Astrostats 2013 Lecture 2
Bayesian time series analysis and stochastic processes

C.A.L. Bailer-Jones
Max Planck Institute for Astronomy, Heidelberg
http://www.mpia.de/~calj/
Last updated: 2013-06-17 22:20

Contents

1 Key points

- Fitting time series model to data is not fundamentally different from fitting other types of data: the basic principle remains to define a generative model for the data, a likelihood, and a prior over the model parameters. Model comparison can be done as before using the evidence, K-fold cross-validation likelihood, or other methods.
- A fully general solution to fitting an arbitrary model to time series data with arbitrary noise models on both the signal and time axes is readily obtainable. It can be solved in general through numerical integration, although some common special cases or approximations render simpler solutions.
- Properly defined, the periodogram or power spectrum is closely related to the posterior probability density function over the frequency parameter of a sinusoidal model.
- A stochastic time series model is one in which the signal evolution itself has a non-deterministic component, even in the absence of measurement noise. A particularly useful model is the Ornstein–Uhlenbeck process, or damped random walk, which has been used to model Brownian motion.

2 Introduction

Bayesian data analysis is a general approach to modelling, and its principles and methods apply equally well to time series data. In this lecture I will consider just single variable time series, in which we are interested in the variation of a single quantity, z, with time, t. Given a set of J events, $D = \{t_j, z_j\}$, we are interested in one or more of the usual three questions:

1. Parameter estimation: Given a model M, which values of its parameters, θ, best describes the data? Or rather, what is the (multidimensional) posterior probability density function (PDF) $P(\theta|D, M)$?
2. Model comparison: Given a set of models $\{M_i\}$, which best explains the data? Or rather, what are the values of the model posterior probabilities $P(M_i|D)$?
3. Prediction: What value do we predict for the time series at some time t'? Or rather, what is $P(z|t = t')$?[1]

3 Sinusoidal model and the Bayesian periodogram

A general model for our data, $f(t)$, we can write as

$$z_j = f(t_j) + \epsilon(t_j) \tag{1}$$

where $\epsilon(t_j)$ is the noise model. A specific sinusoidal model we could write as

$$f(t) = A_1 \cos(\omega t) + A_2 \sin(\omega t) \tag{2}$$

which has three parameters $\theta = (A_1, A_2, \omega)$. (This is appropriate only if the data have zero mean, which we can arrange prior to modelling.) A typical noise model is a zero mean Gaussian, i.e.

$$P(z_j|\theta, M) \;=\; \frac{1}{\sigma_j\sqrt{2\pi}} \exp\left(-\frac{[z_j - f(t_j)]^2}{2\sigma_j^2}\right) \tag{3}$$

where the $\{\sigma_j\}$ could be parameters of the model which we determine via inference, or they may be given. Quite often the noise model is taken to be stationary, in which case we can replace these with a scalar σ, which I will now assume. If we additionally assume the measured times $\{t_j\}$ to be noise-free ("fixed"), then equation 3 is just the likelihood for one data point. Strictly speaking this likelihood should be written $P(z_j|t_j, \theta, M)$, but I make the dependence on the fixed times implicit, and now write the (noisy) measured data as $D = \{z_j\}$.

Assuming that the data have been measured independently, then the total likelihood is the product of the individual likelihoods

$$\begin{aligned} P(D|\theta, M) \;&=\; \prod_j \frac{1}{\sigma_j\sqrt{2\pi}} \exp\left(-\frac{[z_j - f(t_j)]^2}{2\sigma^2}\right) \\ &=\; \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^J \exp\Big(-\frac{1}{2\sigma^2}\sum_j [z_j - f(t_j)]^2\Big)\;. \end{aligned} \tag{4}$$

[1] Posed this way, we have averaged over all models. We might be instead interested in $P(z|t = t', M)$, which has marginalized over the parameters for one model, or even $P(z|t = t', \theta, M)$, for some "optimal" value of θ.

Inference of the model parameters proceeds in the usual way: we adopt a prior PDF and multiply this by the likelihood to get the unnormalized posterior. As this does not have an exact closed form in the θ, we may sample it using some Monte Carlo technique, then use density estimation to plot the resulting posterior PDF and estimate from this summary statistics such as the mean, mode and confidence intervals.

If we marginalize the posterior PDF over all parameters other than ω, then we end up with the one-dimensional (1D) posterior PDF $P(\omega|D, M)$, which is the Bayesian periodogram. This is already a probability density function, so unlike most other periodograms it does not need to be calibrated via (often dubious) significance tests.

Bayesian periodogram

3.1 An analytic Bayesian periodogram[2]

A few simple and not very restrictive approximations to the likelihood allow us to extract an analytic Bayesian periodogram from the above analysis. Expanding the square in equation 4 and substituting in equation 2 we can write the likelihood as

$$P(D|\theta, M) \propto \sigma^{-J} \exp\left(-\frac{G}{2\sigma^2}\right) \tag{5}$$

where I have dropped the numerical constant and where

$$\begin{aligned} G &= \sum_j z_j^2 + \sum_j f(t_j)^2 - 2\sum_j z_j f(t_j) \\ &= \sum_j z_j^2 + \sum_j f(t_j)^2 - 2[A_1 R(\omega) + A_2 I(\omega)] \end{aligned} \tag{6}$$

with

$$R(\omega) = \sum_j z_j \cos(\omega t_j) \tag{7}$$

$$I(\omega) = \sum_j z_j \sin(\omega t_j)\ . \tag{8}$$

Expanding the model term

$$\sum_j f(t_j)^2 = A_1^2 \sum_j \cos^2(\omega t_j) + A_2^2 \sum_j \sin^2(\omega t_j) + 2A_1 A_2 \sum_j \cos(\omega t_j)\sin(\omega t_j)\ . \tag{9}$$

When $J \gg 1$ this can be simplified using the following approximations

$$\sum_j \cos^2(\omega t_j) = \frac{J}{2} + \frac{1}{2}\sum_j \cos(2\omega t_j) \simeq \frac{J}{2} \tag{10}$$

$$\sum_j \sin^2(\omega t_j) = \frac{J}{2} - \frac{1}{2}\sum_j \cos(2\omega t_j) \simeq \frac{J}{2} \tag{11}$$

$$\sum_j \cos(\omega t_j)\sin(\omega t_j) = \frac{1}{2}\sum_j \sin(2\omega t_j) \ll \frac{J}{2}\ . \tag{12}$$

[2]This section follows the development in chapter 2 of Bretthorst (1998), with additional derivations and comments.

In each of these we have the sum of lots of positive and negative terms, and unless we happen to have $\omega t_j = n/2$ for integer n for all t_j, each of these sums is much less than 1, yielding the above approximations. This approximation also requires that $\omega t_j \gg 1$, otherwise we will have $\cos(2\omega t_j) \simeq 1 \; \forall \; i$. Thus in addition to requiring a reasonably large amount of data ($J \gg 1$), our approximation also requires that we have no low frequency variation in the data (the data have been detrended if necessary).

The consequence of these approximations is that

$$\sum_j f(t_j)^2 \simeq \frac{J}{2}(A_1^2 + A_2^2) \tag{13}$$

so we can write G in the likelihood (equation 5) as

$$G \simeq J\overline{z^2} + \frac{J}{2}(A_1^2 + A_2^2) - 2[A_1 R(\omega) + A_2 I(\omega)] \tag{14}$$

where

$$\overline{z^2} = \frac{1}{J}\sum_{j=1}^{j=J} z_j^2 \tag{15}$$

is the mean square of the data. This form for G will prove useful in the next step.

Often we are interested in the posterior PDF over just the frequency. To find this we must multiply the likelihood by the prior PDF and integrate over A_1 and A_2. For simplicity we adopt for both amplitudes a uniform prior which stretches from minus infinity to plus infinity.[3] Note that this is an improper prior, i.e. it does not have a finite integral. This is not a problem here, however, because we are integrating its product with a function which drops to zero as the magnitude of the amplitude increases (the likelihood), so the integral – which is just the integral of equation 5 – will converge. From equation 14 can write the integrand as

$$\exp\left(-\frac{G}{2\sigma^2}\right) = \exp\left(\frac{-J\overline{z^2}}{2\sigma^2}\right) \exp\left(-\frac{JA_1^2}{4\sigma^2} + \frac{R(\omega)A_1}{\sigma^2}\right) \exp\left(-\frac{JA_2^2}{4\sigma^2} + \frac{I(\omega)A_2}{\sigma^2}\right) \tag{16}$$

We make use of the following standard Gaussian integral for each integration

$$\int_{-\infty}^{+\infty} \exp\left(-ax^2 - bx\right) dx = \sqrt{\frac{\pi}{a}} \exp\left(\frac{b^2}{4a}\right) \quad (a > 0) \; . \tag{17}$$

Thus our integral becomes

$$\iint_{-\infty}^{+\infty} \exp\left(-\frac{G}{2\sigma^2}\right) dA_1 dA_2 = \exp\left(\frac{-J\overline{z^2}}{2\sigma^2}\right) 2\sigma\sqrt{\frac{\pi}{J}} \exp\left(\frac{R(\omega)^2}{J\sigma^2}\right) 2\sigma\sqrt{\frac{\pi}{J}} \exp\left(\frac{I(\omega)^2}{J\sigma^2}\right) \tag{18}$$

[3] We may think of this as being "uninformative" because it is uniform. This description is misleading, however, because a uniform distribution can be made non-uniform by various transformations of the parameter. A distribution uniform over A is not uniform over $\log A$. Is this still "uninformative"?

and so the marginalized posterior of ω and σ (implicitly also adopting uniform priors on these) is

$$P(\omega, \sigma|D, M) \propto \sigma^{2-J} \exp\left(-\frac{J\overline{z^2}}{2\sigma^2} + \frac{R(\omega)^2 + I(\omega)^2}{J\sigma^2}\right) . \tag{19}$$

If σ is given we can absorb the first term in the exponent into the proportionality constant and then have the posterior PDF over ω, the (Bayesian) periodogram

Periodogram for known noise

$$P(\omega|D, M) \propto \exp\left(\frac{W(\omega)}{\sigma^2}\right) \tag{20}$$

where

$$W(\omega) \;=\; \frac{1}{J}\left[R(\omega)^2 + I(\omega)^2\right] \;=\; \frac{1}{J}\left|\sum_{j=1}^{j=J} z_j e^{-i\omega t_j}\right|^2 . \tag{21}$$

is the Schuster periodogram. Arthur Schuster intuitively defined the periodogram as $W(\omega)$ 1905. We now see a rigorous basis for it in terms of fitting a single component sinusoidal model to data under some mild assumptions. But the Schuster periodogram itself ignores the noise in the data, whereas the Bayesian periodogram (posterior PDF) does not. Note that the Schuster periodogram is just the magnitude of a discrete Fourier transform of the data, and it is defined for any sampling of the data: nothing we have done demands uniform sampling.

