starrotate Documentation

Release 0.0.1

Ruth Angus

Nov 24, 2019

Contents

starrotate is a tool for measuring stellar rotation periods using Lomb-Scargle (LS) periodograms, autocorrelation functions (ACFs) and Gaussian processes (GPs). It uses the astropy implementation of Lomb-Scargle periodograms, and the exoplanet implementation of fast celerite Gaussian processes.

starrotate is compatible with any light curve with time, flux and flux uncertainty measurements, including Kepler, K2 and TESS light curves. If your light curve is has evenly-spaced (or close to evenly-spaced) observations, all three of these methods: LS periodograms, ACFs and GPs will be applicable. For unevenly spaced light curves like those from the Gaia, or ground-based observatories, LS periodograms and GPs are preferable to ACFs.

CHAPTER 1

Example usage

```
import starrotate as sr

rotate = sr.RotationModel(time, flux, flux_err)
lomb_scargle_period = rotate.LS_rotation()
acf_period = rotate.ACF_rotation()
gp_period = rotate.GP_rotation()
```

CHAPTER 2

User Guide

2.1 Installation

Currently the best way to install *starrotate* is from github.

From source:

```
git clone https://github.com/RuthAngus/starrotate.git
cd starrotate
python setup.py install
```

2.1.1 Dependencies

The dependencies of *starrotate* are NumPy, pandas, h5py, tqdm, emcee, exoplanet, astropy, matplotlib, scipy, and kplr.

These can be installed using pip:

```
conda install numpy pandas h5py tqdm emcee exoplanet astropy matplotlib
scipy kplr
```

or

```
pip install numpy pandas h5py tqdm emcee exoplanet astropy matplotlib
scipy kplr
```

2.2 API documentation

CHAPTER 3

Tutorials

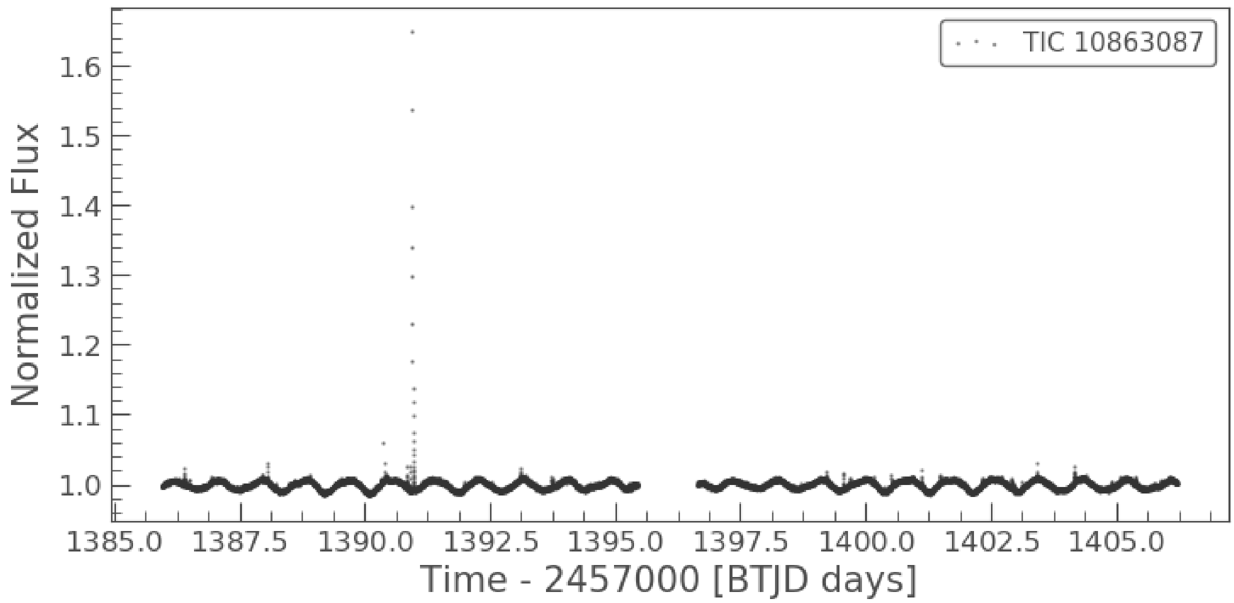
Note: This tutorial was generated from an IPython notebook that can be downloaded here.

