WaveletComp: a glance at the R package

FOM, 2014-11-28

Harald Schmidbauer

Istanbul Bilgi University, Istanbul, Turkey

Angi Rösch

FOM University of Applied Sciences, Munich, Germany

Wavelets

- tool for analyzing the periodic behavior of a time series
- no period pre-assigned
- compromise between period (frequency) and time resolution (periodicity need not be constant)

Our motivation

- investigation of shock dynamics in networks of asset markets
- understanding the periodicity of FX transactions
- SkyAtlas: understanding demand structure
- . . . shortcomings of other wavelet packages
- We needed:
  - selective reconstruction
  - bivariate analysis capabilities
  - flexible plotting
  - tests, based on simulation

Commands

- Simulate a series with variable periods:

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

- Compute its wavelet transform:

```
my.data = data.frame(x = x)
my.w = analyze.wavelet(my.data, "x",
                       loess.span = 0, dt = 1, dj = 1/250,
                       lowerPeriod = 16, upperPeriod = 128,
                       make.pval = T, no.sim = 100)
```

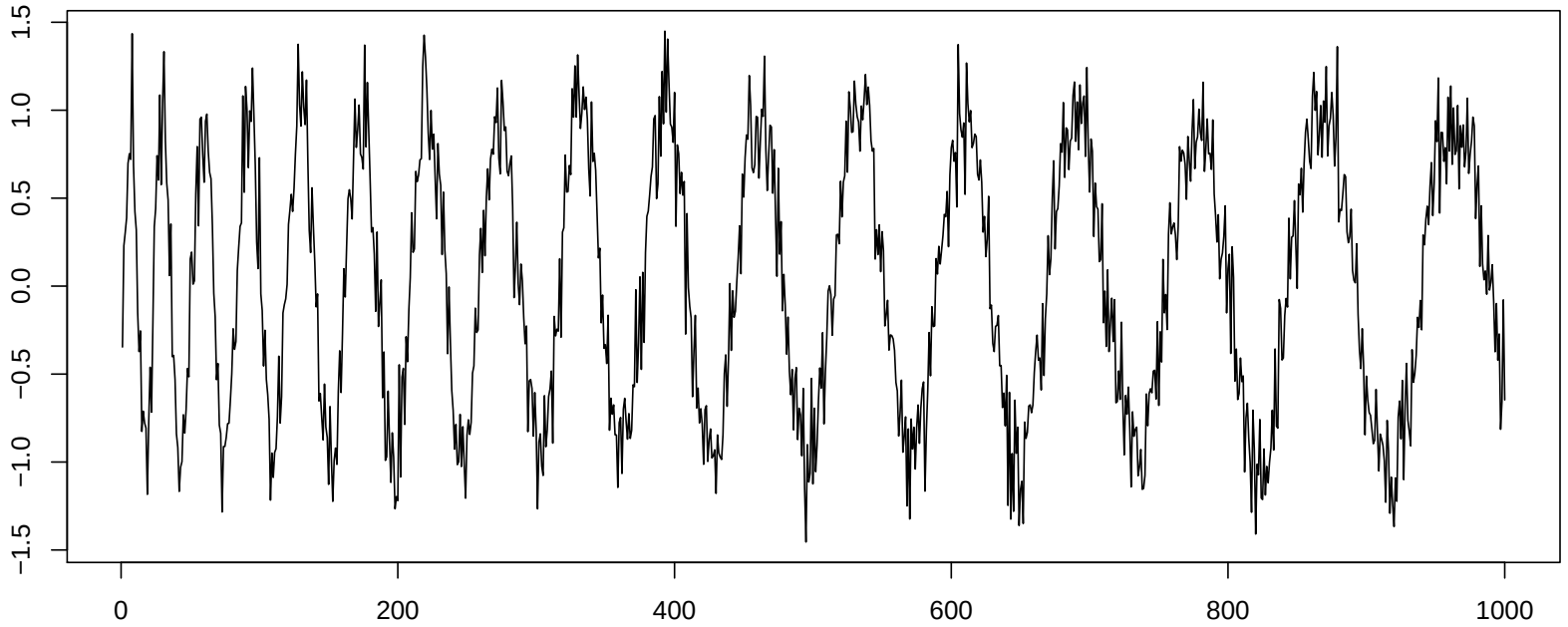
- Plot its wavelet power spectrum:

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

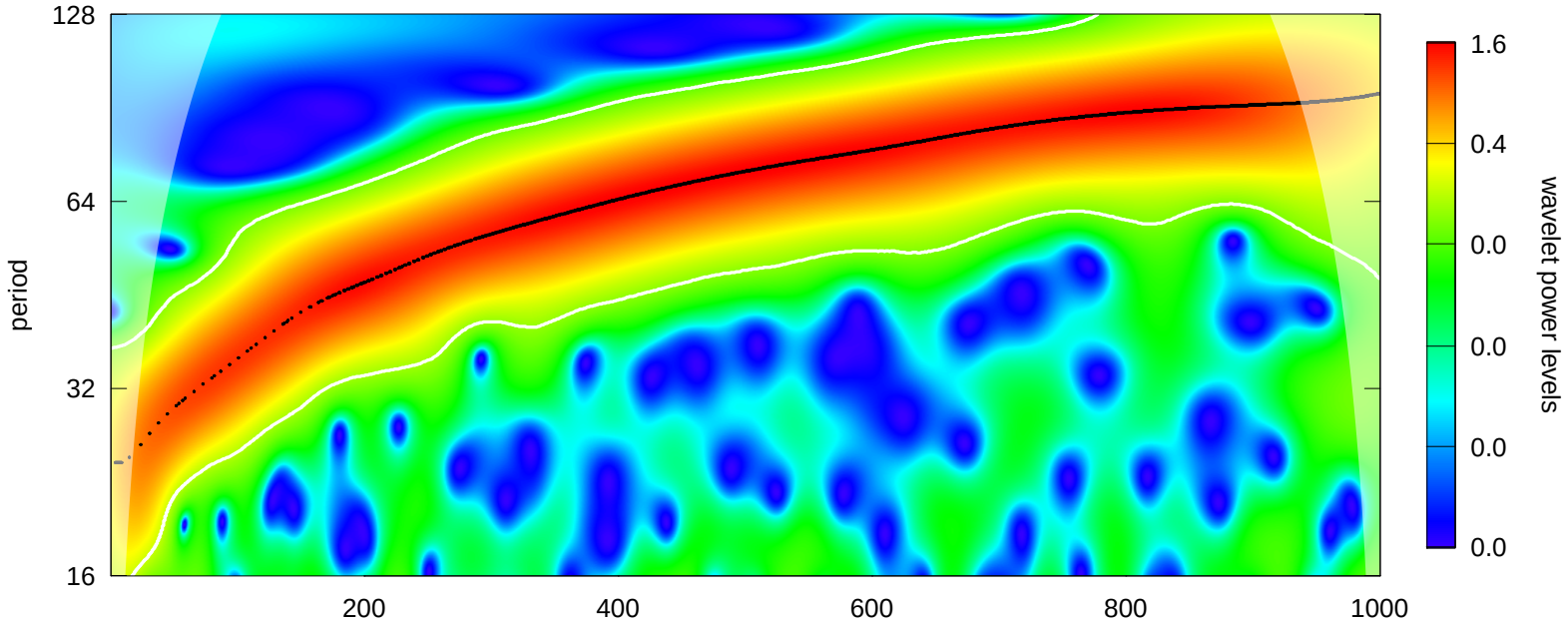
- Reconstruct it:

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

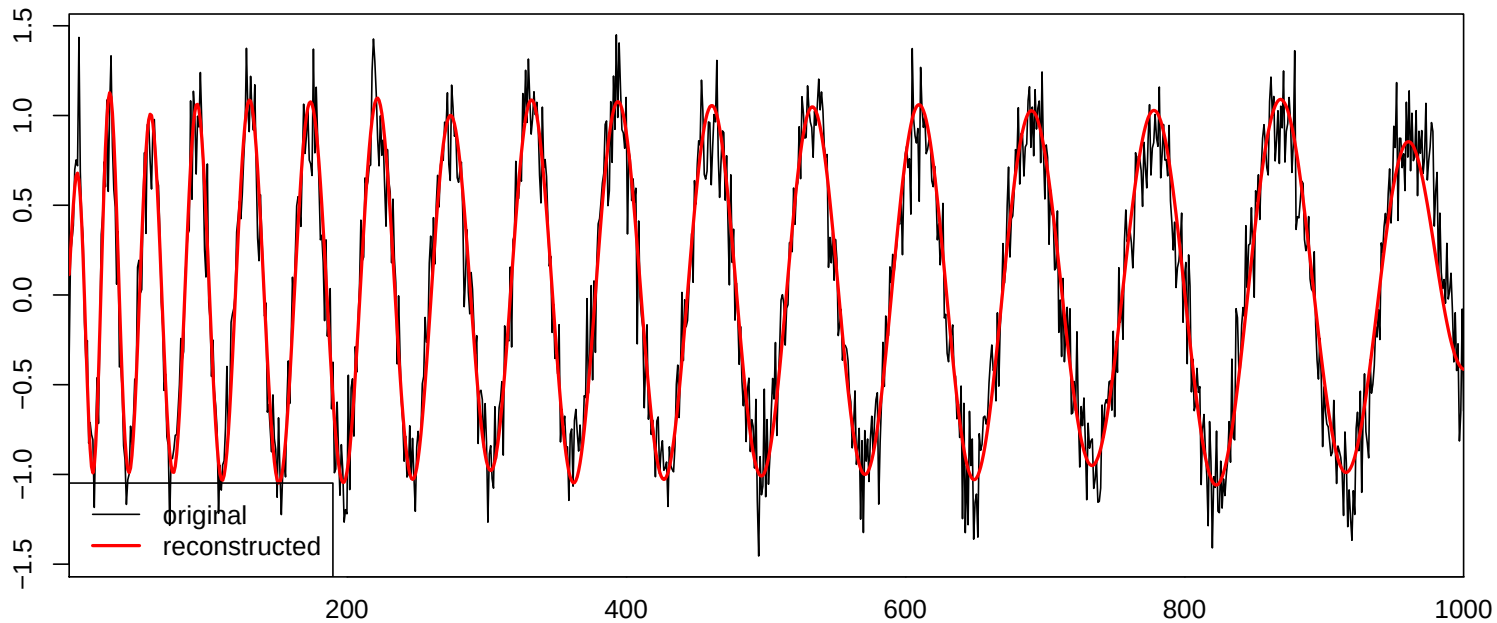
The series



The wavelet power spectrum



Reconstruction ("denoising")



Commands

- Series with two periods superposed:

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

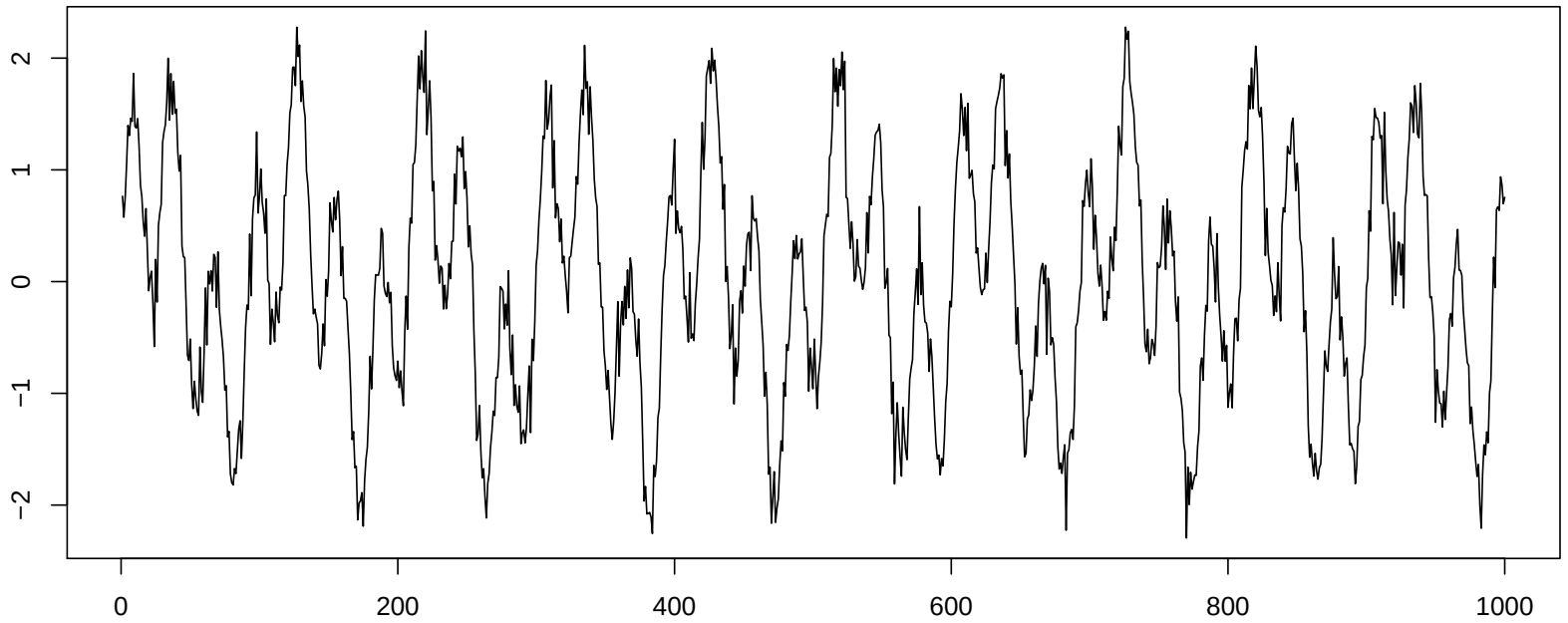
- Wavelet transform, plot:

```
my.data = data.frame(x = x)
my.w = analyze.wavelet(my.data, "x",
                       loess.span = 0, dt = 1, dj = 1/250,
                       lowerPeriod = 16, upperPeriod = 128,
                       make.pval = T, no.sim = 100)
wt.image(my.w, n.levels = 250,
         legend.params = list(lab = "wavelet power levels") )
```

