Spectral Analysis in R

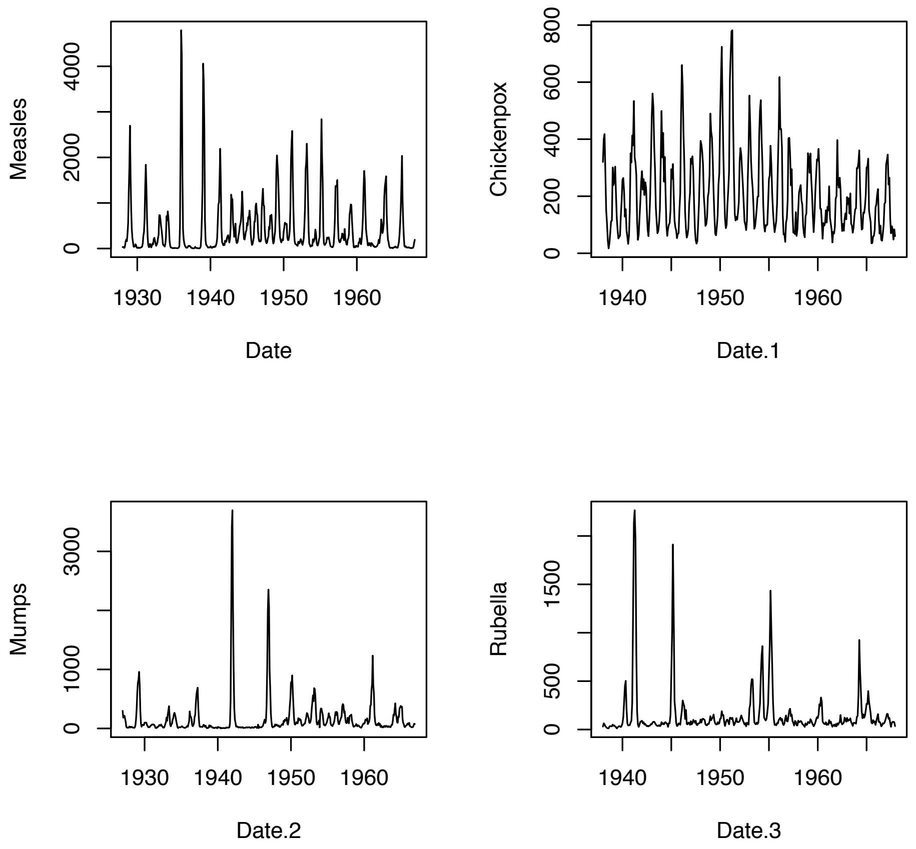
Helen J. Wearing

June 8, 2010

Contents

1 Motivation

Cyclic dynamics are the rule rather than the exception in infectious disease data, which may be due to external forcing by environmental drivers or the inherent periodicity of immunizing (or partially immunizing) infections or a combination of both. As an example, plotted in the figure below are weekly case reports of childhood diseases from Copenhagen, Denmark during the mid-twentieth century.