3.1 A quick starrotate tutorial: measuring the rotation period of a TESS star

In this tutorial we'll measure the rotation period of a TESS target. First we'll download and plot a light curve using the lightkurve package.

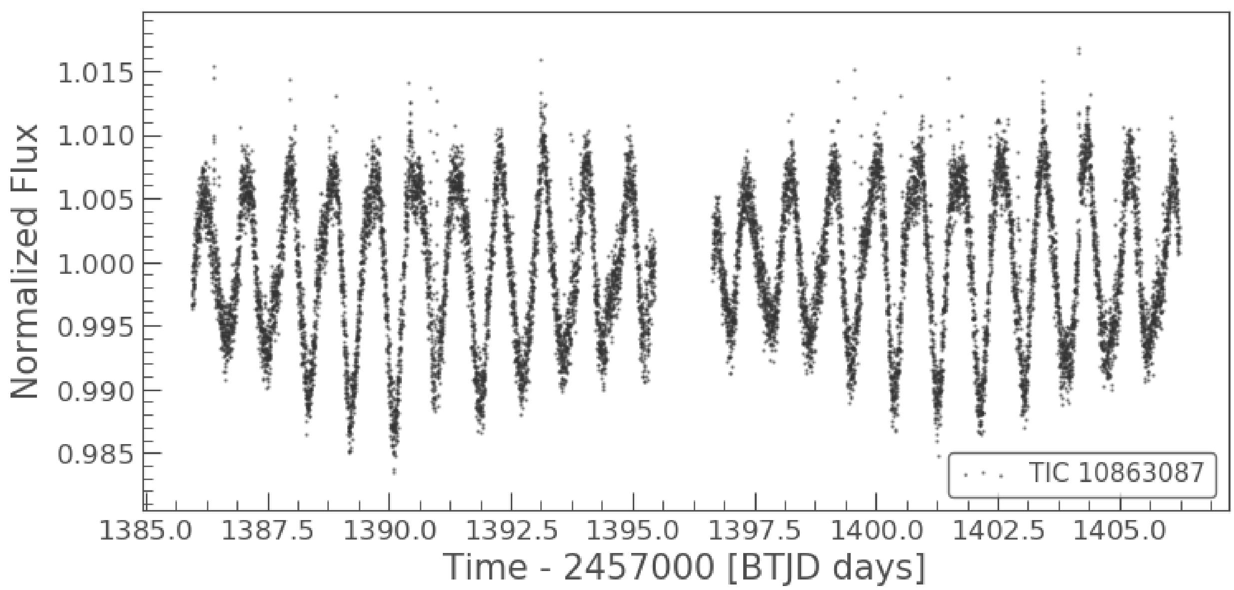
```
import numpy as np
import lightkurve as lk

starname = "TIC 10863087"
lcf = lk.search_lightcurvefile(starname).download()
```

```
lc = lcf.PDCSAP_FLUX
lc.scatter(alpha=.5, s=.5);
```



First of all, let's remove the flares which will limit our ability to measure a rotation period. Let's also get rid of any NaN values in the light curve.

```
no_nan_lc = lc.remove_nans()
clipped_lc = no_nan_lc.remove_outliers(sigma=3)
clipped_lc.scatter(alpha=.5, s=.5);
```

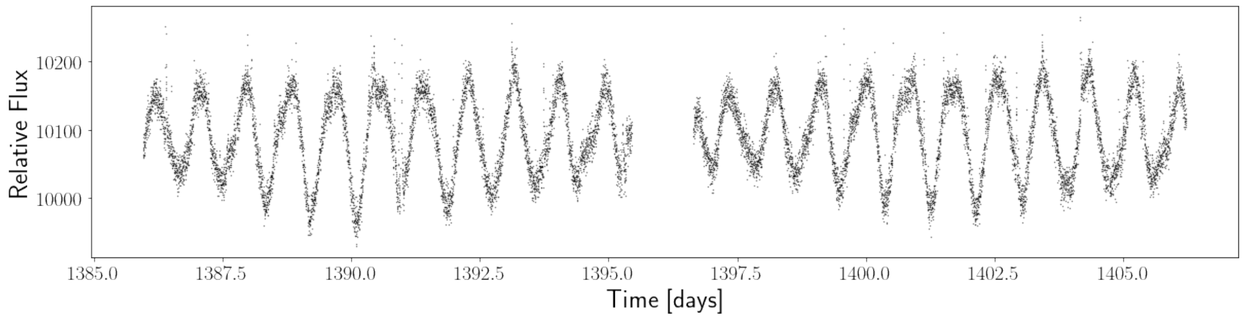


Next, let's import starrotate and set up a RotationModel object.

```
import starrotate as sr

rotate = sr.RotationModel(clipped_lc.time, clipped_lc.flux, clipped_lc.flux_err)
```

