Exoplanet Finder: Unveiling New Worlds Beyond

The Exoplanet Finder project is a thrilling journey into the cosmos, driven by a passion for discovering new worlds beyond our own. By combining the power of data analysis and astronomical insights, this project is dedicated to identifying exoplanets – celestial bodies orbiting stars beyond our solar system. Here's a glimpse into the remarkable process:

Detecting Exoplanets with Precision
Step 1: Data Retrieval
The quest begins with the retrieval of astronomical data using specialized tools like Lightkurve. By pinpointing specific target stars or regions of interest, we gather valuable light curve data from missions like TESS.

Step 2: Light Curve Analysis
The heart of the project lies in the analysis of light curves. We utilize techniques to isolate and study the minute variations in a star's brightness over time. These variations may hold the key to uncovering exoplanetary transits.

Step 3: Stellar Variability Mitigation
To ensure the accuracy of our findings, we employ methods to mitigate stellar variability. This involves the removal of noise and anomalies caused by phenomena other than exoplanetary transits.

Step 4: Periodogram Analysis
By subjecting our pre-processed light curves to periodogram analysis, we search for periodic signals that could indicate the presence of exoplanets. The periodogram reveals potential exoplanetary candidates.

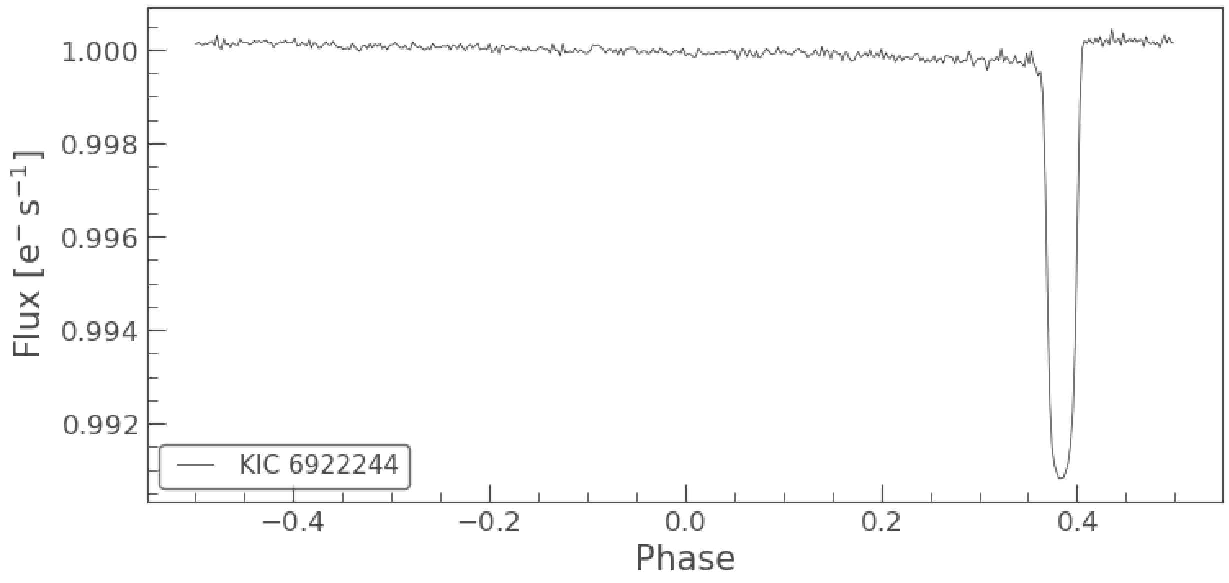
Step 5: Confirmation and Characterization
The most promising candidates undergo rigorous examination to confirm their exoplanetary nature. Additional data and analysis techniques help us characterize these newfound exoplanets, unveiling their properties and orbital dynamics.

Technology at Our Disposal

Python: The primary programming language used for data retrieval, analysis, and visualization.
Lightkurve: A vital tool for working with astronomical time series data from missions like TESS.
NumPy and SciPy: Essential libraries for data manipulation and scientific computing.
Periodogram Analysis: Techniques like Box Least Squares (BLS) are employed for period detection.
Data Visualization: Matplotlib aids in creating informative plots and visualizations.
Stellar Variability Mitigation: Robust techniques are applied to remove noise from light curves.

