starspot Documentation

Release 0.0.1

Ruth Angus

Sep 08, 2020

Contents

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

starspot 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 numpy as np
import starspot as ss

# Generate some data
time = np.linspace(0, 100, 10000)
period = 10
w = 2*np.pi/period
flux = np.sin(w*time) + np.random.randn(len(time))*1e-2 + \
    np.random.randn(len(time))*.01
flux_err = np.ones_like(flux)*.01

rotate = ss.RotationModel(time, flux, flux_err)

# Calculate the Lomb Scargle periodogram period (highest peak in the periodogram).
lomb_scargle_period = rotate.ls_rotation()

# Calculate the autocorrelation function (ACF) period (highest peak in the ACF).
# This is for evenly sampled data only -- time between observations is 'interval'.
acf_period = rotate.acf_rotation(interval=np.diff(time)[0])

# Calculate the phase dispersion minimization period (period of lowest dispersion).
period_grid = np.linspace(5, 20, 1000)
pdm_period = rotate.pdm_rotation(period_grid)

print(lomb_scargle_period, acf_period, pdm_period)
>> 9.99892010582963 10.011001100110011 10.0

# Calculate a Gaussian process rotation period
gp_period = rotate.GP_rotation()
```

CHAPTER 2

User Guide

2.1 Installation

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

From source:

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

2.1.1 Dependencies

The dependencies of *starspot* 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.

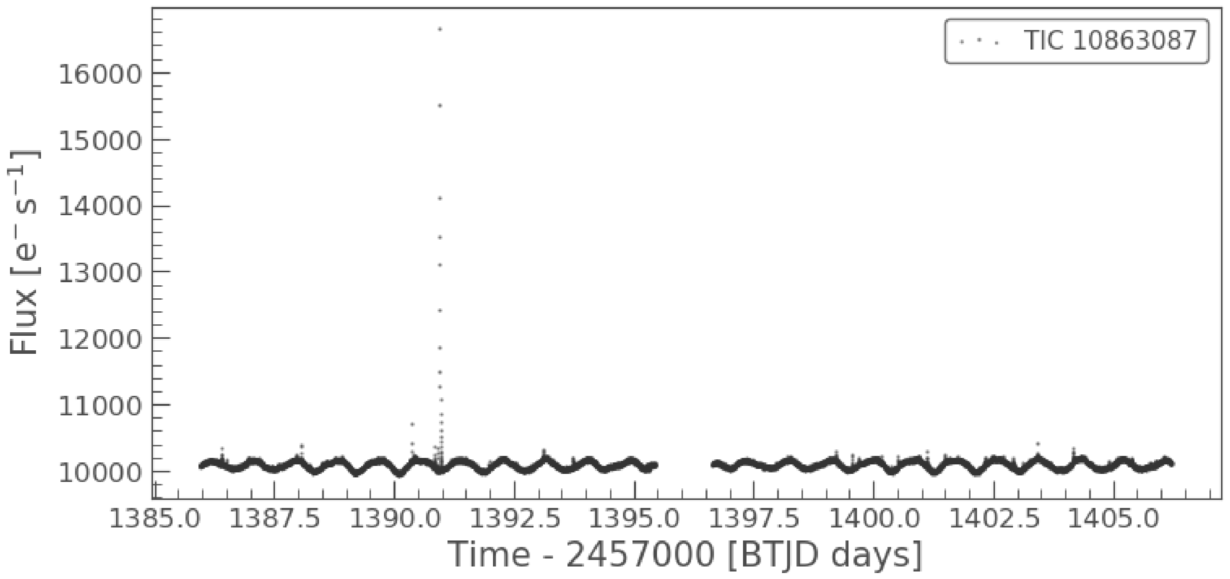
3.1 A quick starspot tutorial: measuring the rotation period of a TESS star

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()
```

