two problems here, due to the data interruption:

- The plot looks as if there were a structural break — although the underlying data-generating model (and even the very realization to be transformed) is the same in Figures 23 and 24.
- The time axis in Figure 24 pretends that data were available throughout July and August. To be sure, fixing the time axis won't make the plot more meaningful.

WaveletComp supports plotting and labeling the time axis. WaveletComp's procedure in functions like `wt.image` is to produce the plot "in one piece", irrespective of time. Then, with `show.date = TRUE`, the time axis is labeled assuming that time proceeds linearly between the epochs of the first and the last observation — an assumption which makes perfect sense if observation epochs are equidistant.

To sum up: The user is responsible for providing input data suitable for wavelet analysis. (This is, of course, true for all the tools of statistical analysis.)

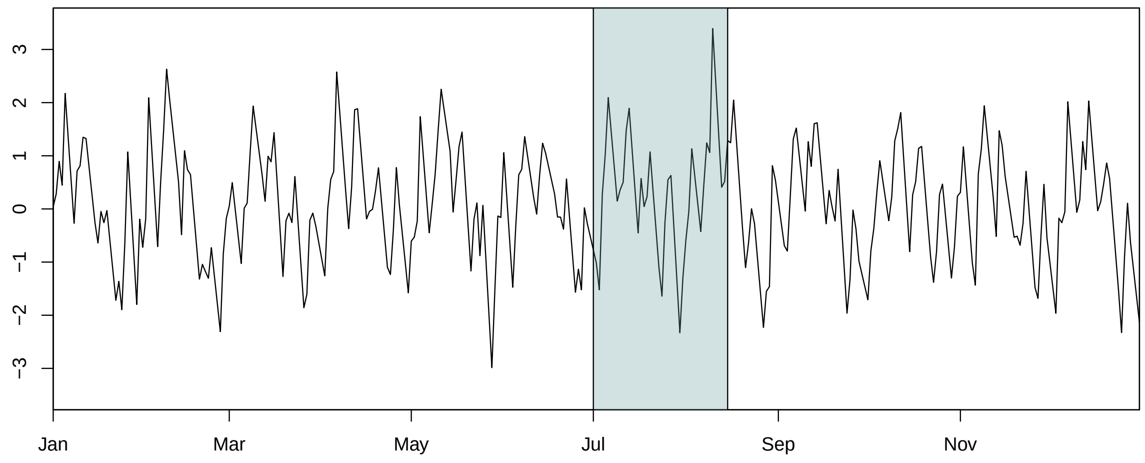


Figure 22: Simulated series with weekly and monthly periods

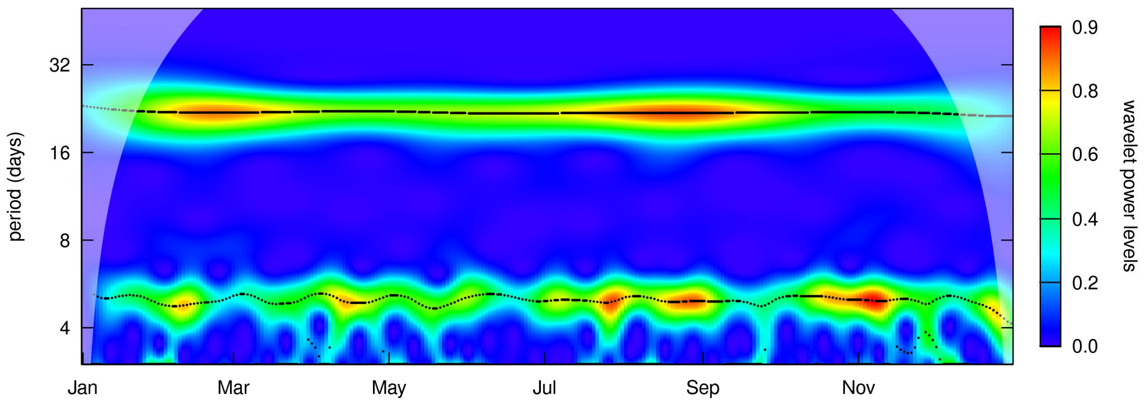


Figure 23: Wavelet power spectrum of the entire simulated series

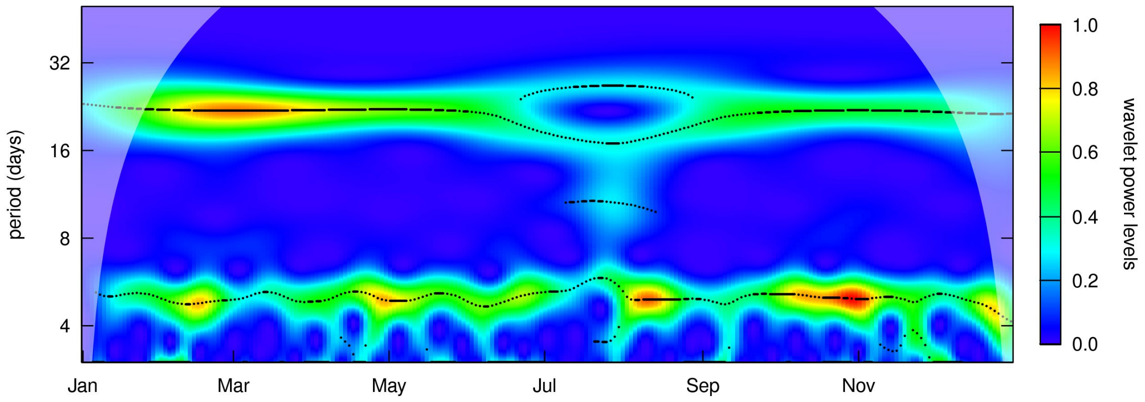


Figure 24: Wavelet power spectrum of a part of the simulated series — this plot is nonsense!

3 Analysis of a bivariate time series

3.1 Example 1: constant periods

The following is a simplified version of an example discussed by Veleda et al. [16]:

```
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 25) represent hourly observations from a 96-day interval. The time axis labels of Figure 25 simply count through the observations; for certain applications, it may be more intuitive to display days (see also Section 2.7). 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 = TRUE, 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 = "", periodlab = "period (days)")
```

This produces the plot in Figure 26. Horizontal arrows pointing to the right indicate that the two series `x` and `y` are in phase at the respective period with vanishing phase differences. Likewise, horizontal arrows pointing to the left indicate that the two series are in anti-phase; this is the case at period 4, due to the opposite sign of `x3` in the definition of `x` and `y`. The arrows are plotted only within white contour lines indicating significance (with respect to the null hypothesis of white noise processes) at the 10% level.

The ridges are not plotted in this image of the cross-wavelet power spectrum; adding them is possible via the argument `plot.ridge = TRUE`, but they will be concealed by the arrows. Finally, a plot of the time-averaged cross-wavelet power is produced by the following command (see also Section 2.4):

```
wc.avg(my.wc, siglvl = 0.01, sigcol = "red", sigpch = 20,
       periodlab = "period (days)")
```

Here, a solid circle (`sigpch=20` by default) indicates that the respective period is significant at the 1% level. (It is also possible to define `siglvl` as a vector.) Figure 27 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); see also the remarks in Section 2.4.

Now that function `analyze.coherency` has been called to produce `my.wc`, individual wavelet power spectra (not shown here) 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` can also be used to obtain results concerning only `x` or `y`.

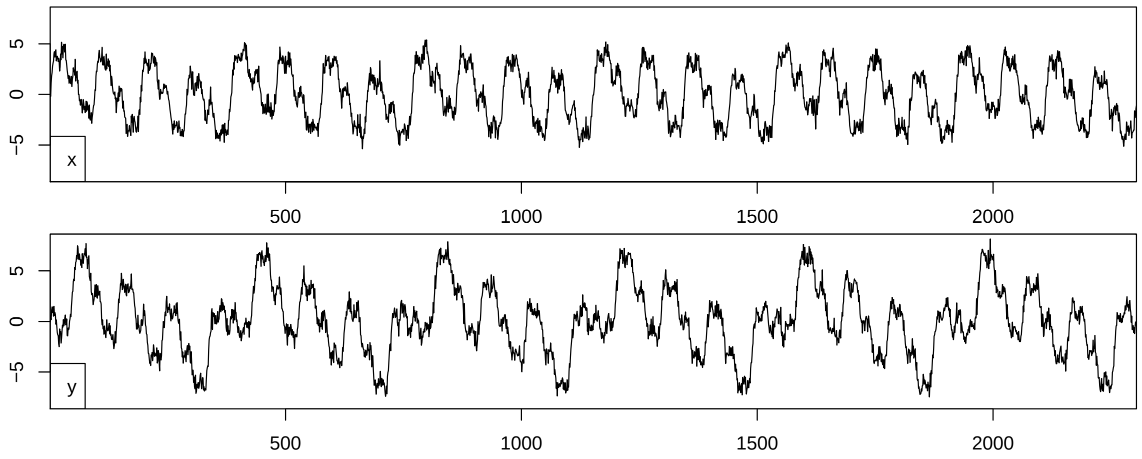


Figure 25: Two series with periods 1, 2, 4, 8, 16 and different amplitudes

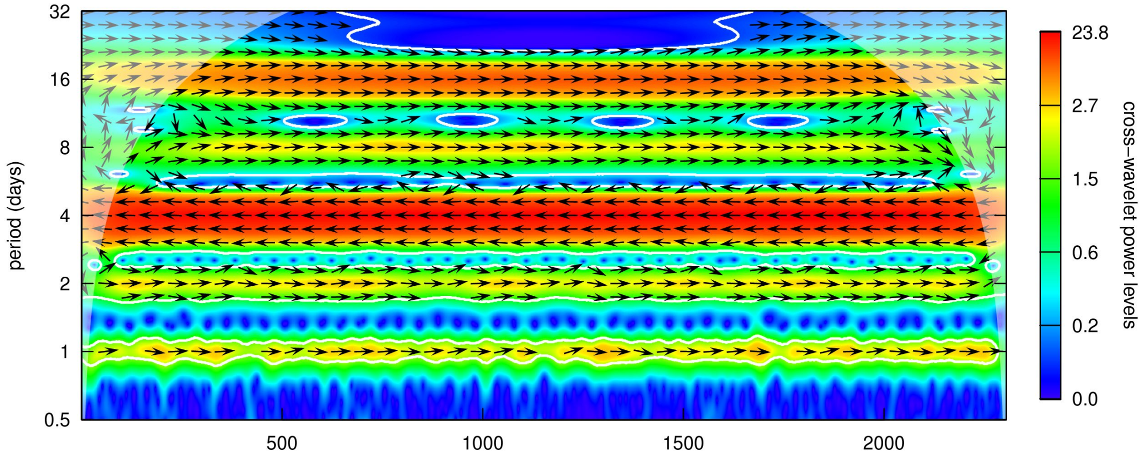


Figure 26: Cross-wavelet power spectrum of x and y

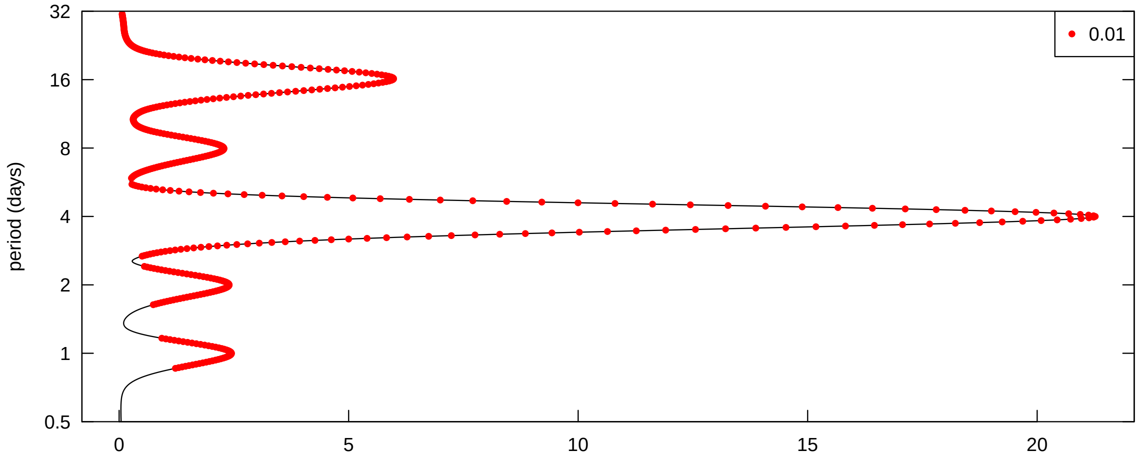


Figure 27: Cross-wavelet average power

3.2 Example 2: variable periods, power spectrum

The time series x shares period 128 with series y within a certain range of time, and y is shifted:

```
xx <- periodic.series(start.period = 64, length = 128*3)
xy <- periodic.series(start.period = 128, length = 2*128*3)
x <- c(xx,xy,xx) + 0.2*rnorm(4*128*3)
y <- periodic.series(start.period = 128, phase = -16, length = 4*128*3) +
     0.2*rnorm(4*128*3)
```

See Figure 28. The cross-wavelet transform of x and y is computed as before:

```
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 = TRUE, 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,  ## default values
         legend.params = list(lab = "cross-wavelet power levels"),
         timelab = "")
```

The result, which is somewhat counterintuitive, is displayed in Figure 29. 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 they tilt away 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, emphasizing joint periodic properties of x and y:

1. Limit the area where arrows are drawn to the region where both *individual* wavelet transforms of x and y show significance (set `which.arrow.sig = "wt"`), and avoid the artifacts of the image in Figure 29 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"),
      timelab = "")
   ```

   This produces the image in Figure 30.

2. Plot wavelet coherence, rather than the cross-wavelet power — like the coefficient of correlation, it adjusts for individual (one-dimensional) power differences in series x and y. This is shown in Section 3.3.

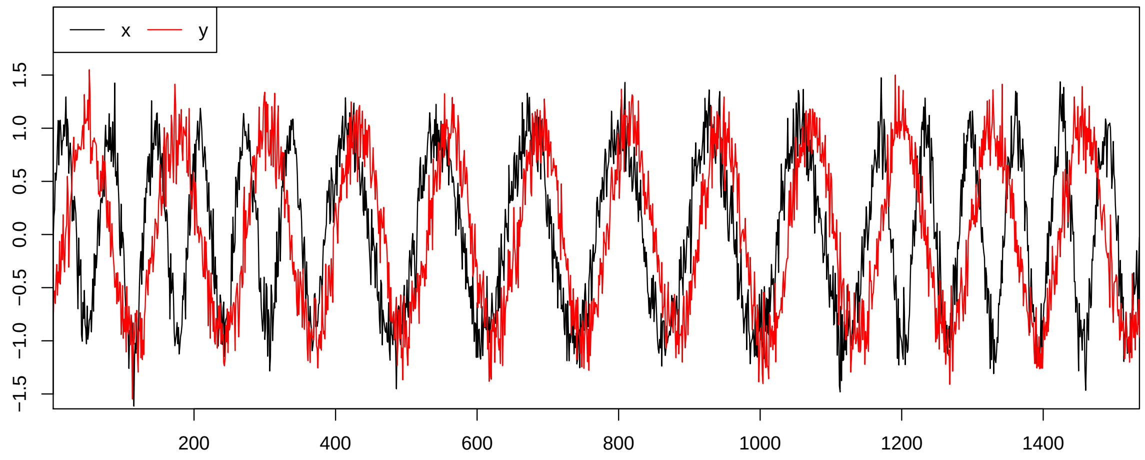


Figure 28: Series with joint period 128 at some time range, but different phase

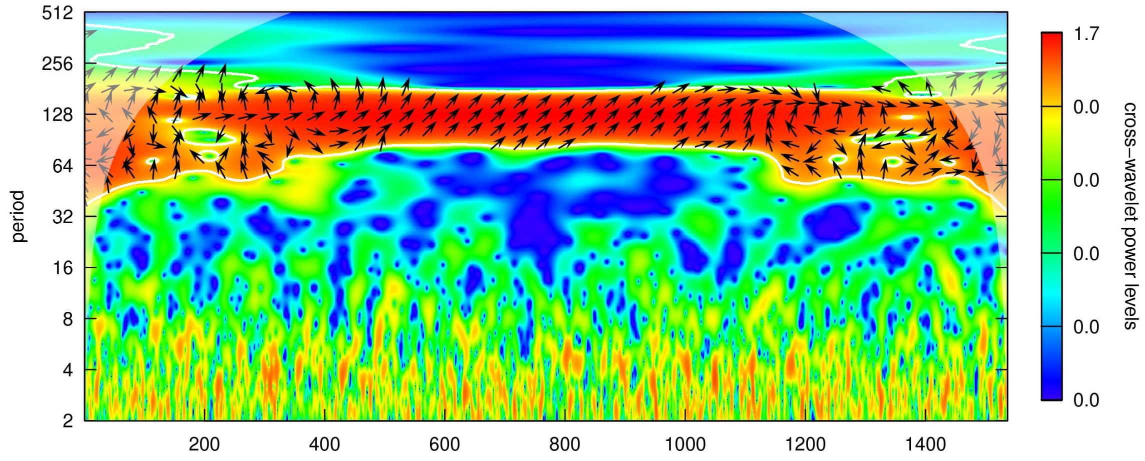


Figure 29: Cross-Wavelet power spectrum of the series

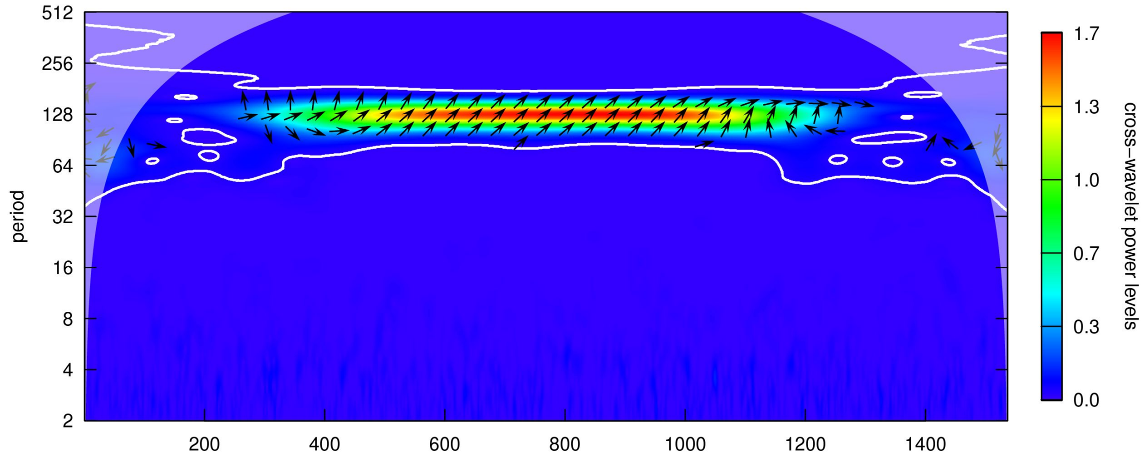


Figure 30: Cross-Wavelet power spectrum of the series with interval color key and restricted arrow area

3.3 Example 3: variable periods, coherence

The advantage of wavelet coherence over wavelet power is that it shows statistical significance only in areas where the series involved actually share significant periods. Its disadvantage may be that it needs careful fine-tuning in computation and plotting. The default setting of `wc.image` is `which.image = "wp"`, where `p` stands for power. Setting `which.image = "wc"` will plot the wavelet coherence:

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

See Figure 31, where `my.wc` is as computed in Section 3.2. The parameters controlling smoothing in the computation of `my.wc` 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. The default smoothing settings have been used in the call of `analyze.coherency`: Bartlett windows in both directions (by choosing `window.type.t = 1`, `window.type.s = 1`), with sizes as follows:

- `window.size.t = 5` in time units of `1/dt`= 1, resulting in size 5 in time direction,
- `window.size.s = 1/4` in units of `1/dj`= 100, resulting in size 25 in scale (or period) direction.

Now let's briefly explore the influence of different smoothing settings on the plot of wavelet coherence. Wavelet power is computed and smoothed simultaneously; changing the settings of window type and/or size therefore implies that `analyze.coherency` has to be run again before calling `wc.image`. (Setting `make.pval = FALSE` in `analyze.coherency` saves computation time, but no arrows indicating phase differences will be plotted.)

Figure 32 results again from smoothing with Bartlett windows, but `window.size.s = 1` now defines a window of length 101 (an even number will be increased by 1) and produces more blurring for low periods (high frequencies):

```
my.wc2 <- analyze.coherency(my.data, my.pair = c("x","y"),
                             loess.span = 0, dt = 1, dj = 1/100,
                             window.type.t = 1, window.type.s = 1,
                             window.size.t = 5, window.size.s = 1,
                             make.pval = TRUE, n.sim = 10)
```

For Figure 33, Boxcar smoothing was used, with the same window sizes as in `my.wc2`:

```
my.wc3 <- analyze.coherency(my.data, my.pair = c("x","y"),
                            loess.span = 0, dt = 1, dj = 1/100,
                            window.type.t = 3, window.type.s = 3,
                            window.size.t = 5, window.size.s = 1,
                            make.pval = TRUE, n.sim = 10)
```

This leads to less granularity in high frequency areas.

In all three images on the facing page, 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 — different smoothing settings will also affect the distribution of coherence levels, and hence color quantiles. Due to repeated simulations (carried out as `make.pval = TRUE`), the white lines indicating significance are slightly different, and so will be the selection of arrows to be plotted. If no smoothing is applied in time direction (`window.type.t = 0`), a wavelet coherence image 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"`.

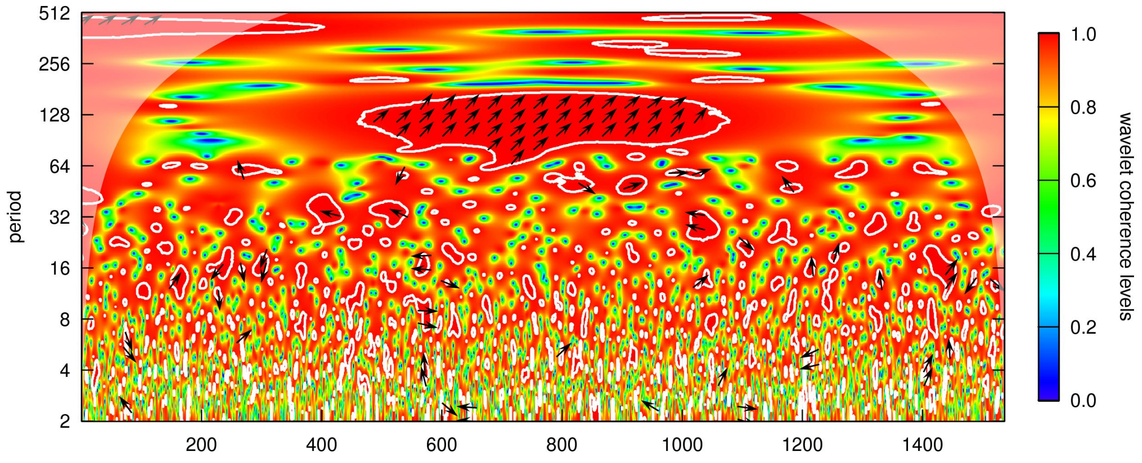


Figure 31: Wavelet coherence, window type: Bartlett, default settings

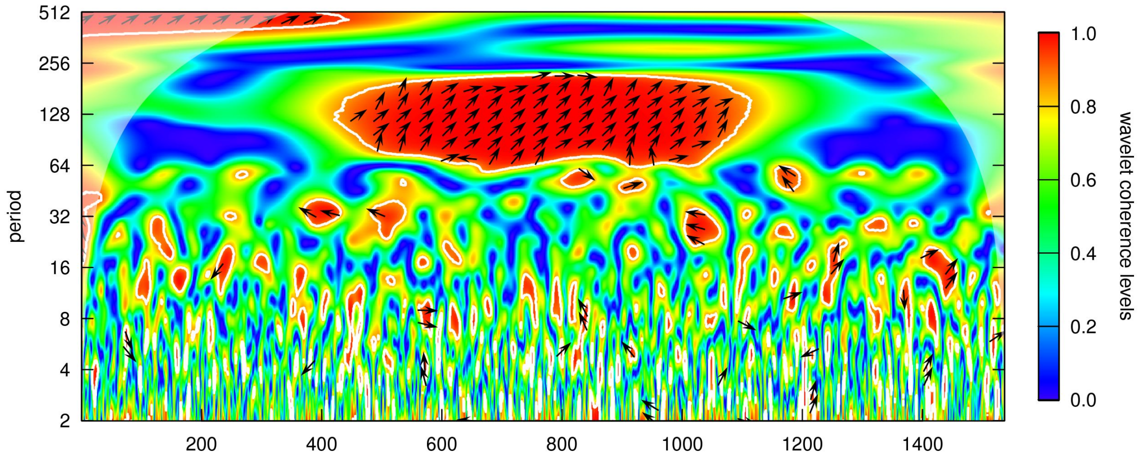


Figure 32: Wavelet coherence, window type: Bartlett, more smoothing along period axis

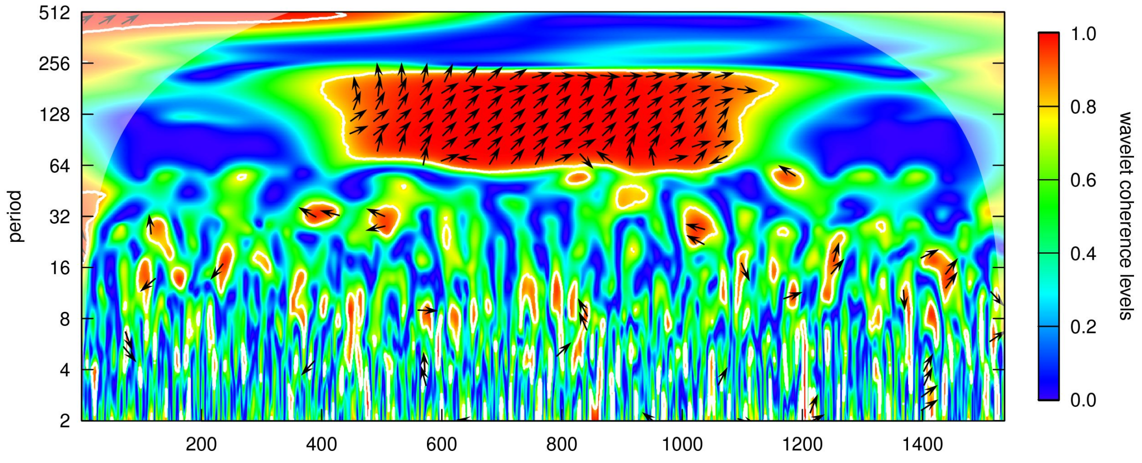


Figure 33: Wavelet coherence, window type: Boxcar

3.4 Example 4: variable periods, phase differences

Continuing the example from Sections 3.2 and 3.3, 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 = TRUE,
              which.sig = "wc",
              siglvl = 0.05,
              phaselim = c(-pi,+pi), ## default if legend.horiz = FALSE
              legend.coords = "topright", legend.horiz = FALSE,
              main = "", sub = "", timelab = "")
```

With the default settings `only.sig = TRUE` 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 34. The plot in Figure 35 has phase axis labels expressed in time units, which may be more intuitive in certain applications; it is produced by:

```
at.ticks <- seq(from = -pi, to = pi, by = pi/4)
label.ticks <- (at.ticks/pi)*(128/2)
wc.sel.phases(my.wc, sel.period = 128,
              only.sig = TRUE,
              which.sig = "wc",
              siglvl = 0.05,
              phaselim = c(-pi,+pi),
              phaselab = "phases (time units)",
              spec.phase.axis = list(at = at.ticks, labels = label.ticks),
              legend.coords = "topright", legend.horiz = FALSE,
              main = "", sub = "", timelab = "")
```

The “global” image of phase differences in Figure 36 is produced by:

```
wc.phasediff.image(my.wc, which.contour = "wc", use.sAngle = TRUE,
                   n.levels = 250, siglvl = 0.1,
                   legend.params = list(lab = "phase difference levels",
                                        lab.line = 3),
                   timelab = "")
```