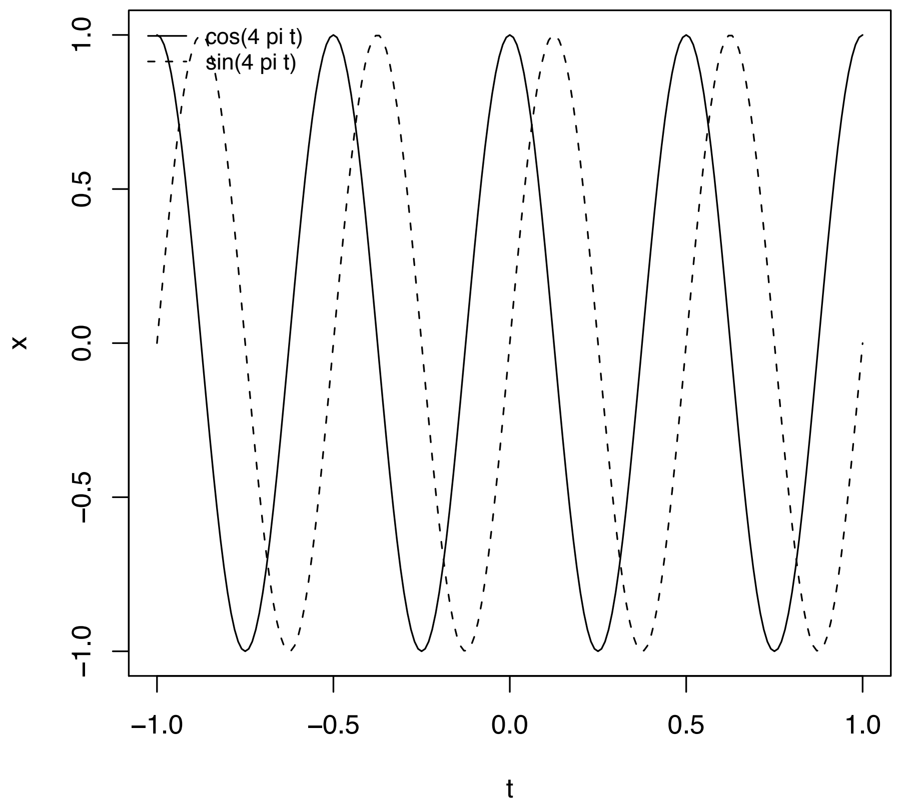


What types of questions might we ask of these data? In this module, we introduce how to estimate the periodicity of time series using spectral analysis. Specifically, we will look at recurrent epidemics from either simulated or real data. We can often use these summary metrics as probes to match model output to data.

2 What is spectral analysis?

In a nutshell: the decomposition of a time series into underlying sine and cosine functions of different frequencies, which allows us to determine those frequencies that appear particularly strong or important.

Let's briefly re-familiarize ourselves with sine and cosine functions!



The **frequency** (f) of a sine or cosine function is typically expressed in terms of the number of cycles per unit time. For example, in the above figure the frequency of each function is 2 cycles per unit time.

The **period** (T) of a sine or cosine function is defined as the length of time required for one full cycle. Thus, it is the reciprocal of the frequency $(T = 1/f)$. In the above figure $T = 1/2$.

Fitting sine waves

One way of viewing spectral analysis is as a linear multiple regression problem, where the dependent variable is the observed time series, and the independent variables are the sine functions of all possible (discrete) frequencies.

Suppose we have a time series x_t of length n, for convenience assume n is even. We can fit a time series regression with x_t as the response and the following $n-1$ predictor variables:

$$\cos\left(\frac{2\pi t}{n}\right), \sin\left(\frac{2\pi t}{n}\right), \ldots, \cos\left(\frac{2(n/2-1)\pi t}{n}\right), \sin\left(\frac{2(n/2-1)\pi t}{n}\right), \cos(\pi t)$$

If we represent the estimated regression coefficients by $a_1, b_1, \ldots, a_{n/2-1}, b_{n/2-1}, a_{n/2}$, respectively, we can write x_t as

$$x_t = a_0 + \sum_{k=1}^{n/2-1} [a_k \cos(2\pi kt/n) + b_k \sin(2\pi kt/n)] + a_{n/2} \cos(\pi t) \tag{1}$$

The cosine parameters, a_k, and sine parameters, b_k, tell us the degree to which the respective functions are correlated with the data. This regression model is a finite Fourier series for a discrete time series.

Note that because the number of coefficients equals the length of the time series, there are no degrees of freedom for error. The intercept term, a_0, is just the mean, $\bar{x}$, of the time series. The lowest possible frequency is one cycle, or 2π radians, per record length (which is $2\pi/n$ radians per sampling interval). A general frequency, in this representation, is k cycles per record length ($2\pi k/n$ radians per sampling interval). The highest frequency is 0.5 cycles per sampling interval (π radians per sampling interval).

We should pay close attention to the **sampling interval** and **record length**. Many time series are of a variable that is continuous in time but is sampled to give a time series at discrete time steps. The sampling interval (or sampling rate) constrains the highest frequency (known as the Nyquist frequency) that we can detect. For example, if we sample every week, we cannot detect cycles less than 2 weeks in length. On the other hand, the length of the time series determines the lowest frequency that we can distinguish.

Periodogram

The periodogram quantifies the contributions of the individual frequencies to the time series regression and is defined as

$$P_k = a_k^2 + b_k^2$$

where P_k is the periodogram value at frequency k (for $k = 1, \ldots, n/2$). The periodogram values can be interpreted in terms of variance of the data at the respective frequency or period. A plot of P_k, as spikes, against k is a Fourier line spectrum. The raw periodogram in R is obtained by joining the tips of the spikes in the Fourier line spectrum to give a continuous plot and scaling it so that the area equals the variance.

Although we have introduced the periodogram in the context of a linear multiple regression, the calculations are usually performed with the fast Fourier transform algorithm (FFT) (and this is what R uses too).

To summarize, spectral analysis will identify the correlation of sine and cosine functions of different frequency with the observed data. If a large correlation (sine or cosine coefficient) is identified, you can conclude that there is a strong periodicity of the respective frequency (or period) in the data.

Let's consider a simple example to clarify the underlying "mechanics" of spectrum analysis in R before we discuss further details of the technique.