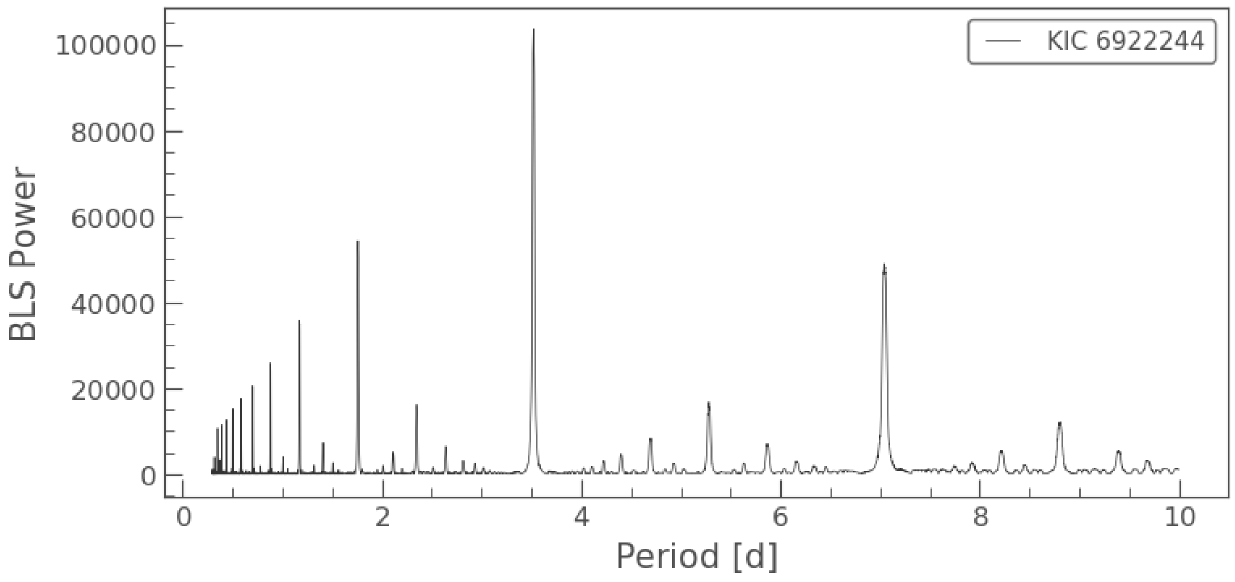
Project:

```
lc.remove_nans().flatten(window_length=401).fold(period=3.5225).bin(binsize=1
0).plot();
```



In [ ]:

```
import numpy as np
periodogram = flat_lc.to_periodogram(method="bls", period=np.arange(0.3, 10,
0.001))
periodogram.plot();
```



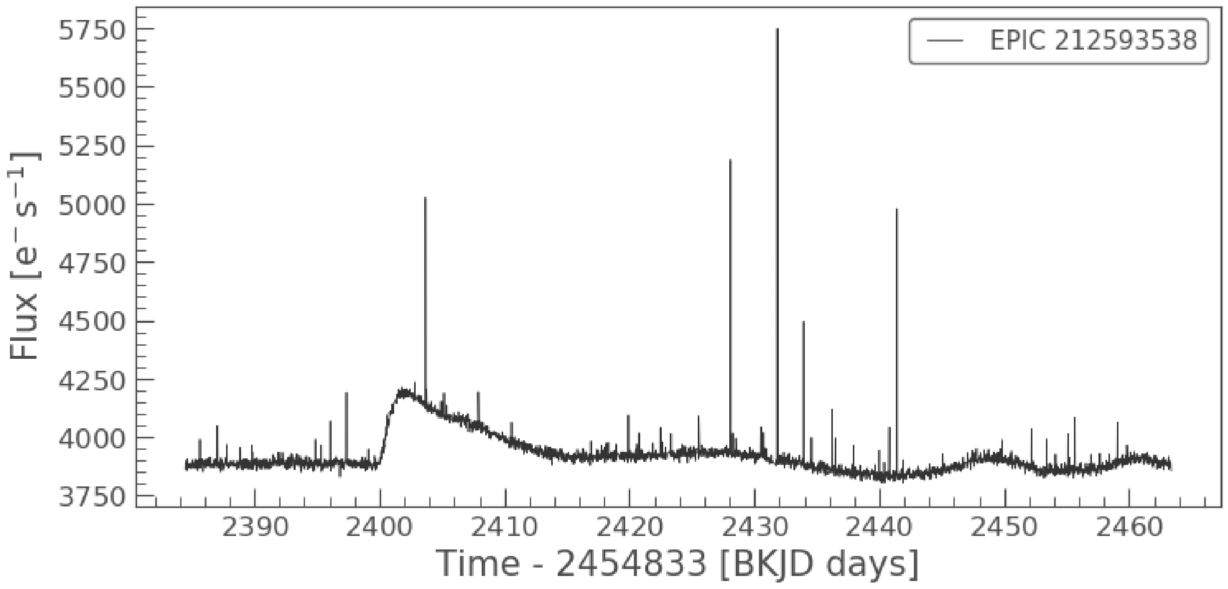
In [ ]:

```
best_fit_period = periodogram.period_at_max_power
print('Best fit period: {:.5f}'.format(best_fit_period))
```

```
Best fit period: 3.52200 d
```