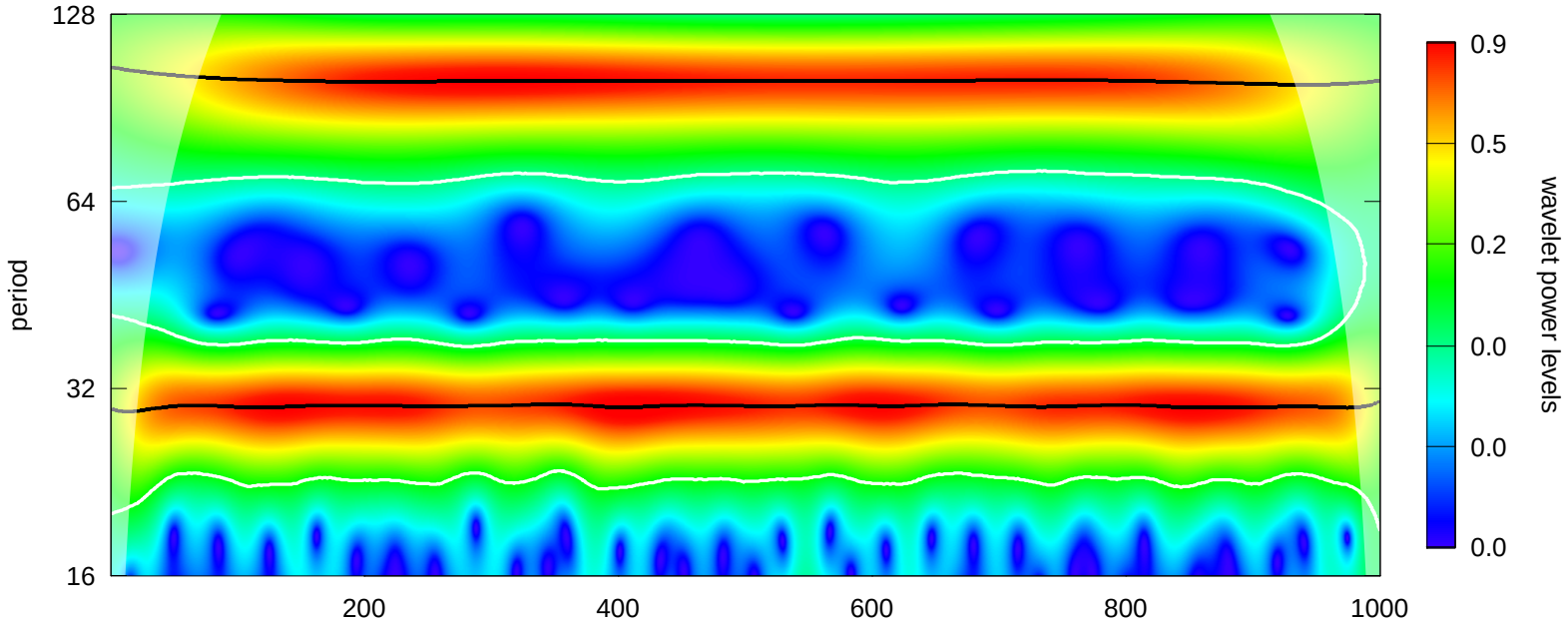
- Reconstruction, using only period 100:

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

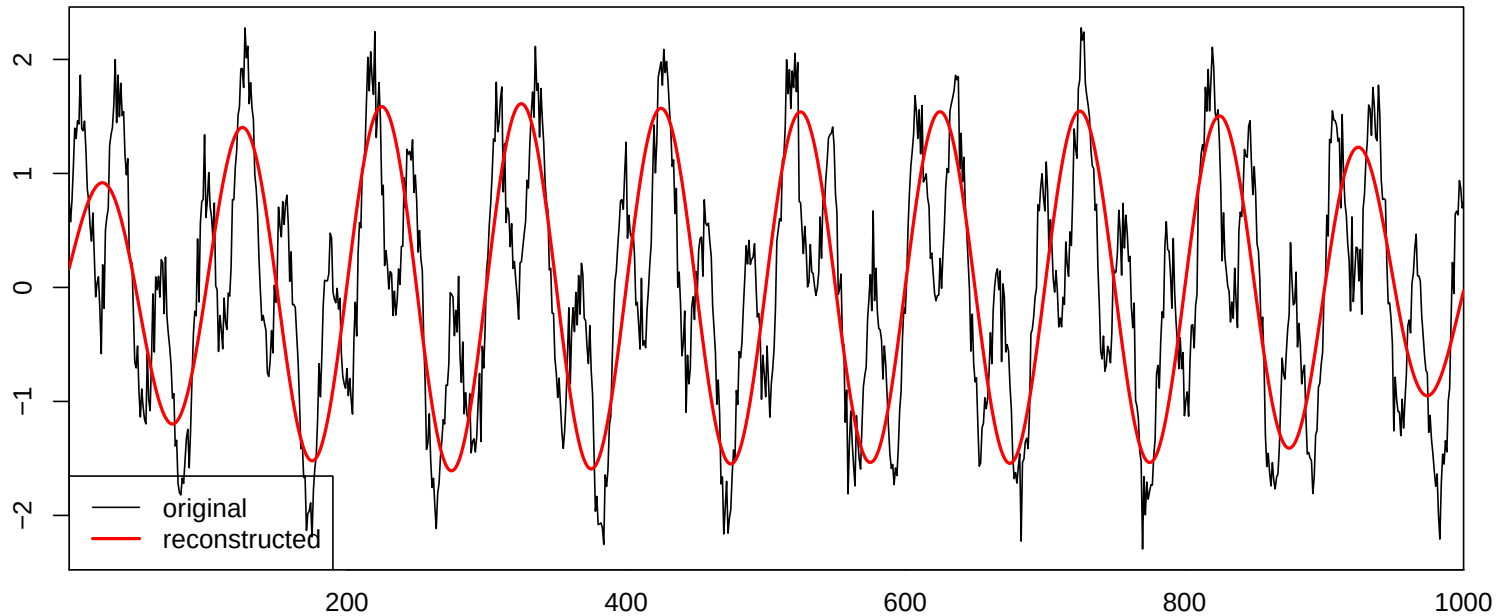
The series



The wavelet power spectrum



Reconstruction, using only period 100



Commands

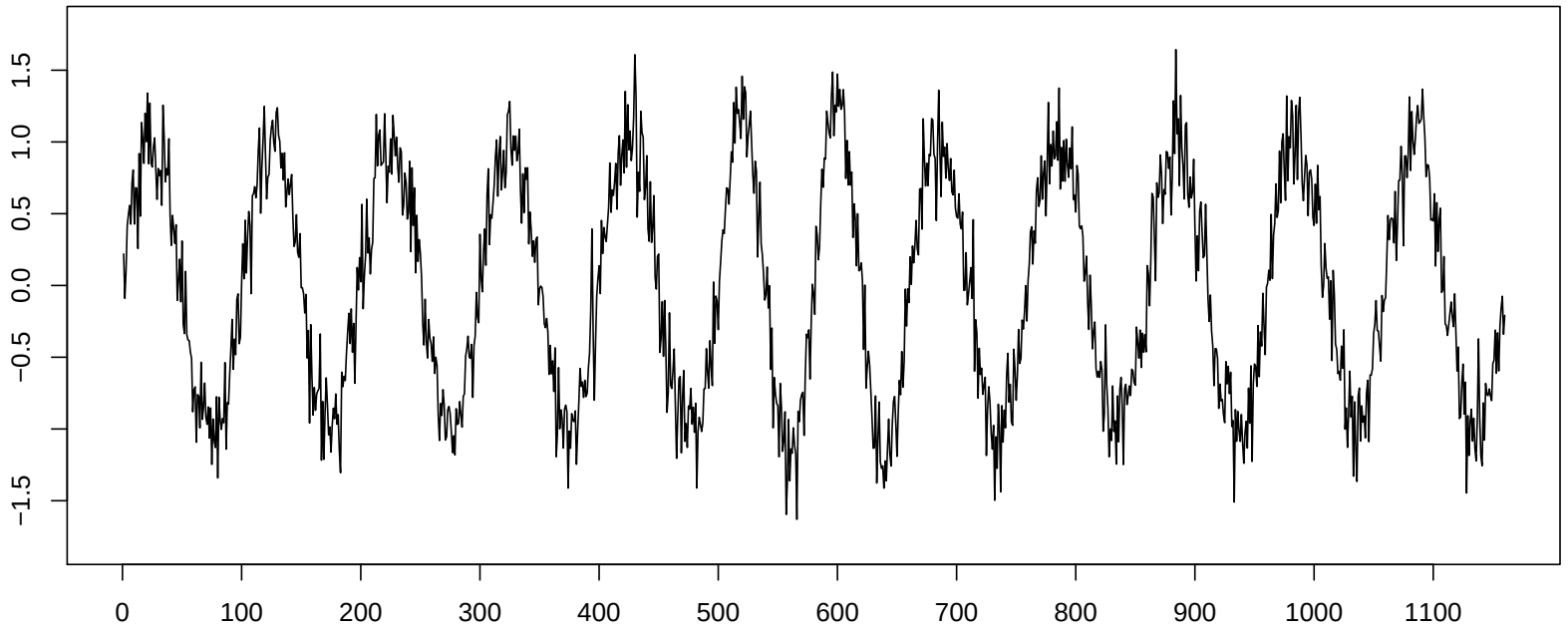
- Simulate a series:

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

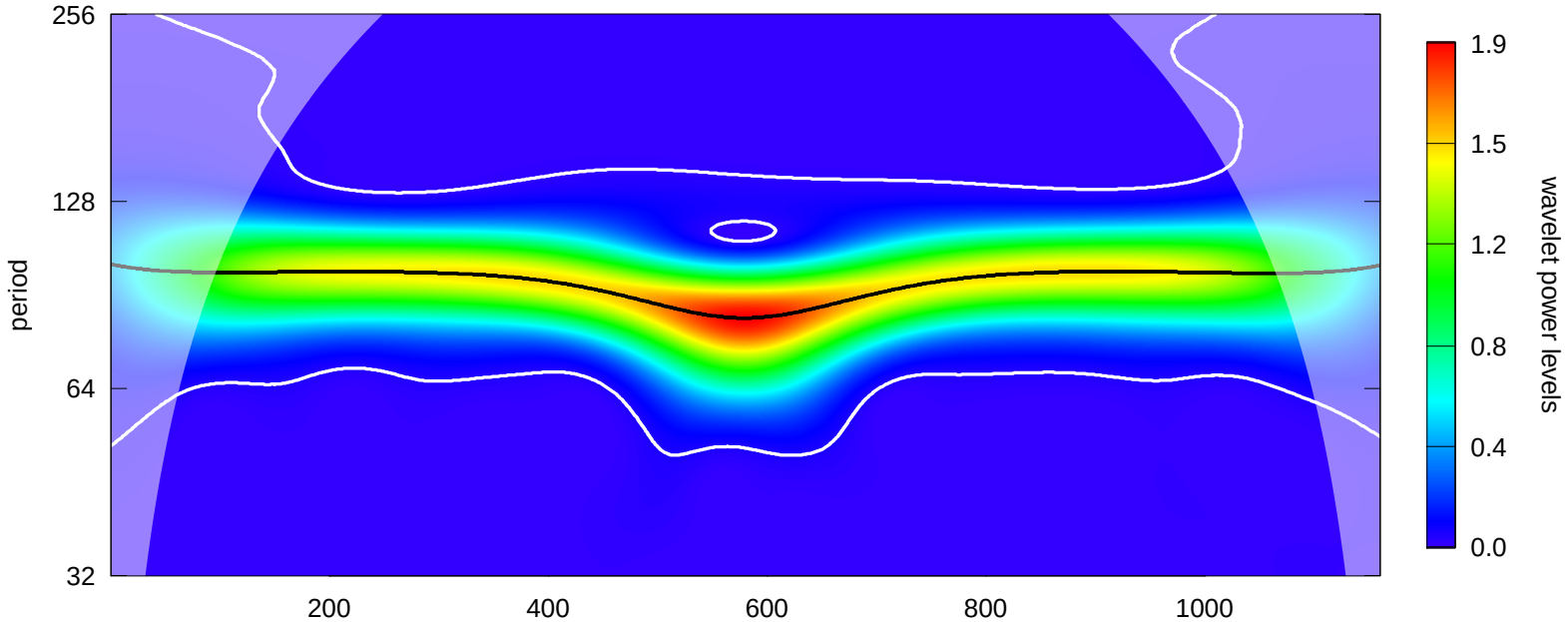
- Wavelet transform, plot:

```
my.w = analyze.wavelet(my.data, "x",
                       loess.span = 0, dt = 1, dj = 1/250,
                       lowerPeriod = 32, upperPeriod = 256,
                       make.pval = T, no.sim = 100)
wt.image(my.w, color.key = "interval", n.levels = 250,
         legend.params = list(lab = "wavelet power levels", label.digits = 2))
```