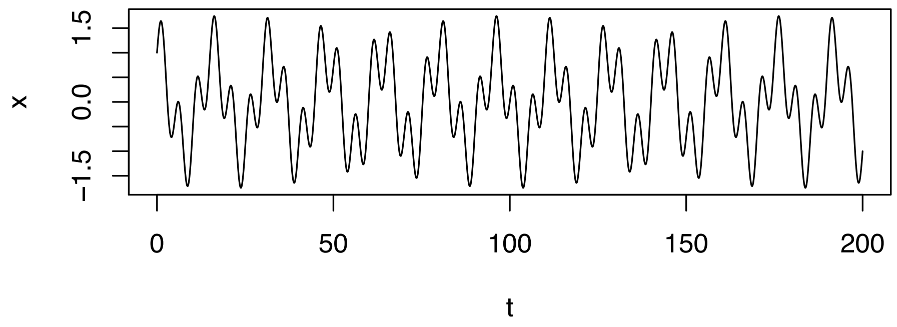
Simple Example

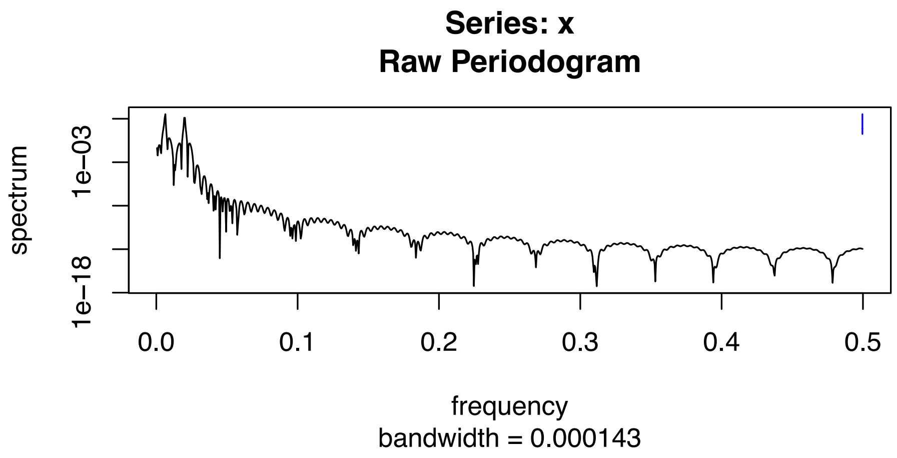
We will create a simple time series, and then see how we can extract the frequency information using spectral analysis. First, create a time variable t and then specify the time-dependent variable x:

```
> t <- seq(0,200,by=0.1)
> x <- cos(2*pi*t/16) + 0.75*sin(2*pi*t/5)
```

The variable x is made up of two underlying periodicities: the first at a frequency of 1/16 or period of 16 (one observation completes 1/16'th of a full cycle, and a full cycle is completed every 16 observations) and the second at a frequency of 1/5 (or period of 5). The cosine coefficient (1.0) is larger than the sine coefficient (0.75).

```
> par(mfrow=c(2,1))
> plot(t,x,'l')
> spectrum(x)
```





The R command `spectrum` calculates the periodogram and automatically plots it against frequency.

There are three technical points we should briefly discuss (and some we won't but feel free to ask further questions if you have any):

- pre-processing of the data
- smoothing of the periodogram
- how to make R output better looking and give more intuitive estimates of the spectral density!

Preparing the Data for Analysis

Usually, we want to subtract the mean from the time series. Otherwise the periodogram and density spectrum will mostly be "overwhelmed" by a very large value for the first cosine coefficient (a_0). In R , the `spectrum` function goes further and automatically removes a linear trend from the series before calculating the periodogram. It seems appropriate to fit a trend and remove it if the existence of a trend in the underlying stochastic process is plausible. Although this will often be the case, there may be cases in which you prefer not to remove a fitted trend and this can be accomplished using `spec.pgram`, which gives the user more control over certain arguments.