We can also plot the light curve using the plot_lc function in starrotate:

```
rotate.plot_lc()
```



Now let's measure a rotation period for this star using the astropy implementation of the Lomb-Scargle periodogram. This algorithm fits a single sinusoid to the light curve and reports the squared amplitude of the sinusoid over a range of frequencies (1/periods).

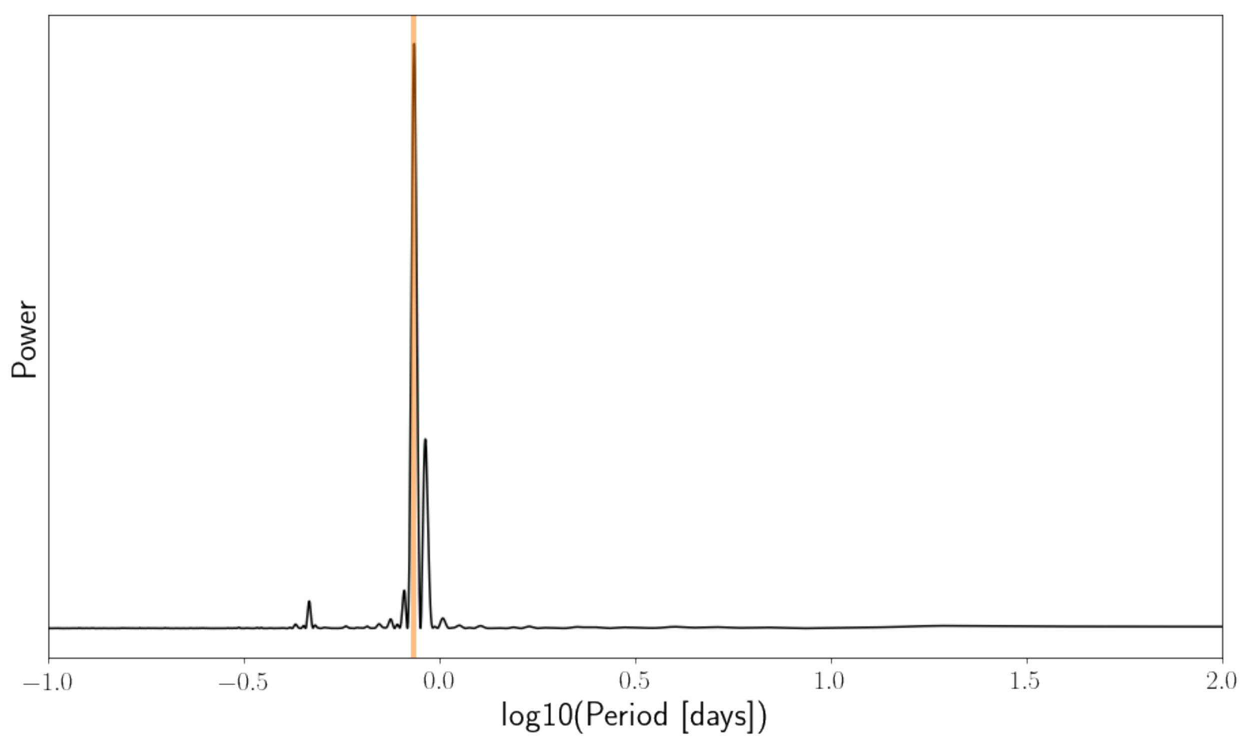
```
ls_period = rotate.LS_rotation()
```

```
ls_period
```

```
0.8607640045552087
```

We measured a rotation period of 0.86 days by finding the period of the highest peak in the periodogram. Let's plot the periodogram.

```
rotate.pgram_plot()
```



Now let's calculate an ACF and measure a rotation period by finding the highest peak.