Schuster periodogram

If σ is instead unknown, then by adopting a prior over σ, multiplying this by the right-hand-side of equation 19 (which is proportional to the likelihood) and then integrating over σ, we will get the posterior over ω for the case of unknown noise. A convenient prior is the Jeffrey's prior, $P(\sigma) = 1/\sigma$, which is again an improper prior (and is equivalent to a prior uniform in $\log\sigma$). We have

$$P(\omega|D, M) \;\propto\; Q \;=\; \int_0^\infty \sigma^{1-J} \exp\left(-\frac{J\overline{z^2}}{2\sigma^2} + \frac{W(\omega)}{\sigma^2}\right) d\sigma \tag{22}$$

Writing

$$a = \frac{J\overline{z^2}}{2} - W(\omega) \tag{23}$$

and making the substitution $x = 1/\sigma$, we can write this as

$$Q \;=\; \int_0^\infty x^{J-3} \exp\left(-ax^2\right) dx \; . \tag{24}$$

This is a standard integral, with result

$$Q \;=\; \frac{1}{2}\Gamma\Big(\frac{J-2}{2}\Big) a^{(2-J)/2} \quad (a > 0, J > 2) \; . \tag{25}$$

Absorbing the terms not involving a into the proportionality constant, we can write the posterior PDF over ω as

Periodogram for unknown noise

$$P(\omega|D, M) \;\propto\; \left(\frac{J\overline{z^2}}{2} - W(\omega)\right)^{(2-J)/2} \; . \tag{26}$$

The condition $a > 0$ implies $\frac{J\overline{z^2}}{2} > W(\omega)$. As the maximum value of $W(\omega)$ is $\overline{z^2}$, this condition is always met provided $J > 2$, a condition we already required earlier. The fact that $a > 0$ also ensures that the right-hand-side of equation 26 is always defined and is positive, a requirement for a probability density function.

In summary, we have derived analytic forms for the Bayesian periodogram for the case of constant noise standard deviation, where this is known (equation 20) and unknown (equation 26). These are valid for any time sampling. In both cases we have assumed that we have a reasonably large amount of data, and that the data are zero mean and lacking any low frequency trend. We will use them in the next section.

Bretthorst (1998) goes on to show (in his chapter 3) how the above can be extended to a model with multiple sinusoidal components, or indeed any other set of basis functions. He shows that the simplification to "analytic" forms for the PDF – similar to those derived above – can be achieved through a diagonalization of these basis functions (i.e. forming a new orthogonal basis set through linear combinations of the original set). This accommodates whatever time sampling we have in the data.

4 Parameter estimation with the sinusoidal model

The following R code shows how we can implement the ideas in the previous sections. It makes use of the functions defined in the appendix in section B. Plots generated by this code are shown in Figures 1 to 4.

The parameters are A_1, A_2, and ν, where $\omega = 2\pi\nu$. There are called `theta` in the code.

Concerning the priors, I adopt Gaussian distributions for the two amplitude parameters, with zero mean, and standard deviations on the scale of the amplitude of the data. As the frequency must be positive, I use a gamma distribution for its prior. The gamma distribution is characterized by two parameters, shape and scale. I adopt a shape parameter greater than 1 to ensure that very low frequencies are suppressed (I use 1.5), and a scale parameter which yields non-vanishing probabilities for the highest frequencies I expect. In real problems, we are likely to have some idea of this from the nature of the phenomenon, or from what the measurements may have been sensitive to. Thus in this implementation there are three "hyperparameters" (parameters of the prior PDF), the Gaussian standard deviation for each of A_1 and A_2 and the gamma scale for frequency. There are called `alpha` in the code.

R file: `Rcode/exp_sinusoidal.R`

```
# C.A.L. Bailer-Jones
# Astrostats 2013
# This file: exp_sinusoidal.R
# R code for experimenting with the sinusoidal model

source("sinusoidal.R")
source("monte_carlo.R") # provides metrop() and make.covariance.matrix()

set.seed(200)

# Generate a time series from a sinusoidal model
```

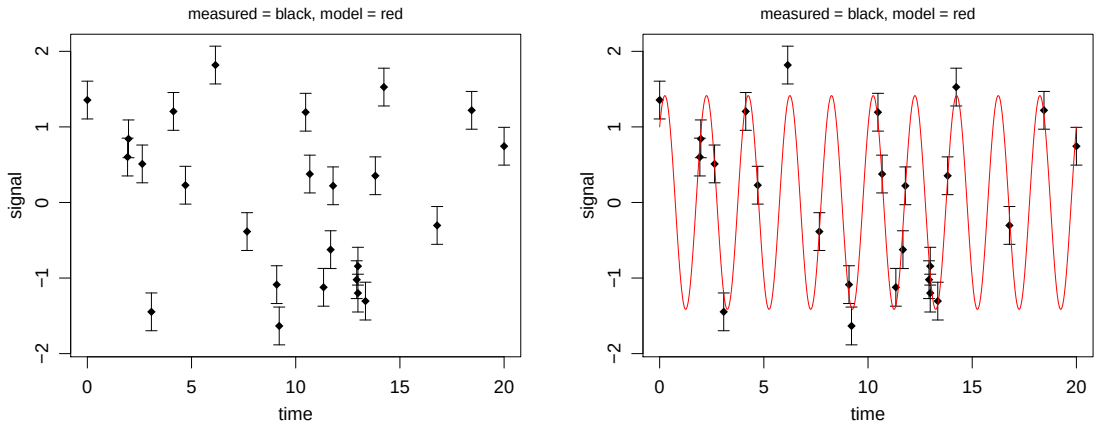

Figure 1: Data set (black) generated from the sinusoidal model (red), equation 2. The true parameters are $A_1 = 1$, $A_2 = 1$, $\omega = 2\pi * 0.5$. The same data are shown in both panels.

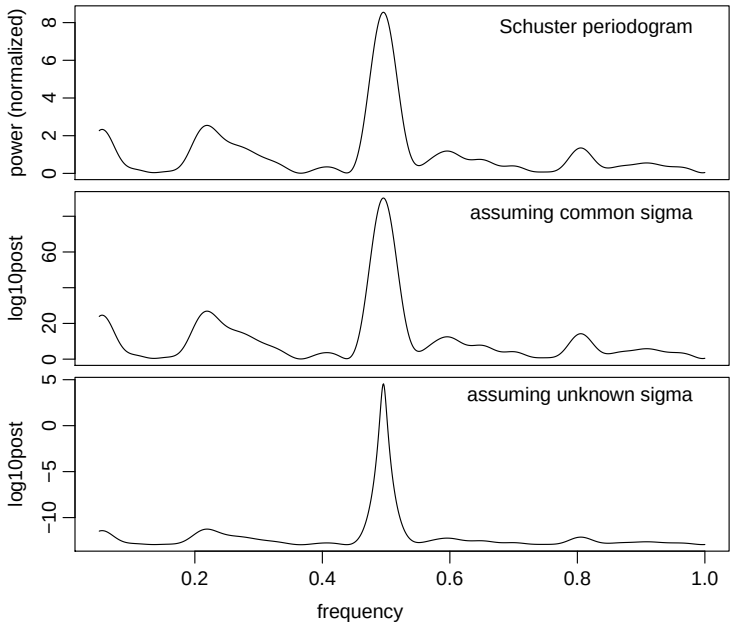

Figure 2: Periodograms of the data shown in Fig. 1: Schuster periodogram (top; equation 21), the logarithm of the (unnormalized) posterior PDF over frequency (i.e. Bayesian periodogram) for the case of known noise standard deviation (middle; equation 19) and unknown noise standard deviation (bottom; equation 26). Because the posteriors are shown on a logarithmic scale, they correspond to a much narrower peak and thus a much better determined frequency than the plain periodogram.

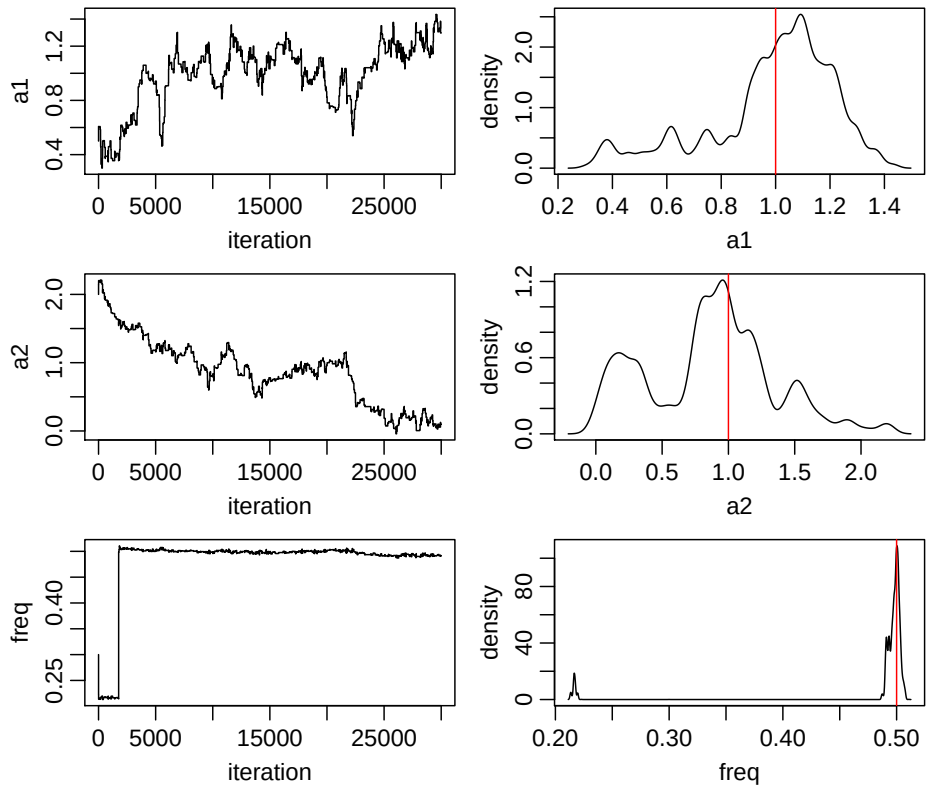

Figure 3: Example of a posterior PDF sampling of the sinusoidal model for the data shown in Fig. 1. The panels in the left column show the chains for the three parameters, those in the right the 1D PDFs generated from the samples via density estimation. The vertical red lines indicate the true parameters. In order to test the convergence I intentionally initialized the MCMC far further from the true values than one could estimate by eyeballing the data (for the amplitudes) or from the Schuster periodogram. We should therefore reject the early samples in the chain (the "burn-in"). The chains shown here are not particularly good because the acceptance rate was rather low (around 0.02). Note that the frequency is nonetheless well determined – by which I mean a narrow PDF, not proximity to the true values – whereas the amplitudes are less well determined.

