time_series_predictor.py

September 16, 2018

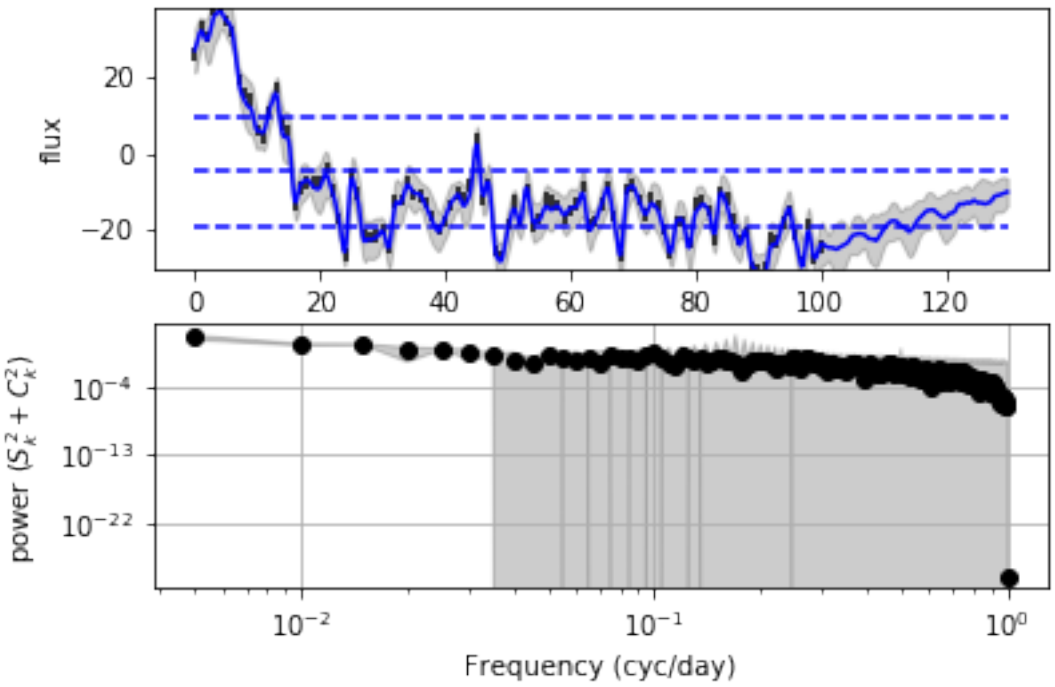
Modeling time series variations David Starkey

A sample of code similar to the bike challenge to forecast future (and current) behaviour of a time-evolving system.

In [1]:
```
#standard python modules
import numpy as np
import matplotlib.pylab as plt

#custom modules to generate synthetic variability and load the rw mod code
import myfitrw_092018 as mfr
import mylcgen as mlc
```

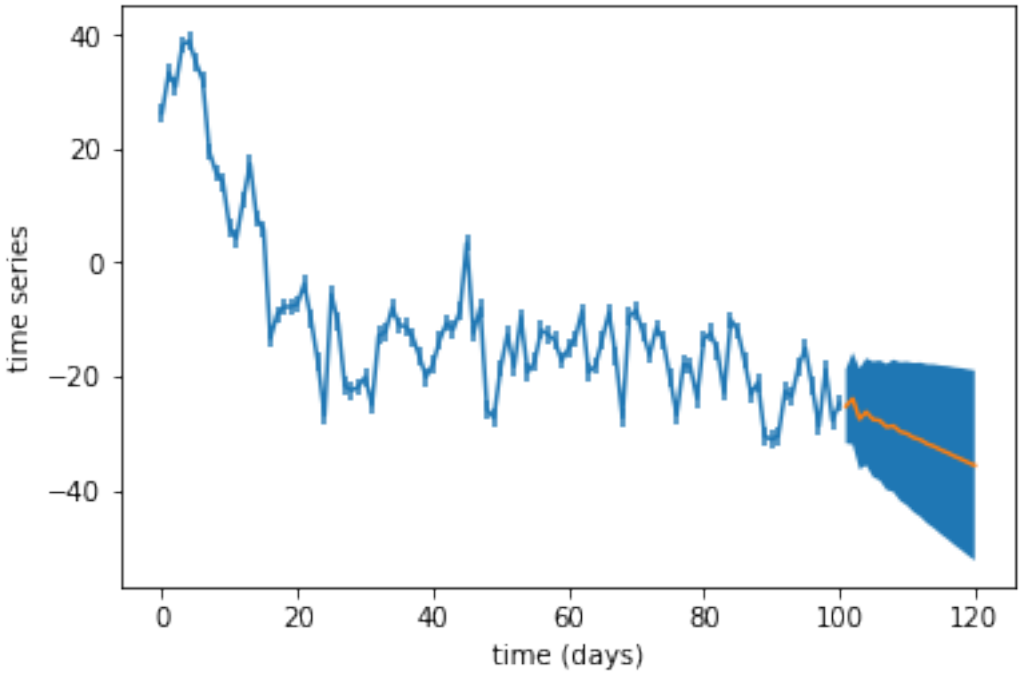
In [5]:
```
#generate some synthetic observations
dat = mlc.mylcgen(datfile='',p0=1.0,f0=0.1,a=-2,b=-2,tlo=0,thi=100,dt=1.0,ploton=0,ise
t = dat[:,0]
x = dat[:,1]
ndat = np.shape(t)[0]

#add noise
std = np.std(x)
x = np.random.randn(ndat)*0.1*std + x
sig = np.ones(ndat)*std*0.1


fit = mfr.fitrw([t],[x],[sig],floin=1./200,fhiin=1.0,
            plot_tit='show',dtresin=-1,nits = 1000,tplotlims=[0,130,0.2],extra_f=[])
```

```
covariance
(401, 401) (401,) (401,)
```



Can also use ARMA models for forecasting.

```
In [8]: from statsmodels.tsa.arima_model import ARIMA
        model = ARIMA(x, order=(5,1,0))
        model_fit = model.fit(disp=0)

        #forecast 20 days ahead
        nforecast = 20
        a = model_fit.forecast(steps=nforecast)
        predictions = a[0]
        sig_p = a[1]

        #plot the result
        fig = plt.figure()
        ax1 = fig.add_subplot(111)
        ax1.errorbar(t,x,sig)
        dx = np.mean(t[1:]-t[:-1])

        xfc = dx*np.arange(ndat,ndat+nforecast,1)
        ax1.plot(xfc,predictions)
        ax1.fill_between(xfc,predictions-sig_p,predictions+sig_p)
        ax1.set_xlabel('time (days)')
        ax1.set_ylabel('time series')
        plt.show()
```


40
20
0
−20
−40
time series
0
20
40
60
80
100
120
time (days)