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

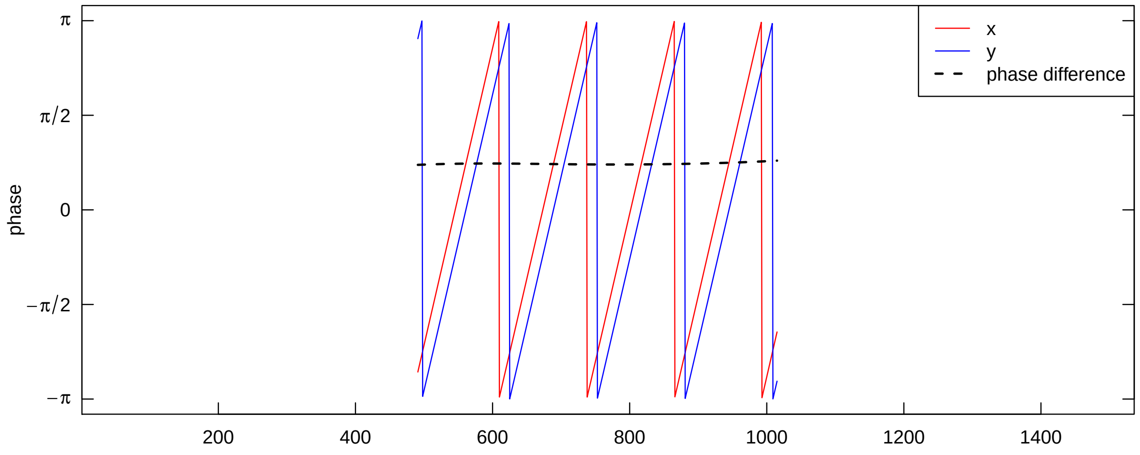


Figure 34: Phases and phase differences at period 128, default phase axis

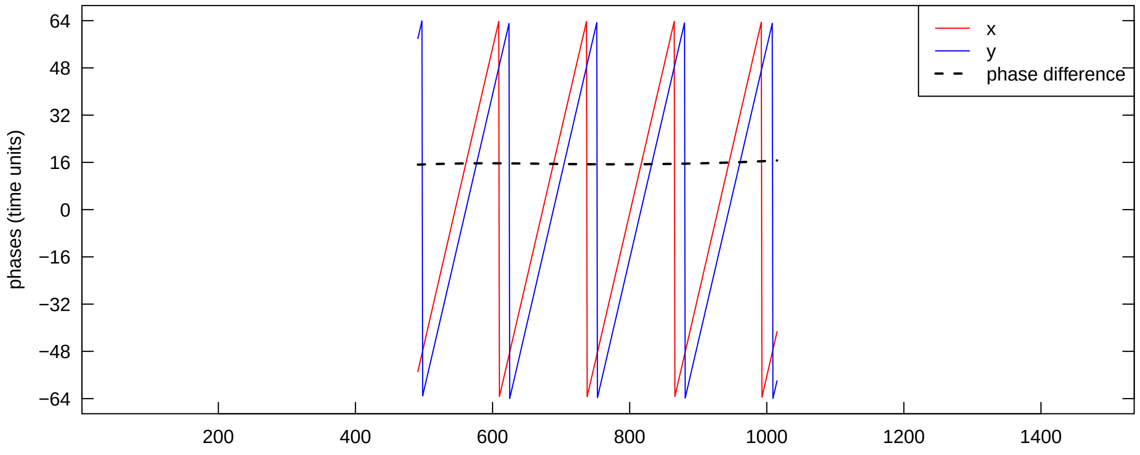


Figure 35: Phases and phase differences at period 128, custom phase axis

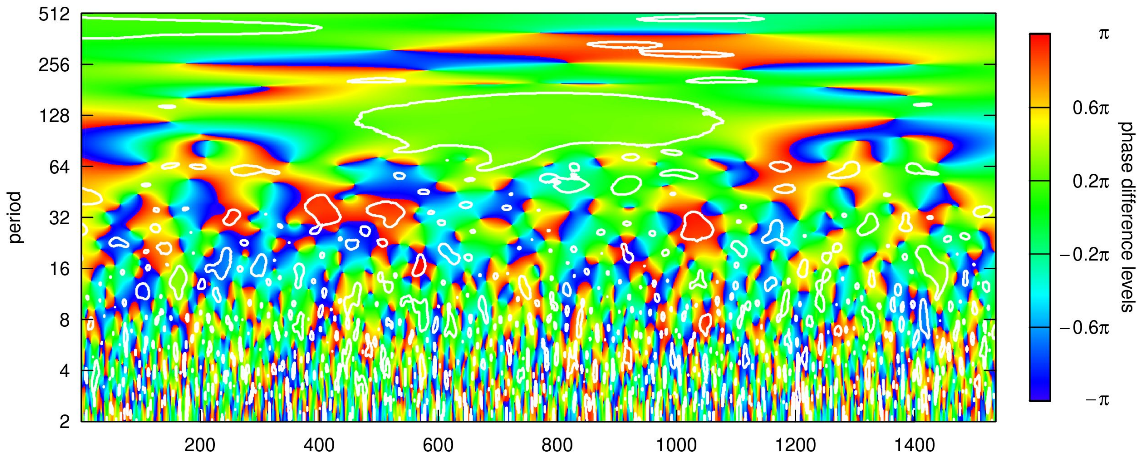


Figure 36: Phase difference image

3.5 More about phase differences

The timely structure of the following example is similar to the 96-day hourly example in Section 3.1:

```
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 share period 4 (days) in the middle, with the period of x growing linearly (see Figure 37). The cross-wavelet power spectrum is computed and plotted as follows:

```
my.data <- data.frame(x = x, y = y)

my.wc <- analyze.coherency(my.data, my.pair = c("x","y"),
                           loess.span = 0,
                           lowerPeriod = 0.5, upperPeriod = 32,
                           dt = 1/24, dj = 1/100,
                           make.pval = TRUE, n.sim = 10)

wc.image(my.wc, n.levels = 250,
         siglvl.contour = 0.1, siglvl.arrow = 0.05,  ## default values
         legend.params = list(lab = "cross-wavelet power levels"),
         timelab = "",
         periodlab = "period (days)")
```

See Figure 38. 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 Sections 3.2 and 3.3), and the wavelet coherence image would be helpful again (no plot is shown here).

The phase and phase difference image, shown in Figure 39, can be obtained via:

```
wc.sel.phases(my.wc, sel.period = 4, only.sig = TRUE, siglvl = 0.05,
              which.sig = "wt",
              legend.coords = "topleft",
              phaselim = c(-pi, +pi + 1),
              main = "", sub = "", timelab = "")
```

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 when a constant period is selected. The concave part of the dashed line in Figure 39 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.

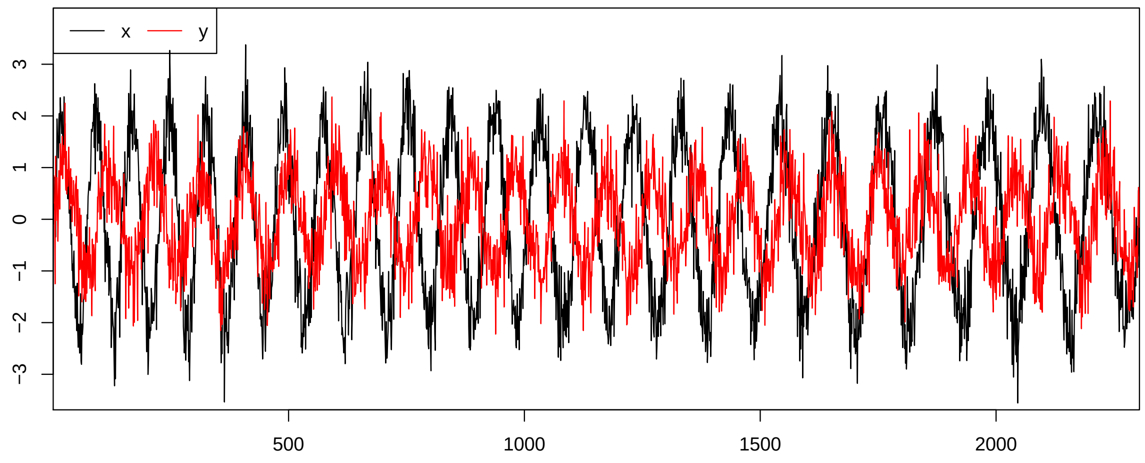


Figure 37: Two series, overlapping periods

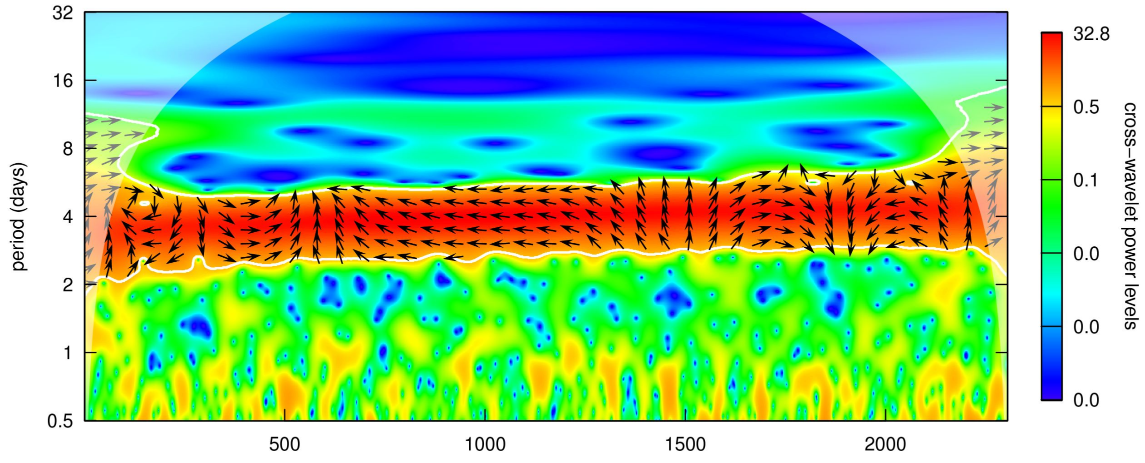


Figure 38: Cross-wavelet power spectrum: significant periods are increasing

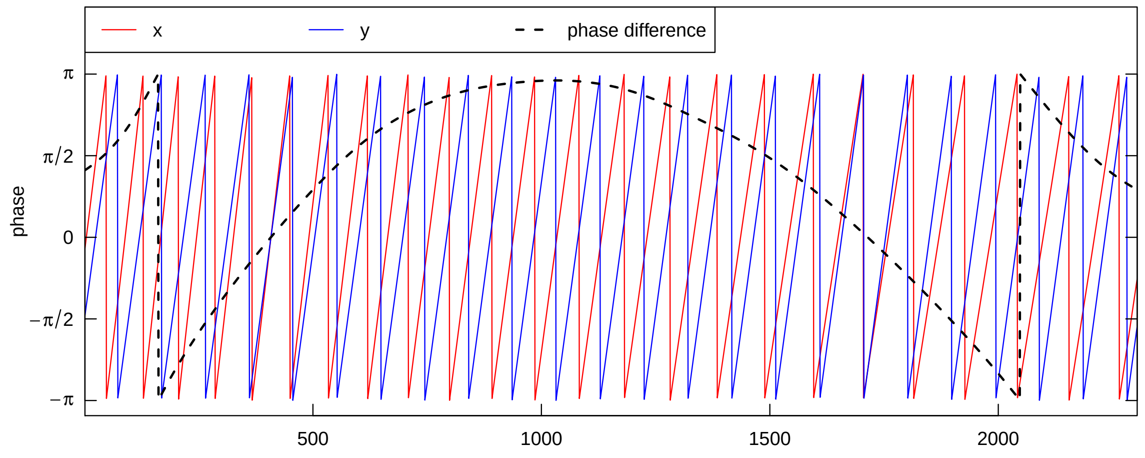


Figure 39: Phases and phase differences: asymmetry due to increasing period

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; see Figure 40. 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, 3)
                 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
```

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",
                        dt = 1/(12*24))
```

The image 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 41: There is a 1-day period, interrupted by weekends, and ridges indicating 3-to-4-day and 7-day periods.

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 and measure periodicity in terms of active days:

```
my.data.a <- FXtrade.transactions[FXtrade.transactions$active == TRUE, ]
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 = 1/50,         # resolution along period axis
            lowerPeriod = 1/8, # lowest period of interest: 3 hours
            make.pval = TRUE,  # draws white lines indicating significance
            n.sim = 10)        # higher number will give smoother white lines
```

As we have excluded weekends, it would be a poor idea to set `show.date = TRUE` and `date.format = "%F %T"` when calling `wt.image` — the time axis would falsely include weekends again! (See Section 2.8.) It is more appropriate to define time axis labels in detail:

```
at <- seq(1, nrow(my.data.a), by = 12*24) # every active day at 00:00:00
labels <- strftime(as.POSIXct(my.data.a$date[at],
                   format="%F %T", tz = "GMT"), format ="%b %d")
wt.image(my.w.a, n.levels = 250, periodlab = "period (active days)",
         legend.params = list(lab = "wavelet power levels"),
         spec.time.axis = list(at = at, labels = labels))
```

The resulting wavelet power spectrum is shown in Figure 42.

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

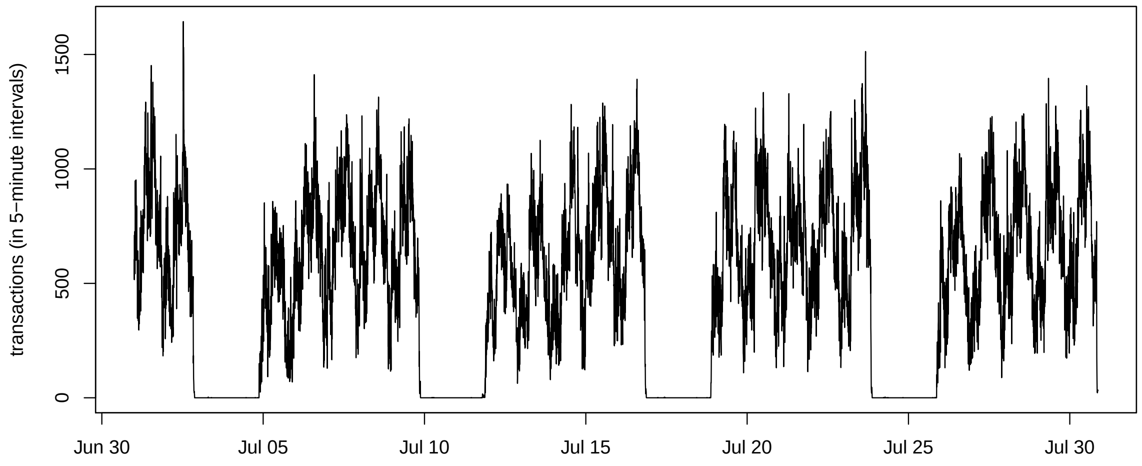


Figure 40: Number of transactions in five-minute intervals

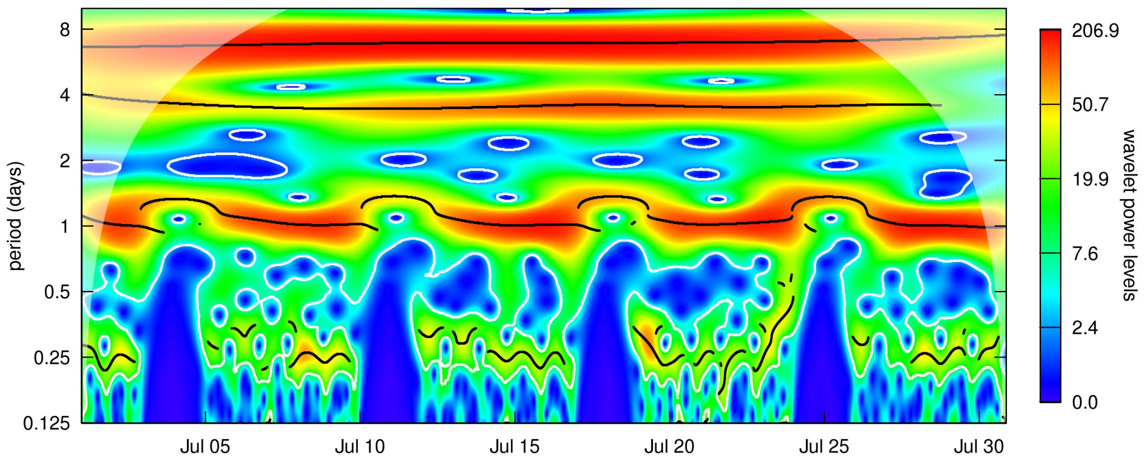


Figure 41: Transactions (original series), the wavelet power spectrum

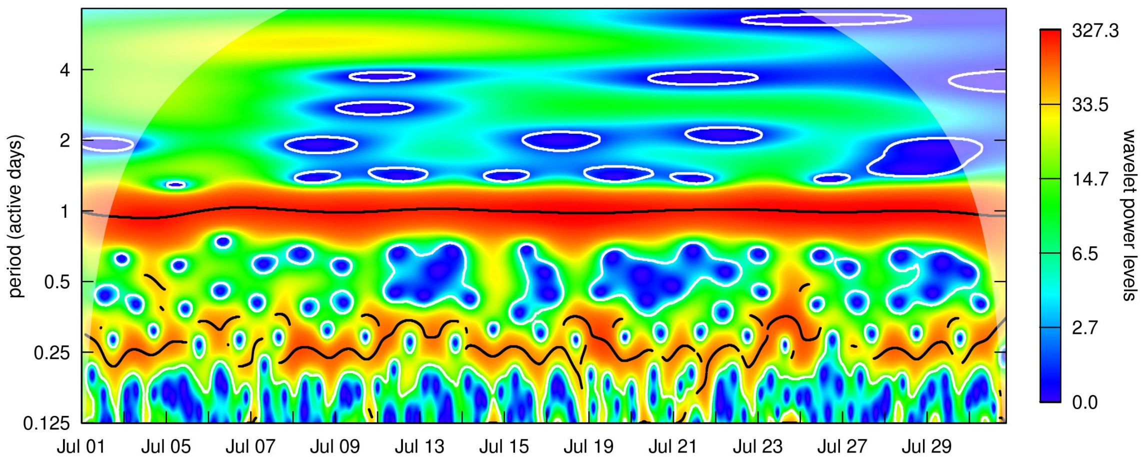


Figure 42: Transactions (active time only), the wavelet power spectrum

4.3 Reconstruction

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

```
my.rec.a <- reconstruct(my.w.a, plot.waves = FALSE,
                        spec.time.axis = list(at = at, labels = labels))
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 = TRUE` — many periods are significant. Of course, a further option can be chosen in `reconstruct`, such as `only.ridge = TRUE`, 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 43) 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 == TRUE] <- 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 43. 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 44; 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 45 (again scaled down to transactions per minute) — with observations from only four weeks at hand, the air is still too thin for the law of large numbers to tidy up efficiently.

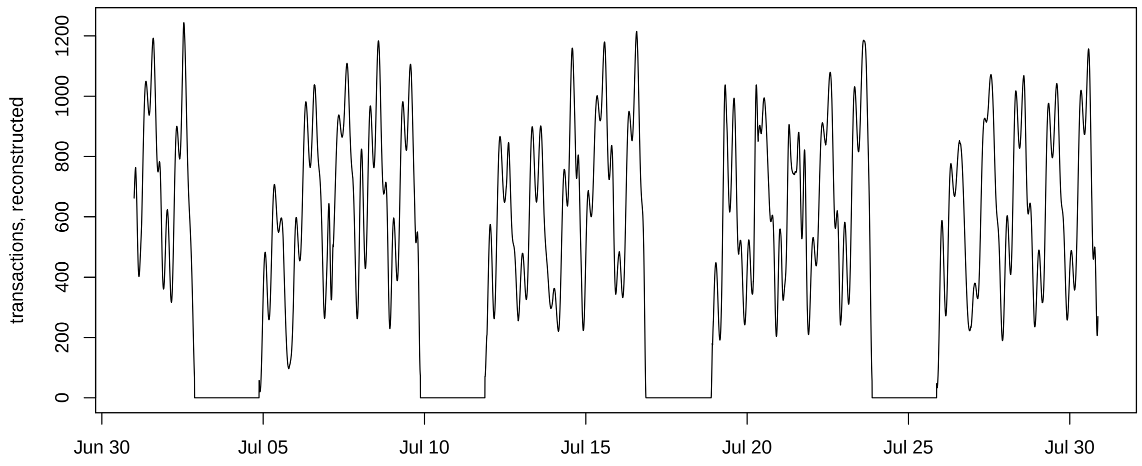


Figure 43: Number of transactions, reconstructed

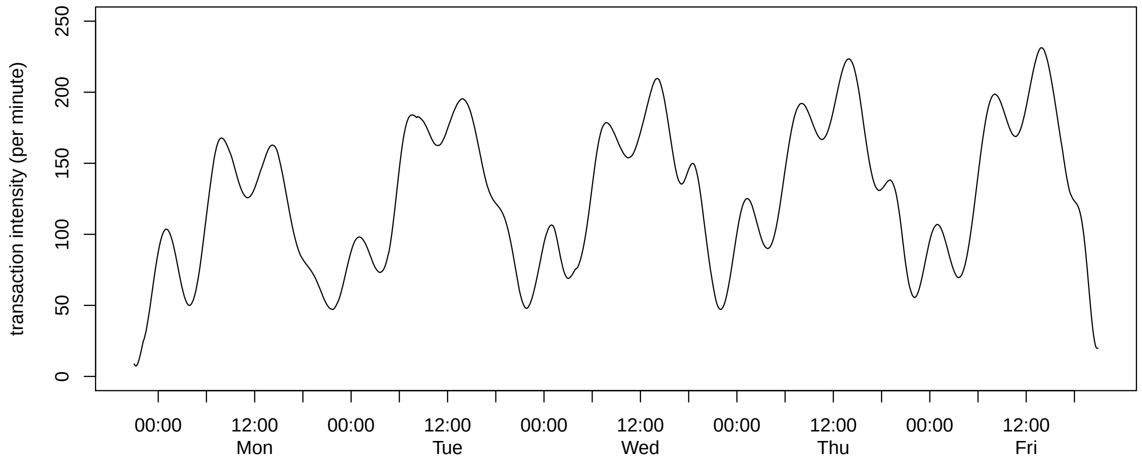


Figure 44: Transaction intensity estimate

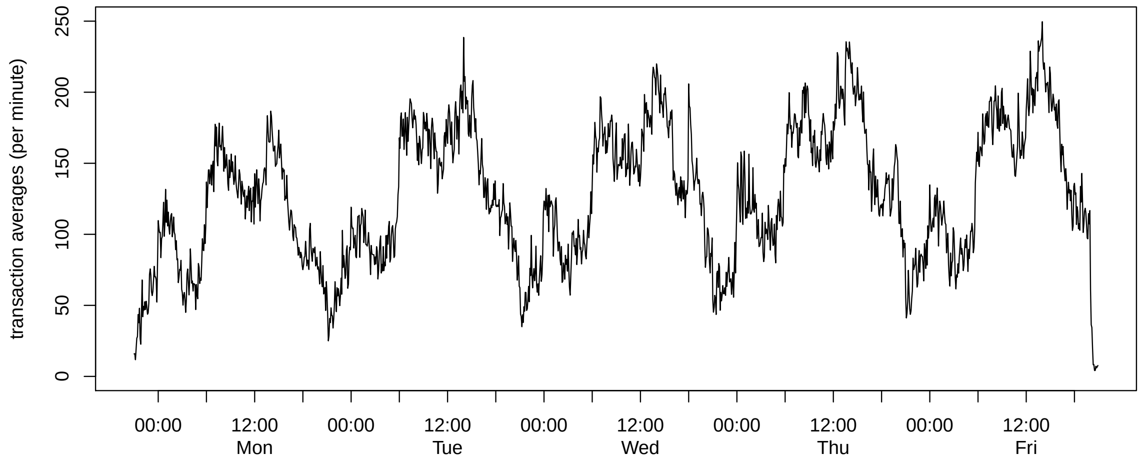


Figure 45: Transactions: weekly averages

5 Example: Marriages in Turkey

5.1 The series of marriages

Data set marriages.Turkey contains the monthly 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 46.

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 = TRUE, n.sim = 10)
wt.image(my.w, n.levels = 250,
         legend.params = list(lab = "wavelet power levels"),
         periodlab = "period (months)", show.date = TRUE, 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 limit 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 = TRUE, n.sim = 10)
wt.image(my.w, n.levels = 250,
         legend.params = list(lab = "wavelet power levels"),
         periodlab = "period (months)", show.date = TRUE, timelab = "",
         spec.period.axis = list(at = c(8, 10, 12, 14, 16)),
         graphics.reset = FALSE)
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 48. We have to leave the plot open for further input in order to plot the horizontal line at 12 by setting `graphics.reset = FALSE`. Also observe that coordinates on the period axis are given in terms of $\log_2$ values.

[9]The series was retrieved 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 December 2013). See http://www.tuik.gov.tr/VeriTabanlari.do?vt_id=21&ust_id=109 (accessed Oct 1, 2014).

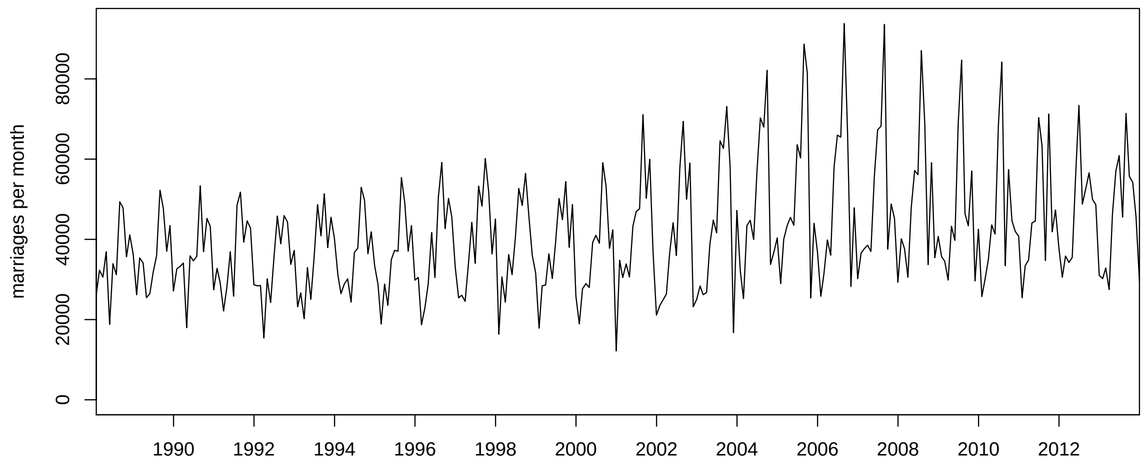


Figure 46: The series of monthly marriages ("Sunday-adjusted")

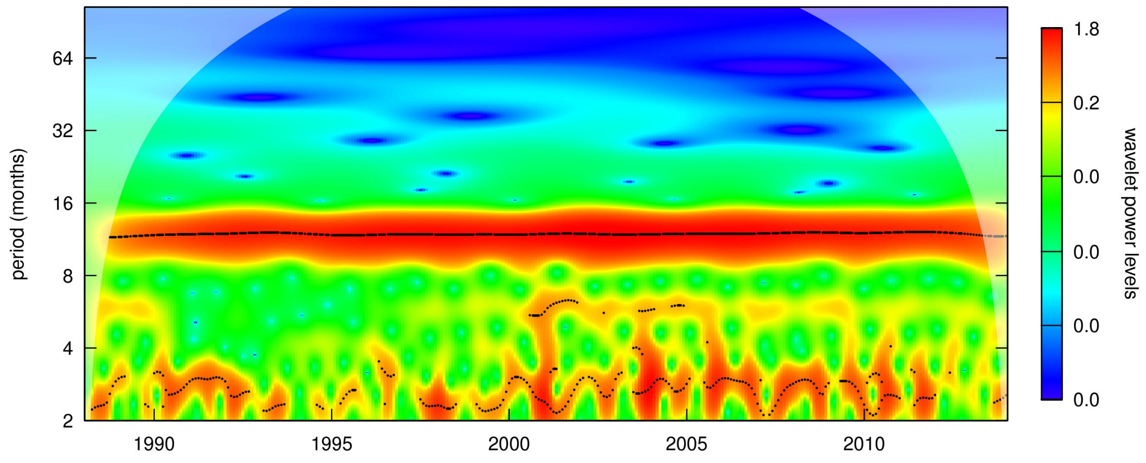


Figure 47: Marriages, the wavelet power spectrum

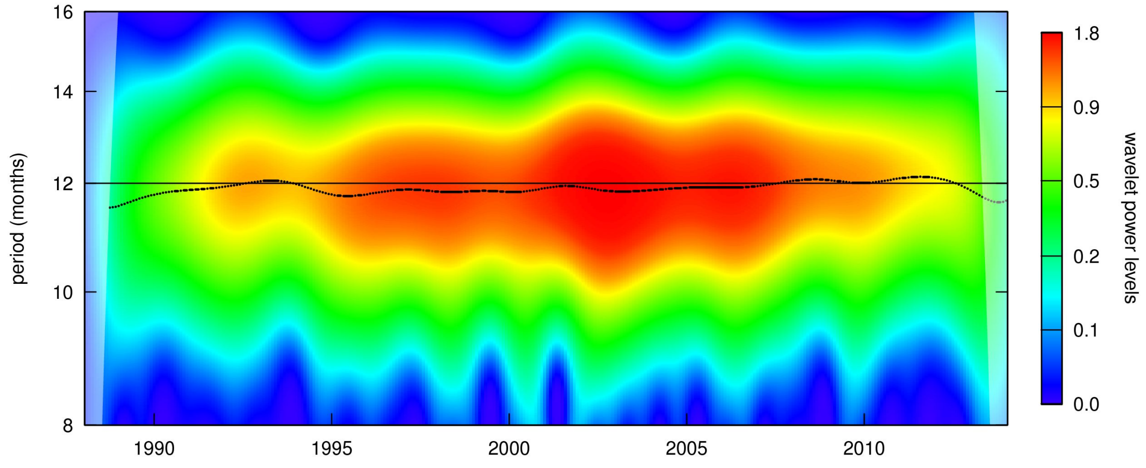


Figure 48: Marriages, the wavelet power spectrum, periods 8 – 16

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 <- colSums(ridge)
ridge[ridge == 0] <- NA
```

This ridge constitutes the black line in Figure 49. It deviates slightly from the hypothesized 12-month period. 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 49: The timing of marriages in Turkey may be influenced by Islamic festivities. 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 earlier 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 48 and 49? 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 50. The wavelet power spectrum plot in Figure 51 results from:

```
my.w.s <- analyze.wavelet(my.data.s, "y",
                          loess.span = 0, lowerPeriod = 8, upperPeriod = 16,
                          dt = 1, dj = 1/1000, make.pval = FALSE)
wt.image(my.w.s, n.levels = 250,
         legend.params = list(lab = "wavelet power levels"),
         periodlab = "period (months)",
         spec.period.axis = list(at = c(8, 10, 12, 14, 16)),
         show.date = TRUE, timelab = "", graphics.reset = FALSE)
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 49. 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.