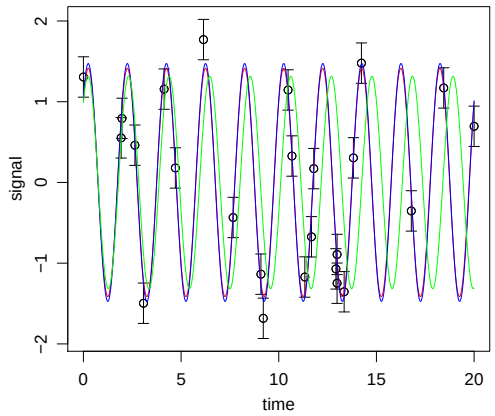

Figure 4: Predicted data at the mode (in blue) and mean (in green) of the posterior PDF in Figure 3. Overplotted are the data (black points) and true model (in red).

```
sinmod <- list(a1=1, a2=1, freq=0.5)
eventTimes <- sort(c(0, 20, runif(n=23, min=0, max=20)))
ysigma <- 0.25
obsdata <- gen.sinusoidal(sinmod=sinmod, eventTimes=eventTimes, ysigma=ysigma)

# Plot data and overplot with true sinusoidal model
oplot.obsdata.sinusoidal(obsdata=obsdata, sinmod=sinmod, ploterr=TRUE, fname="sindat1.pdf")
oplot.obsdata.sinusoidal(obsdata=obsdata, sinmod=NULL,   ploterr=TRUE, fname="sindat2.pdf")

# Model assumes data are zero mean
obsdata[,3] <- obsdata[,3] - mean(obsdata[,3])

# Calculate and plot Schuster periodogram and derived posterior PDFs over frequency
schuster(obsdata=obsdata, minFreq=0.05, maxFreq=1, NFreq=1e3, fname="schuster.pdf")

# For the following inference we sample over the three parameters
# theta = c(a1, a2, freq).  The hyperparameters of the prior PDFs are
# alpha = c(sd_a1, sd_a2, scale_freq), with mean_a1, mean_a2, and
# shape_freq fixed in the definitions of the functions.
alpha <- c(1, 1, 1) # c(sd_a1, sd_a2, scale_freq)

# Use MCMC to sample posterior for sinusoidal model: c(a1, a2, freq)
sampleCov <- make.covariance.matrix(sampleSD=c(0.05, 0.05, 0.1), sampleCor=0)
thetaInit <- c(0.5, 2, 0.3)
postSamp <- metrop(func=logpost.sinusoidal, thetaInit=thetaInit, Nburnin=0,
                   Nsamp=3e4, verbose=1e3, sampleCov=sampleCov,
                   obsdata=obsdata, alpha=alpha)

# Plot MCMC chains and use density estimation to plot 1D posterior PDFs from these.
thetaTrue <- as.numeric(sinmod)
parnames <- c("a1", "a2", "freq")
pdf("sindat_mcmc.pdf", width=7, height=6)
par(mfrow=c(3,2), mar=c(3.0,3.0,0.5,0.5), oma=c(1,1,1,1), mgp=c(1.8,0.6,0), cex=1.0)
for(p in 3:5) { # columns of postSamp
  plot(1:nrow(postSamp), postSamp[,p], type="l", xlab="iteration", ylab=parnames[p-2])
```

```
  postDen <- density(postSamp[,p], n=2^10)
  plot(postDen$x, postDen$y, type="l", xlab=parnames[p-2], ylab="density")
  abline(v=thetaTrue[p-2], col="red")
}
dev.off()

# Find MAP solution and mean solutions and overplot on data
posMAP     <- which.max(postSamp[,1]+postSamp[,2])
(thetaMAP  <- postSamp[posMAP, 3:5])
(thetaMean <- apply(postSamp[,3:5], 2, mean)) # Monte Carlo integration
locplot <- function(theta, col) {
  tsamp <- seq(from=min(eventTimes), to=max(eventTimes), length.out=1e3)
  omegat <- 2*pi*theta[3]*tsamp
  modpred <- theta[1]*cos(omegat) + theta[2]*sin(omegat)
  lines(tsamp, modpred, col=col, lw=1)
}
pdf("sindat_fits.pdf", width=6, height=5)
par(mfrow=c(1,1), mgp=c(2.0,0.8,0), mar=c(3.5,4.0,1.0,1.0), oma=c(0,0,0,0), cex=1.2)
plotCI(obsdata[,1], obsdata[,3], xlab="time", ylab="signal", uiw=obsdata[,4], gap=0)
locplot(theta=as.numeric(sinmod), col="red")
locplot(theta=thetaMAP,  col="blue")
locplot(theta=thetaMean, col="green")
dev.off()
```

5 A general method for time series modelling[4]

In section 3 I made two implicit assumptions. First, I assumed that the process itself was deterministic. Randomness arose only on account of measurement noise. Yet some processes are intrinsically random (or stochastic), even if there is no measurement noise. These are discussed further in section 6. Second, I assumed that the times at which the data were obtained were subject to no uncertainty. This is not true in general, and often not in practice, e.g. astronomical ages derived from photometric redshifts or stellar clusters, or geological ages from the fossil record.

For this more general case we first need a more general notation. Let t and z now correspond to the *true* time and signal (respectively), not to the *measured* values. Instead, for each event j, our measurement of its time of the event, t_j, is s_j with a standard deviation (estimated measurement uncertainty) σ_{s_j}, and our measurement of the signal of the event, z_j, is y_j with a standard deviation (estimated measurement uncertainty) σ_{y_j}.

For shorthand I write $D_j = (s_j, y_j)$, the (noisy) data for one event, and $\sigma_j = (\sigma_{s_j}, \sigma_{y_j})$. The *measurement model* (or noise model) describes the probability of observing the measured values for a single event given the true values and the estimated uncertainties: it gives $P(D_j|t_j, z_j, \sigma_j)$. The σ_j are considered fixed parameters of the measurement model.

M is a stochastic *time series model* with parameters θ. It specifies $P(t_j, z_j|\theta, M)$, the probability of observing an event at time t_j with signal z_j.

To do inference we need to derive the likelihood, $P(D|\sigma, \theta, M)$, where $D = \{D_j\}$ and $\sigma =$

[4]This section is an introduction to the method explained in detail in Bailer-Jones (2012).

$\{\sigma_j\}$. (We can also write this as $P(D|\theta, M)$ because σ is fixed.) We will see below how to derive it in terms of the measurement model and time series model, but let's first examine these a little more.

5.1 Measurement model

If t and z have no bounds and the measurement uncertainties are standard deviations, then an appropriate choice for the measurement model is a two-dimensional Gaussian in the variables (s_j, y_j) for event j. If we assume no covariance between the variables then this reduces to the product of two 1D Gaussians

$$P(D_j|t_j, z_j, \sigma_j) = \frac{1}{\sqrt{2\pi}\sigma_{s_j}} e^{-(s_j - t_j)^2/2\sigma_{s_j}^2} \; \frac{1}{\sqrt{2\pi}\sigma_{y_j}} e^{-(y_j - z_j)^2/2\sigma_{y_j}^2} \; . \tag{27}$$

(The two terms are normalized with respect to s_j and y_j respectively.) If we had other information about the measurement, e.g. asymmetric error bars, strictly positive signals, or uncertainties which are not standard deviations, then we should adopt a more appropriate distribution.

5.2 Time series model

We can write the time series model as the product of two components

$$P(t_j, z_j|\theta, M) \; = \; P(z_j|t_j, \theta, M) P(t_j|\theta, M) \tag{28}$$

which I will refer to as the signal and time components respectively.

The process may be subject to a stochastic component (which has nothing to do with measurement noise). In that case it is useful to express the signal component itself using two independent subcomponents: (1) the stochastic model of the variation in the signal, z; (2) a deterministic function which defines the time-dependence of its mean.

The stochastic subcomponent might be a Gaussian

$$P(z_j|t_j, \theta, M) \; = \; \frac{1}{\sqrt{2\pi}\lambda} e^{-(z_j - \eta[t_j])^2/2\lambda^2} \tag{29}$$

where $\theta = (\eta, \lambda)$ are the "parameters" of the distribution: $\eta[t_j]$ is the expected true signal at true time t_j; λ^2 is a parameter which reflects the degree of stochasticity (in this case the variance) of the process.

For a sinusoidal process, the deterministic subcomponent can be written as before

$$\eta \; = \; A_1 \cos(\omega t) + A_2 \sin(\omega t) \tag{30}$$

which has parameters (A_1, A_2, ω).

The final term in equation 28, $P(t_j|\theta, M)$, reflects the stochasticity in the event times themselves. This is used to model the probability distribution over the times of events which are considered non-deterministic in the problem context, such as supernovae or asteroid impacts. If there is no stochasticity in the event times, then a uniform distribution should be adopted.

5.3 Likelihood

The probability of observing data D_j from time series model M with parameters θ when the uncertainties are σ_j, is $P(D_j|\sigma_j, \theta, M)$, the *event likelihood*. This is obtained by marginalizing over the true, unknown event time and signal

$$\begin{aligned} P(D_j|\sigma_j, \theta, M) &= \iint_{t_j, z_j} P(D_j, t_j, z_j|\sigma_j, \theta, M)\, dt_j dz_j \\ &= \iint_{t_j, z_j} P(D_j|t_j, z_j, \sigma_j, \theta, M) P(t_j, z_j|\sigma_j, \theta, M)\, dt_j dz_j \\ &= \iint_{t_j, z_j} \underbrace{P(D_j|t_j, z_j, \sigma_j)}_{\text{Measurement model}}\ \underbrace{P(t_j, z_j|\theta, M)}_{\text{Time series model}}\, dt_j dz_j \end{aligned} \tag{31}$$

where the time series model and its parameters drop out of the first term because D_j is independent of this once conditioned on the true variables, and the measurement model (via σ_j) drops out of the the second term because it has nothing to do with the predictions of the time series model. For specific, but common, situations, this two-dimensional integral can be approximated by a 1D integral or even a function evaluation.

If we have a set of J events for which the ages and signals have been estimated independently of one another, then the probability of observing all these, the *likelihood*, is

$$P(D|\sigma, \theta, M) = \prod_j P(D_j|\sigma_j, \theta, M)\ . \tag{32}$$

Armed with the likelihood for our given measurement model, time series model, and measured data, and adopting suitable priors over the model parameters, we can now calculate all the usual things we need for Bayesian inference (posterior PDFs, evidences etc.), typically using MCMC methods to sample the likelihood. Specific models and an application to brown dwarf light curves can be found in Bailer-Jones (2012). This also makes use of stochastic models, which we now turn to.

6 Stochastic processes

A stochastic process is one in which some aspect of the system evolves randomly. Normally we do not take this definition to include the effects of measurement noise, as then almost all observed processes would appear stochastic. Rather, we take stochastic to mean that the evolution of the system itself has a non-deterministic component.

If $\{z_j\}$ is the state of a stochastic system at times $\{t_j\}$, then the multidimensional PDF $P(\{z_j\}, \{t_j\})$ completely describes the system. The conditional PDF of future events ($j > 0$) in terms of past events can then be written

$$P(z_{j+1}, t_{j+1}; z_{j+2}, t_{j+2}; \ldots | z_0, t_0; z_{j-1}, t_{j-1}; \ldots) = \frac{P(\{z_j\}, \{t_j\})}{P(z_0, t_0; z_{j-1}, t_{j-1}, \ldots)} \tag{33}$$

where for convenience I've adopted the notation such that $\ldots \geq t_{j+1} \geq t_j \geq t_{j-1} \geq \ldots$. A simple stochastic system is one in which all events are independent,

$$P(\{z_j\}, \{t_j\}) = \prod_j P(z_j, t_j) \tag{34}$$

an example of which is a pure noise process.

In this section I will use the notation $\mathcal{N}(x;\mu,V)$ to indicate a random number drawn from a Gaussian PDF with mean μ and variance V. The x indicates that a different random number is drawn for every x, i.e. $\mathcal{N}(x;\mu,V)$ is statistically independent of $\mathcal{N}(x';\mu,V)$ if $x \neq x'$. $\mathcal{N}(x)$ is a shorthand for $\mathcal{N}(x;\mu=0,V=1)$.

6.1 Markov processes

A Markov process is a specific type of stochastic process in which the future value of the state variable is independent of the past values conditioned on the present value, i.e.

$$P(z_{j+1},t_{j+1};z_{j+2},t_{j+2};\ldots|z_0,t_0;z_{j-1},t_{j-1};\ldots) \;=\; P(z_{j+1},t_{j+1};z_{j+2},t_{j+2};\ldots|z_0,t_0)\;. \quad (35)$$

Such "one step memory" processes are useful in practice, because a chain of past dependencies can then be written as a probability conditional on only one previous time step, e.g.

$$P(z_1,t_1;z_2,t_2|z_0,t_0) \;=\; P(z_2,t_2|z_1,t_1)P(z_1,t_1|z_0,t_0) \quad (36)$$

Markov processes are widely applicable, because many stochastic processes can be reduced to a Markov process. For example, if we had a "two step memory" process, we could convert it into two Markov processes.