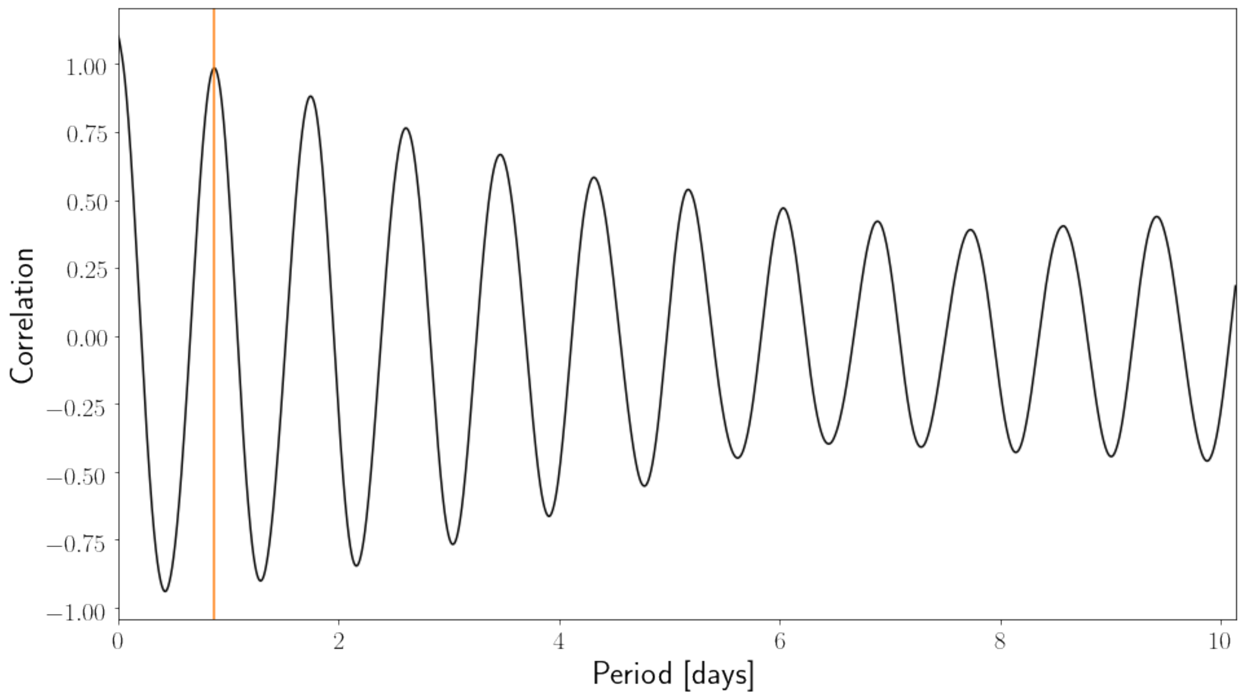
```
tess_cadence = 1./24./30.  # This is a TESS 2 minute cadence star.
acf_period = rotate.ACF_rotation(tess_cadence)
```

```
/Users/rangus/projects/starrotate/starrotate/rotation_tools.py:158:␣
↪FutureWarning: Using a non-tuple sequence for multidimensional indexing␣
↪is deprecated; use arr[tuple(seq)] instead of arr[seq]. In the future␣
↪this will be interpreted as an array index, arr[np.array(seq)], which will␣
↪result either in an error or a different result.
  acf = np.fft.ifft(f * np.conjugate(f), axis=axis)[m].real
```

```
acf_period
```

```
0.8763888888888888
```

```
rotate.acf_plot()
```



This method estimates a period of 0.88 days, which is very close to the periodogram method. It is important to note that the LS periodogram method and the ACF method are not independent, i.e. if you measure a certain rotation period with one, you are likely to measure the same rotation period with the other. These two methods should not be used as independent 'checks' to validate a measured rotation period.