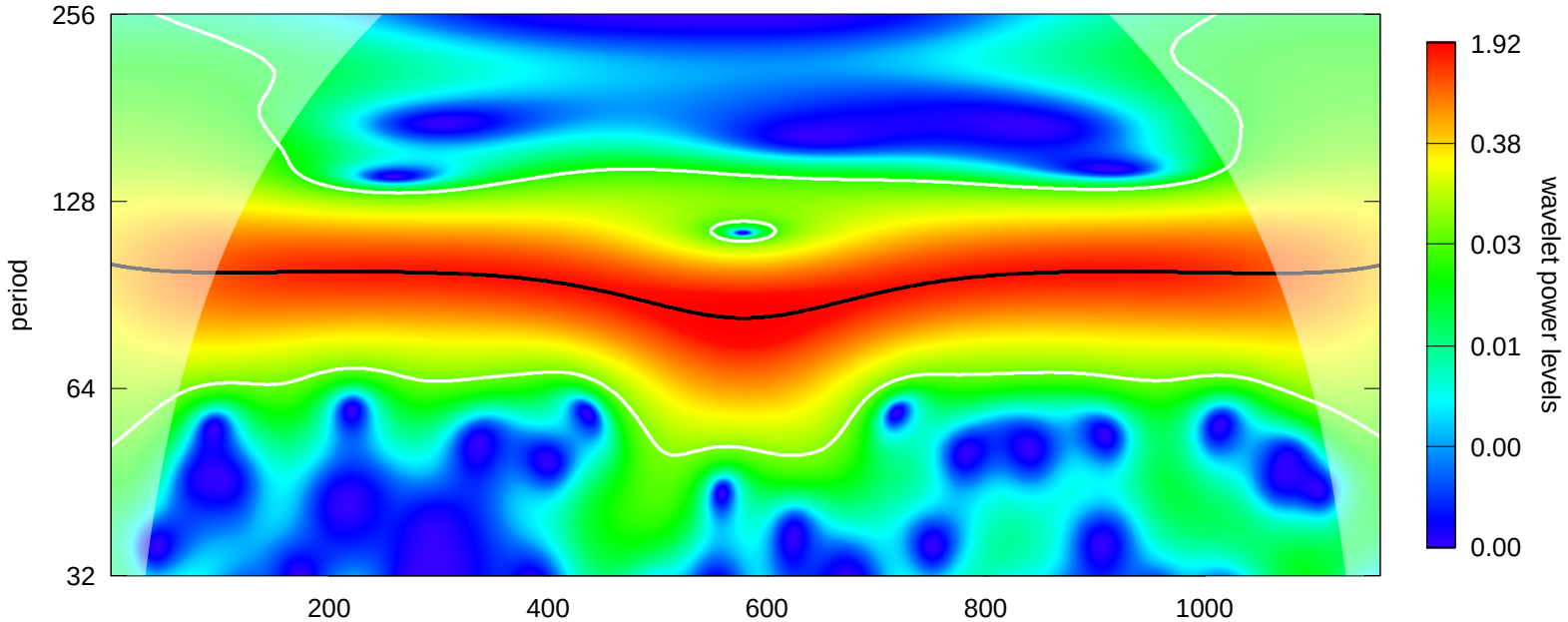
The series



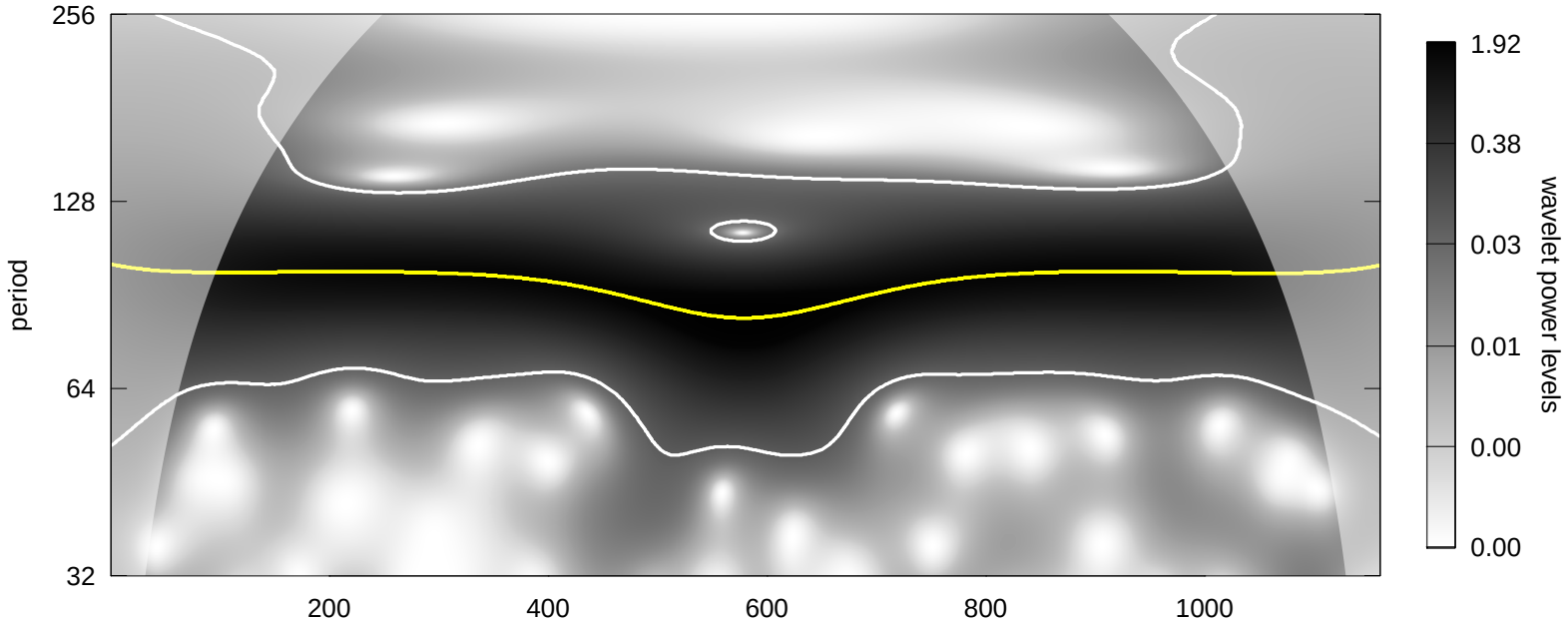
The wavelet power spectrum, `color.key = "interval"`