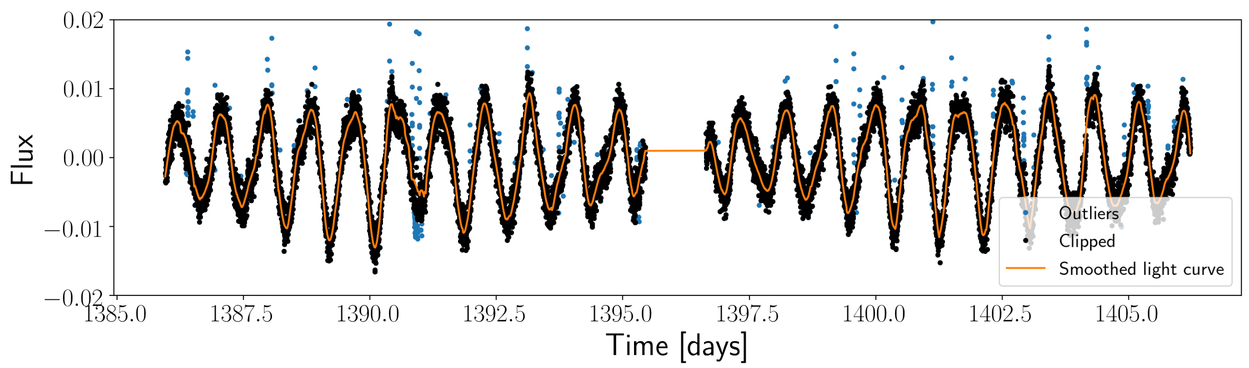
```
Warning: 31% (6168/19692) of the cadences will be ignored due to the quality mask␣
↪(quality_bitmask=175).
```

```
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 by sigma clipping.

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

# Calculate the median so that we can median-normalize.
med = np.median(lc.flux)

# Do an initial sigma clip to remove big outliers.
m = ss.sigma_clip(lc.flux/med - 1, nsigma=6)
x, y, yerr = lc.time[m], lc.flux[m]/med - 1, lc.flux_err[m]/med

# Then a sigma clip using a Sav-Gol filter for smoothing
mask, smooth = ss.filter_sigma_clip(x, y, window_length=199)

time, flux, flux_err = x[mask], y[mask], yerr[mask]

plt.figure(figsize=(16, 4), dpi=200)
plt.plot(lc.time, lc.flux/med-1, ".", label="Outliers")
plt.plot(time, flux, "k.", label="Clipped")
plt.plot(x, smooth, label="Smoothed light curve")
plt.xlabel("Time [days]")
plt.ylabel("Flux");
plt.ylim(-.02, .02);
plt.legend(loc="lower right", fontsize=15);
```

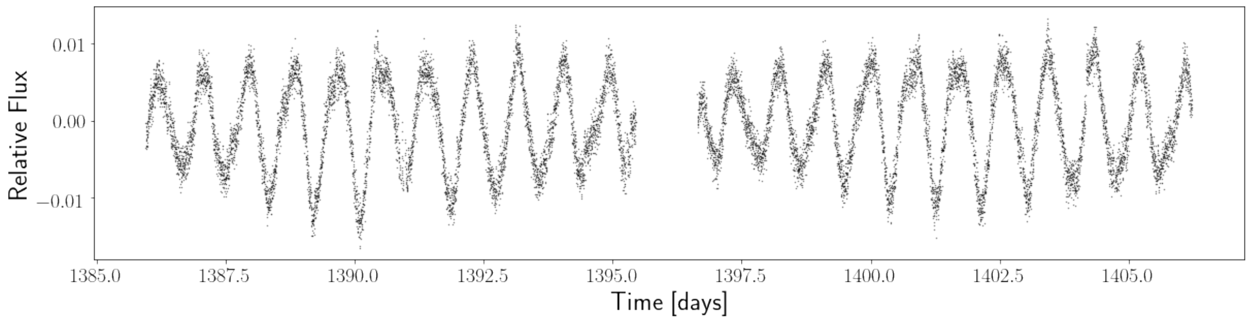


Next, let's import starspot and set up a RotationModel object.

```
import starspot as ss

rotate = ss.RotationModel(time, flux, flux_err)
```

We can also plot the light curve using the lc_plot function in starspot:

```
rotate.lc_plot()
```



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).