6.2 Describing continuous stochastic processes (Langevin equation)

A general first order differential equation which describes the evolution of a continuous state variable $z(t)$ is

$$\frac{dz}{dt} \;=\; A(z,t)\;. \quad (37)$$

If A is a deterministic function then z is a deterministic process. If we want the differential equation to describe a stochastic process, then we may add to the right-hand-side of this a stochastic term. The differential equation is then referred to as a Langevin equation. It turns out that in order to keep this equation self-consistent, we are limited in what kind of stochastic term we can add. If $z(t)$ is to describe a *continuous* Markov process, then the following three conditions must apply, with $dz = z(t+dt) - z(t)$,

1. dz depends only on z, t and dt (this makes it Markov),
2. dz depends smoothly on z, t and dt,
3. z is continuous in the sense that $dz \to 0$ as $dt \to 0$ for all t and z.

It turns out that the general form of the Langevin equation which satisfies this is

$$dz \;=\; A(z,t)dt + \mathcal{N}(t)\sqrt{D(z,t)dt} \quad (38)$$

where A and D are any smooth functions and D is non-negative. A and D are sometimes called the *drift* and *diffusion* terms respectively. The presence of the second term on the right-hand-side makes this equation a stochastic differential equation. The square-root of dt may be unfamiliar. We could write the second term instead as $\mathcal{N}(t;0,dt)\sqrt{D(z,t)}$, in which $\mathcal{N}(t;0,dt)$ is a Gaussian random variable with variance dt.

6.3 Ornstein–Uhlenbeck process

The Ornstein–Uhlenbeck (OU) process is a particular but widely-used description of a stochastic process. It has a Langevin equation with constant drift and diffusion terms

$$dz = -\frac{1}{\tau}z dt + c^{1/2}\mathcal{N}(t;0,dt) \quad (t>0) \tag{39}$$

where τ and c are positive constants, the *relaxation time* (dimension t) and the *diffusion constant* (dimension z^2t^{-1}) respectively. The OU process can also be seen as the continuous-time analogue of the discrete-time AR(1) (autoregressive) process, and so is sometimes referred to as the (or a) CAR(1) process. It turns out that in the context of Brownian motion, $z(t)$ is a good description of the velocity of the particle.

The OU process is stationary, Gaussian and Markov.[5] The solution of equation 39 is the PDF of $z(t)$ given $z_0 = z(t{=}t_0)$ for any $t > t_0$

$$\begin{aligned} P(z|t,z_0,t_0) &= \mathcal{N}(z;\mu_z,V_z) \quad \text{where} \\ \mu_z &= z_0\upsilon \\ V_z &= \frac{c\tau}{2}(1-\upsilon^2) \\ \upsilon &= e^{-(t-t_0)/\tau} \ . \end{aligned} \tag{40}$$

The relaxation time determines the time scale over which the mean and variance change. The diffusion constant determines the amplitude of the variance. The OU process $z(t)$ is a mean-reverting process: for $t - t_0 \gg \tau$ the mean tends towards zero and the variance asymptotes to $c\tau/2$ (for finite τ).

It follows from equation 40 that $z(t)$ is the sum of $z_0\upsilon$ and a random number drawn from a zero-mean Gaussian with time-dependent variance. Thus given the event $z_{j-1}(t_{j-1})$, we can write down an update equation (or generative model) for the state at the next time step, $z_j(t_j)$, as

update equation for OU process

$$z_j = z_{j-1}\upsilon + \mathcal{N}(z)\sqrt{V_z} \tag{41}$$

where now

$$\upsilon = e^{-(t_j - t_{j-1})/\tau} \tag{42a}$$

$$V_z = \frac{c\tau}{2}(1-\upsilon^2) \ . \tag{42b}$$

For a given sequence of time steps, $(t_0, t_1, \ldots)$, we can use this to simulate an OU process. Because the time series is stochastic and must be calculated at discrete steps, then even for a fixed random number seed, the generated time series depends on the actual sequence of steps. Some examples are shown in Fig. 5.

We can then show that z_j has a Gaussian distribution with mean and variance

$$\mu[z_j] = \mu[z_{j-1}]\upsilon \tag{43a}$$

$$V[z_j] = V[z_{j-1}]\upsilon^2 + V_z \tag{43b}$$

[5] It is actually the only stochastic process which has these three properties.

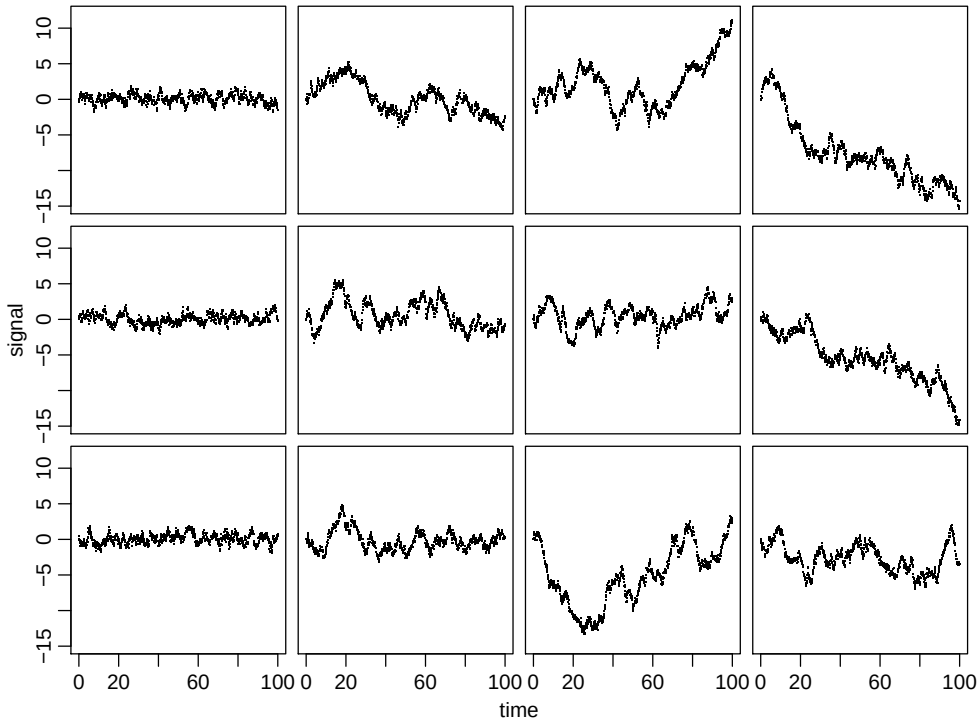

Figure 5: Examples of time series generated from the OU process using the update equation (equation 41). The columns from left to right have $\tau = (1, 10, 100, 1000)$ respectively. The rows differ only in the random number sequence used in the update equation. The other parameters are fixed in all cases to $c = 1$, $\mu[z_1] = 0$, $V[z_1] = 0$. All time series are calculated with the same uniform time step (0.1) and all panels have the same scales. Plot generated by R code `Rcode/plot_OUprocess.R`.

respectively. This is subtly different from the update equation, because it concerns the expected distribution of z_j in terms of the expected distribution at the previous time step. It has an additional variance term beyond V_z, because the uncertainty (variance) in z_j is increased by the uncertainty in z_{j-1}.

6.4 Wiener process

The Wiener process can be considered as a special case of the OU process in which the relaxation time scale is infinite, $\tau \to \infty$ in equation 39, i.e. there is no damping. Equivalently $A = 0$ in equation 38. With $\upsilon = e^{-\Delta t/\tau}$, $\upsilon \to 1$ and[6] $\tau(1 - \upsilon^2) \to (\Delta t)^2$. The PDF of $z(t)$

[6]Using a Taylor expansion

$$\begin{aligned}
\tau(1 - e^{-2(\Delta t)/\tau}) &= \tau\left[1 - \left(1 - \frac{2\Delta t}{\tau} + \frac{1}{2!}\left(\frac{2\Delta t}{\tau}\right)^2 - \ldots\right)\right] \\
&= \tau\left[\frac{2\Delta t}{\tau} - \frac{1}{2!}\left(\frac{2\Delta t}{\tau}\right)^2 + \ldots\right] \\
&= 2\Delta t \;\text{ as }\; \tau \to \infty\,.
\end{aligned}$$

remains Gaussian as in equation 40, but the mean and variance become (with $\Delta t = t - t_0$)

$$\begin{aligned} \mu_z &= z_0 \\ V_z &= c(t - t_0) \ . \end{aligned} \tag{44}$$

There is no damping, so the variance of the state variable grows monotonically with time, and this is not a stationary process. Yet as the expectation value of the state variable is constant, this Wiener process is referred to as being *driftless*. The Wiener process is sometimes taken to describe (in a somewhat idealized form) the position of a particle undergoing Brownian motion.

6.5 Historical note

The Langevin equation and the Wiener and OU processes all evolved from various approaches to modelling Brownian motion. Einstein modelled Brownian motion with a kinematic approach, solving a partial differential equation of the PDF of the particle position (what we today would call a special case of the Fokker-Planck equation). Langevin instead adopted a dynamic approach, applying Newton's second law to a typical particle, and writing

$$\frac{dz}{dt} = \beta z + \eta(t) \ . \tag{45}$$

where z is the velocity of the particle, with constant β and where the stochastic force, $\eta(t)$, is subject to $\langle \eta(t) \rangle = 0$ and $\langle \eta(t)\eta(t') \rangle = c\,\delta(t - t')$ for some constant c. Einstein and Langevin essentially arrived at the same result by different methods, but both approaches had issues concerning the assumption and approximations. Ornstein and Uhlenbeck generalized and improved upon them.

7 Modelling the OU process

This is a little technical. An qualitative summary is given in section 8

Given a set of data $D = \{D_j\}$ with measurement uncertainties $\sigma = \{\sigma_j\}$, we can in principle calculate the likelihood and thus posterior PDF over the model parameters of a stochastic process. This is, however, not quite so straight forward even for a Markov process, because the time series model, $P(z_j, t_j|\theta, M)$, which appears in the equation for the likelihood (equation 32), now depends on the PDF at the previous time step. A relatively simple solution can nonetheless be obtained if we can assume uncertainties on the times to be negligible, and if both the signal measurement model and the stochastic part of the time series model are Gaussian. The latter is the case for the OU process. The result is a recurrence relation in which each event likelihood is a Gaussian distribution with mean and variance depending on both the signal uncertainties for the present and previous events and the variance of the OU process (which depends on the time step size). A full explanation and derivation can be found in section A.2 of Bailer-Jones (2012). Below I just give the result, which we will then use to analyse some data.

Starting from equation 31, making the above assumptions, and using the properties of the OU process, we find that we can write the event likelihood as a Gaussian

$$\begin{aligned} P(D_j|\sigma_j,\theta,M) &= \frac{1}{T_2-T_1}\int_{z_j} P(y_j|z_j,\sigma_{y_j})P(z_j|t_j,\theta,M)\,dz_j \\ &= \frac{1}{T_2-T_1}\frac{1}{\sqrt{2\pi V[y_j]}}\exp\left(\frac{-(y_j-\mu[y_j])^2}{2V[y_j]}\right) \quad (T_1<t_j<T_2) \end{aligned} \tag{46}$$

which has mean and variance

$$\mu[y_j] = 0 \quad +\mu[z_j] \tag{47a}$$

$$V[y_j] = \sigma_{y_j}^2 + V[z_j] \tag{47b}$$

respectively, with $\mu[z_j]$ and $V[z_j]$ given by equation 43. For simplicity I have additionally assumed that the time component of the time series model, $P(t_j|\theta,M)$, is uniform from T_1 to T_2.

