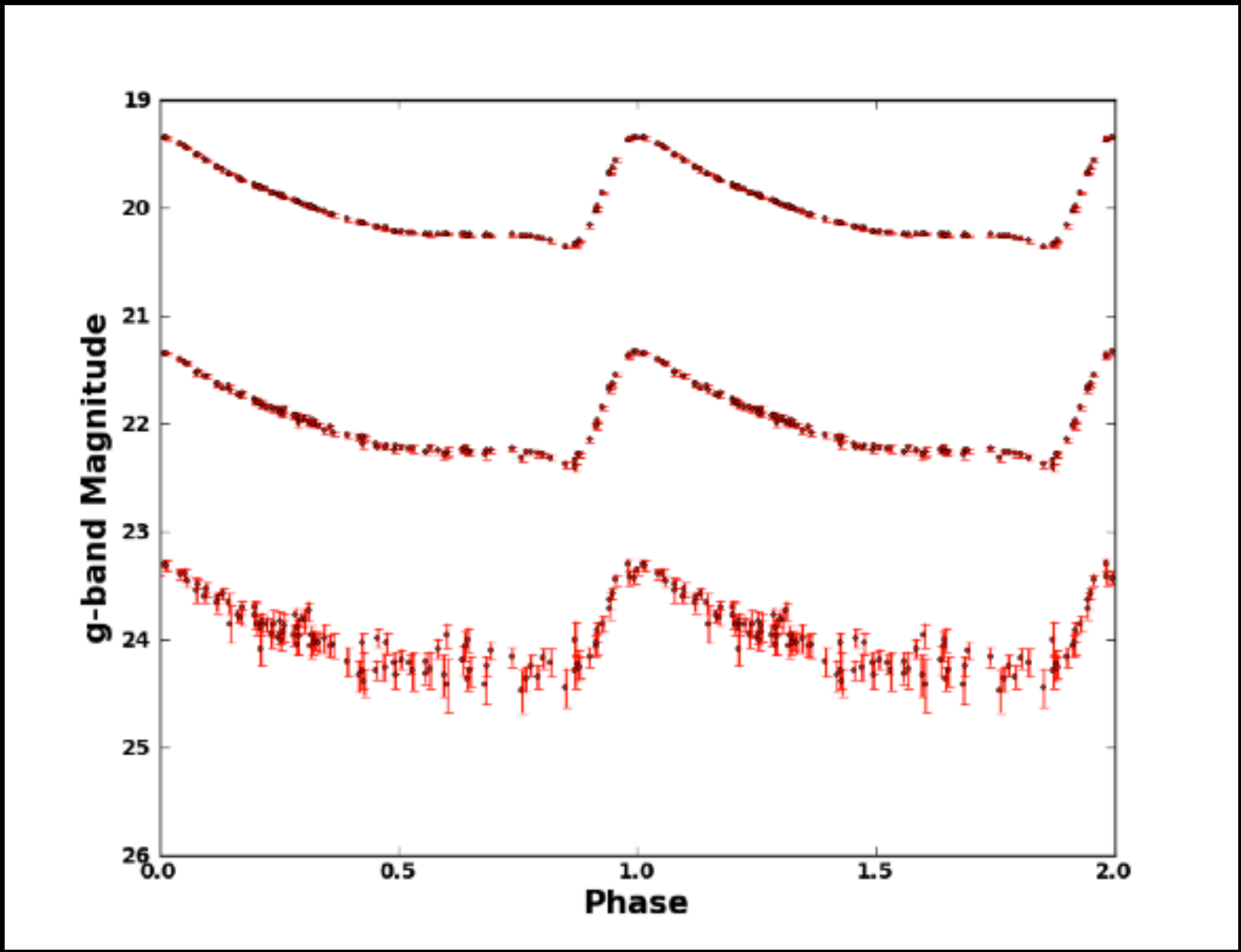
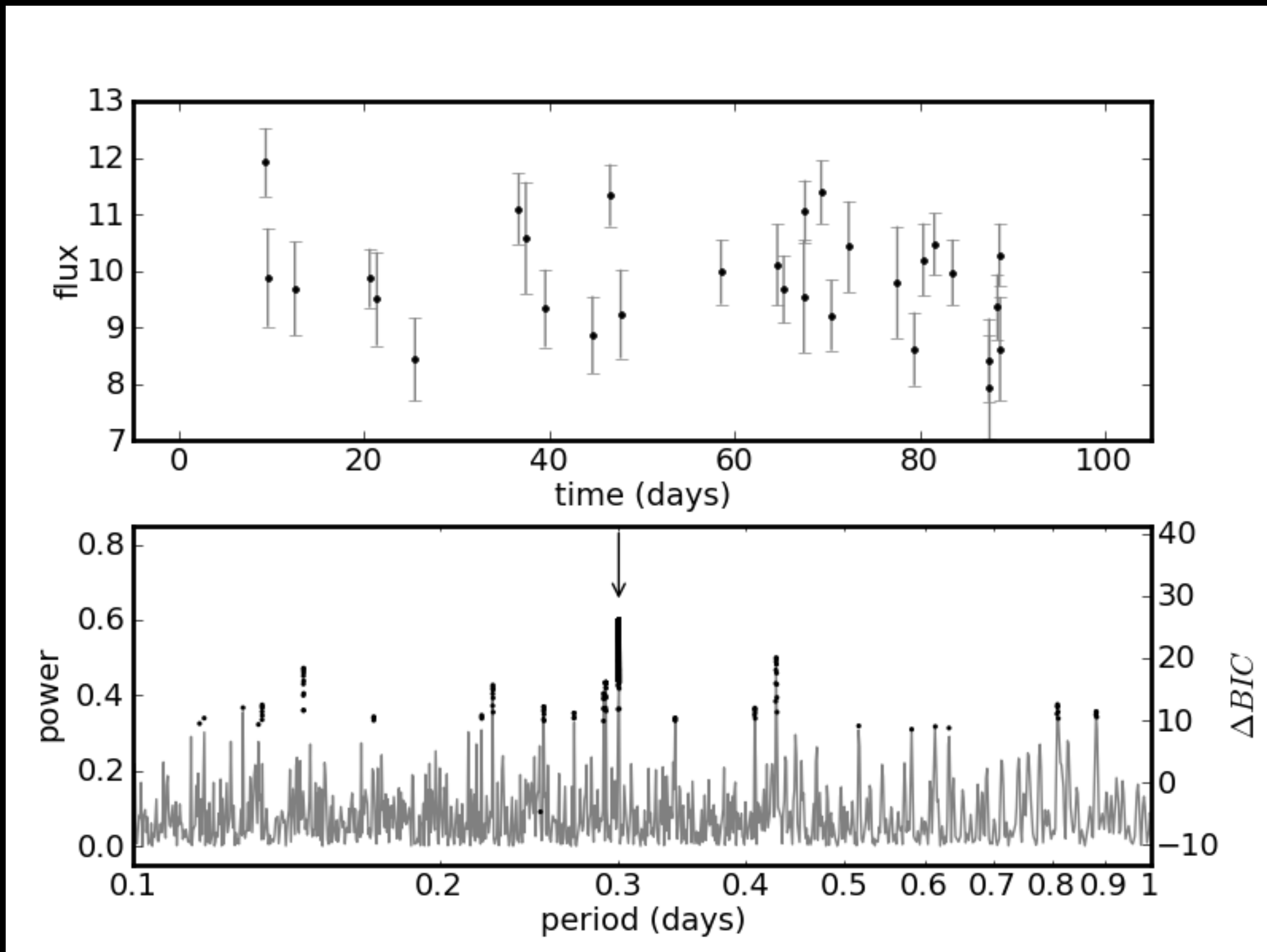
Lecture Three: Time Series Analysis