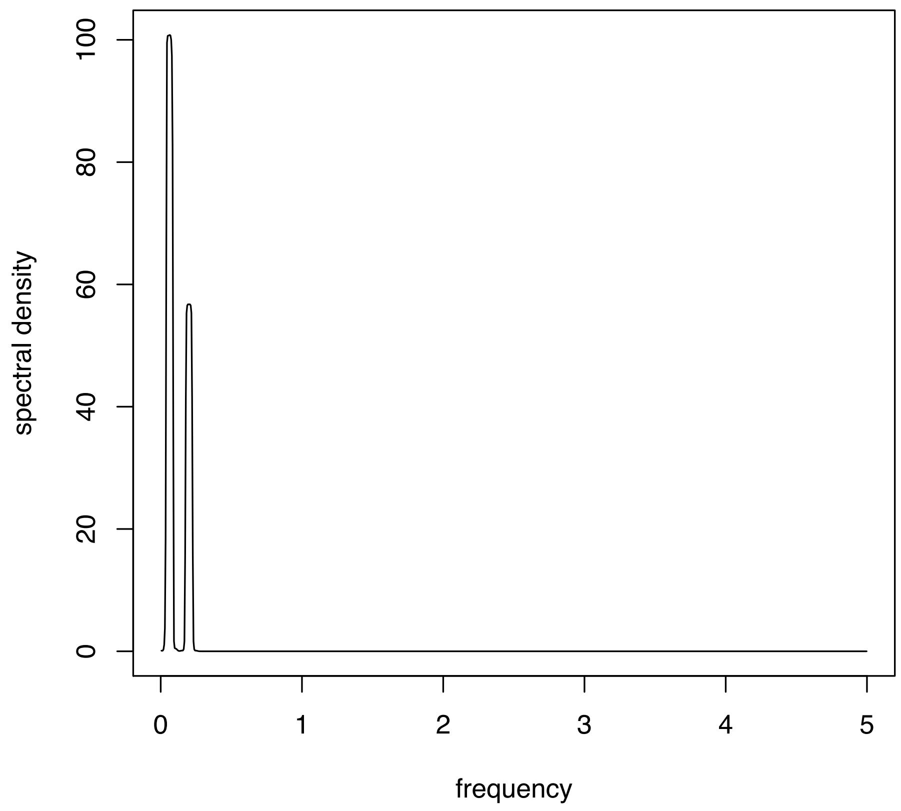
Smoothing

The periodogram distributes the variance over frequency, but it has two drawbacks. The first is that the precise set of frequencies is arbitrary, in as much as it depends on the record length. The second is that the periodogram does not become smoother as the length of the time series increases but just includes more spikes packed closer together. The remedy is to smooth the periodogram, and one way to do this is by using a smoothing kernel of spikes before joining the tips. The smoothed periodogram is also known as the sample spectrum. However, the smoothing will reduce the heights of peaks, and excessive smoothing will blur the features we are looking for. It is a good idea to consider spectra with different amounts of smoothing, and this is made easy for us with the R function `spectrum`. The argument `span` is the number of spikes in the kernel. An alternative method for computing a smoothed spectrum is to calculate the Fourier line spectrum for a number of shorter sub-series of the time series and average the line spectra of the subseries.

Spectral analysis in R

The `spectrum` function defaults to a logarithmic scale for the spectrum, but we can change this by setting the log parameter to ”no”. The default frequency axis is in cycles per sampling interval. It is more intuitive to convert the frequency axis to cycles per unit time, we can do this by extracting the frequency values that R returns and dividing by the length of the sampling interval. We should also multiply the spectral density by 2 so that the area under the periodogram actually equals the variance of the time series.

```
> del<-0.1 # sampling interval
> x.spec <- spectrum(x,log="no",span=10,plot=FALSE)
> spx <- x.spec$freq/del
> spy <- 2*x.spec$spec
> plot(spy~spx,xlab="frequency",ylab="spectral density",type="l")
```