As this is a Markov process, the likelihood depends on the previous time step through $\mu[z_{j-1}]$ and $V[z_{j-1}]$ appearing in equation 43. These we must estimate using the data at that previous time step. How, for any time step j (such as $j-1$), do the values of $\mu[z_j]$ and $V[z_j]$ depend on y_j and σ_{y_j}? This is equivalent to asking what is the posterior PDF over z_j using both the data at t_j and the prior value of z_j (which is the posterior from the previous time step). This is given by Bayes theorem. It turns out that this posterior PDF over z_j is a Gaussian with mean and variance

$$\mu'[z_j] = \frac{y_j V[z_j] + \mu[z_j]\sigma_{y_j}^2}{V[z_j]+\sigma_{y_j}^2} \tag{48a}$$

$$V'[z_j] = \frac{V[z_j]\sigma_{y_j}^2}{V[z_j]+\sigma_{y_j}^2} \tag{48b}$$

respectively, where the prime symbol is used to distinguish these posterior moments from the prior ones in equation 43. It is these quantities which we then use at the *next* event as the estimates of the mean and variance of z. Thus, at iteration (event) j, when we evaluate equation 47)and hence the likelihood), we use $\mu'[z_{j-1}]$ and $V'[z_{j-1}]$ as our estimates of $\mu[z_{j-1}]$ and $V[z_{j-1}]$. This is how we introduce a dependence on the previous measurement (the Markov property). We then calculate the mean and variance of the posterior for z_j using equation 48, and will then use these in the next iteration. The result is a recurrence relation for the posterior PDF of z_j. At each time step we can siphon off the relevant quantities in order to calculate the event likelihood (equation 46).

To initialize the process we must specify initial values $\mu[z_1]$ and $V[z_1]$. We use these in equation 47 to calculate $\mu[y_1]$ and $V[y_1]$ and hence the likelihood for the first event, y_1, from equation 46. We then calculate the posterior moments using equation 48. For the next event, $j=2$, these posterior moments are assigned to $\mu[z_{j-1}]$ and $V[z_{j-1}]$ in equation 43 and the likelihood calculated. The procedure is iterated through all the events.

8 Parameter estimation with the OU process

The following R code estimates the posterior PDF for an OU process using the ideas in the previous section. It makes use of the functions defined in the appendix in section C.

I first simulate an OU process at a series of time steps using the update equation (equation 41). The mean and standard deviation of the state variable are shown as the magenta points/error bars in the left panel of Figure 6. This is the true process: there is no measurement noise. From each of these a value is drawn and measurement noise added: this is shown as the black point/error bar. These, together with value for the OU process parameters, are then used to predict the process. The mean and variance of the predicted PDF at each time step is shown with the red points/error bars in the right panel of Figure 6. As a Markov process, each of these predictions uses the data at just the previous data point. This "prior" prediction of event j is then combined with the data at event j to make a "posterior" prediction (equation 48; not plotted), which is propagated by equation 43 to give the prior prediction for the next event.

The R code uses MCMC to sample the posterior PDF over the OU process parameters. At each parameter sample, the whole OU process is predicted and the likelihood calculated as described in section 7. An example of the sampling is show in Figure 7.

The OU process as defined has four parameters, c, τ, $\mu[z_1]$, and $V[z_1]$. The latter two are taken as fixed in the following. In practice we may choose to set $\mu[z_1]$ to the initial value of the time series, and $V[z_1] = 0$, which I do here. Concerning the priors, I use the gamma distributions for both c and τ, which forces their values to be positive. To suppress vanishingly small values, I set the shape parameter of the gamma distribution to be 1.5. The scale parameters I set as follows. For τ, I set it to tenth of the duration of the time series. This is somewhat arbitrary, so sensitivity to it should be checked. As the long-term variance of an OU process is $c\tau/2$, then I set the scale for c to be this asymptotic value, $2\varsigma^2/\tau$, where ς is the standard deviation of the time series. Thus there are two "hyperparameters" (parameters of the prior PDF), called `alpha` in the code.

R file: `Rcode/exp_OUprocess.R`

```
# C.A.L. Bailer-Jones
# Astrostats 2013
# This file: exp_OUprocess.R
# R code for experimenting with modelling the OU process

source("OUprocess.R")
source("monte_carlo.R")

set.seed(102)

# Generate a time series from an OU process
ouproc <- list(diffcon=1, relax=1, zstartmean=0, zstartsd=0)
eventTimes <- sort(c(0, 20, runif(n=48, min=0, max=20)))
ysigma <- 0.1
temp <- gen.OUprocess(ouproc=ouproc, eventTimes=eventTimes, ysigma=ysigma)
obsdata  <- temp$obsdata
procDist <- temp$procDist

# Plot data, ... first overplotting the distributions for each point from the true process
# i.e. the mean and sd of the update equation
oplot.obsdata.OUprocess(obsdata=obsdata, procDist=procDist, fname="ouprocdat1.pdf")
# ... then overplotting the prior PDF evaluated for the OU process at the given parameters
oplot.obsdata.OUprocess(obsdata=obsdata, ouproc=ouproc, fname="ouprocdat2.pdf")
```

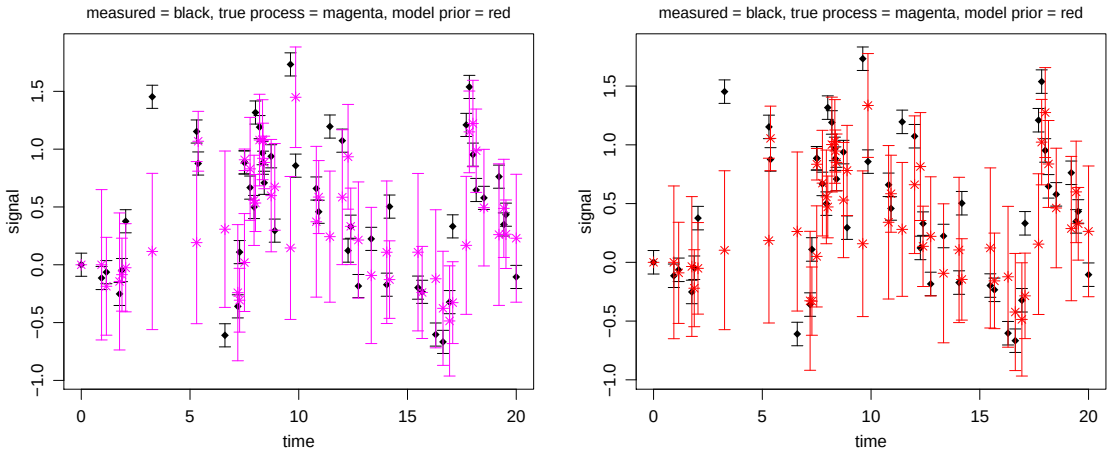

Figure 6: Data set (black) generated from the OU process using update equation 41. The black error bars characterize the Gaussian measurement noise. The same data are shown in both panels. The magenta points/bars in the left panel show the mean and standard deviation of the update equations as given in equation 42. These error bars indicate the intrinsic variance of the process itself: they are *not* measurement noise. Each data point (black point) has been generated by drawing a random number from the Gaussian distribution characterized by the magenta point/bar, and then adding Gaussian measurement noise (zero mean, standard deviation 0.1). The red points/error bars in the right panel show the mean and variance of the prior predicted PDF of each time step (i.e. not using the measurement at that time step), as given by equation 43 (using the true values of the OU process parameters). Note that the red error bars cannot be smaller than the magenta ones; they are very similar in this example. The magenta and red means differ. The true parameters of the generative model are $c = 1$, $\tau = 1$ as well as the initial conditions $\mu[z_1] = 0$, $V[z_1] = 0$.

```
# Model assumes data are zero mean
obsdata[,3] <- obsdata[,3] - mean(obsdata[,3])

# For the following inference, we fix the OU process parameters
# ouprocFixed <- c(zstartmean, zstartsd) and just sample over the
# parameters theta = c(diffcon, relax). The scale hyperparameters on the
# gamma priors are specified by alpha = c(scale_diffcon, scale_relax)
ouprocFixed <- c(0,0) # c(zstartmean, zstartsd)
alpha <- c(1,1) # c(scale_diffcon, scale_relax)
# Or, using the prescription in the lecture
# trange <- diff(range(obsdata[,1]))/10
# alpha <- c(2*var(obsdata[,3])/trange, trange)

# Use MCMC to sample posterior for OU process: c(diffcon, relax)
sampleCov <- make.covariance.matrix(sampleSD=c(0.1, 0.1), sampleCor=0)
thetaInit <- c(3,3)
set.seed(150)
postSamp <- metrop(func=logpost.OUprocess, thetaInit=thetaInit, Nburnin=0,
                   Nsamp=1e4, verbose=1e3, sampleCov=sampleCov,
                   obsdata=obsdata, ouprocFixed=ouprocFixed, alpha=alpha)
```

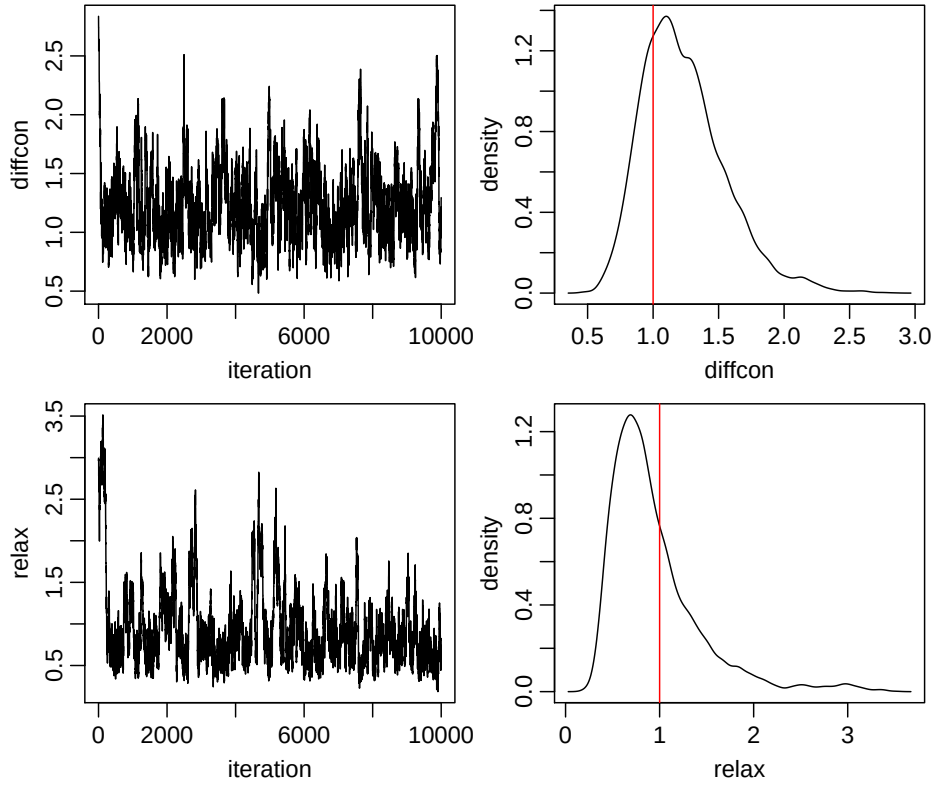

Figure 7: Example of a posterior PDF sampling of the OU process model for the data shown in Fig. 6. The panels in the left column show the chains for the two parameters c ("diffcon") and τ ("relax"). The panels on the right show the 1D PDFs generated from the samples via density estimation. The vertical red lines indicate the true parameters.

```
# Plot MCMC chains and use density estimation to plot 1D posterior PDFs from these.
thetaTrue <- as.numeric(ouproc[1:2])
parnames <- c("diffcon", "relax")
pdf("ouprocdat_mcmc.pdf", width=7, height=6)
par(mfrow=c(2,2), mar=c(3.0,3.0,0.5,0.5), oma=c(1,1,1,1), mgp=c(1.8,0.6,0), cex=1.0)
for(p in 3:4) { # columns of postSamp
  plot(1:nrow(postSamp), postSamp[,p], type="l", xlab="iteration", ylab=parnames[p-2])
  postDen <- density(postSamp[,p], n=2^10)
  plot(postDen$x, postDen$y, type="l", xlab=parnames[p-2], ylab="density")
  abline(v=thetaTrue[p-2], col="red")
}
dev.off()
```