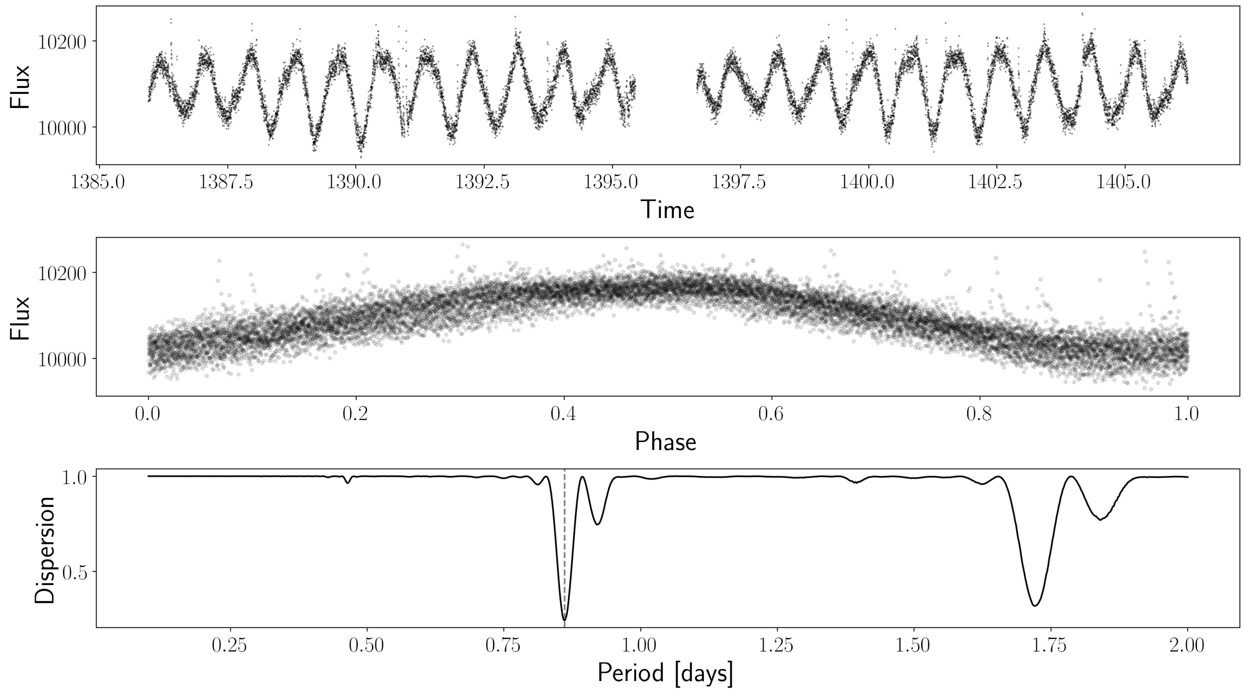
Now, let's calculate a rotation period using the Phase Dispersion Minimization algorithm of Stellingwerf (1978). This function will return the period with the lowest phase dispersion.

```
period_grid = np.linspace(.1, 2, 1000)
pdm_period = rotate.PDM(period_grid, pdm_nbins=10)  # Set the number of bins to 10
print(pdm_period)
```

```
100%|| 1000/1000 [00:05<00:00, 176.93it/s]
```

```
0.8607607607607607
```

```
rotate.PDM_plot()
```



The Lomb-Scargle periodogram, ACF, and phase dispersion arrays are accessible via:

```
# Lomb-Scargle periodogram
period_array = 1./rotate.freq
power_array = rotate.power

# Autocorrelation function
ACF_array = rotate.acf
lag_array = rotate.lags

# Phase-dispersion minimization
phi_array = rotate.phis  # The 'dispersion' plotted in the lower panel above.
period_grid = period_grid  # We already defined this above.
```

These could come in handy because it might be useful to calculate various peak statistics. We can do that with the get_peak_statistics() function in rotation_tools, e.g.

```
import starrotate.rotation_tools as rt

# Get peak positions and heights, in order of highest to lowest peak.
peak_positions, peak_heights = rt.get_peak_statistics(1./rotate.freq, rotate.power)
print(peak_positions[0]) # This is the period of the highest peak (which is the␣
↪default LS period)
```

```
0.8607640045552087
```

For the ACF peak statistics, we might choose either the highest peak as the period (default in starrotate):

```
# Get peak positions and heights, in order of highest to lowest peak.
acf_peak_positions, acf_peak_heights = rt.get_peak_statistics(rotate.lags, rotate.acf,
↪ sort_by="height")
print(acf_peak_positions[0])
```

```
0.8763888888888888
```

Or the first peak:

```
# Get peak positions and heights, in order of lags.
acf_peak_positions, acf_peak_heights = rt.get_peak_statistics(rotate.lags, rotate.acf,
↪ sort_by="position")
print(acf_peak_positions[0])
```

```
0.8763888888888888
```

In this example the first and the highest peak are the same.