The wavelet power spectrum, `color.key = "quantile"`



The wavelet power spectrum, grayscale



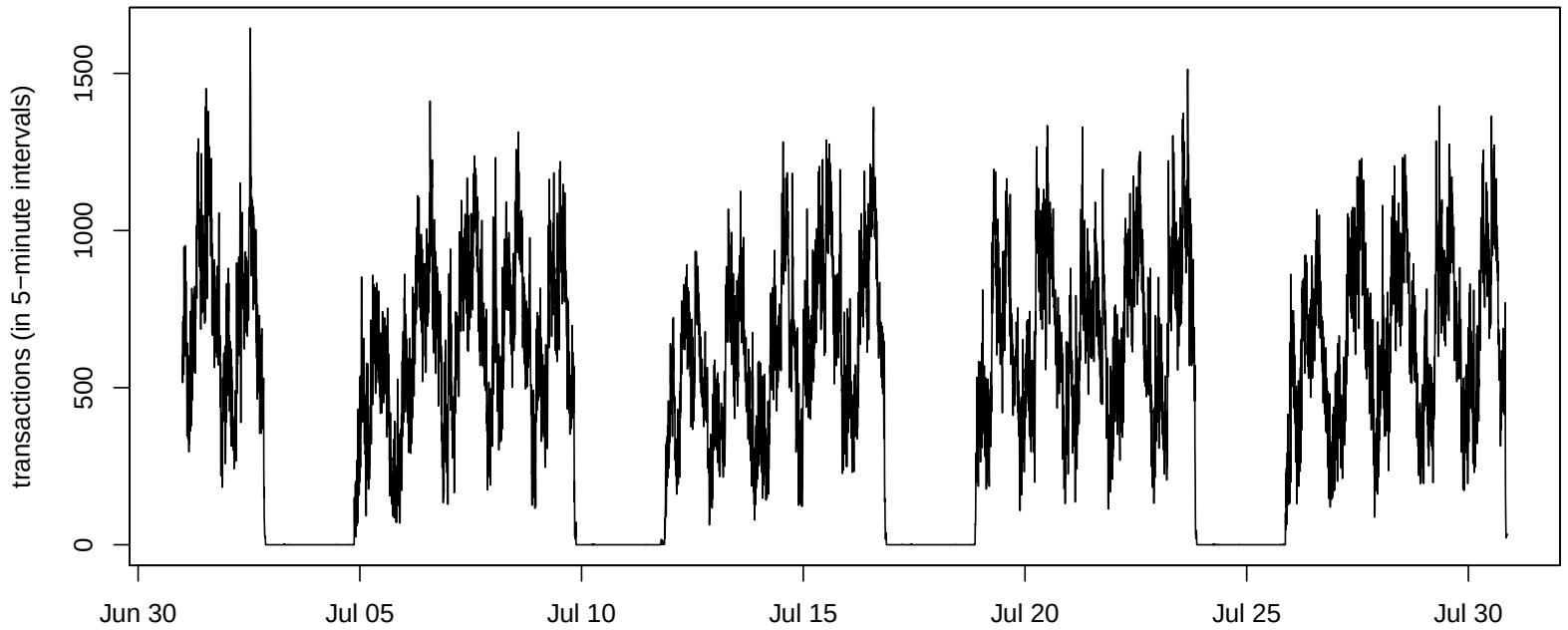
Data

- worldwide number of registered USD/euro transactions, July 2010
- data:

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

- active cycle: Sunday, 21:00, to Friday, 20:55.

Number of transactions



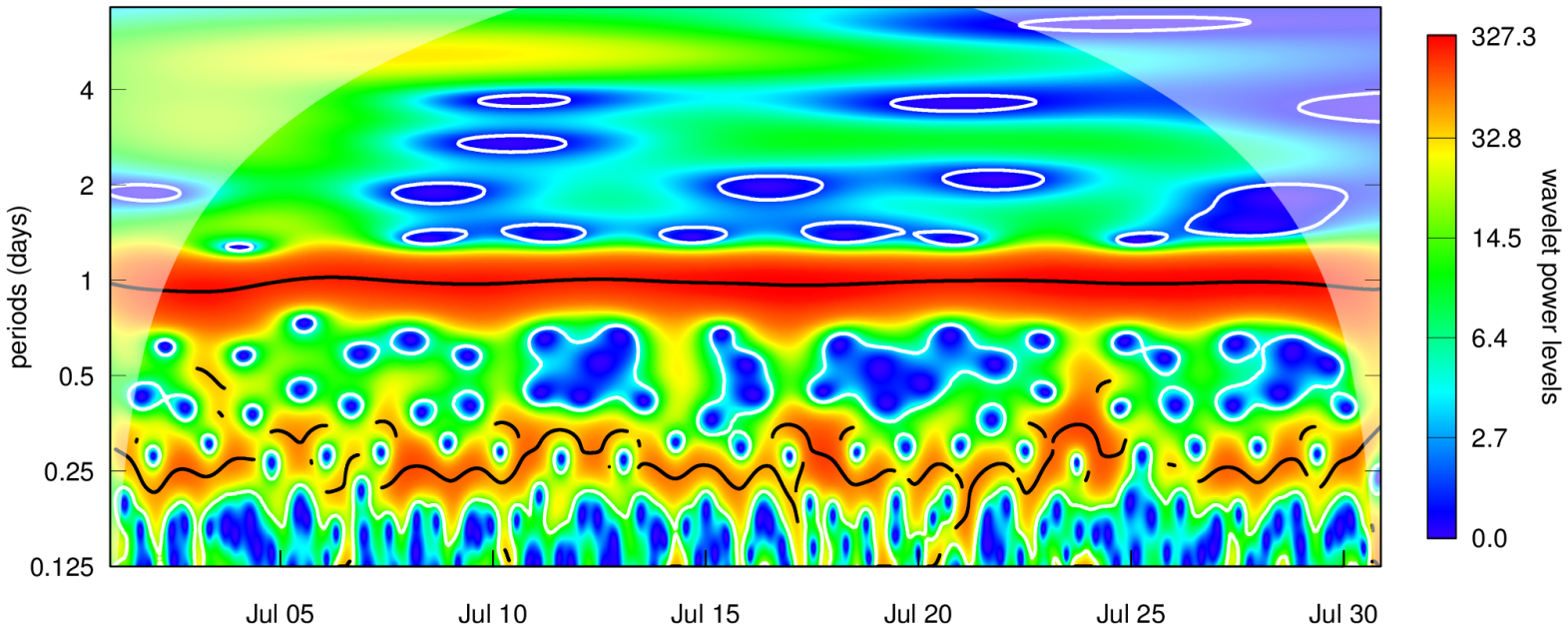
Analysis

- Analysis:

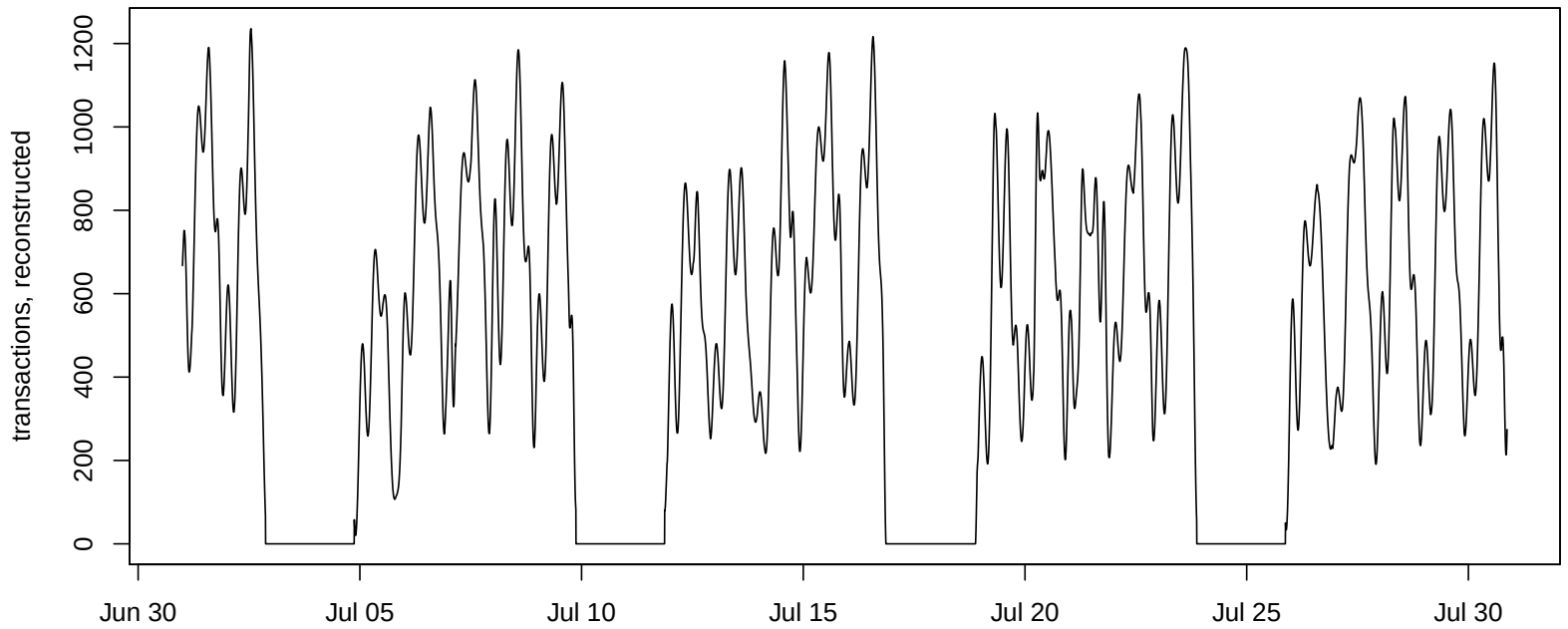
```
my.data.a = FXtrade.transactions[FXtrade.transactions$active == T, ]
my.w.a = analyze.wavelet(
           my.data.a, "transactions",
           loess.span = 0.0,  # no detrending required
           dt = 1/(12*24),    # one day has 12*24 5-minute time slots
           dj = 1/250,        # resolution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

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

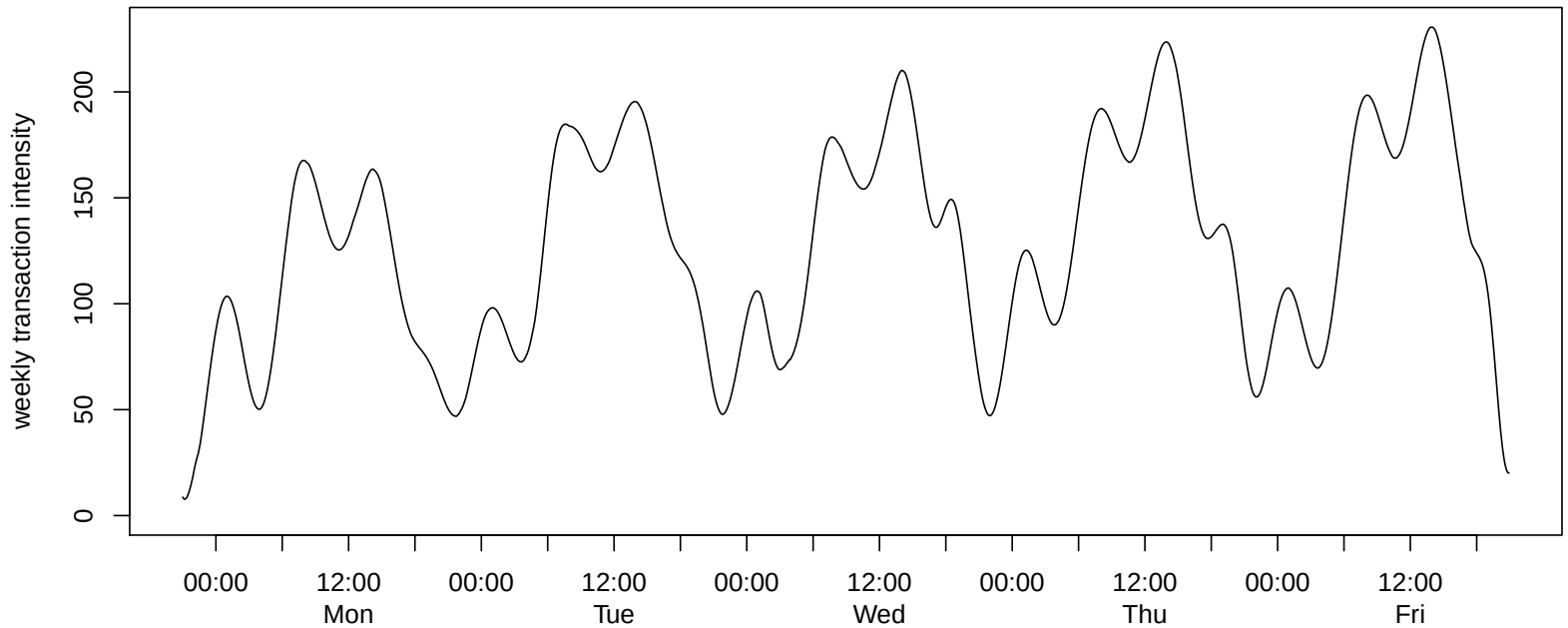
Wavelet power spectrum



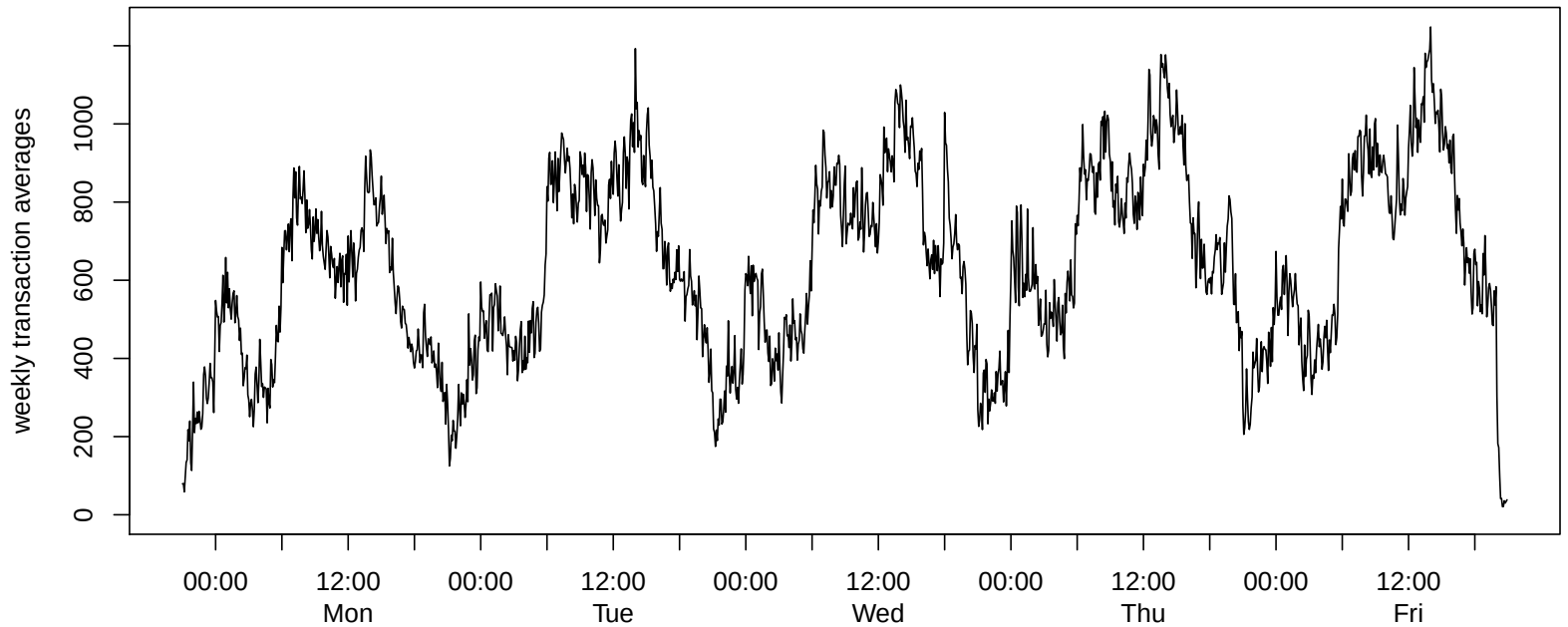
Wavelet reconstruction



Intensity estimation



The naive way...

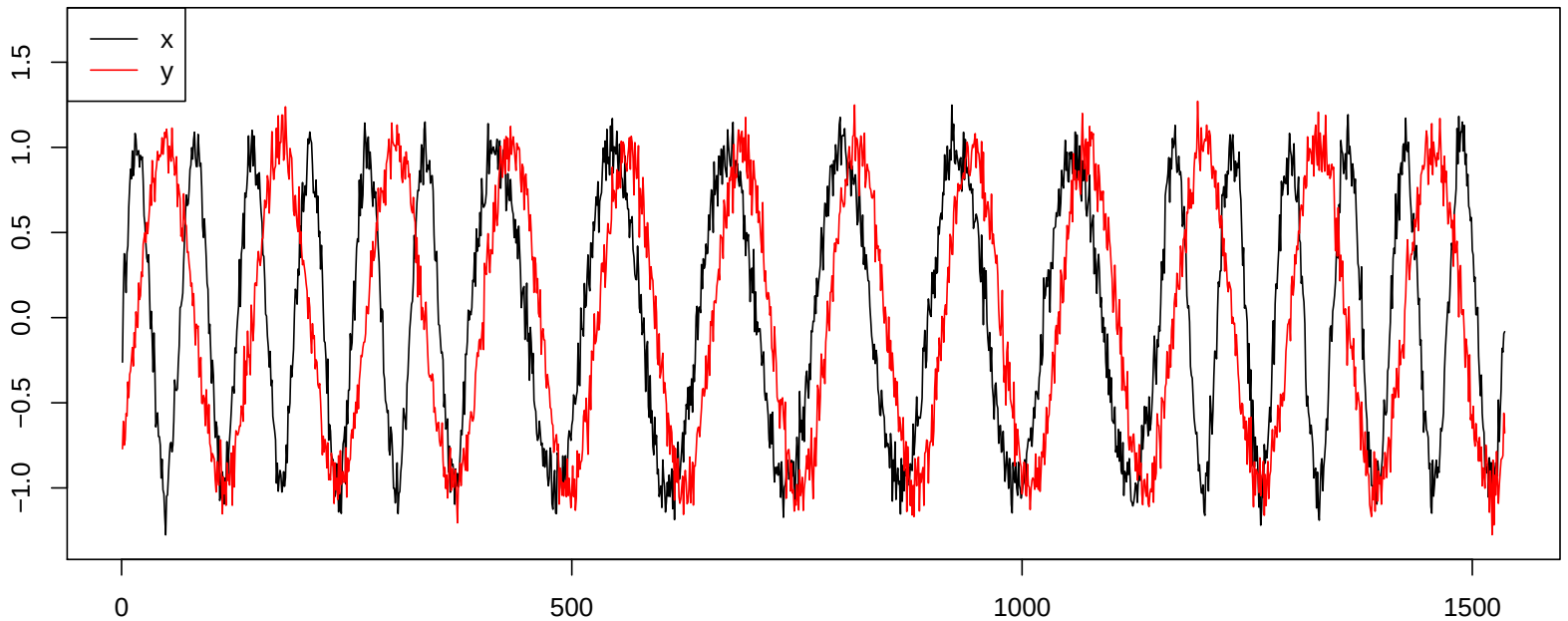


Commands

- Simulate two series:

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

The series



Wavelet transformation, plotting

- Compute cross-wavelet transform:

```
my.data = data.frame(x = x, y = y)
my.wc = analyze.coherency(my.data, my.pair = c("x","y"),
                          loess.span = 0,
                          dt = 1, dj = 1/100,
                          make.pval = T, no.sim = 500)
```

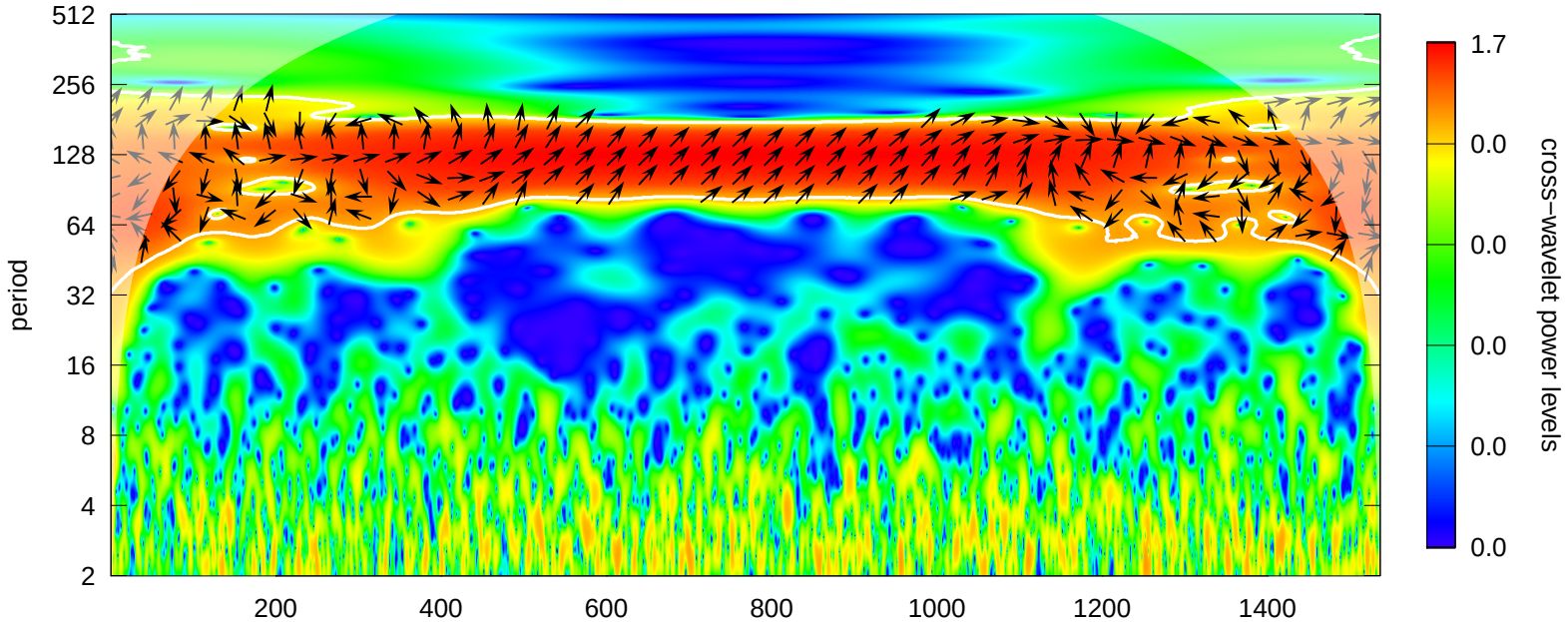
- Plot cross-wavelet power spectrum:

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

- Plot wavelet coherence:

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

Cross-wavelet power spectrum



Wavelet coherence spectrum