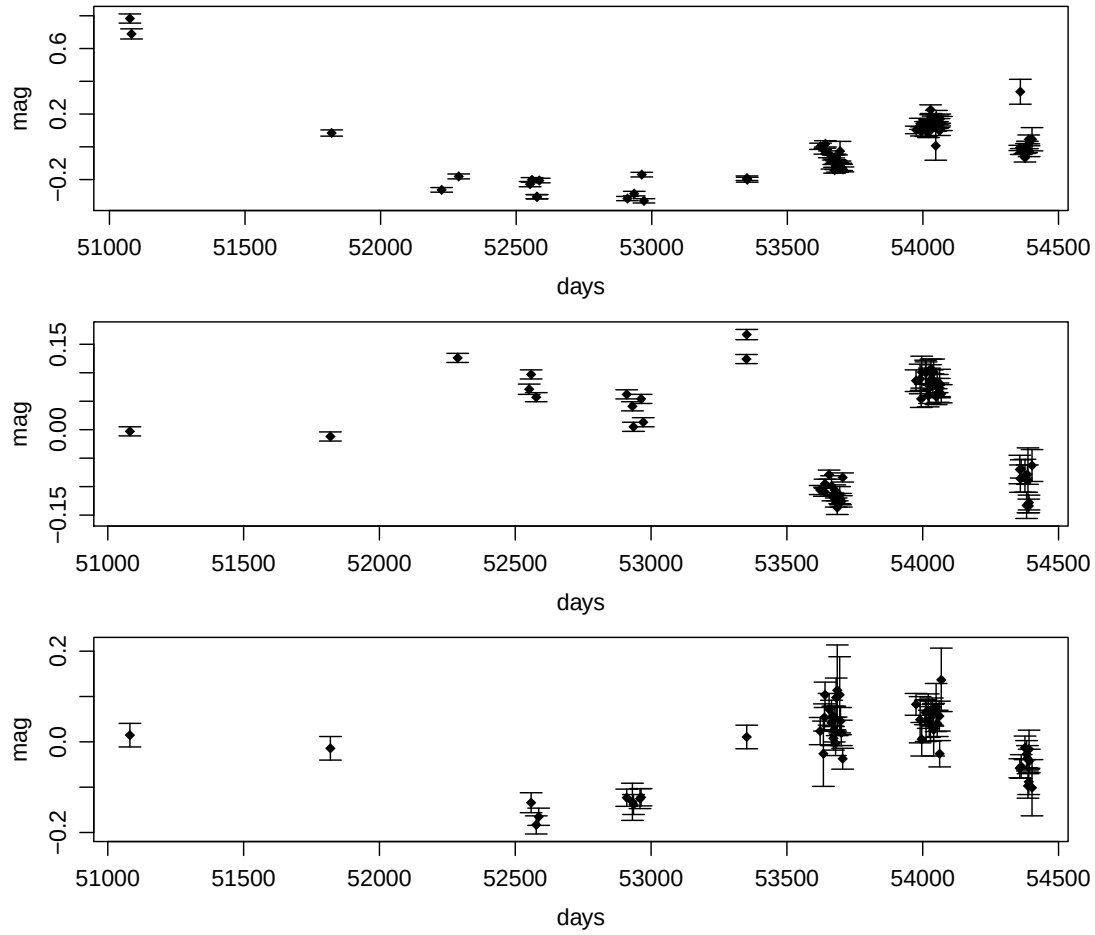

Figure 8: Three g-band quasar light curves. From top to bottom their ID numbers are: 1001265, 1002162, 208035.

9 Comparison of sinusoidal and OU process models on real data

A number of different studies have demonstrated that a large fraction of quasar light curves can be described better by the OU process than by several other deterministic or stochastic processes. One particular study which uses the Bayesian evidence for this purpose is Andrae et al. (2013). In the code below we calculate the evidence and K-fold CV likelihood for the sinusoidal and OU process models on three light curves from this study (Figure 8).

R file: `Rcode/model_comparison_2.R`

```
# C.A.L. Bailer-Jones
# Astrostats 2013
# This file: model_comparison_2.R
# R code to calculate evidence and K-fold CV likelihood for sinusoidal
# and OU process models on a real data set.
```

```
library(gplots) # for plotCI()
source("sinusoidal.R")
source("OUprocess.R")
source("monte_carlo.R")
source("kfoldCV.R")

########## Read in data

# Files:
#obsdata <- read.table("../data/g-band-LC_1001265.dat", header=FALSE)
#obsdata <- read.table("../data/g-band-LC_1002162.dat", header=FALSE)
obsdata <- read.table("../data/g-band-LC_208035.dat", header=FALSE)
# Models assumes data are zero mean
obsdata[,3] <- obsdata[,3] - mean(obsdata[,3])
plotCI(obsdata[,1], obsdata[,3], uiw=obsdata[,4], err="y", xlab="time", ylab="signal",
       pch=18, type="n", gap=0, sfrac=0.01)

########## Set priors and fixed parameters

zrange <- diff(range(obsdata[,3]))
alphaSinusoidal <- 0.5*c(zrange/2, zrange/2, 1/1000) # c(sd_a1, sd_a2, scale_freq)
#
ouprocFixed <- c(obsdata[,3], 0) # c(zstartmean, zstartsd)
tscale <- diff(range(obsdata[,1]))/10
alphaOUprocess <- 0.5*c(2*var(obsdata[,3])/tscale, tscale) # c(scale_diffcon, scale_relax)

########## Calculate evidences

set.seed(100)
# sinusoidal model
Nsamp <- 1e4
priorSamples <- sampleprior.sinusoidal(Nsamp, alphaSinusoidal)
logLike <- vector(length=Nsamp)
for(i in 1:Nsamp) {
  logLike[i] <- loglike.sinusoidal(priorSamples[i,], obsdata)
}
evSM <- mean(10^logLike)
# OUprocess
Nsamp <- 1e4
priorSamples <- sampleprior.OUprocess(Nsamp, alphaOUprocess)
logLike <- vector(length=Nsamp)
for(i in 1:Nsamp) {
  logLike[i] <- loglike.OUprocess(priorSamples[i,], obsdata, ouprocFixed)
}
evOU <- mean(10^logLike)
#
cat("Bayes factor [OUprocess/sinusoidal] = ", evOU/evSM, "\n")
cat("log10 Bayes factor [OUprocess - sinusoidal] = ", log10(evOU/evSM), "\n")
cat("log10 Evidences [OUprocess, sinusoidal] = ", log10(evOU), log10(evSM), "\n")

########## Calculate K-fold CV likelihoods

set.seed(100)
# sinusoidal model: c(a1, a2, freq)
sampleCov <- make.covariance.matrix(sampleSD=alphaSinusoidal/50, sampleCor=0)
```

```
thetaInit <- alphaSinusoidal
kcvSM <- kfoldcv(Npart=10, obsdata=obsdata, logpost=logpost.sinusoidal,
                 loglike=loglike.sinusoidal, sampleCov=sampleCov, thetaInit=thetaInit,
                 Nburnin=1e3, Nsamp=1e4, alpha=alphaSinusoidal)
# OUprocess model: c(diffcon, relax)
sampleCov <- make.covariance.matrix(sampleSD=alphaOUprocess, sampleCor=0)
thetaInit <- alphaOUprocess
kcvOU <- kfoldcv(Npart=10, obsdata=obsdata, logpost=logpost.OUprocess,
                 loglike=loglike.OUprocess, sampleCov=sampleCov, thetaInit=thetaInit,
                 Nburnin=1e3, Nsamp=1e4, ouprocFixed=ouprocFixed, alpha=alphaOUprocess)
#
cat("log10 K-fold CV likelihood [OUprocess, sinusoidal]", kcvOU, kcvSM, "\n")
cat("Difference log10 K-fold CV likelihood [OUprocess - sinusoidal]", kcvOU - kcvSM, "\n")

######### Calculate posterior PDFs

# should really check the MCMC in the K-fold CV likelihood by plotting samples and
# posterior PDFs for the data partitions used there. But get a good idea by sampling
# the whole data set, as done here.

# sinusoidal model: c(a1, a2, freq)
sampleCov <- make.covariance.matrix(sampleSD=alphaSinusoidal/25, sampleCor=0)
thetaInit <- alphaSinusoidal
postSamp <- metrop(func=logpost.sinusoidal, thetaInit=thetaInit, Nburnin=1e3,
                   Nsamp=1e4, verbose=1e3, sampleCov=sampleCov,
                   obsdata=obsdata, alpha=alphaSinusoidal)
parnames <- c("a1", "a2", "freq")
par(mfrow=c(3,2), mar=c(3.0,3.0,0.5,0.5), oma=c(1,1,1,1), mgp=c(1.8,0.6,0), cex=1.0)
for(p in 3:5) { # columns of postSamp
  plot(1:nrow(postSamp), postSamp[,p], type="l", xlab="iteration", ylab=parnames[p-2])
  postDen <- density(postSamp[,p], n=2^10)
  plot(postDen$x, postDen$y, type="l", xlab=parnames[p-2], ylab="density")
}
```

10 Exercises

R code to help you perform these exercises has been supplied above in the relevant sections. Recall that the analyses above assume the data to have zero mean signal, so you'll need to subtract the mean (even if you generate a finite amount of data from a zero mean process).

1. **Sinusoidal model 1**. Simulate data from a sinusoidal model (equation 2) and add some Gaussian noise (see section 4). Calculate and plot the Schuster periodogram (equation 21) as well as the (unnormalized) log posterior PDF over frequency (i.e. the Bayesian periodogram), for the two cases of known common noise (equation 20) and unknown noise (equation 26). Experiment with changing the amount of data, the amplitudes and the frequency in relation to the time span.

2. **Sinusoidal model 2**. Taking one of your noisy simulated sinusoidal data sets, use an MCMC algorithm to infer the model parameters (i.e. sample the posterior). (See section 4.)

3. **OU process 1**. Using the code in section 8, simulate data from an OU process using the update equations 41 and 42 for specific values of the two model parameters, τ and c. Get a feel for how changing the parameters alters the properties of the process the parameters. For fixed parameters, also get a feel for how much the time series changes just by changing the random number seed. (In addition to plotting the generated data points, plot the distributions – as mean and error bar – of the update equations.)

4. **OU process 2**. Simulate data from an OU process as in the previous exercise and add some Gaussian noise. Using the code in section 8, infer the model parameters (i.e. sample the posterior).

5. **Astronomical variability**. Look at the astronomical time series provided and examined in section 9. Visual inspection suggests this may show periodic variability, but maybe a stochastic model explains the data better? Adopting suitable priors, use the Bayesian evidence to compare a periodic model with an OU process model. (You may also use the K-fold CV likelihood, but it can take a while to run). Infer the posterior PDFs over the parameters of the two models to check whether there is a dominant solution. For comparison you may also want to calculate the various periodograms) for the two models, discussed in section 3.