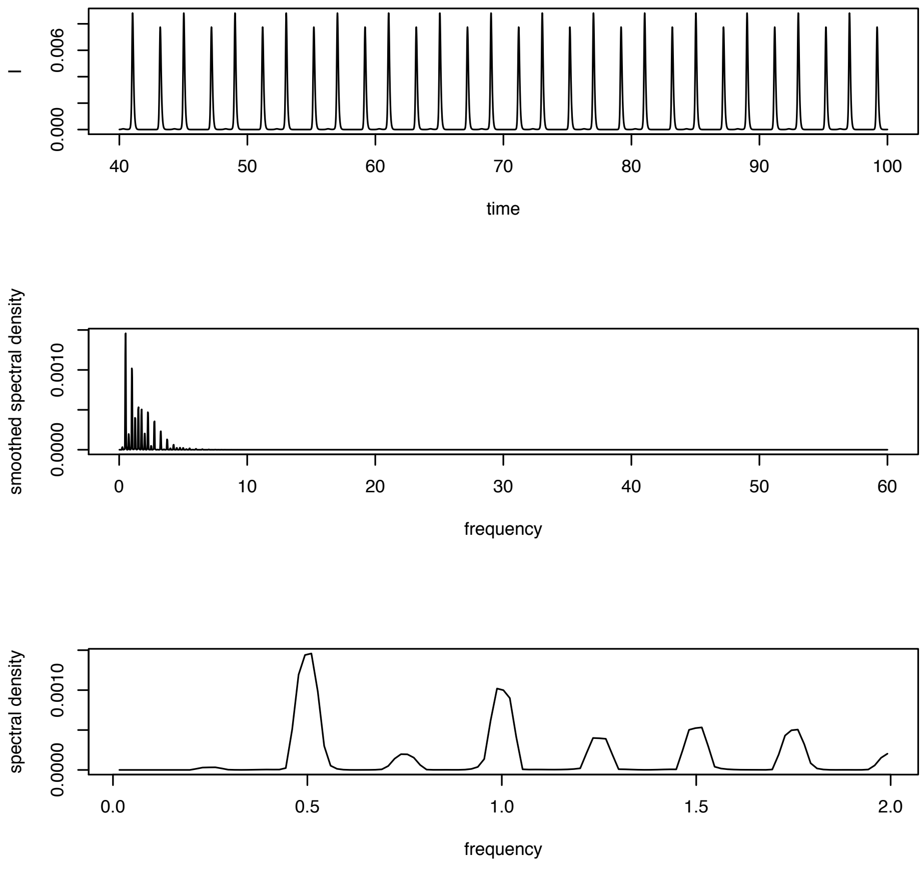
3 Assessing periodicity of model output

Let's now look at how all this works on simulated data. We will start by simulating the seasonal SIR model that was introduced yesterday. First, specify the model

```
> require(deSolve)
> seasonal.sir.model <- function (t, x, params) {
+   with(
+        as.list(c(x,params)),
+        {
+          beta <- beta0*(1+beta1*cos(2*pi*t))
+          dS <- mu*(N-S)-beta*S*I/N
+          dI <- beta*S*I/N-(mu+gamma)*I
+          dR <- gamma*I-mu*R
+          res <- c(dS,dI,dR)
+          list(res)
+        }
+        )
+ }
```

Then we simulate the model using `lsoda` and calculate the periodogram on the last part of the time series (after discarding transients).

```
> times <- seq(0,100,by=1/120)
> params <- c(mu=1/50,N=1,beta0=1000,beta1=0.4,gamma=365/13)
> xstart <- c(S=0.06,I=0.001,R=0.939)
> out <- as.data.frame(lsoda(xstart,times,seasonal.sir.model,params,rtol=1e-12,hmax=1/120))
> par(mfrow = c(3,1))
> plot(I~time,data=out,type='l',subset=time>=40)
> Iend<-subset(out,time>=40,select=c(I))
> del<-1/120
> x.spec <- spectrum(Iend,span=5,log="no",plot=FALSE)
> spx <- x.spec$freq/del
> spy <- 2*x.spec$spec
> plot(spy~spx,xlab="frequency",ylab="smoothed spectral density",type="l")
> plot (spy~spx, subset=spx<=2,xlab="frequency",ylab="spectral density",type = "l") #Zoom-in on low fre
> dom.freq=spx[which.max(spy)] #Extract the dominant frequency
```



Exercise 1. Explore the effects of changing amplitude of seasonality, β_1, on the periodicity of this model. Be careful to distinguish between transient and asymptotic dynamics. What happens if you log transform the simulated data and then apply the spectrum?

****Exercise 2.** Construct a figure that illustrates the relationship between β_1 and the dominant period of the output.

Stochastic model

As you saw yesterday, the dynamics of the deterministic SIR model without seasonality are damped oscillations toward an equilibrium. In the stochastic version, you probably saw a lot of extinction because the populations you looked at were small and there was no import parameter. Below is some code to simulate the stochastic SIR model using the tau-leap method for a population of 1 million and with a small import parameter ν. What you find is that demographic stochasticity amplifies the intrinsic oscillations of the system and we observe sustained cycles.

The tau-leap code describing the SIR model with births and deaths:

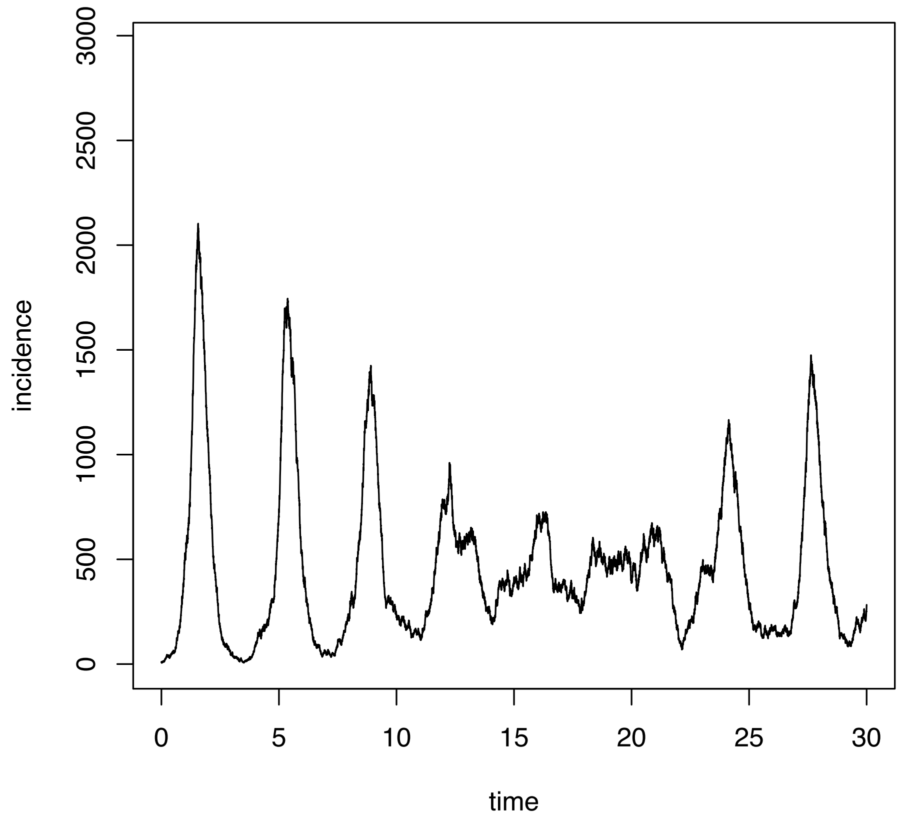
```
> sir.birth.death.onestep.tauleap <- function (x, params) {
+ S <- x[2]
+ I <- x[3]
+ R <- x[4]
+ N <- S+I+R
+ with(
+      as.list(params),
+      {
+        births<-min(S,rpois(1,mu*N*tau))
+        Sdeaths<-min(S,rpois(1,mu*S*tau))
+        Ideaths<-min(I,rpois(1,mu*I*tau))
+        Rdeaths<-min(R,rpois(1,mu*R*tau))
+        dSI <- min(S,rpois(1,beta*S*(I/N+nu)* tau))
+        dIR <- min(I,rpois(1,gamma*I * tau))
+        new.sir<-cbind(S +births-Sdeaths- dSI , I -Ideaths+ dSI - dIR, R -Rdeaths+ dIR)
+        cbind(tau,new.sir)
+        }
+      )
+ }
```

As before, we set parameters and loop through the process:

```
> sir.birth.death.model <- function (x, params, nstep) {
+   X <- array(dim=c(nstep+1,4))
+   colnames(X) <- c("time","S","I","R")
+   X[1,] <- x
+   for (k in 1:nstep) {
+     X[k+1,] <- x <- sir.birth.death.onestep.tauleap(x,params)
+   }
+   X
+ }
```

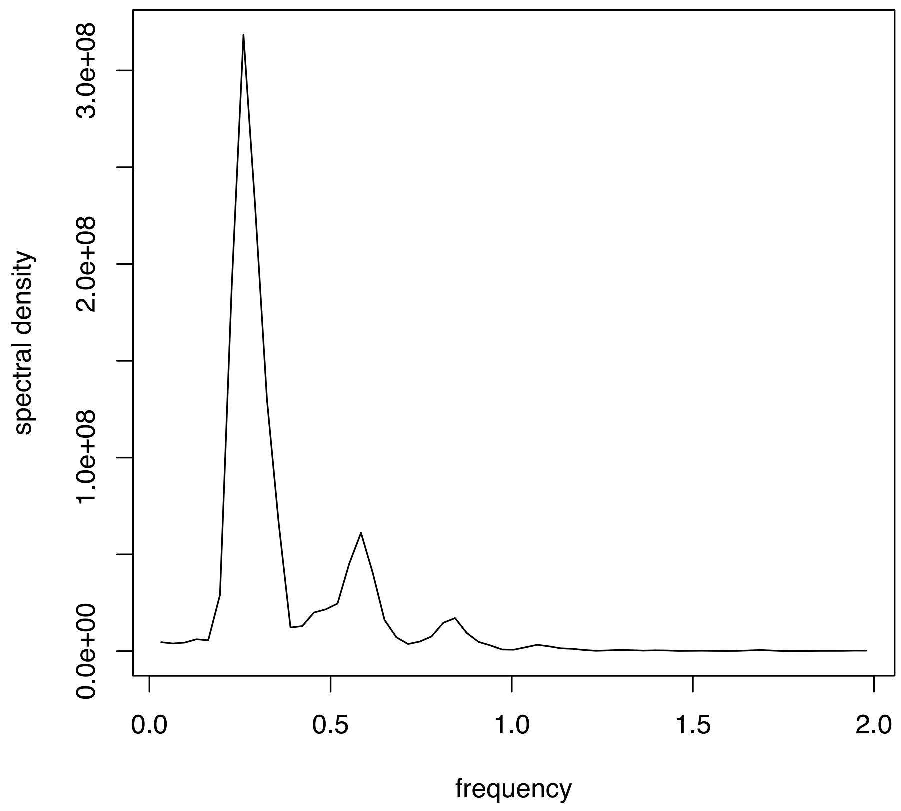
Now let's simulate and plot the resulting time series:

```
> set.seed(38499583)
> nsims <- 1
> pop.size <- 1000000
> I0 <- 10
> S0 <- round(0.1*pop.size)
> nstep <- round(30*365)
> xstart <- c(time=0,S=S0,I=I0,R=pop.size-I0-S0)
> params <- c(mu=0.014,beta=300,gamma=365/13,nu=0.000001,tau=1/365)
> x <- as.data.frame(sir.birth.death.model(xstart,params,nstep))
> x$cum.time <- cumsum(x$time)
> max.time<-max(x$cum.time)
> max.y<-1.4*max(x$I)
> plot(I~cum.time,data=x,xlab='time',ylab='incidence',col=1,
+      xlim=c(0,max.time),ylim=c(0,max.y),type='l')
```