"If your experiment needs statistics, you ought to have done a better experiment."
Ernest Rutherford

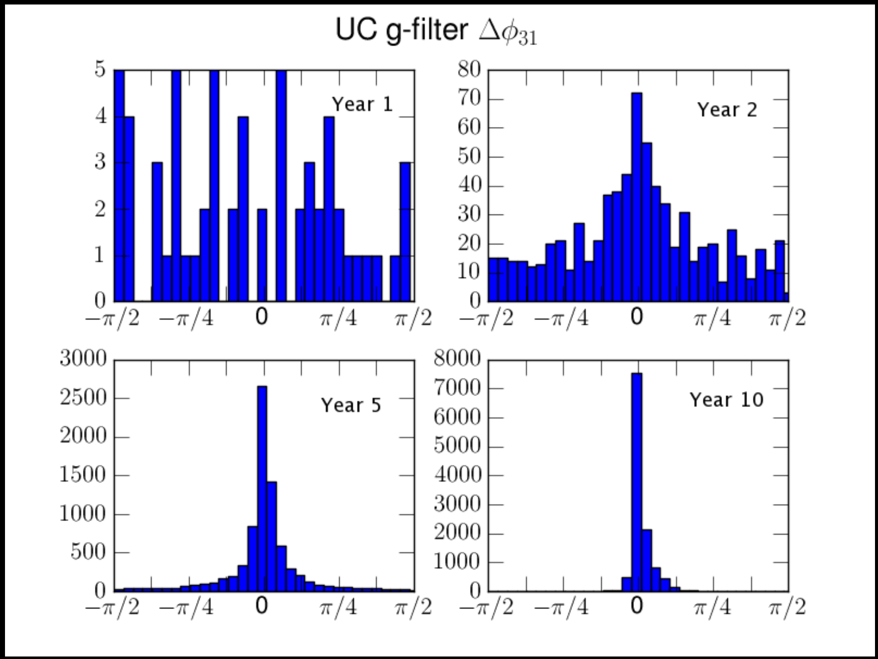
Time domain data (a day at a time)



Time domain data (a day at a time)



Sampling, noise, and baseline are all important



Lightcurve shape as a proxy for metalicity (phases in a Fourier series). Noise in period determination (sparse sampling) reflected in the metalicity accuracy

What is a light curve?

$$y(t) = \sum_{m=1}^{M} \alpha_m G(t|\theta_m),$$

G(t|θ) are functions (uneven sampling, period or non-periodic)
For example G(t) = sin(wt), or G(t) = exp(-Bt)

Fourier Analysis

Fourier transform $$H(f) = \int_{-\infty}^{\infty} h(t) \exp(-i2\pi f t)\, dt,$$

Inverse Fourier transform $$h(t) = \int_{-\infty}^{\infty} H(f) \exp(i2\pi f t)\, df,$$

Numerical Recipes define this with a minus sign
Note also the packing of the arrays

Convolution, deconvolution, filtering, correlation and autocorrelation, power spectrum are easy for evenly sampled, high signal-to-noise data. Low signal-to-noise and uneven sampling Bayesian analyses can can be better

Power Spectrum (PSD)

$$\mathrm{PSD}(f) \equiv |H(f)|^2 + |H(-f)|^2.$$

Power Spectrum is the amount of power in the frequency interval f ➔f+df

$h(t) = \sin(2\pi t/T)$ FT $PSD(f) = \delta(f = 1/T)$

$$P_{tot} \equiv \int_0^\infty \mathrm{PSD}(f)\,df = \int_{-\infty}^\infty |h(t)|^2\,dt;$$

Total power is the same whether computed in frequency or time domain (Parsevals Theorem)

Fourier Analysis in Python

```
import numpy as np
from scipy import fftpack

# compute PSD using simple FFT
N = len(data)
df = 1. / (N * dt)
PSD = abs(dt * fftpack.fft(data)[:N / 2]) ** 2
f = df * np.arange(N / 2)
```