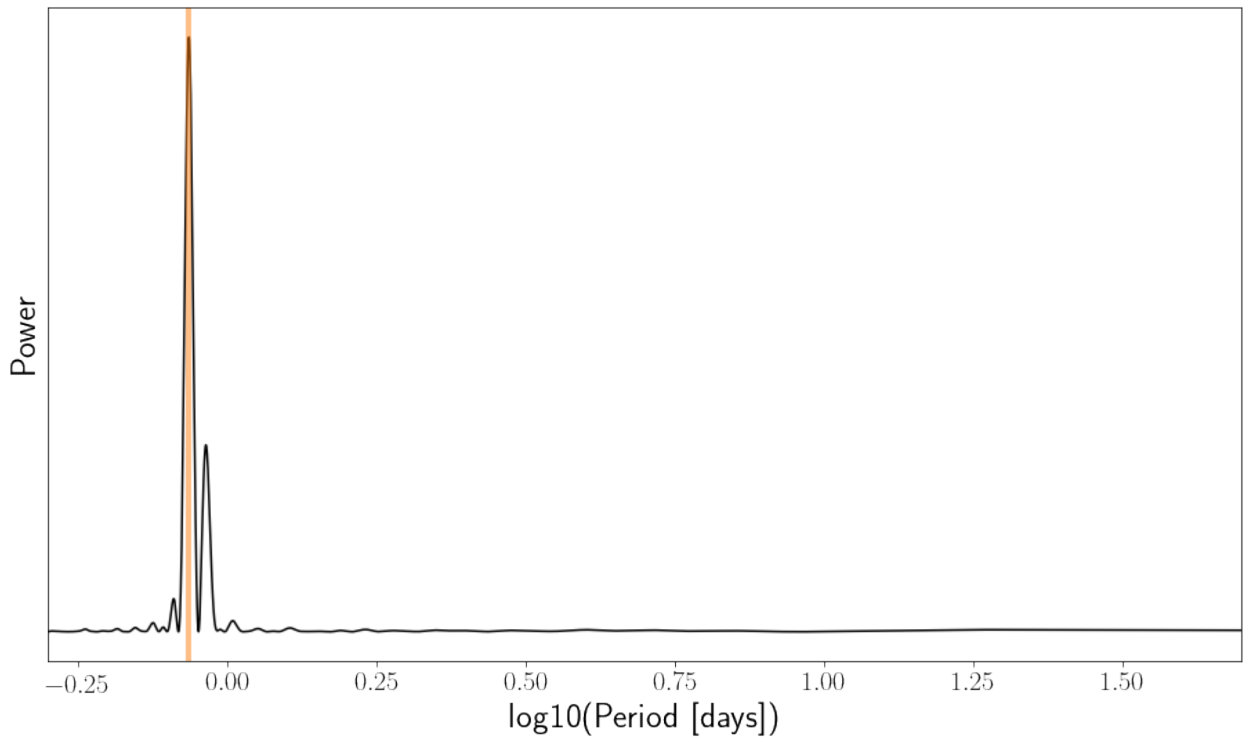
```
ls_period = rotate.ls_rotation()
```

```
ls_period
```

```
0.860808017577187
```

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.ls_plot();
```



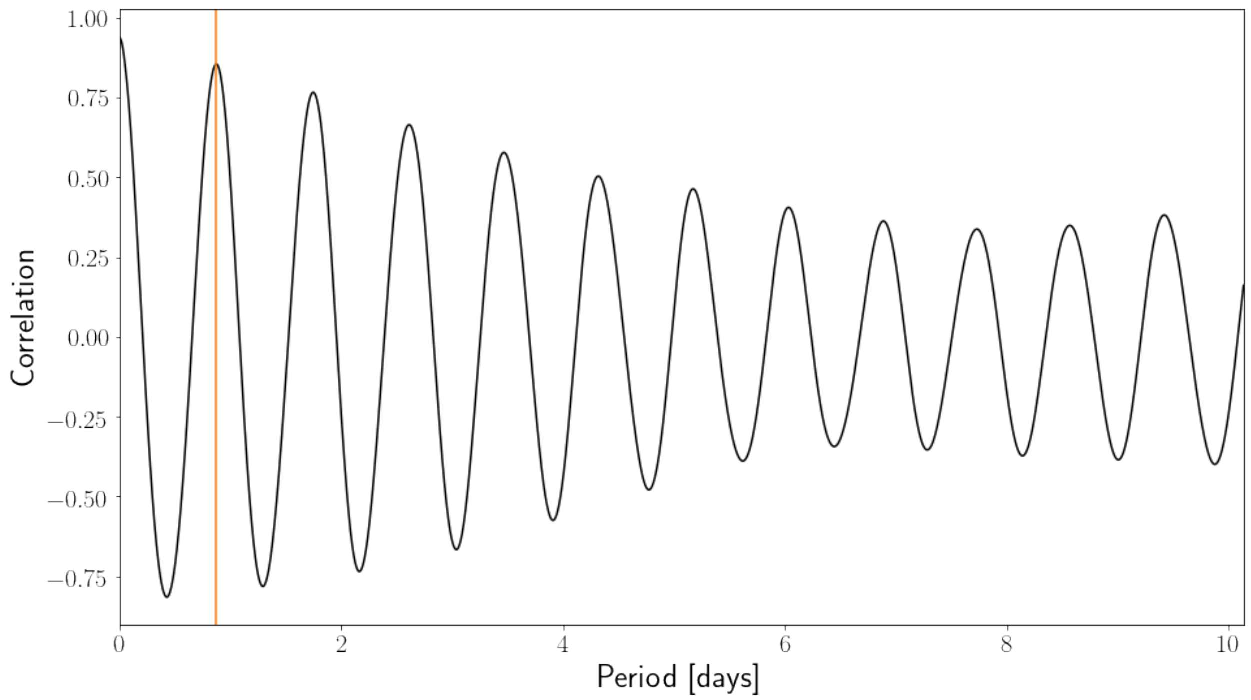
Now let's calculate an ACF and measure a rotation period by finding the highest peak.

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