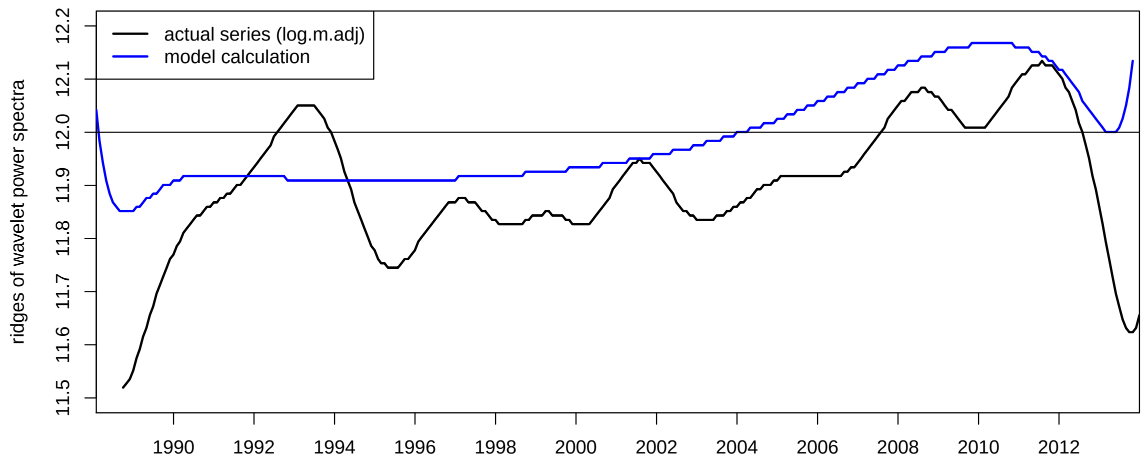


Figure 49: Ridge of power spectra

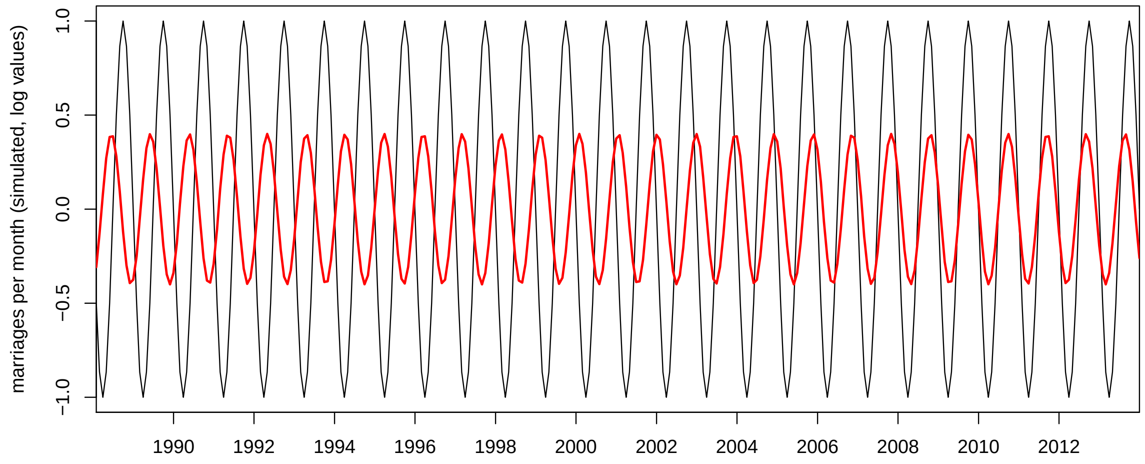


Figure 50: Simulation of marriages, with superposed lunar calendar (red)

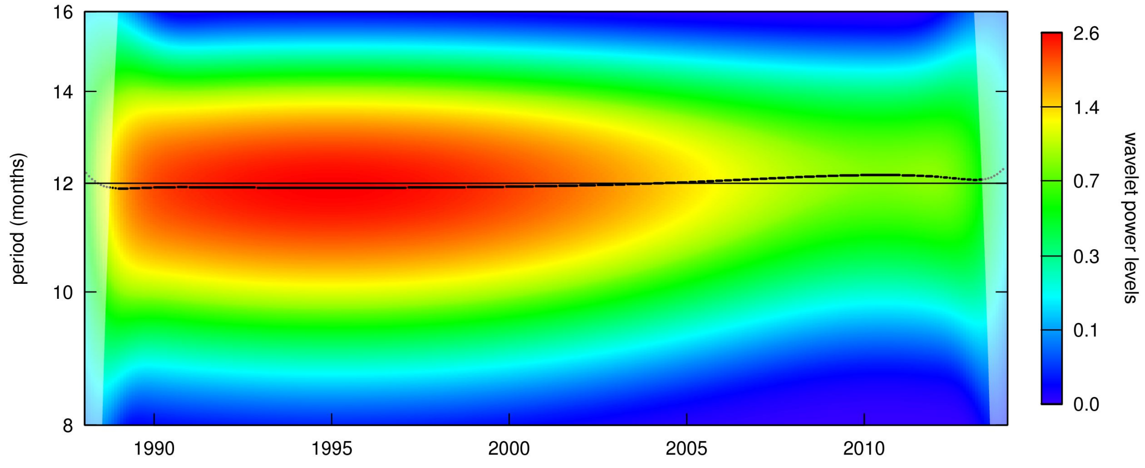


Figure 51: Power spectrum of simulated series

6 Example: Weather and radiation

6.1 Weather and radiation data from Mannheim-Rheinau

Data set weather.radiation.Mannheim contains ten years (from January 2005 through December 2014; 3652 values) of daily average temperature, humidity, and radiation readings from Mannheim-Rheinau (Germany):[10]

```
> data(weather.radiation.Mannheim)
> head(weather.radiation.Mannheim, 3)
        date temperature humidity     radiation
1 2005-01-01         6.5       90         0.096
2 2005-01-02         6.6       71         0.095
3 2005-01-03         4.8       75         0.094
```

Here, temperature is measured in °C, humidity is given as percentage, and radiation means the ambient gamma dose rate (ODL), measured in μSv/h (= microsievert per hour). [11] The three series are plotted in Figure 52.

6.2 Seasonality in weather and radiation data

It can be seen with the naked eye that the three series are periodic, periodicity being strongest in the case of temperature and weakest in the case of radiation. For a systematic analysis (here, temperature):

```
my.w <- analyze.wavelet(weather.radiation.Mannheim, "temperature",
                        loess.span = 0,
                        dt = 1, dj = 1/50,
                        lowerPeriod = 32, upperPeriod = 1024,
                        make.pval = TRUE, n.sim = 10)
```

The wavelet power spectrum can be plotted as follows:

```
wt.image(my.w, color.key = "interval", n.levels = 250,
         legend.params = list(lab = "wavelet power levels"),
         periodlab = "period (days)",
         show.date = TRUE, date.format = "%F", timelab = "")
```

This code draws labels on the time axis, as shown in the case of radiation in Figure 53. For the other two plots, the time axis is unlabeled:

```
wt.image(my.w, color.key = "interval", n.levels = 250,
         legend.params = list(lab = "wavelet power levels"),
         periodlab = "period (days)",
         label.time.axis = FALSE)
```

The plots in Figure 53 are not quite satisfactory: The three series' wavelet transforms have different maximum powers (as might have been guessed from Figure 52). The plots in Figure 53 thus cannot be directly compared. It will be better to render the spectra such that the correspondence between color and power level is the same across the three plots. To obtain the maximum power:

```
max.power <- max(my.w$Power)
```

(where `my.w` refers to series `temperature`). The maximum power is approximately 1.531654.

[10]Radiation data retrieved in 2015 from ODL-INFO (URL: http://odlinfo.bfs.de/download.php) of the German Federal Office for Radiation Protection (in German: Bundesamt für Strahlenschutz, BfS); weather data from the German Meteorological Office (in German: Deutscher Wetterdienst, DWD; URL: ftp://ftp-cdc.dwd.de/pub/CDC/observations_germany/climate/daily/kl/historical/). — We owe this example to our former student Nadiya Appelhans.

[11]The ambient gamma dose rate is an equivalent dose representing the stochastic health effects of low levels of ionizing radiation on the human body. According to BfS, radioactivity is to be found everywhere in the environment. It may be of natural or artificial origin. — We drew Mannheim randomly from a set of places for which weather and radiation data were available; there is no conspicuous radiation in Mannheim.

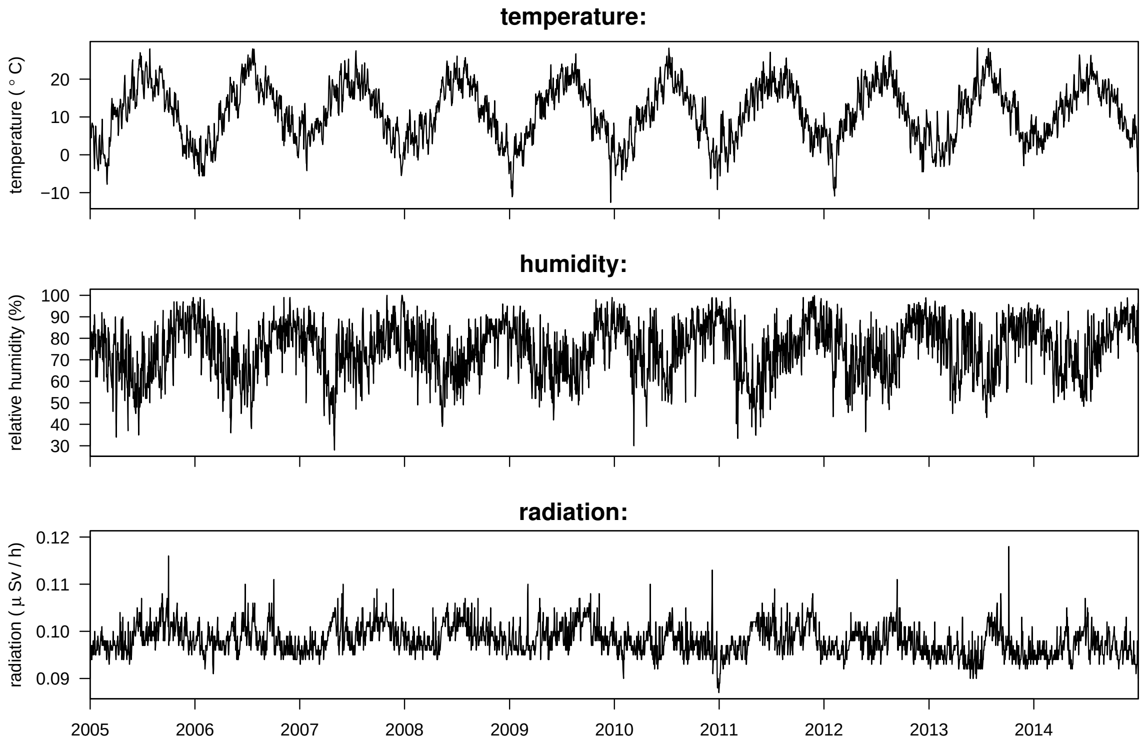


Figure 52: Temperature, humidity, radiation: time series

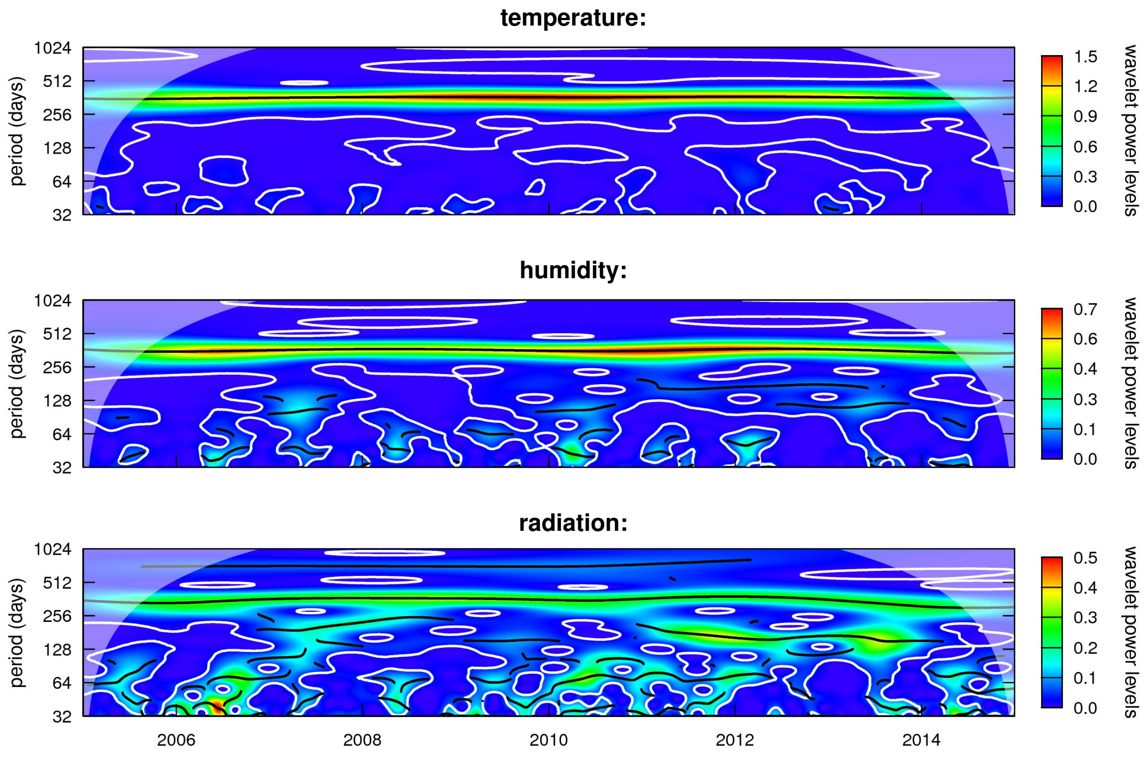


Figure 53: Temperature, humidity, radiation: wavelet power spectra; Version 1

The code for plotting the spectra with power ranging from zero to `max.power` is:

```
wt.image(my.w, color.key = "interval", n.levels = 250,
         legend.params = list(lab = "wavelet power levels"),
         periodlab = "period (days)",
         maximum.level = 1.001 * max.power,
         show.date = TRUE, date.format = "%F", timelab = "")
```

For an unlabeled time axis, substitute `show.date = FALSE`. Observe that `max.power` is made slightly larger by multiplication with 1.001 because the reported `max.power` value may not match R's internal accuracy; omitting this step may lead to an error. — The result is shown in Figure 54.