How do we deal with sampled data

Sampling of the data – you just cant get away from it…

$$H_k = \sum_{j=0}^{N-1} h_j \exp[-i2\pi jk/N],$$

Uniformly sampled

$$h_j = \frac{1}{N} \sum_{k=0}^{N-1} H_k \exp[i2\pi jk/N]$$

FFT *O*(NlogN) rather than N^2 (numpy.fft and scipy.fft)

Sampling frequencies: Nyquist

What is the relation between continuous and sampled FFTs

Nyquist sampling theorem

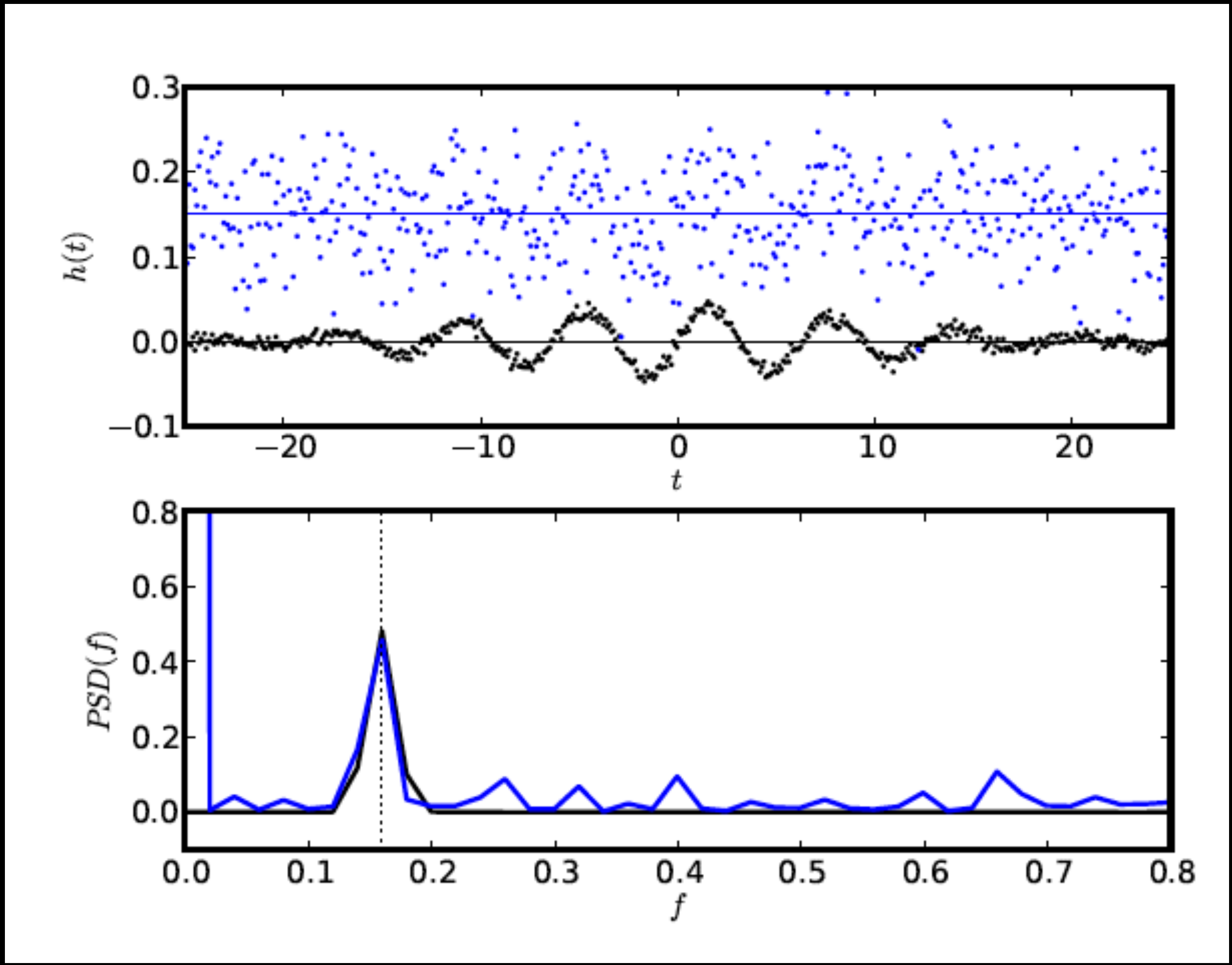
- For *band-limited* data ($H(f)=0$ for $|f| > f_c$)
  (the band limit or Nyquist frequency)

- There is some resolution limit in time $t_c = 1/(2\, f_c)$ below which $h(t)$ appears smooth