```
acf_period
```

```
0.8749999999999999
```

```
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.

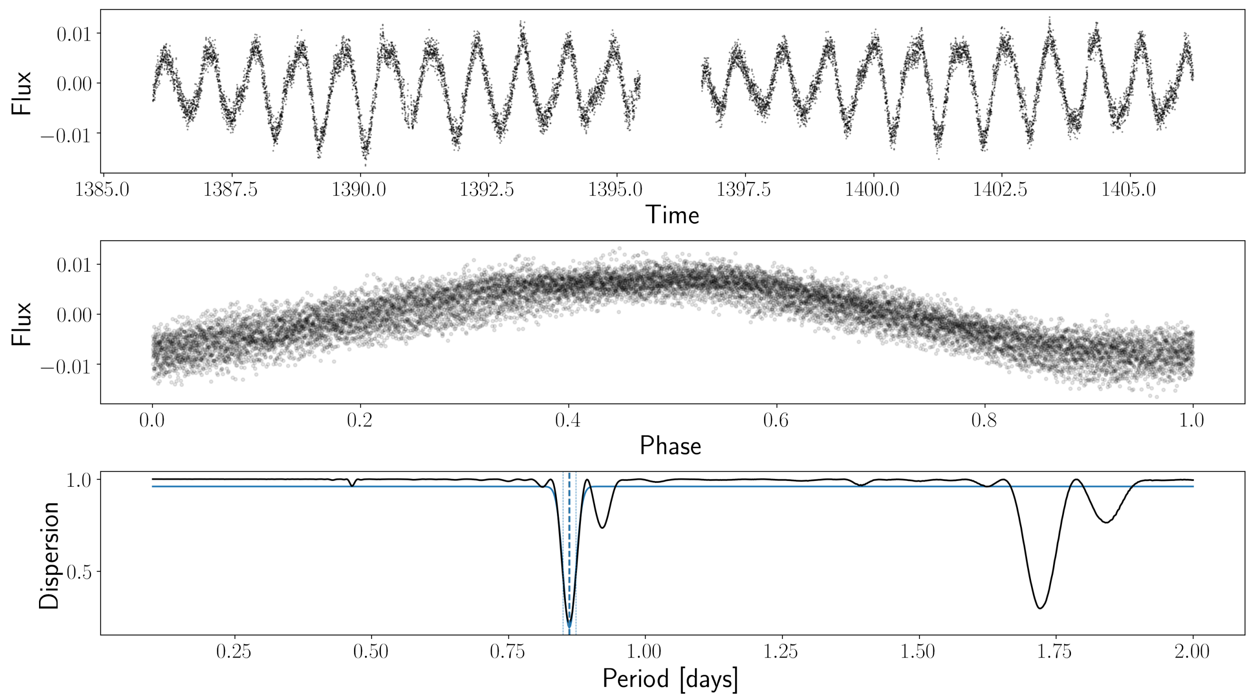
Now, let's calculate a rotation period using the Phase Dispersion Minimization algorithm of Stellingwerf (1978). This function will return the period with the lowest phase dispersion. It also fits a Gaussian to the dispersion curve in order to estimate the uncertainty. This Gaussian is shown in blue in the lower panel.

```
period_grid = np.linspace(.1, 2, 1000)

# Set the number of bins to 10
pdm_period, period_err = rotate.pdm_rotation(period_grid, pdm_nbins=10)
print(pdm_period, period_err)
```

```
100%|| 1000/1000 [00:05<00:00, 168.10it/s]
```