It makes sense to alter some details of these plots:

1. Make the gradation more explicit, especially for low powers, by plotting the square root of power (WaveletComp allows for plotting power, raised to any positive exponent, rather than power itself.[12])

2. Plot time axes similar (with respect to ticks and labels) to those in Figure 52.

3. On the period axis, add a label for period 365 to emphasize the one-year period; draw horizontal lines.

Here is the WaveletComp plotting code to accomplish this:

```
wt.image(my.w, color.key = "interval", n.levels = 250,
         legend.params = list(lab = "wavelet power levels", label.digits = 2),
         periodlab = "periods (days)", maximum.level = 1.25,
         # Concerning item 1 above --- plot the square root of power:
         exponent = 0.5,
         # Concerning item 2 above --- time axis:
         show.date = TRUE, date.format = "%F", timelab = "",
         spec.time.axis = list(at = c(paste(2005:2014, "-01-01", sep = "")),
                               labels = c(2005:2014)),
         timetcl = -0.5,  # draws outward ticks
         # Concerning item 3 above --- period axis:
         spec.period.axis =
                list(at = c(32, 64, 128, 365, 1024)),
         periodtck = 1, periodtcl = NULL  # draws horizontal lines
         )
```

Here, the choice `maximum.level = 1.25` was found appropriate since it matches the square root (`exponent = 0.5`) of maximum power: $\sqrt{1.531654} \approx 1.24$. In order to avoid non-equidistant labels on the color bar, we also set `label.digits = 2` (overturning the default value of 1). If no labels on the time axis are desired, simply set `labels = rep("", 10)` in `spec.time.axis`. The result is shown in Figure 55.

It turns out that environmental radiation is a more complex phenomenon than temperature or humidity. It does have a significant one-year period, albeit with a low power, but many other periods are significant as well.

[12]This is a common procedure in wavelet analysis, see for example Percival and Walden [12].

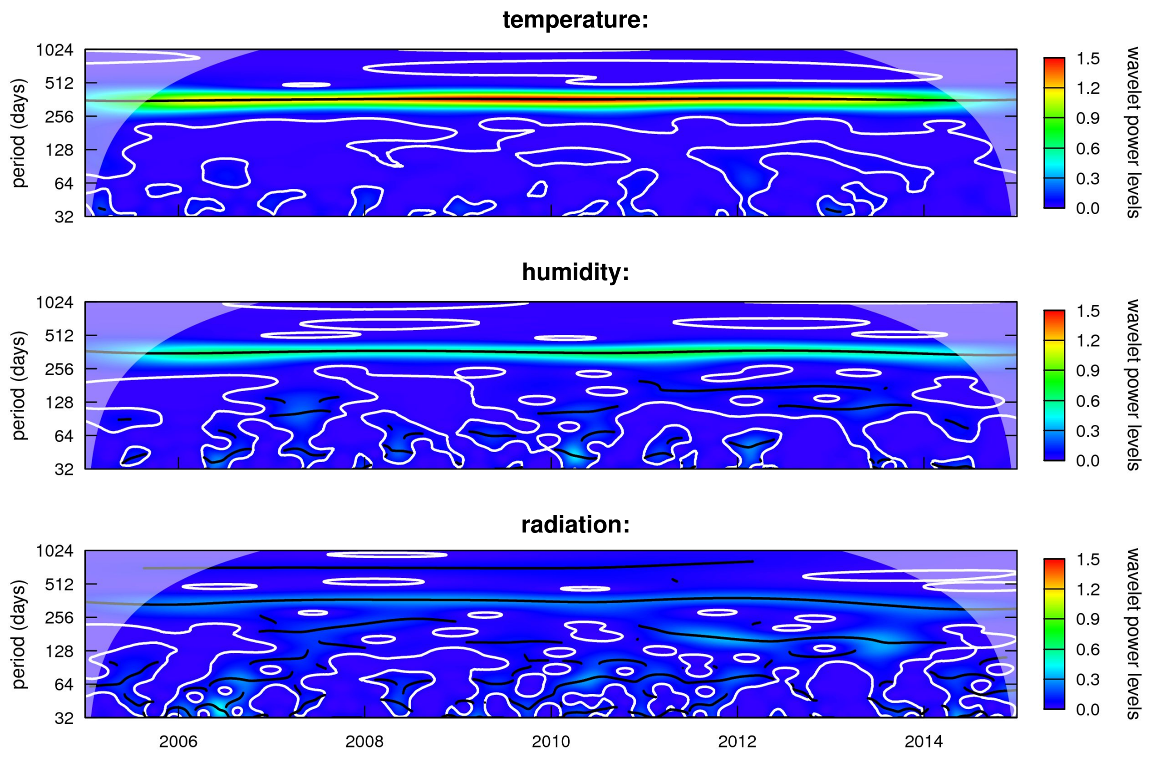


Figure 54: Temperature, humidity, radiation: wavelet power spectra; Version 2

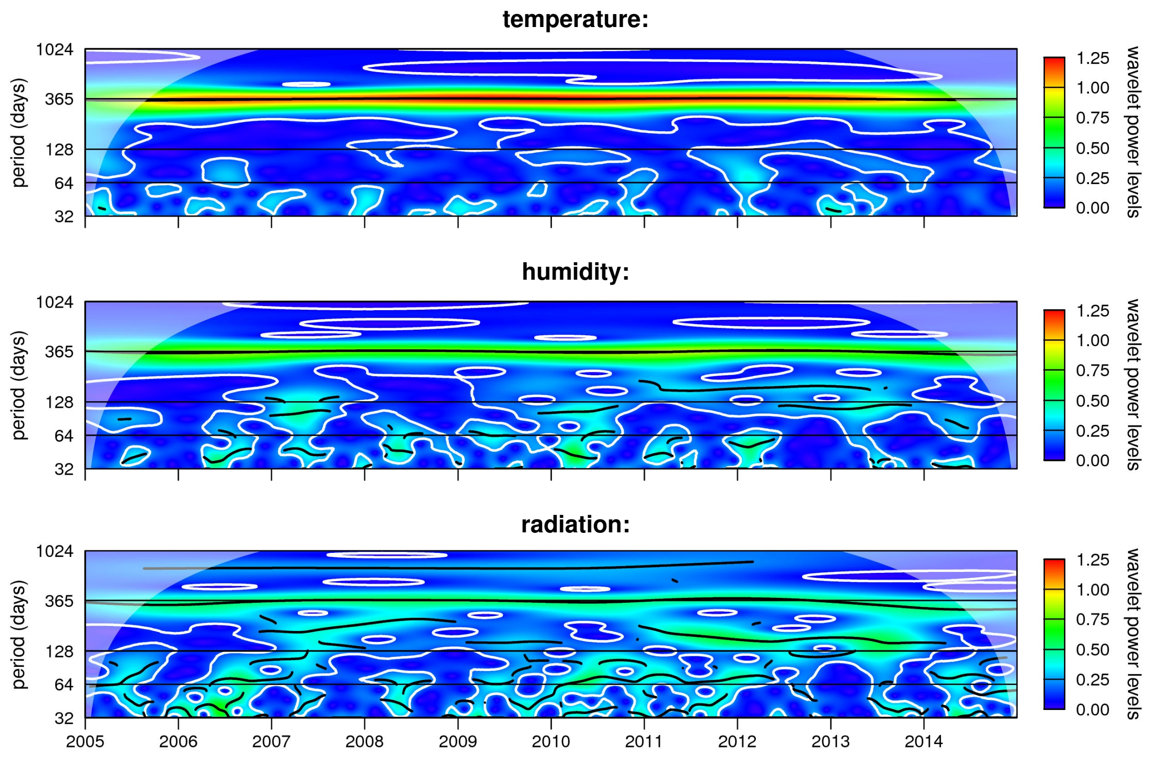


Figure 55: Temperature, humidity, radiation: wavelet power spectra; Version 3

6.3 Cross-wavelet transformation of weather and radiation data

Each pair from the series temperature, humidity and radiation can be analyzed jointly with respect to its wavelet coherency; this will reveal which series is leading at given time and period (in case of joint significance):

```
my.wc <- analyze.coherency(weather.radiation.Mannheim,
                           my.pair = c("temperature", "humidity"),
                           loess.span = 0,
                           dt = 1, dj = 1/50,
                           lowerPeriod = 32, upperPeriod = 1024,
                           make.pval = TRUE, n.sim = 10)
max.power <- max(my.wc$Power.xy)  # for plotting
save(my.wc, file = "cross_wavelet_transform_temperature_over_humidity")
```

The last line above is a reminder that object `my.wc` can be stored in a file; it may be practical to detach wavelet transformation (which may take some computation, or rather: simulation time) from plotting. The code for plotting the cross-wavelet power spectrum is very similar to the one in Section 6.2:

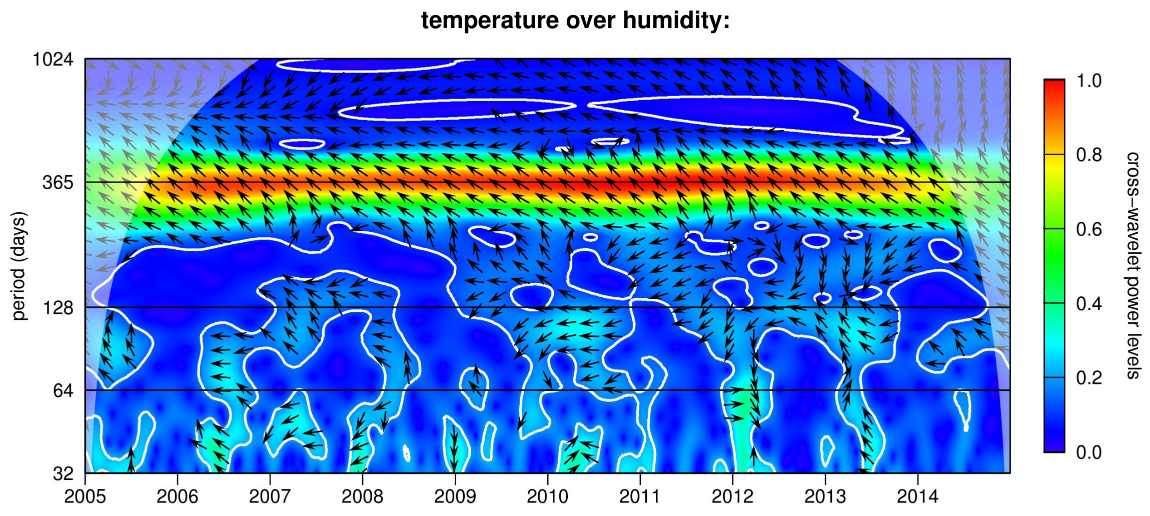
```
load("cross_wavelet_transform_temperature_over_humidity")
exponent <- 0.5
wc.image(my.wc, n.levels = 250,
         legend.params = list(lab = "cross-wavelet power levels"),
         color.key = "interval",
         maximum.level = (1.001*max.power)^exponent, exponent = exponent,
         # time axis:
         label.time.axis = TRUE, show.date = TRUE,
         spec.time.axis = list(at = paste(2005:2014, "-01-01", sep = ""),
                               labels = 2005:2014),
         timetcl = -0.5, # outward ticks
         # period axis:
         periodlab = "period (days)",
         spec.period.axis = list(at = c(32, 64, 128, 365, 1024)),
         periodtck = 1, periodtcl = NULL)
```

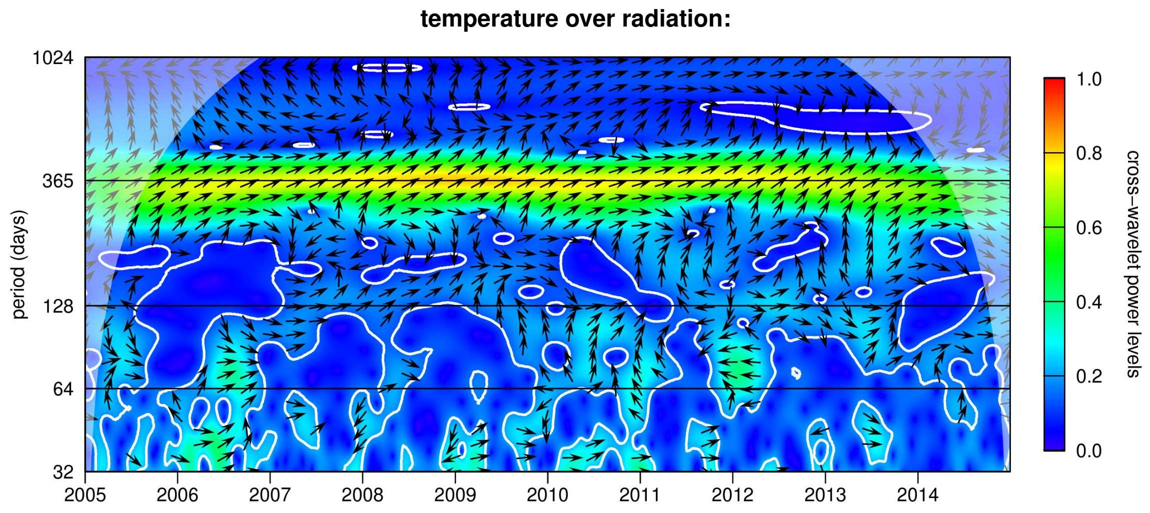
The result is shown in Figure 56. Using function `wt.image`, the object `my.wc`, as defined above, can also serve to plot the power spectrum of a univariate series: for temperature, the command is `wt.image(my.wc, my.series = 1,...)`, and for radiation, substitute `my.series = 2`; this may save run time.