Calculating the spectra:

```
> Iend<-subset(x,select=c(I))
> x.spec <- spectrum(Iend,span=3,log="no",plot=FALSE)
> spx <- x.spec$freq/params[5]
> spy <- 2*x.spec$spec
> plot (spy~spx, subset=spx<=2,xlab="frequency",ylab="spectral density",type = "l")
> dom.freq=spx[which.max(spy)]
```



We see that most of the variance of the time series is described by the low frequencies (long periods), as we would expect from looking at the simulated data.

4 Assessing periodicity of real data

Everything we have looked at in the context of spectral analysis and simulated data can also be applied to real data. The data that was plotted at the beginning of this tutorial from Copenhagen is available in the file ”Copenhagen.csv”.

Exercise 3. Use what you have learned to analyze the periodicity of the Copenhagen data. Read in the data. Choose one of the diseases and plot the time series. Calculate the spectrum. Can you uncover periodic patterns in the time series?

Probe matching

Statistics from spectral analysis (such as the dominant period) can be used to compare simulated time series to observed time series. This type of model fitting can be done using a variety of descriptive statistics, which are often referred to as ”probes”. The model most similar to the data, as measured by these probes, is considered to be the most likely candidate to represent the mechanism underlying the cycles **????**. Although such statistics are using only a subset of the information in the data, they are often good enough to distinguish between different dynamical regimes.

5 Other details and extensions

Confidence Intervals / Significance

Although the spectrum of a time series is innately useful for describing the distribution of variance as a function of frequency, sometimes we would like to know how the sample spectrum for a given time series differs from that of some known generating process. We would also like to assess the statistical significance of peaks in the spectrum. Significance can be evaluated only by reference to some standard of comparison. The question is ”significantly different than what?”. A standard for comparison is a null model, and is usually theoretically-based, but can be data-based. The simplest null model is white noise, which has an even distribution of variance over frequency. The white noise spectrum is consequently a horizontal line. Variance is not preferentially concentrated in any particular frequency range. However, in testing for significance of spectral peaks, white noise may be inappropriate. Positive autocorrelation in a time series can skew its frequency concentration toward the low-frequency side of the spectrum. One option for dealing with this is to use the theoretical spectrum of an autoregressive process as the null model.

Cross-spectrum

The cross-spectrum is an extension of spectral analysis to the simultaneous analysis of two time series. Briefly, the purpose of cross-spectral analysis is to uncover the correlations between two series at different frequencies. For example, disease incidence may be related to certain environmental variables. If we looked at the cross-spectrum of the two time series, we may find a periodicity in an environmental variable that is ahead ”in phase” of the disease cycles.