A Further reading

Andrae R., Kim D.-W., Bailer-Jones C.A.L., 2013, *Assessment of stochastic and deterministic models of 6 304 quasar lightcurves from SDSS Stripe 82*, A&A in press
`http://arxiv.org/abs/1304.2863`

Bailer-Jones C.A.L., 2012, *A Bayesian method for the analysis of deterministic and stochastic time series*, A&A 546, A89
See `http://www.mpia.de/~calj/ctsmod.html` for more information and software

Bretthorst G.L., 1998, *Bayesian spectrum analysis and parameter estimation*, Lecture notes in statistics vol. 48, Springer
Out of print, but available from `http://http://bayes.wustl.edu/glb/book.pdf`

Gardiner C., 2009, *Stochastic methods*, Springer, 4th edition
A definitive work, but is hard going after the first few chapters

Gillespie D.T., 1996, *Exact numerical simulation of the Ornstein–Uhlenbeck process and its integral*, Phys. Rev. E, 54, 2084

Gillespie D.T., 1996b, *The mathematics of Brownian motion and Johnson noise*, Am. J. Phys. 64, 225
A nice introduction to stochastic processes

Gregory P.C., 2005, *Bayesian logical data analysis for the physical sciences*, Cambridge University Press
Chapter 13 is on spectral analysis

Scott M., 2011, *Applied stochastic processes in science and engineering*, unpublished book, `http://www.math.uwaterloo.ca/~mscott/Little_Notes.pdf`

B R functions for the sinusoidal model

R file: `Rcode/sinusoidal.R`

```
# C.A.L. Bailer-Jones
# Astrostats 2013
# This file: sinusoidal.R
# R functions related to the single frequency sinusoidal model

# An event is described by a 4 element vector c(s, s.sd, y, y.sd)
#   where s is the time and y is the signal and s.sd and y.sd their
#   uncertainties, respectively.
# Nevents is the number of events
# obsdata is a matrix of Nevents (rows) and 4 ordered columns c(s, s.sd, y, y.sd)

# Given sinmod and times eventTimes (vector), generate data from a sinusoidal model,
# optionally with additional Gaussian measurement noise of standard deviation ysigma.
# sigma can be a vector of length eventTimes, or a scalar. Default value is zero.
# Return in format obsdata.
gen.sinusoidal <- function(sinmod, eventTimes, ysigma=0) {
  Nevents <- length(eventTimes)
  obsdata <- matrix(data=0, nrow=Nevents, ncol=4)
  obsdata[,1] <- eventTimes
  obsdata[,2] <- 0
  obsdata[,4] <- ysigma
  omegat <- 2*pi*sinmod$freq*obsdata[,1]
  obsdata[,3] <- sinmod$a1*cos(omegat) + sinmod$a2*sin(omegat)
  obsdata[,3] <- obsdata[,3] + rnorm(Nevents, mean=0, sd=obsdata[,4])
  return(obsdata)
}

# Plot obsdata, and optionally overplot a sinusoidal model with parameters sinmod.
# trange=c(tmin,tmax) can be supplied, otherwise (if it's NULL) is calculated from the data.
# If ploterr=TRUE, plot error bars.
# If fname is supplied, save the plot in a PDF with this name.
oplot.obsdata.sinusoidal <- function(obsdata, sinmod=NULL, trange=NULL,
                                     ploterr=TRUE, fname=NULL) {
  library(gplots)
  if(!is.null(fname)) pdf(fname, width=6, height=5)
  par(mfrow=c(1,1), mgp=c(2.0,0.8,0), mar=c(3.5,4.0,1.0,1.0), oma=c(0,0,2.0,0), cex=1.2)
  if(is.null(trange)) {
    trange <- range(c(obsdata[,1]+obsdata[,2], obsdata[,1]-obsdata[,2]))
  }
  if(!is.null(sinmod)) {
    tmod <- seq(from=min(trange), to=max(trange), length.out=1e3)
    omegat <- 2*pi*sinmod$freq*tmod
    ymod <- sinmod$a1*cos(omegat) + sinmod$a2*sin(omegat)
    ymin <- min(ymod, obsdata[,3]-obsdata[,4])
    ymax <- max(ymod, obsdata[,3]+obsdata[,4])
  } else {
    ymin <- min(obsdata[,3]-obsdata[,4])
    ymax <- max(obsdata[,3]+obsdata[,4])
  }
  plot(obsdata[,1], obsdata[,3], xlim=trange, ylim=c(ymin,ymax),  pch=18,
```

```
       xlab="time", ylab="signal")
  if(ploterr) plotCI(obsdata[,1], obsdata[,3], uiw=obsdata[,4], err="y", type="n",
                     gap=0, sfrac=0.01, add=TRUE)
  if(!is.null(sinmod)) {
    lines(tmod, ymod, col="red")
  }
  mtext("measured = black, model = red", padj=-1)
  if(!is.null(fname)) dev.off()
}

# Calculate the Schuster periodogram of obsdata at NFreq uniformly spaced
# frequencies between minFreq and maxFreq.
# Plot this as well as the unnormalized posterior assuming
# (1) a common sigma for all the data equal to the mean of obsdata[,4].
# (2) sigma is unknown
# as well as the other assumptions noted in the script for the
# approximations to hold.
# If fname is supplied, save the plot in a PDF with this name.
schuster <- function(obsdata=obsdata, minFreq=NULL, maxFreq=NULL, NFreq=NULL, fname=NULL) {
  cat("mean, sd of data =", mean(obsdata[,3]), sd(obsdata[,3]), "\n")
  J <- nrow(obsdata)
  sampFreq <- seq(from=minFreq, to=maxFreq, length.out=NFreq)
  schuster <- vector(mode="numeric", length=length(sampFreq))
  for(f in 1:length(sampFreq)) {
    omegat <- 2*pi*sampFreq[f]*obsdata[,1]
    R <- sum(obsdata[,3]*cos(omegat))
    I <- sum(obsdata[,3]*sin(omegat))
    schuster[f] <- (R^2 + I^2)/J
  }
  schusterNorm <- schuster/( sum(schuster)*(maxFreq-minFreq)/NFreq ) # set area to unity
  #
  if(!is.null(fname)) pdf(fname, width=7, height=6)
  par(mfrow=c(3,1), mar=c(0.5,3.5,0,0), oma=c(3.0,0,1,1), mgp=c(2.0,0.6,0), cex=1.1)
  plot(sampFreq, schusterNorm, xaxt="n", ylab="power (normalized)", type="l")
  text(max(sampFreq), 0.9*max(schusterNorm), "Schuster periodogram", pos=2)
  # posterior periodogram with known common sigma
  logpost <- (1/log(10))*schuster/mean(obsdata[,4])^2
  plot(sampFreq, logpost, xaxt="n", ylab="log10post", type="l")
  text(max(sampFreq), 0.9*max(logpost), "assuming common sigma", pos=2)
  # posterior periodogram with unknown sigma
  # print(sum(obsdata[,3]^2)/2 - schuster) # check: are always positive
  posterior <- (sum(obsdata[,3]^2)/2 - schuster)^((2-J)/2)
  plot(sampFreq, log10(posterior), ylab="log10post", type="l")
  text(max(sampFreq), 0.7*max(log10(posterior)), "assuming unknown sigma", pos=2)
  #
  mtext("frequency", side=1, line=1.5, outer=TRUE, cex=1.1)
  if(!is.null(fname)) dev.off()
}

# Return log10(unnormalized posterior) of the sinusoidal model
# (see notes on the functions called)
logpost.sinusoidal <- function(theta, obsdata, alpha, ind=NULL) {
  logprior <- logprior.sinusoidal(theta, alpha)
  if(is.finite(logprior)) { # only evaluate model if parameters are sensible
    return( loglike.sinusoidal(theta, obsdata, ind) + logprior )
```

```
  } else {
    return(-Inf)
  }
}

# Return log10(likelihood) of the sinusoidal model on rows ind of
# obsdata (to within an additive constant)
# with parameters theta = c(a1, a2, freq)
# ... is needed to pick up the unwanted alpha passed by kfoldcv(),
loglike.sinusoidal <- function(theta, obsdata, ind=NULL, ...) {
  if(is.null(ind)) ind <- 1:nrow(obsdata)
  omegat <- 2*pi*theta[3]*obsdata[ind,1]
  modpred <- theta[1]*cos(omegat) + theta[2]*sin(omegat)
  logEventLike <- (1/log(10))*dnorm(x=obsdata[ind,3], mean=modpred,
                                    sd=sqrt(obsdata[ind,4]), log=TRUE)
  return( sum(logEventLike) )
}

# Return log10(unnormalized prior) of the sinusoidal model
# with parameters theta = c(a1, a2, freq) and selected prior
# hyperparameters alpha
logprior.sinusoidal <- function(theta, alpha) {
  a1Prior <- dnorm(x=theta[1], mean=0, sd=alpha[1])
  a2Prior <- dnorm(x=theta[2], mean=0, sd=alpha[2])
  freqPrior <- dgamma(x=theta[3], shape=1.5, scale=alpha[3])
  return( sum(log10(a1Prior), log10(a2Prior), log10(freqPrior)) )
}

# return Nsamp samples from prior
# (is consistent with logprior.sinusoidal)
sampleprior.sinusoidal <- function(Nsamp, alpha) {
  a1 <- rnorm(n=Nsamp, mean=0, sd=alpha[1])
  a2 <- rnorm(n=Nsamp, mean=0, sd=alpha[2])
  freq <- rgamma(n=Nsamp, shape=1.5, scale=alpha[3])
  return(cbind(a1, a2, freq))
}
```