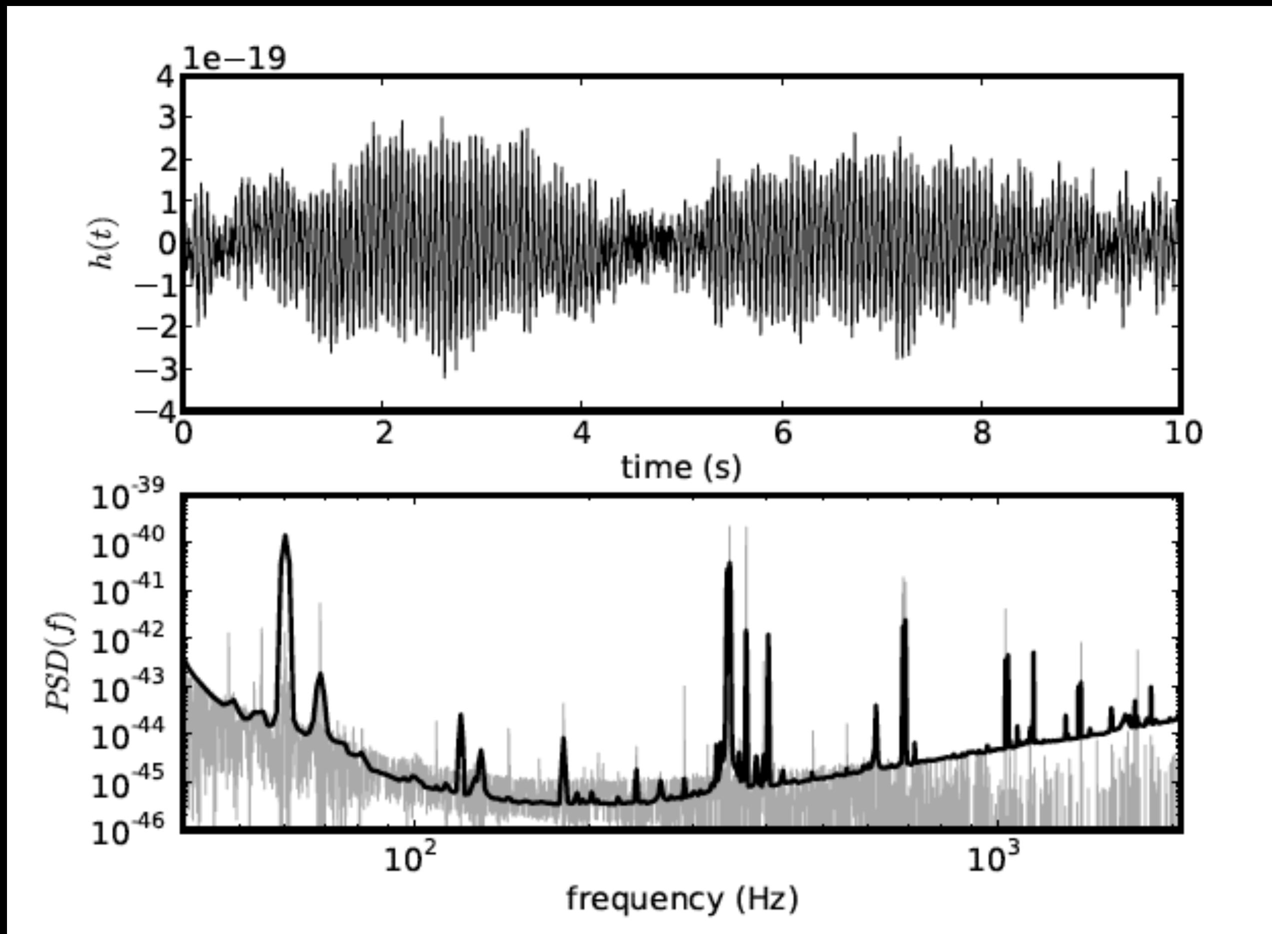
$$h(t) = \frac{\Delta t}{t_c} \sum_{k=-\infty}^{k=\infty} h_k \frac{\sin[2\pi f_c(t - k\Delta t)]}{2\pi f_c(t - k\Delta t)}.$$

We can now reconstruct $h(t)$ from evenly sampled data when $\delta t < t_c$ (Shannon interpolation formula or sinc shifting)

Estimating the PSD



Welch transform



Compute FFT for multiple overlapping windows on the data

Welch’s transform in Python

```
from matplotlib import mlab

# compute PSD using Welch's method
# this uses overlapping windows to reduce noise
PSDW, fW = mlab.psd(data, NFFT=4096, Fs = 1. / dt)
```

Filtering Data

Filtering decreases the information (even if visually you are suppressing the noise) and you should consider fitting to the raw data when fitting models

Low pass filter suppress frequencies with $f > f_c$

$$\hat{Y}(f) = Y(f)\,\Phi(f),$$

Could set $f>f_c$ to zero in $\phi(f)$ but this causes ringing

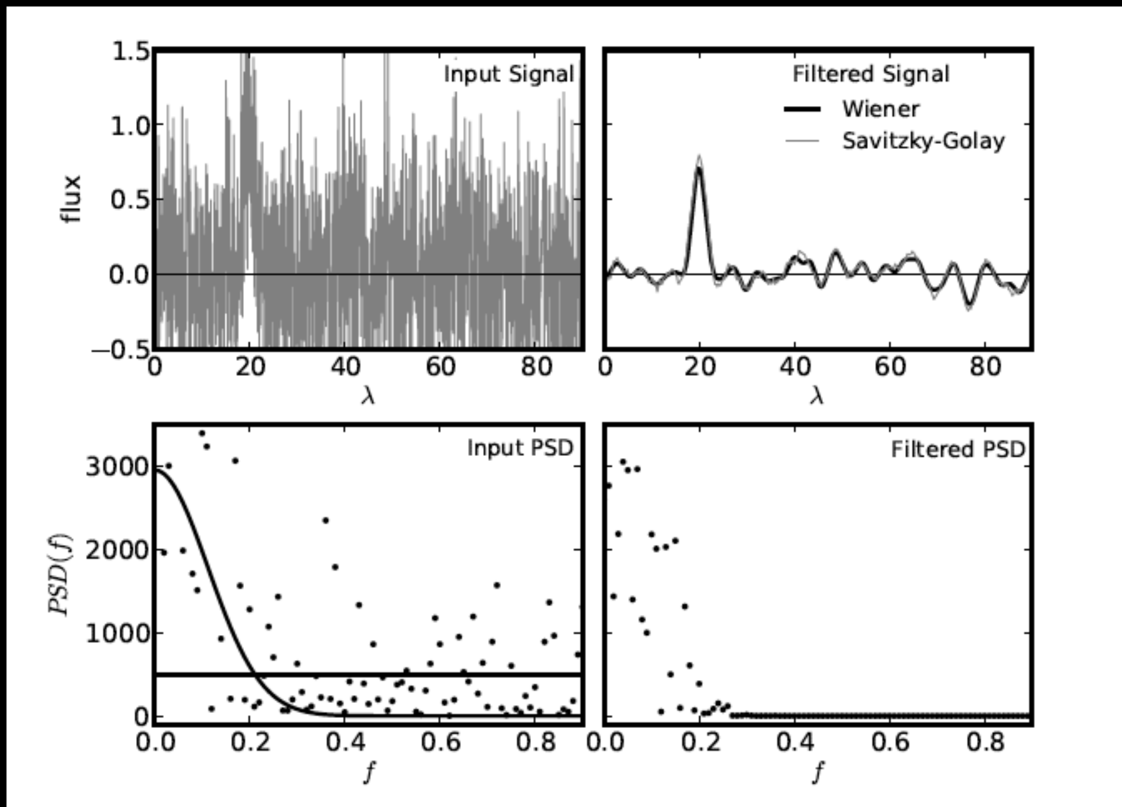
Optimal filter is Wiener filter (minimize MISE $\hat{Y} - Y$)