Focusing on period 365 days, the arrows in the cross-wavelet power spectra in Figure 56 can then be interpreted with the help of Figure 2 (and remembering that a full circle corresponds to one year):

- Temperature and humidity are out of phase, with humidity leading by roughly 1/8 year: Humidity reaches its trough about 6 to 7 weeks before temperature reaches its maximum. (An example will be given in Section 7 showing how to extract angle information from `my.wc`.)
- Temperature and radiation are in phase; temperature is leading by less than 6 weeks.
- Humidity and radiation are again out of phase; humidity is leading by about 3 months.

The next section shows, among other things, how the series of phase differences can be extracted from an object of class “analyze.coherency” (such as `my.wc` above).





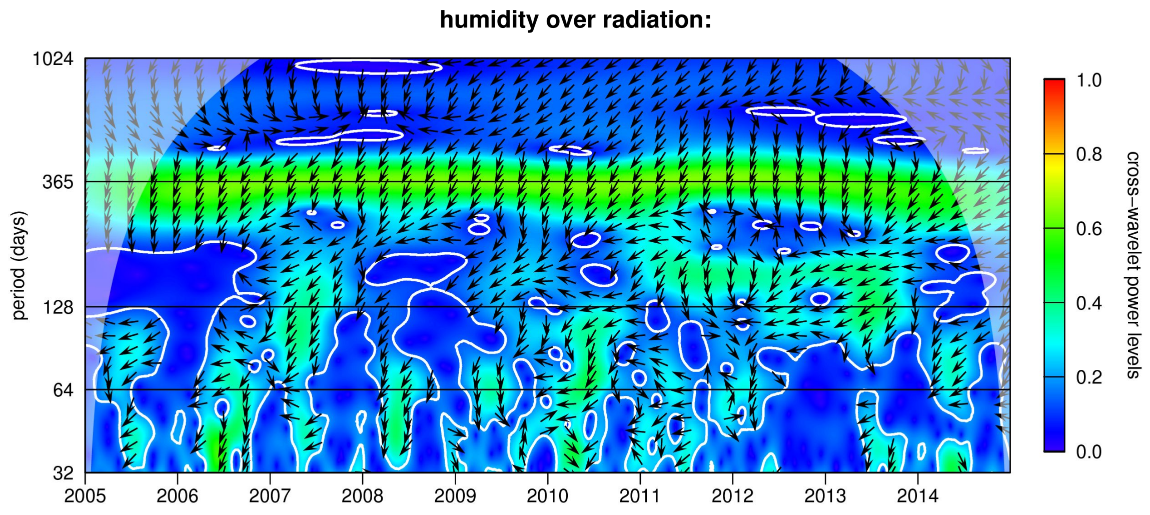


Figure 56: Cross-wavelet power spectra, weather and radiation

7 Example: The 2016 US presidential election and media on Instagram

7.1 Media uploads to Instagram the week before the 2016 US presidential election

This example is concerned with the number of candidate-related hourly media uploads to Instagram in the week before the 2016 US presidential election. Two prominent candidates were Hillary Clinton (Democratic Party) and Donald Trump (Republican Party). The election took place on Tuesday, 2016-11-08. While Hillary Clinton was widely expected to win, Donald Trump finally became the 45th president of the United States by winning the Electoral College.

Media on Instagram are annotated with hashtags which can be used to determine whether a candidate-related upload is neutral or in support of a candidate (for example, #makeamericagreatagain for Trump, #hillary2016 for Clinton) or opposing a candidate (for example, #dumptrump for Trump, #neverhillary for Clinton).[13] In this way, four hourly time series were obtained: `trump.pos` and `clinton.pos`, the number of positive or neutral postings; and `trump.neg` and `clinton.neg`, the number of negative postings. The four hourly series of media uploaded to Instagram are shown in Figure 57.

Data set USelection2016.Instagram contains one week (170 hours from Sunday, 2016-10-30 23:00:00 EDT, through Sunday, 2016-11-06 23:00:00 EST; the shaded area in Figure 57) of each series. The data set looks like this:

```
> data(USelection2016.Instagram)
> head(USelection2016.Instagram, 5)
                     date trump.pos clinton.pos trump.neg clinton.neg
1 2016-10-30 23:00:00 EDT   4382909     1916694    634311      301505
2 2016-10-31 00:00:00 EDT   4383903     1917219    634448      301610
3 2016-10-31 01:00:00 EDT   4384490     1917600    634550      301683
4 2016-10-31 02:00:00 EDT   4384894     1917801    634588      301710
5 2016-10-31 03:00:00 EDT   4385057     1917877    634591      301736
```

The date column gives date and time (format: `"%F %T"`) and a label EDT (Eastern Daylight Time) or EST (Eastern Standard Time); daylight saving time ended 2016-11-06 at 2:00 a.m. when clocks were moved back to 1:00 a.m. EST. The time stamp `"2016-11-06 01:00:00"` therefore occurs twice, once with EDT and once again with EST. This peculiarity poses a challenge to be handled with care.

We compute hourly differences of the series, that is, the net increase in the number of media during the hour before a given epoch:

```
my.data <- apply(USelection2016.Instagram[, 2:5], FUN = "diff", MAR = 2)
my.data <- data.frame(date = USelection2016.Instagram$date[-1], my.data)
```

The resulting data set contains the time series to be wavelet-transformed:

```
> head(my.data, 5)
                     date trump.pos clinton.pos trump.neg clinton.neg
2 2016-10-31 00:00:00 EDT       994         525       137         105
3 2016-10-31 01:00:00 EDT       587         381       102          73
4 2016-10-31 02:00:00 EDT       404         201        38          27
5 2016-10-31 03:00:00 EDT       163          76         3          26
6 2016-10-31 04:00:00 EDT       183          42        -1          15
```

The time series of differences are plotted in Figure 58. No labels are shown on the y-axis — actually, the wavelet transform of a time series is independent of its magnitude. There is obviously a 24-hour period in each of these series, reflecting Instagram users' daily routine: busily posting media in the afternoons and evenings. Are there any other important constituents in these series? (In our context, this question means: Are there significant periods, other than the obvious 24-hour period?) Are the series in sync? (The answer to this question is relative to a period to be selected.) Is one series ahead of the other, in our context: Was Clinton ahead, or was Trump ahead, or were they in sync? The cross-wavelet transform helps us answer these questions.

[13] For details, see Schmidbauer, Rösch and Stieler [14]

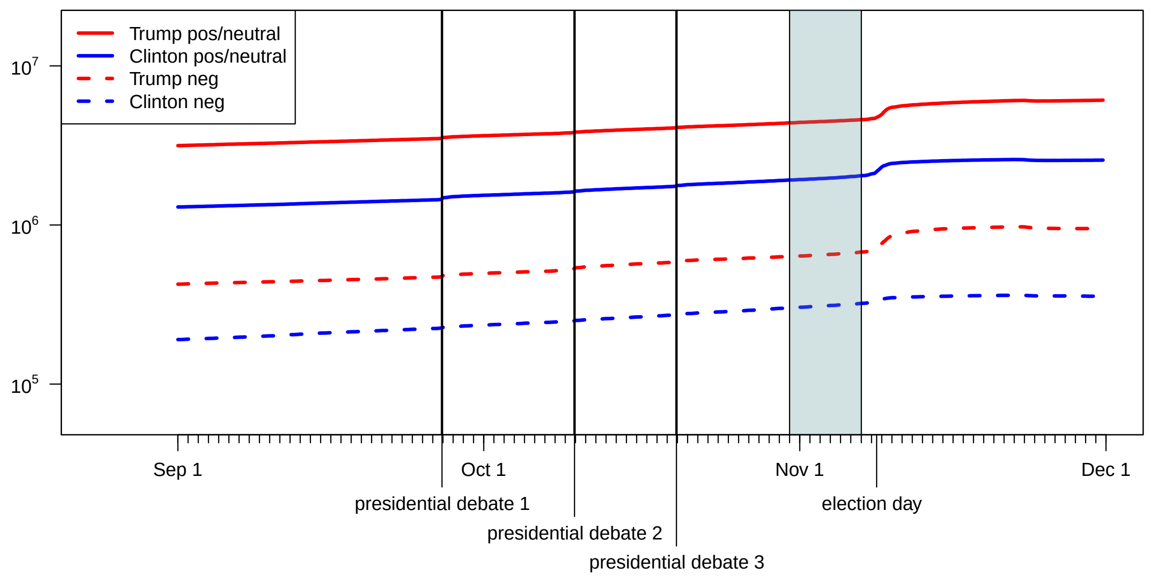


Figure 57: Number of media posted on Instagram, Sep – Nov 2016

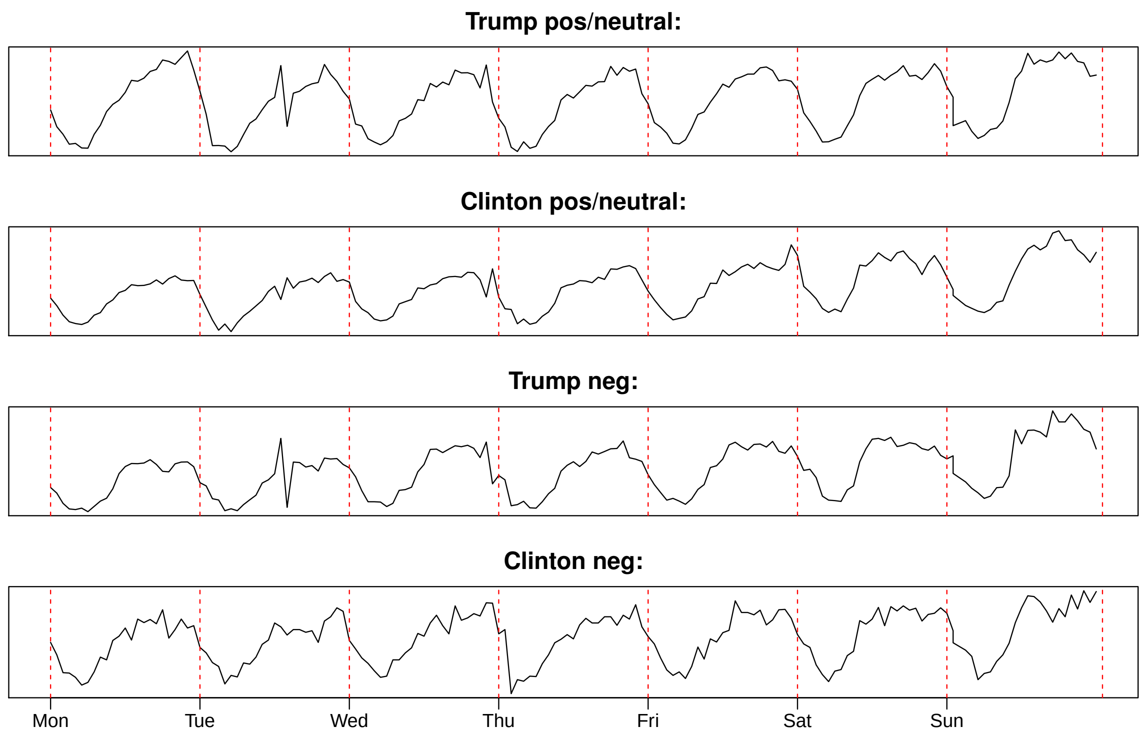


Figure 58: Hourly increase in media, week before election day (2016-11-08)

7.2 Trump vs. Clinton — who was in the lead?

Six pairs can be chosen from the four series of differences in `my.data`. The following analysis focuses on two pairs: Trump pos/neutral vs. Clinton pos/neutral; Trump neg vs. Clinton neg. To compute the cross-wavelet transform:[14]

```
my.wc <- analyze.coherency(my.data, my.pair = c("trump.pos", "clinton.pos"),
                           dt = 1, dj = 1/100,
                           lowerPeriod = 6, upperPeriod = 36,
                           make.pval = TRUE, n.sim = 10)
```

In order to find out which periods are important, we first look at average cross-wavelet powers:

```
wc.avg(my.wc, exponent = 0.5,
       periodlab = "period (hours)",
       minimum.level = 0, maximum.level = 1.25,
       spec.avg.axis = list(at = seq(0, 1.25, by = 0.25)),
       spec.period.axis = list(at = c(12, 24)),
       periodtck = 1, periodtcl = NULL)
```

See Figure 59. This reveals that the series in question also have a significant 12-hour period in common. This is hardly visible in the plots of Figure 58.[15]

WaveletComp provides many ways to fine-tune the image of the cross-wavelet power spectrum. The plots in Figure 60 use the default palette and show arrows, which indicate the phase difference, extending over an area slightly larger than the area of significance, marked by the white lines. For this arrow style, we set `p = 0`, which defines the area to be filled with arrows according to power (and not p-value, which is the default setting), and `lvl = 0.3`, which means arrows are plotted wherever the square root (we set again: `exponent = 0.5`) of the cross-wavelet power is at least 0.3:

```
at <- seq(from = as.POSIXct("2016-10-31 00:00:00", tz = "EST5EDT"),
          to = as.POSIXct("2016-11-06 00:00:00", tz = "EST5EDT"), by = "days")
labels <- format(at, format = "%a")
wc.image(my.wc, n.levels = 250,
         p = 0,  lvl = 0.3,                    # defines where to plot arrows
         legend.params = list(lab = "cross-wavelet power levels",
                              label.digits = 2),
         color.key = "i",
         maximum.level = 1.25, exponent = 0.5,
         label.time.axis = TRUE,
         show.date = TRUE, date.format = "%F %T", date.tz = "EST5EDT",
         spec.time.axis = list(at = at, labels = labels),
         timetcl = -0.5, # outward ticks
         periodlab = "period (hours)",
         spec.period.axis = list(at = c(12, 24)),
         periodtck = 1, periodtcl = NULL)
```

It is important to state the time zone explicitly: `date.tz = "EST5EDT"`, so that the date column in `my.date` is converted correctly into a POSIXct class object.