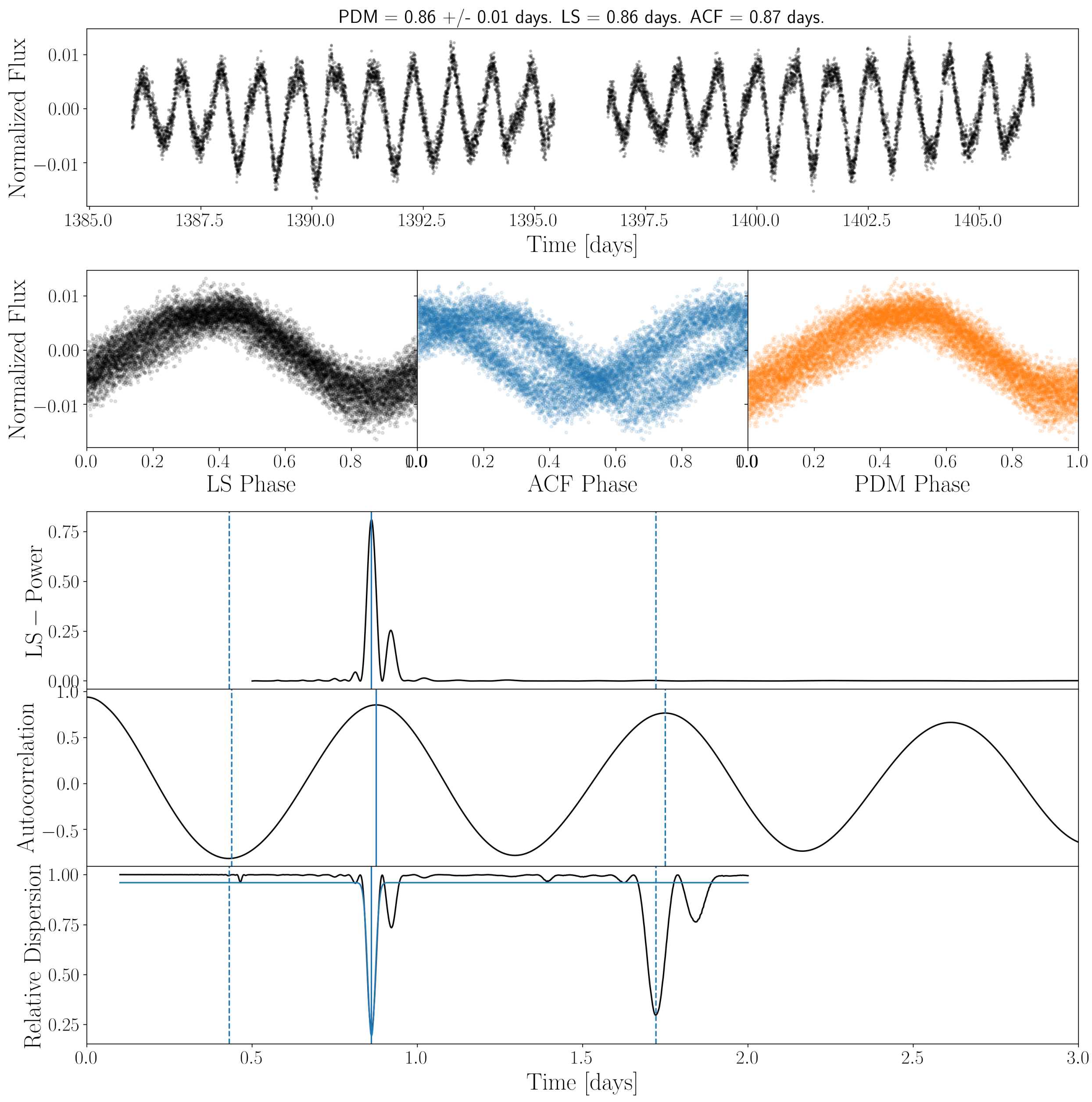
```
0.8607607607607607 0.011651757899962235
```

```
rotate.pdm_plot();
```



Now we can plot all these methods together in a single figure, with the light curve folded on the three different periods using big_plot:

```
# Provide the list of methods and xlimits for the bottom panels
fig = rotate.big_plot(["ls", "acf", "pdm"], method_xlim=(0, 3));
```



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.

```
# Get peak positions and heights, in order of highest to lowest peak.
peak_positions, peak_heights = ss.get_peak_statistics(1./rotate.freq, rotate.power)

# This is the period of the highest peak (which is the default LS period)
print(peak_positions[0])
```

```
0.860808017577187
```

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 = ss.get_peak_statistics(rotate.lags,
                                                               rotate.acf,
                                                               sort_by="height")
print(acf_peak_positions[0])
```

```
0.8749999999999999
```

Or the first peak:

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

```
0.8749999999999999
```

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

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

Index