Signal

$$\Phi(f) = \frac{P_S(f)}{P_S(f) + P_N(f)}.$$

Noise

Wiener Filtering



Usually fit signal and noise to PSD (assumes uncorrelated noise)

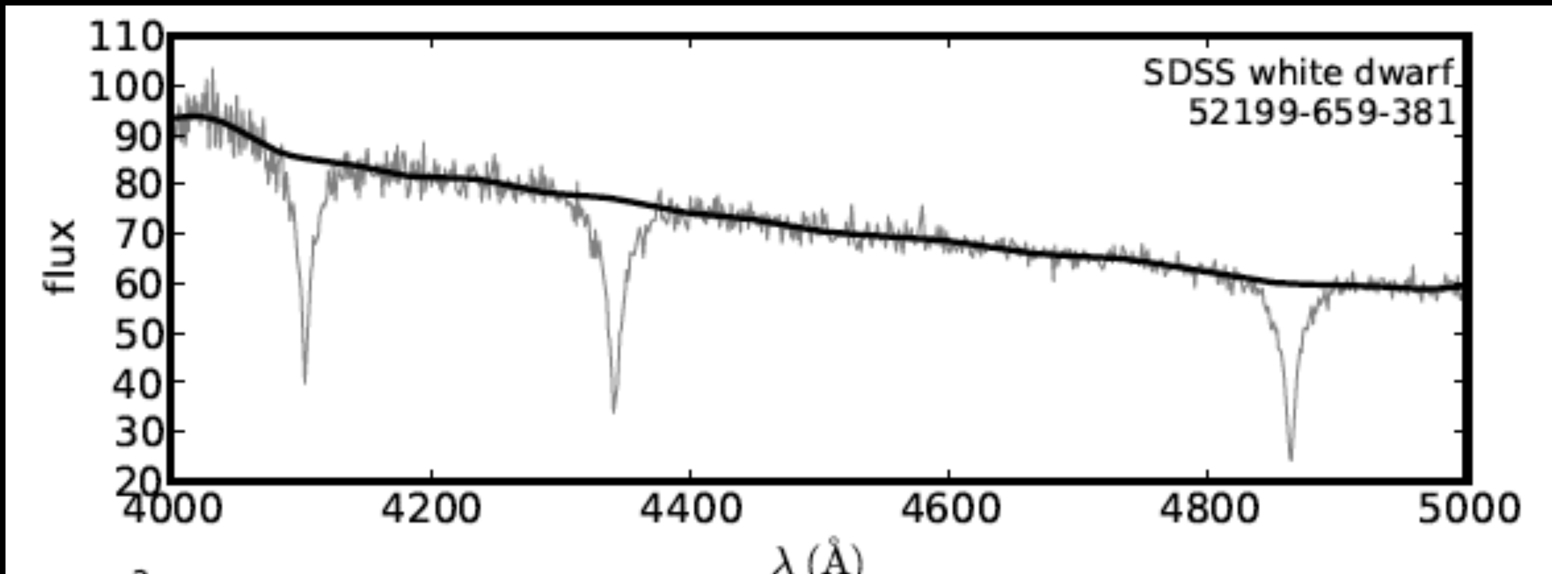
$$PSD_Y(f) = P_S(f) + P_N(f)$$

An interesting relation to kernel density estimation

Wiener filtering in Python

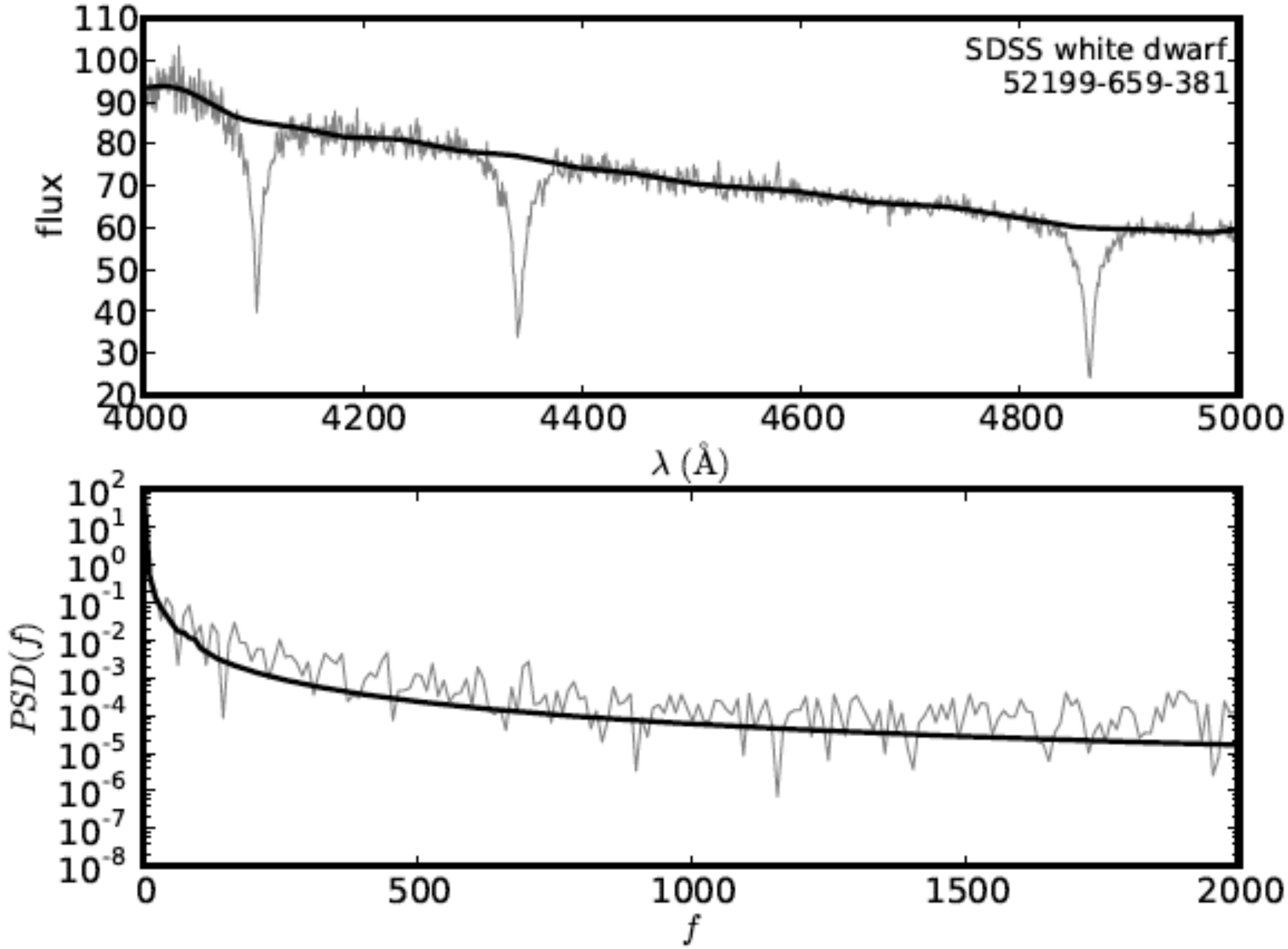
```
import numpy as np
from scipy import optimize, fftpack

# compute the PSD

# Set up the Wiener filter:
# fit a model to the PSD consisting of the sum of a Gaussian and white noise
signal = lambda x, A, width: A * np.exp(-0.5 * (x / width) ** 2)
noise = lambda x, n: n * np.ones(x.shape)

func = lambda v: np.sum((PSD - signal(f, v[0], v[1]) - noise(f, v[2])) ** 2)
v0 = [5000, 0.1, 10]
v = optimize.fmin(func, v0)

P_S = signal(f, v[0], v[1])
P_N = noise(f, v[2])

# define Wiener filter
Phi = P_S / (P_S + P_N)

h_smooth = fftpack.ifft(Phi * HN)
```

Minimum component filtering



Used for the case of fitting the baseline (continuum)

- Mask regions of signal and fit low order polynomial model to unmask regions
- Subtract the low order model and FFT the signal
- Remove high frequencies using a low pass filter
- Inverse FT and add the baseline fit