C R functions for the OU process

R file: `Rcode/OUprocess.R`

```
# C.A.L. Bailer-Jones
# Astrostats 2013
# This file: OUprocess.R
# R functions related to the OU process

# An event is described by a 4 element vector c(s, s.sd, y, y.sd)
#   where s is the time and y is the signal and s.sd and y.sd their
#   uncertainties, respectively.
# Nevents is the number of events
# obsdata is a matrix of Nevents (rows) and 4 ordered columns c(s, s.sd, y, y.sd)
# ouproc is a named list of the parameters of the OU process
```

```
#   ouproc = list(diffcon, relax, zstartmean, zstartsd)
#   zstartmean and zstartsd are the initial conditions.

# Given ouproc and times eventTimes (vector), generate data from an OU process
# using the update equations.
# If ysigma>0 add Gaussian measurement noise with this standard deviation.
#   ysigma can be a vector of length eventTimes, or a scalar. Default value is zero.
# Return a two element list:
# obsdata
# procDist, a dataframe with Nevents rows and 2 columns:
#   mean and variance of the process itself at each time step
gen.OUprocess <- function(ouproc, eventTimes, ysigma=0) {
  Nevents <- length(eventTimes)
  obsdata <- matrix(data=0, nrow=Nevents, ncol=4)
  zMean    <- vector(mode="numeric", length=Nevents)
  zVar     <- vector(mode="numeric", length=Nevents)
  obsdata[,1] <- eventTimes
  obsdata[,2] <- 0
  zMean[1] <- ouproc$zstartmean
  zVar[1]  <- ouproc$zstartsd
  obsdata[1,3] <- ouproc$zstartmean + rnorm(1, mean=0, sd=ouproc$zstartsd)
  for(j in 2:Nevents) {
    nu <- exp(-(obsdata[j,1]-obsdata[j-1,1])/ouproc$relax)
    Vz <- (ouproc$diffcon*ouproc$relax/2)*(1-nu^2)
    zMean[j] <- obsdata[j-1,3]*nu
    zVar[j]  <- Vz
    obsdata[j,3] <- rnorm(1, mean=zMean[j], sd=sqrt(zVar[j]))
  }
  if(ysigma>0) { # add measurement noise
    obsdata[,4] <- ysigma
    obsdata[,3] <- obsdata[,3] + rnorm(Nevents, mean=0, sd=obsdata[j,4])
  }
  return(list(obsdata=obsdata, procDist=data.frame(zMean=zMean, zVar=zVar)))
}

# Plot obsdata, and optionally overplot either (if defined)
# (1) points with mean and variance given by procDist, or
# (2) predicted priors of an OU process evaluated on these data with parameters ouproc.
# If both are defined then only (1) is plotted. (This will usually be used to
# plot the mean and sd of the update formula used to generate obsdata.)
# The data are plotted in black, procDist in magenta, the predicted priors in red.
# trange=c(tmin,tmax) can be supplied, otherwise (if it's NULL) is calculated from the data.
# If ploterr=TRUE, plot error bars on the data.
# If fname is supplied, save the plot in a PDF with this name.
# NOTE: This will give warnings if the error bars are too small to plot
oplot.obsdata.OUprocess <- function(obsdata, procDist=NULL, ouproc=NULL, trange=NULL,
                                    ploterr=TRUE, fname=NULL) {
  library(gplots)
  if(!is.null(fname)) pdf(fname, width=6, height=5)
  if(is.null(trange)) {
    trange <- range(c(obsdata[,1]+obsdata[,2], obsdata[,1]-obsdata[,2]))
  }
  if(!is.null(ouproc)) {
    procParam <- eval.OUprocess(ouproc, obsdata)
  }
```

```
  if(!is.null(procDist)) {
    procParam <- procDist
  }
  par(mfrow=c(1,1), mgp=c(2.0,0.8,0), mar=c(3.5,3.5,1.0,1.0), oma=c(0,0,1,0), cex=1.1)
  if(!is.null(procParam)) {
    ymin <- min(c(procParam$zMean-sqrt(procParam$zVar), obsdata[,3]-obsdata[,4]))
    ymax <- max(c(procParam$zMean+sqrt(procParam$zVar), obsdata[,3]+obsdata[,4]))
  } else {
    ymin <- min(obsdata[,3]-obsdata[,4])
    ymax <- max(obsdata[,3]+obsdata[,4])
  }
  plot(obsdata[,1], obsdata[,3], xlim=trange, ylim=c(ymin,ymax),  pch=18,
       xlab="time", ylab="signal")
  if(ploterr) plotCI(obsdata[,1], obsdata[,3], uiw=obsdata[,4], err="y", type="n",
                     gap=0, sfrac=0.01, add=TRUE)
  if(!is.null(procDist)) {
    points(obsdata[,1], procDist$zMean, pch=8, col="magenta")
    plotCI(obsdata[,1], procDist$zMean, uiw=sqrt(procDist$zVar), err="y", type="n",
           col="magenta", gap=0, sfrac=0.01, add=TRUE)
  }
  if(!is.null(ouproc) && is.null(procDist)) {
    points(obsdata[,1], procParam$zMean, pch=8, col="red")
    plotCI(obsdata[,1], procParam$zMean, uiw=sqrt(procParam$zVar), err="y", type="n",
           col="red", gap=0, sfrac=0.01, add=TRUE)
  }
  mtext("measured = black, true process = magenta, model prior = red", padj=-1)
  if(!is.null(fname)) dev.off()
}

# Evaluate an OU process of given parameters using given obsdata.
# Specifially, return a ncol(obsdata) x 2 dataframe with named columns
# zMean and zVar which give the mean and variance of the prior PDF
# of the OU process variable at each event.
# Note: probably slower than it should be due to presence of loop
eval.OUprocess <- function(ouproc, obsdata) {
  Nevents  <- nrow(obsdata)
  zMean    <- vector(mode="numeric", length=Nevents)
  zVar     <- vector(mode="numeric", length=Nevents)
  j <- 1
  zMean[j] <- ouproc$zstartmean
  zVar[j]  <- ouproc$zstartsd^2
  for(j in 1:Nevents) {
    if(j > 1) {
      # warning: divide by zero if relax=0 (controlled by prior)
      nu <- exp(-(obsdata[j,1]-obsdata[j-1,1])/ouproc$relax)
      Vz <- (ouproc$diffcon*ouproc$relax/2)*(1-nu^2)
      zMean[j] <- zMeanPost*nu # + ouproc$offset*(1-nu)
      zVar[j]  <- zVarPost*nu^2 + Vz
    }
    if(j < Nevents) { # calculate posterior
      ysigsq    <- obsdata[j,4]^2
      denom     <- ysigsq + zVar[j]
      if(denom==0) { # controlled by data/prior
        stop("Both ysigsq and zVar are zero, so posterior moments are undefined")
      }
```

```
      zMeanPost <- (obsdata[j,3]*zVar[j] + zMean[j]*ysigsq)/denom
      zVarPost  <- zVar[j]*ysigsq/denom
    }
  }
  return(data.frame(zMean=zMean, zVar=zVar))
}

# Return log10(unnormalized posterior) of the OU process
# (see notes on functions called)
logpost.OUprocess <- function(theta, obsdata, ouprocFixed, alpha, ind=NULL) {
  logprior <- logprior.OUprocess(theta, alpha)
  if(is.finite(logprior)) { # only evaluate model if parameters are sensible
    return( loglike.OUprocess(theta, obsdata, ouprocFixed, ind) + logprior )
  } else {
    return(-Inf)
  }
}

# Return log10(likelihood) of the OU process defined by parameters
# theta = c(diffcon, relax) and ouprocFixed = c(zstartmean, zstartsd)
# for events ind in the time series obsdata (to within an additive
# constant). We require the full set of data in order to propagate the
# estimates of the process parameters along the chain of
# events. However, the likelihood is calculated and returned only for
# the events specified in ind. (This can be used for calculating the
# k-fold CV likelihood, for example.) If ind=NULL (the default), the
# likelihood for all the data is calculated.
# ... is needed to pick up the unwanted alpha passed by kfoldcv().
# Note: Function is much slower than other loglike functions, presumably
# due to eval.OUprocess()
loglike.OUprocess <- function(theta, obsdata, ouprocFixed, ind=NULL, ...) {
  if(is.null(ind)) ind <- 1:nrow(obsdata)
  ouproc <- list(diffcon=theta[1], relax=theta[2], zstartmean=ouprocFixed[1],
                 zstartsd=ouprocFixed[2])
  procParam <- eval.OUprocess(ouproc, obsdata)
  yMean <- procParam$zMean
  yVar  <- obsdata[,4]^2 + procParam$zVar
  logEventLike <- (1/log(10))*dnorm(x=obsdata[ind,3], mean=yMean[ind],
                                    sd=sqrt(yVar[ind]), log=TRUE)
  return( sum(logEventLike) )
}

# Return log10(unnormalized prior) unnormalized prior of the OU process
# with parameters theta = c(diffcon, relax) and selected prior
# hyperparameters alpha.
# zstartmean and zstartsd are assumed fixed.
logprior.OUprocess <- function(theta, alpha) {
  diffconPrior <- dgamma(x=theta[1], shape=1.5, scale=alpha[1])
  relaxPrior   <- dgamma(x=theta[2], shape=1.5, scale=alpha[2])
  return( sum(log10(diffconPrior), log10(relaxPrior)) )
}

# return Nsamp samples from prior
# (is consistent with logprior.OUprocess)
sampleprior.OUprocess <- function(Nsamp, alpha) {
```

```
  diffcon <- rgamma(n=Nsamp, shape=1.5, scale=alpha[1])
  relax   <- rgamma(n=Nsamp, shape=1.5, scale=alpha[2])
  return(cbind(diffcon, relax))
}
```

D Spectral analysis

D.1 General concepts

The auto-covariance function of the signal $z(t)$ is

auto-covariance

$$C(t') = \lim_{T\to\infty} \frac{1}{T} \int_0^T z(t)z(t+t')\,dt = \langle z(t)z(t+t')\rangle \ . \tag{49}$$

If $z(t)$ is stationary, then the auto-covariance is independent of t.

The power spectrum of a stationary process $z(t)$ is defined as

power spectrum

$$S(\omega) = \lim_{T\to\infty} \frac{1}{2\pi T}|\phi(\omega)|^2 \tag{50}$$

where $\phi(\omega)$ is the Fourier transform of the signal

$$\phi(\omega) = \int_{-T}^{T} z(t)e^{-i\omega t}\,dt \ . \tag{51}$$

Taking the limit $T \to \infty$, it can then be shown that

$$\begin{aligned} S(\omega) &= \frac{1}{\pi}\int_{-\infty}^{\infty} C(t)e^{-i\omega t}\,dt \\ &= \frac{2}{\pi}\int_0^{\infty} C(t)\cos(\omega t)\,dt \end{aligned} \tag{52}$$

the second line following by taking just the real part and with $0 < \omega < \infty$, and

$$C(t) = \int_0^{\infty} S(\omega)\cos(\omega t)\,d\omega \ . \tag{53}$$

This shows that the power spectrum is the Fourier transform (real part, positive frequencies) of the auto-covariance function, and is known as the *Wiener–Khintchine theorem*. Setting $t' = 0$ in equation 49 and $t = 0$ in equation 53 we see that

$$\langle z(t)^2\rangle = \int_0^{\infty} S(\omega)\,d\omega \ . \tag{54}$$

Thus we see that the power spectrum, $S(\omega)$, is the portion of intensity of the signal with frequencies between ω and $\omega + d\omega$. Note that it gives the frequency spectrum of $\langle z(t)^2\rangle$ and not of $z(t)$ itself.

Comparing equation 21 with equation 50 we see that the Schuster periodogram is a discrete approximation of the power spectrum over a finite time scale.

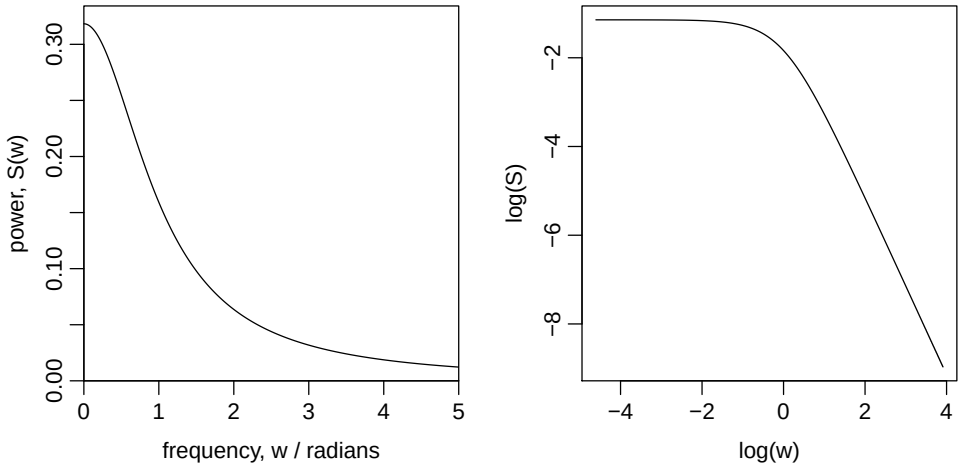

Figure 9: Power spectrum for an OU process with $\tau = c = 1$ on a linear scale (left) and a log-log scale (right). Plot generated by R code `Rcode/plot_OUprocess.R`.

D.2 OU process

The auto-covariance and power spectrum of a fully relaxed OU process, one for which $t - t_0 \gg \tau$, is

$$C(t) = \frac{c\tau}{2} e^{-t/\tau} \tag{55}$$

$$S(\omega) = \frac{1}{\pi} \frac{c\tau^2}{1 + (\omega\tau)^2} \tag{56}$$

This power spectrum is shown in Fig. 9. The turn-over in the log-log power spectrum occurs at $\omega \sim 1/\tau$. For much smaller frequencies the slope is zero; for much larger frequencies the slope is -2 (these limits reached for any values of c and τ).