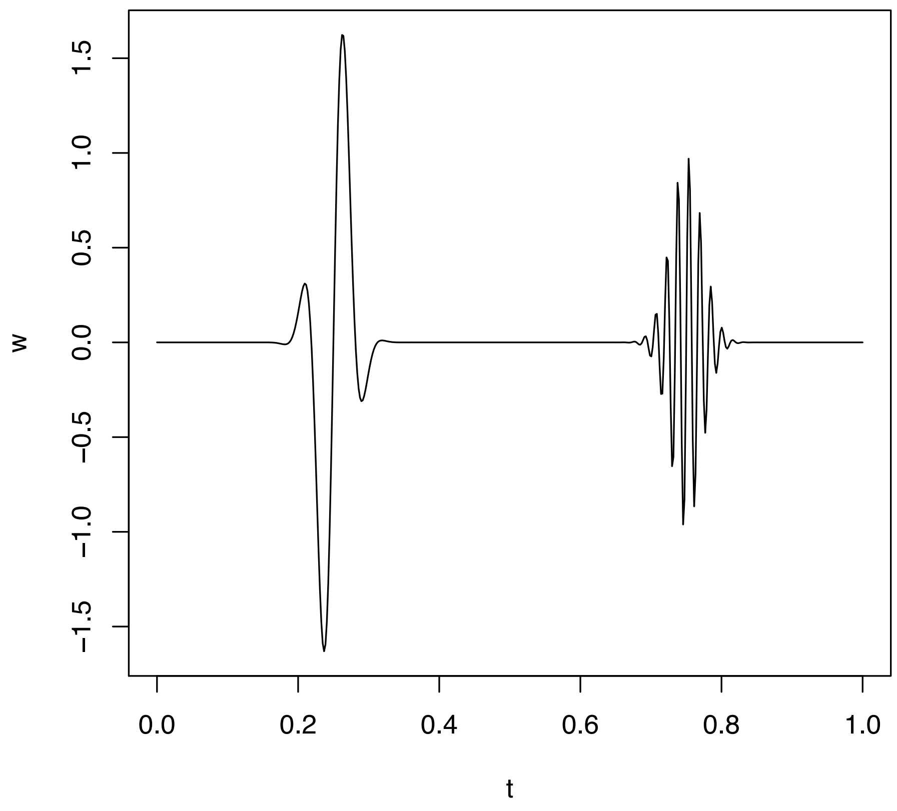
Nonstationarity and wavelets

Spectral analysis is appropriate for the analysis of stationary time series and for identifying periodic signals that are corrupted by noise. However, spectral analysis is not suitable for non-stationary applications, instead wavelets have been developed to summarize the variation in frequency composition through time.

To do wavelet analysis in `R` you will need to install the package `Rwave`.

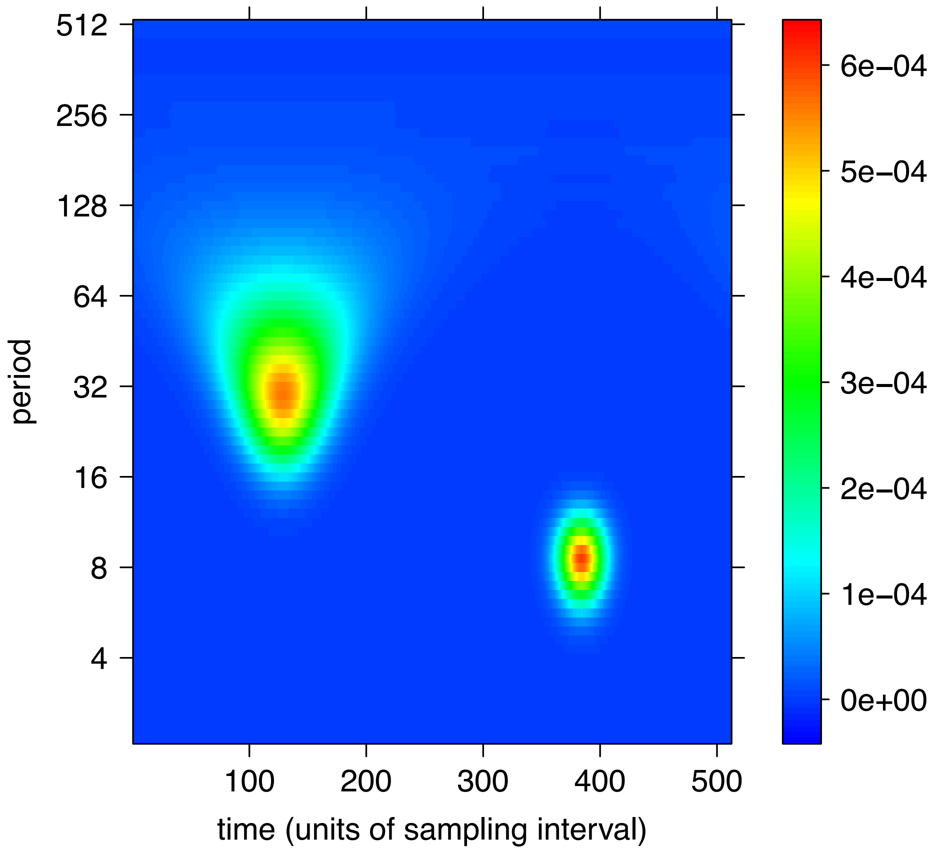
The following demonstrates a somewhat contrived example that illustrates the power of wavelet analysis.

```
> t = seq(0,1,len=512)
> w = 2 * sin(2*pi*16*t)*exp(-(t-.25)^2/.001)
> w= w + sin(2*pi*64*t)*exp(-(t-.75)^2/.001)
> w = ts(w,deltat=1/512)
> plot(t,w,'l')
```



Now for the wavelet transform of this time series (the functions in the file ”mk.cwt.R” help produce prettier graphs and are courtesy of Christian Gunning):

```
> require(Rwave)
> require(lattice)
> source("mk.cwt.R")
> tmp<-mk.cwt(w,noctave = floor(log2(length(w)))-1,nvoice=10)
> print(plot.cwt(tmp,xlab="time (units of sampling interval)"))
```



The intensity of the colormap represents the variance of the time series that is associated with particular frequencies (y-axis) through time (x-axis). As we can see, wavelet analysis is able to detect frequencies that are localized in time, and therefore if the dominant period of a time series changes over time, wavelets can be used to detect this transition.