SDSS white dwarf
52199-659-381
flux
λ (Å)
PSD(f)
f

Is there signal in my noise?

Hypothesis testing – are the data consistent with a stationary process

Simple example:

$$y(t) = A\sin(wt)$$

$$\mathrm{var}(t) = \sigma^2 + \frac{A^2}{2}$$

X^2 of the data (assuming A=0)

$$\chi^2_{dof} = N^{-1}\sum_j (y_j/\sigma)^2$$

Minimum detected variability amplitude

$$A > 2.9\sigma/N^{1/4}$$

Is there a periodicity in the data?

$$y(t) = A\,\sin(\omega t + \phi)$$

$$y(t) = a\,\sin(\omega t) + b\,\cos(\omega t)$$

Non-linear in w and Φ

Linear in all but w

Consider it a least-squares problem – without requiring evenly sampled data

$$L \equiv p(\{t, y\}|\omega, a, b, \sigma) = \prod_{j=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-[y_j - a\sin(\omega t_j) - b\cos(\omega t_j)]^2}{2\sigma^2}\right).$$

Periodogram

FFT Space

$$\mathrm{PSD}(f_k) = 2\left(\frac{T}{N}\right)^2 \left[\left(\sum_{j=0}^{N-1} h_j \cos(2\pi f_k t_j)\right)^2 + \left(\sum_{j=0}^{N-1} h_j \sin(2\pi f_k t_j)\right)^2\right].$$

Time Space

$$P(\omega) = \frac{1}{N}[R^2(\omega) + I^2(\omega)].$$

$$I(\omega) = \sum_{j=1}^{N} y_j \sin(\omega t_j), \quad R(\omega) = \sum_{j=1}^{N} y_j \cos(\omega t_j),$$