The arrows (see also Figure 2) now reveal that both pairs under scrutiny are more or less in sync at the 24-hour period, but there is a very important difference between the pairs at the 12-hour period: Trump pos/neutral is leading over Clinton pos/neutral, while Clinton neg is leading over Trump neg most of the time. It looks like Trump supporters and Clinton opponents were eager to post media, while Trump opponents and Clinton supporters were sluggish. In view of election forecasts and the age structure of Instagram users, these results come as a surprise.

[14] An analogous transform has to be computed with `my.pair = c("trump.neg", "clinton.neg")`.

[15] The 12-hour period is not generally significant in series of media uploads to Instagram, see Schmidbauer, Rösch and Stieler [14].

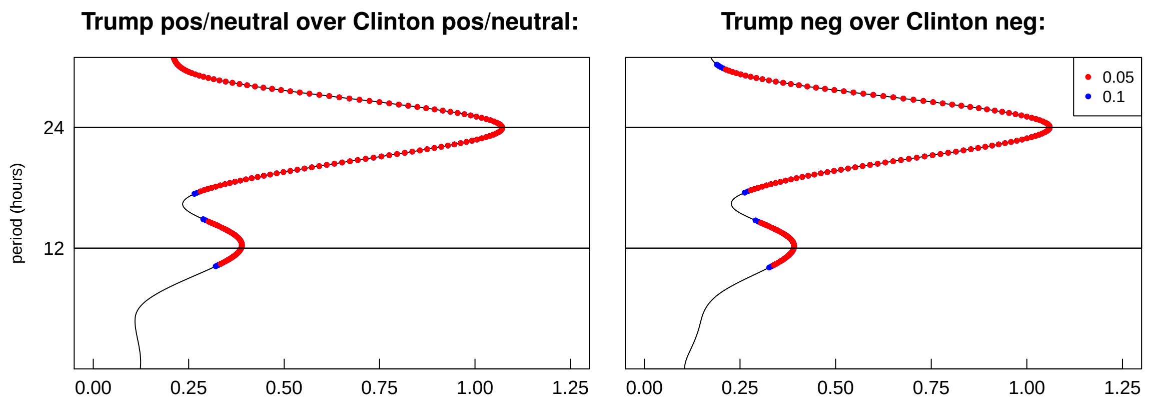


Figure 59: Average cross-wavelet power, week before election day (2016-11-08)

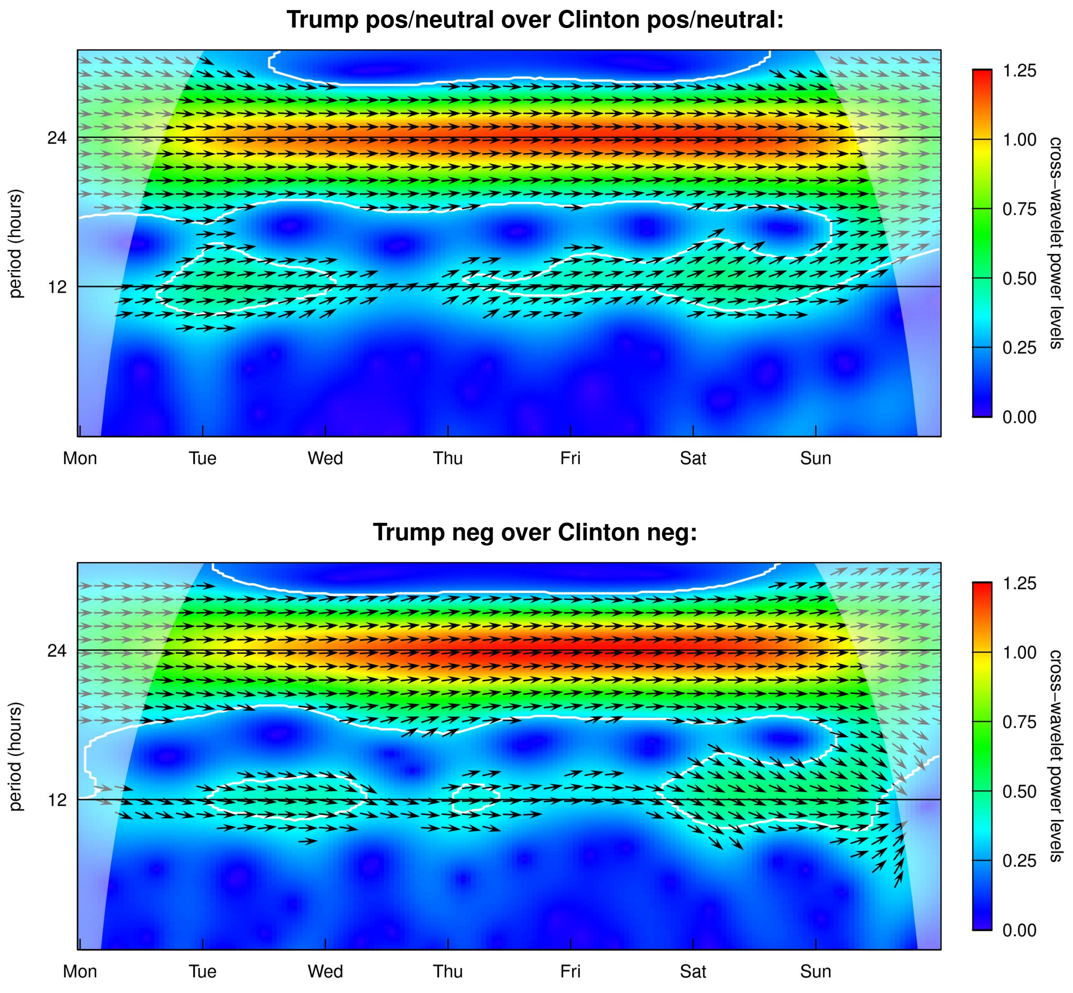


Figure 60: Cross-wavelet power spectra, week before election day (2016-11-08)

7.3 Analyzing the phase difference

What exactly is going on at the 12-hour period? The arrow angles plotted in the cross-wavelet power spectra (Figure 60) can give a hint on the phase difference between the two series at any period between 6 and 36 hours at any point in time that week. For a more detailed analysis, WaveletComp provides the means to plot, using an object of class analyze.coherency, phases of the two series and phase differences between the series at a given period, here: 12 hours. The angle is expressed in radians, ranging from $-\pi$ to $+\pi$. At the 12-hour period, π corresponds to 6 hours. To make the plot more intuitive, we furnish its phase axis with labels expressing angles in hours. In the following code, `at.ph` and `labels.ph` define the location and labels on the phase axis:

```
at.ph <- seq(-pi, pi, by = pi/6)
labels.ph <- seq(-6, 6, by = 1)
at.t <- seq(from = as.POSIXct("2016-10-31 00:00:00", tz = "EST5EDT"),
            to = as.POSIXct("2016-11-06 00:00:00", tz = "EST5EDT"), by = "days")
labels.t <- format(at.t, format = "%a")
wc.sel.phases(my.wc, sel.period = 12, siglvl = 1,
              spec.phase.axis = list(at = at.ph, labels = labels.ph),
              timelab = "", phaselab = "phase (hours)",
              show.date = TRUE,  date.format = "%F %T", date.tz = "EST5EDT",
              spec.time.axis = list(at = at.t, labels = labels.t),
              only.coi = TRUE)
abline(h = 0)
```

The upper plot of Figure 61, referring to the case Trump pos/neutral over Clinton pos/neutral, confirms that Trump pos/neutral is leading over Clinton pos/neutral throughout the week: the red line is *always* preceding the blue one, and the phase difference (the dashed line) is always positive. Donald Trump's supporters were ahead of Hillary Clinton's supporters. In the case of Trump neg over Clinton neg, however, things are reversed: Trump neg is lagging behind Clinton neg most of the time! In this sense, Hillary Clinton's opponents were ahead of Donald Trump's opponents.

For an even finer analysis, WaveletComp allows to extract the time series of angles at period 12 hours from `my.wc`. To that end, we first need to locate the period in `my.wc` closest to 12 (this depends on `dj`; see Section 2.1):

```
row.closest.to.12 <- which.min((my.wc$Period - 12)^2)
```

The relevant angle series can now be extracted, and we can compute the lead time in minutes:

```
## determine angles:
angle.series <- my.wc$Angle[row.closest.to.12,]
## omit 6 values at the beginning and 6 values at the end (outside of coi):
angle.series[1:6] <- NA
angle.series[(length(angle.series) - 5):length(angle.series)] <- NA
## convert to minutes: 2*pi corresponds to 12 hours, because period = 12:
lead.time.in.minutes <- 60 * (12 * (angle.series / (2*pi)))
```

A plot of the lead time (or phase difference) is shown in Figure 62. (Remember that we also need to compute `my.wc` with `my.pair = c("trump.neg", "clinton.neg")`.) Taking means, it turns out that Trump supporters were on average 35 minutes ahead of Clinton supporters in posting media on Instagram in the week before the election, while Trump opponents were lagging on average about 22 minutes behind Clinton opponents. *Why* did this happen? Wavelet analysis cannot answer this question. However, with this evidence from wavelet analysis at hand, we can ask very specific questions.

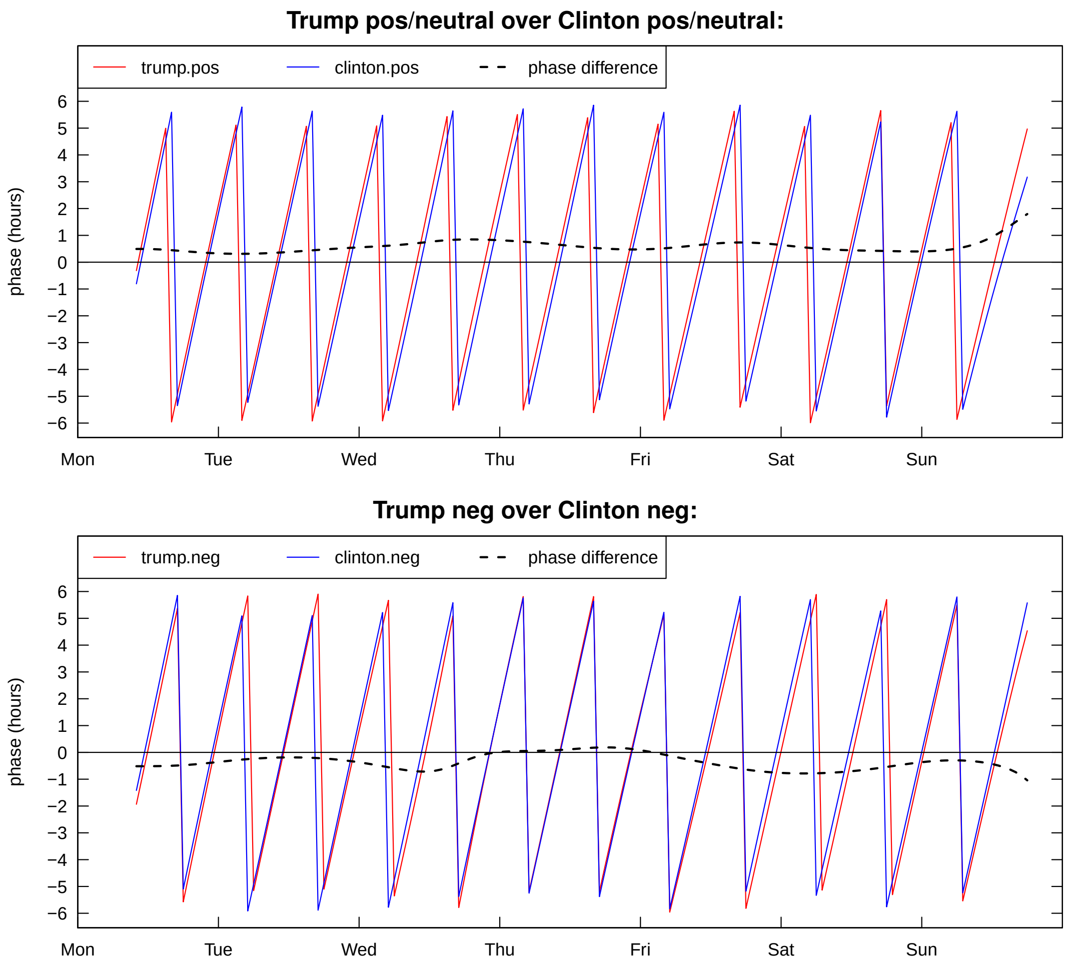


Figure 61: Phases and phase difference at period 12 hours, week before election day (2016-11-08)

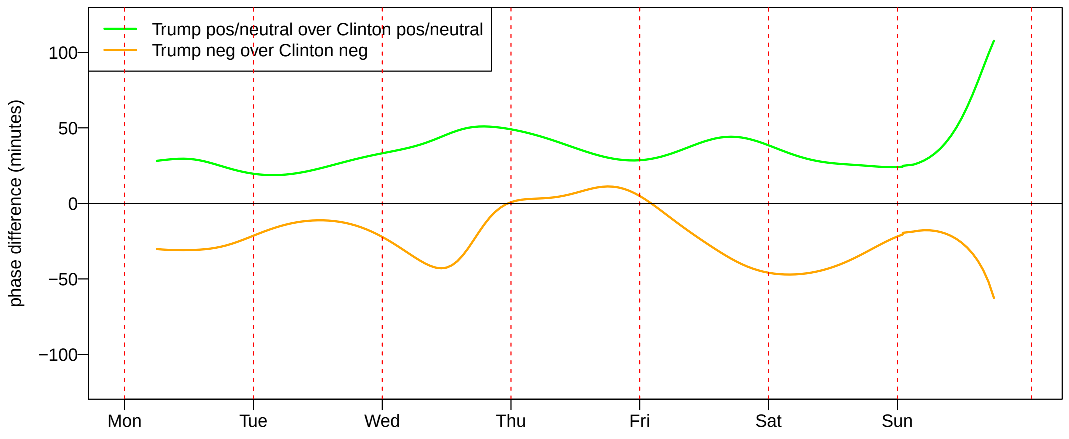


Figure 62: Phase difference at period 12 hours, week before election day (2016-11-08)

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

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

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

[8] 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.

[9] 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.

[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] Percival D.B., and Walden A.T., 2000. *Wavelet Methods for Time Series Analysis.* Cambridge University Press, New York.

[13] R Core Team, 2018. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

[14] Schmidbauer H., Roesch A., and Stieler F., 2018. The 2016 US presidential election and media on Instagram: Who was in the lead? *Computers in Human Behavior* 81, 148–160. doi: https://doi.org/10.1016/j.chb.2017.11.021

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

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