Finally, let's measure a rotation period with the exoplanet implementation of a celerite Gaussian process. This part takes a little while to run.

```
gp_results = rotate.GP_rotation()
```

```
success: False
initial logp: -54886.575693444596
final logp: -53829.8979901813
sampling...
```

```
Sampling 4 chains: 100%|| 308/308 [01:22<00:00,  1.05s/draws]
Sampling 4 chains: 100%|| 108/108 [00:20<00:00,  5.15draws/s]
Sampling 4 chains: 100%|| 208/208 [00:16<00:00,  6.30draws/s]
Sampling 4 chains: 100%|| 408/408 [00:26<00:00,  6.52draws/s]
Sampling 4 chains: 100%|| 808/808 [02:00<00:00,  3.12draws/s]
Sampling 4 chains: 100%|| 1608/1608 [2:02:57<00:00,  4.79draws/s]
Sampling 4 chains: 100%|| 4608/4608 [03:32<00:00, 21.66draws/s]
Sampling 4 chains: 100%|| 208/208 [00:13<00:00,  7.44draws/s]
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [mix, logdeltaQ, logQ0, logperiod, logamp, logs2, mean]
Sampling 4 chains:  54%|    | 4303/8000 [02:08<01:50, 33.49draws/s]
The number of effective samples is smaller than 25% for some parameters.
```

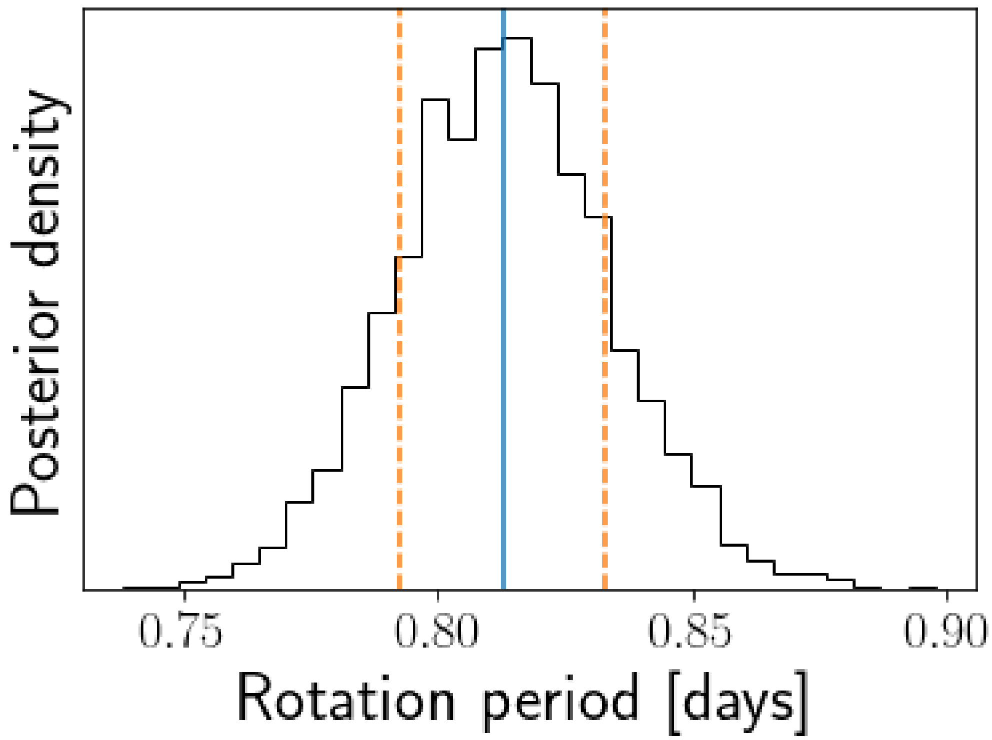
We can print the resulting GP rotation period and its associated uncertainties:

```
print("GP period = {0:.2f} + {1:.2f} - {2:.2f}".format(rotate.gp_period, rotate.errp,␣
↪rotate.errm))
```

```
GP period = 0.81 + 0.02 - 0.02
```

And plot the posterior PDF:

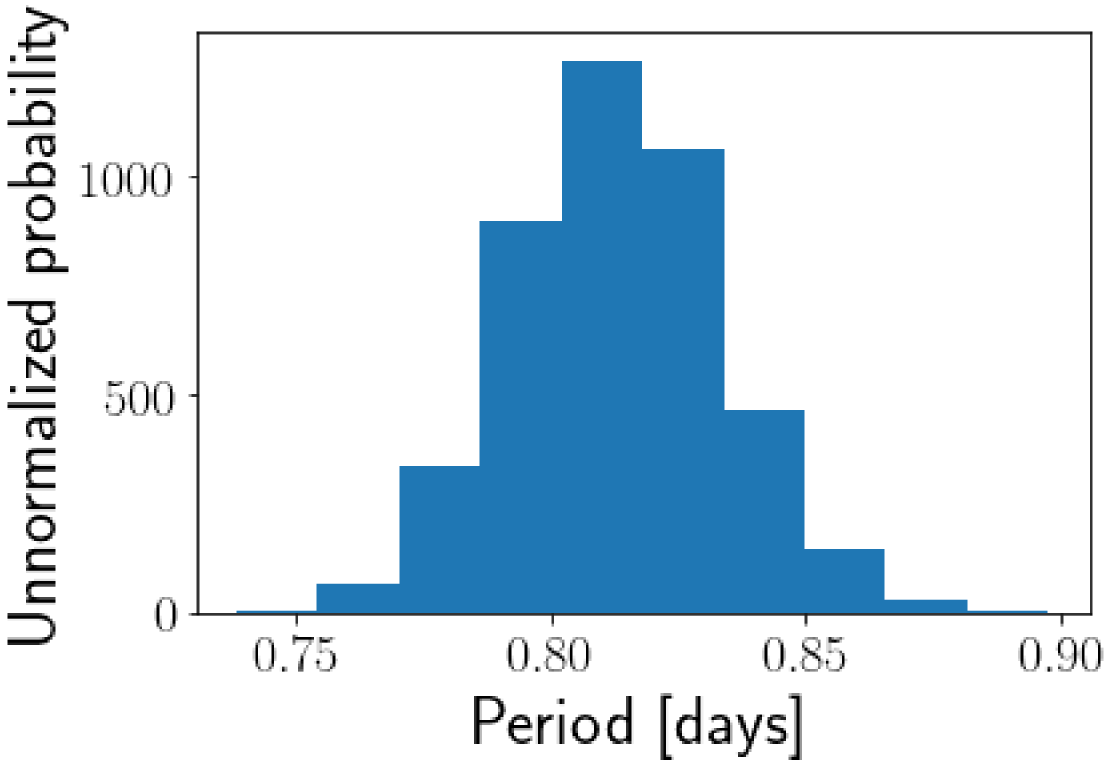
```
rotate.plot_posterior()
```



We can also plot this manually:

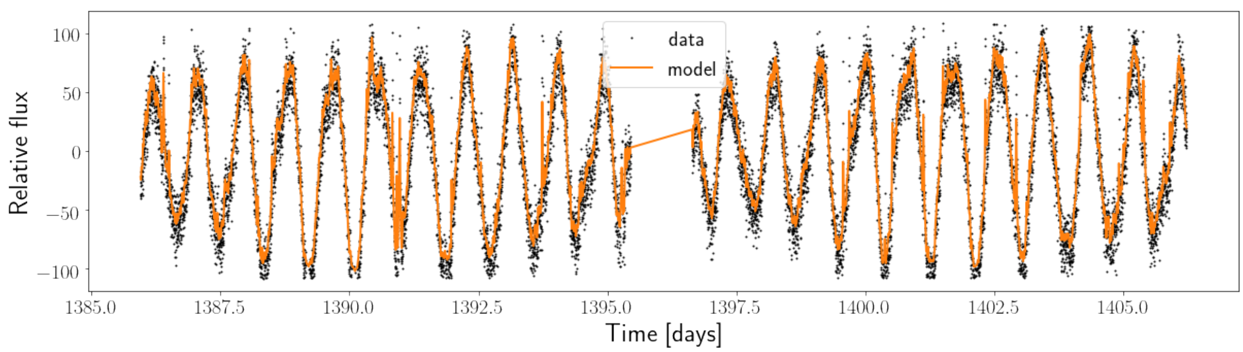
```
import matplotlib.pyplot as plt
%matplotlib inline

plt.hist(rotate.period_samples);
plt.xlabel("Period [days]")
plt.ylabel("Unnormalized probability")
```

```
Text(0, 0.5, 'Unnormalized probability')
```



And we can plot the posterior prediction:

```
rotate.plot_prediction()
```



You can see that there are still outliers in the light curve produced by flares which is affecting the GP fit. A better outlier removal algorithm would improve this fit!

CHAPTER 4

License & attribution

If you make use of this code, please cite this package and its dependencies. You can find more information about how and what to cite in the citation documentation.

- search

Python Module Index

s

Index

S