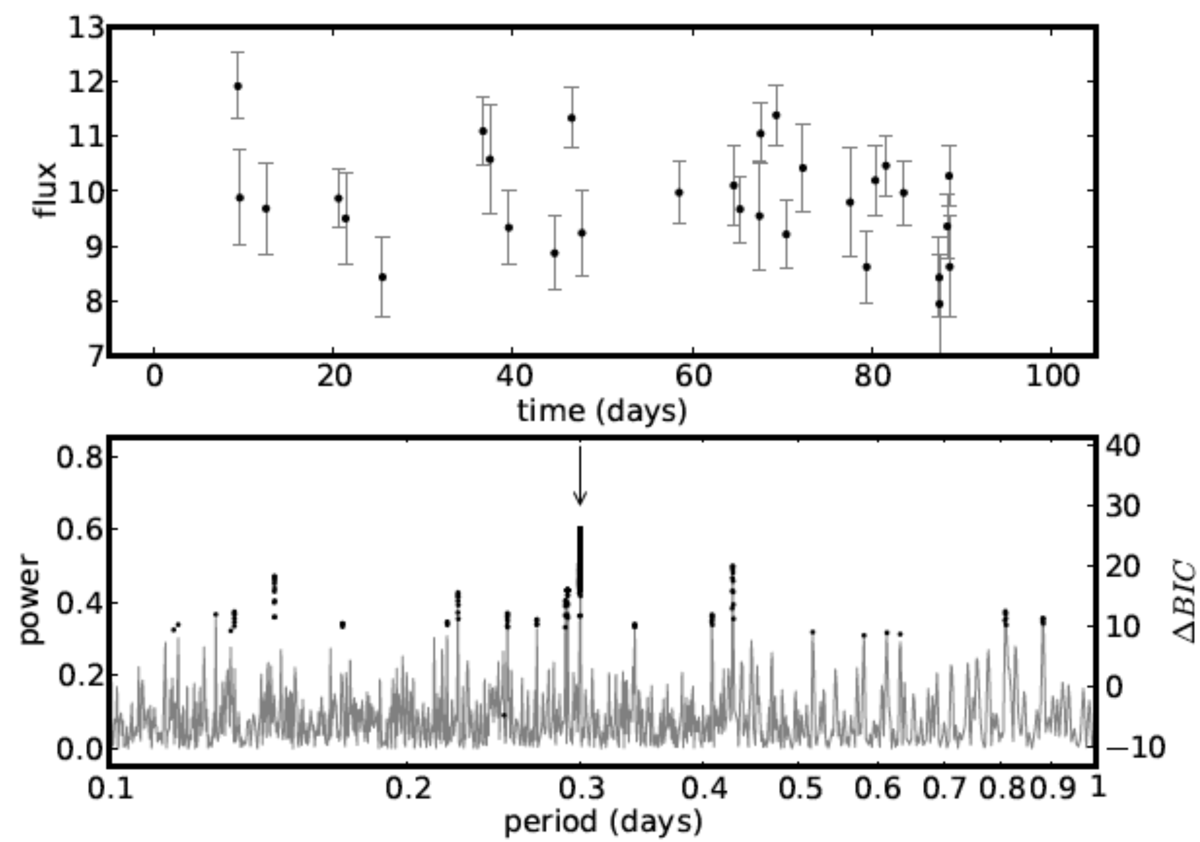
Lomb-Scargle Periodogram

$$P_{LS}(\omega) = \frac{1}{V}\left[\frac{R^2(\omega)}{C(\omega)} + \frac{I^2(\omega)}{S(\omega)}\right],$$

$$R(\omega) = \sum_{j=1}^{N} w_j\,(y_j - \overline{y})\,\cos[\omega(t_j-\tau)], \quad I(\omega) = \sum_{j=1}^{N} w_j\,(y_j - \overline{y})\,\sin[\omega(t_j-\tau)],$$

$$C(\omega) = \sum_{j=1}^{N} w_j\,\cos^2[\omega(t_j-\tau)], \quad S(\omega) = \sum_{j=1}^{N} w_j\,\sin^2[\omega(t_j-\tau)].$$

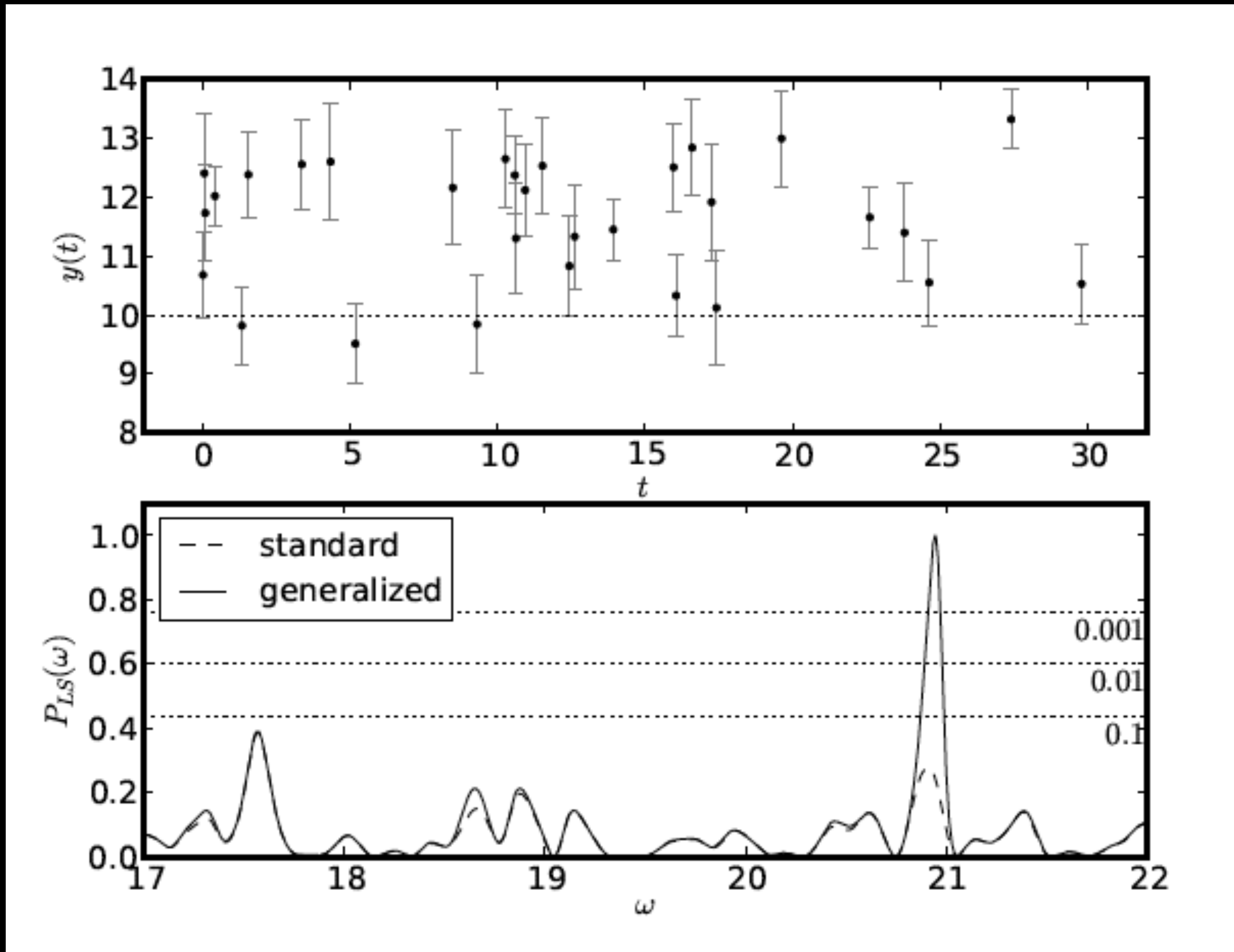
Generalized for heteroscedastic errors but still corresponds to a single sinusoidal model. Model is non-linear in frequency so we typically maximize that through a grid search



flux
time (days)
power
ΔBIC
period (days)

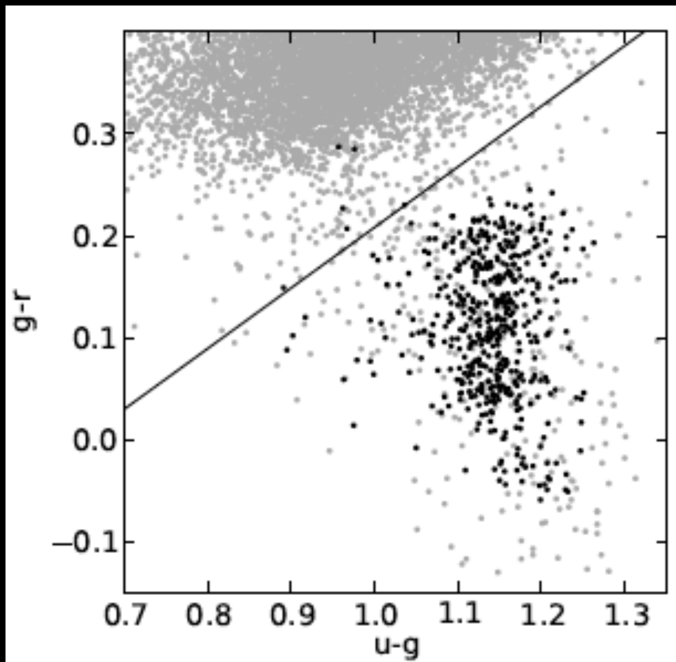
```
import numpy as np

from astroML.periodogram import lomb_scargle, search_frequencies
# generate data
t = np.random.randint(100, size=N) + 0.3 + 0.4 * np.random.random(N)
y = 10 + np.sin(2 * np.pi * t / P)
dy = 0.5 + 0.5 * np.random.random(N)

y_obs = y + np.random.normal(0, dy)

period = 10 ** np.linspace(-1, 0, 1000)
omega = 2 * np.pi / period

sig = np.array([0.1, 0.01, 0.001])
PS, z = lomb_scargle(t, y_obs, dy, omega, modified=True, significance=sig)

omega_sample, PS_sample = search_frequencies(t, y_obs, dy,
                                             n_save=100,
                                             LS_kwargs=dict(modified=True))
```

Generalized Lomb-Scargle

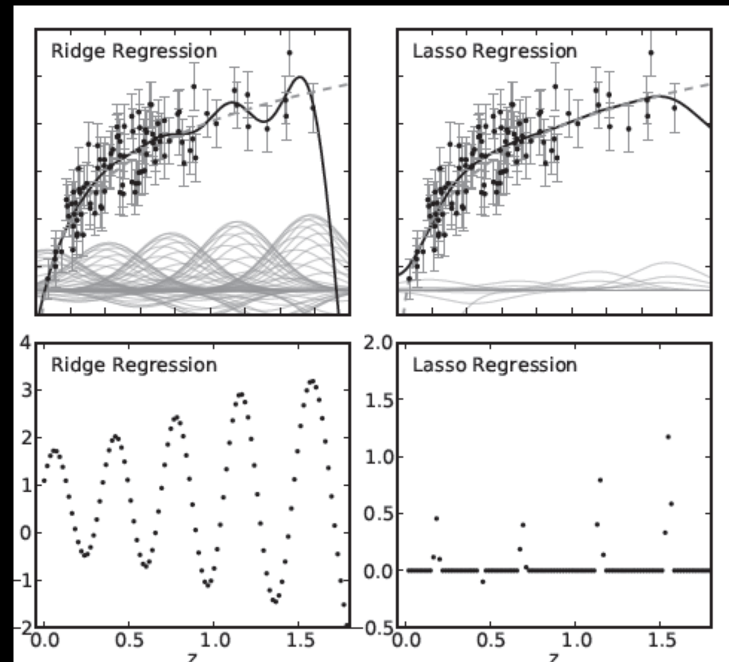


LS assumes a zero mean (calculated from the data) which can bias the results. We can however add this as a term to the analysis

Where next...



Classification



Regression

Where next...

Read the book – send comments/corrections/ suggestions
ajc@astro.washington.edu