In [ ]:

```
tpf = search_targetpixelfile('EPIC 212593538', campaign=6).download()
lc = tpf.to_lightcurve(aperture_mask='all')
lc.plot();
```



In [ ]:

```
from lightkurve import search_targetpixelfile
search_result = lk.search_targetpixelfile('Pi Mensae', mission='TESS', sector=1)
tpf = search_result.download(quality_bitmask='default')
```

In [ ]:

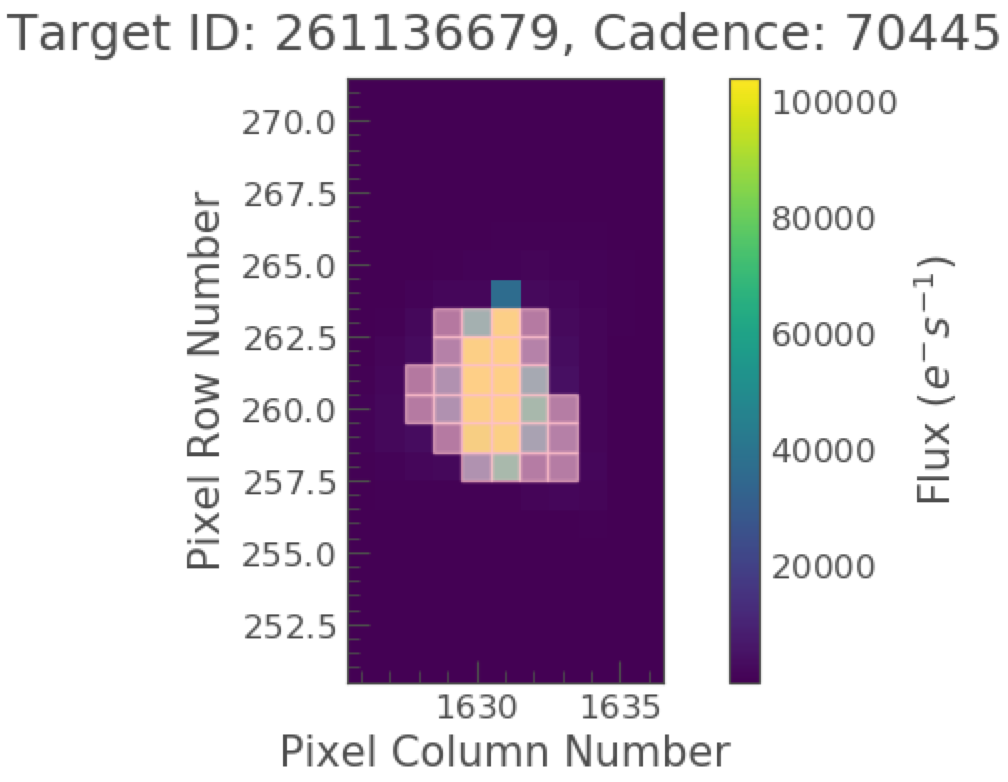
```
tpf.mission
tpf.targetid
```

Out[ ]:

```
261136679
```

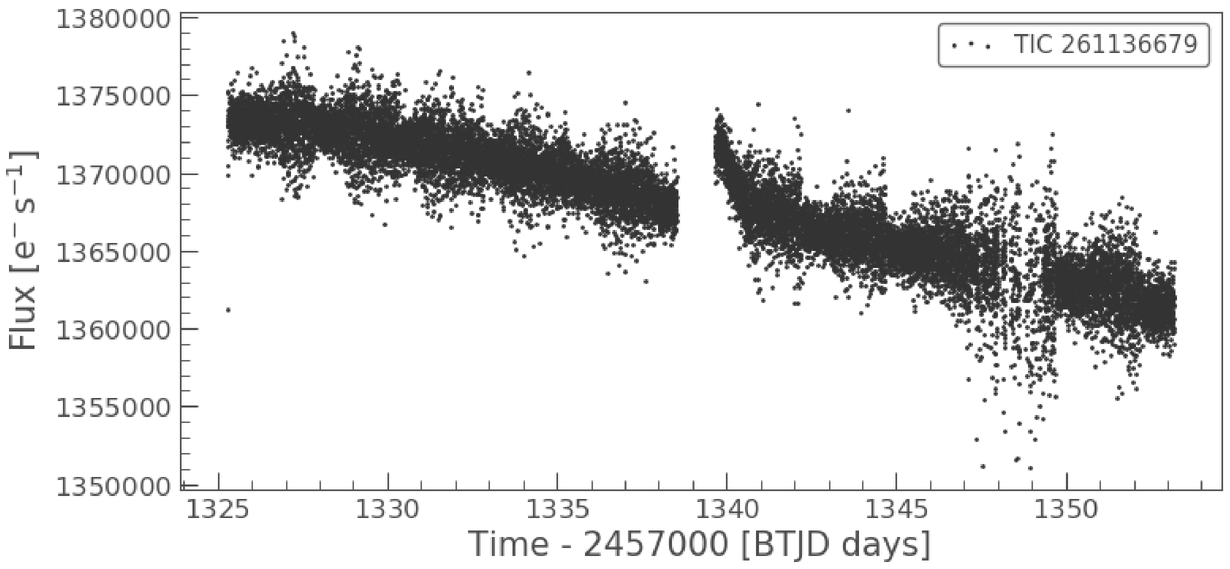
In [ ]:

```
tpf.plot(aperture_mask=tpf.pipeline_mask);
```



In [ ]:

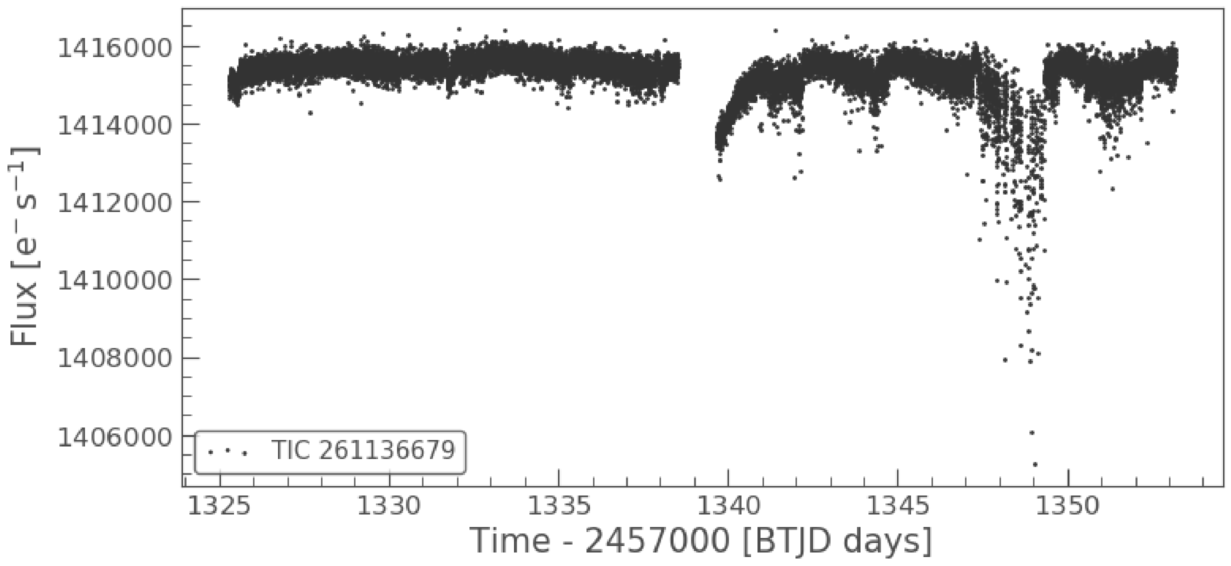
```
lc = tpf.to_lightcurve()
lc.scatter();
```



In [ ]:

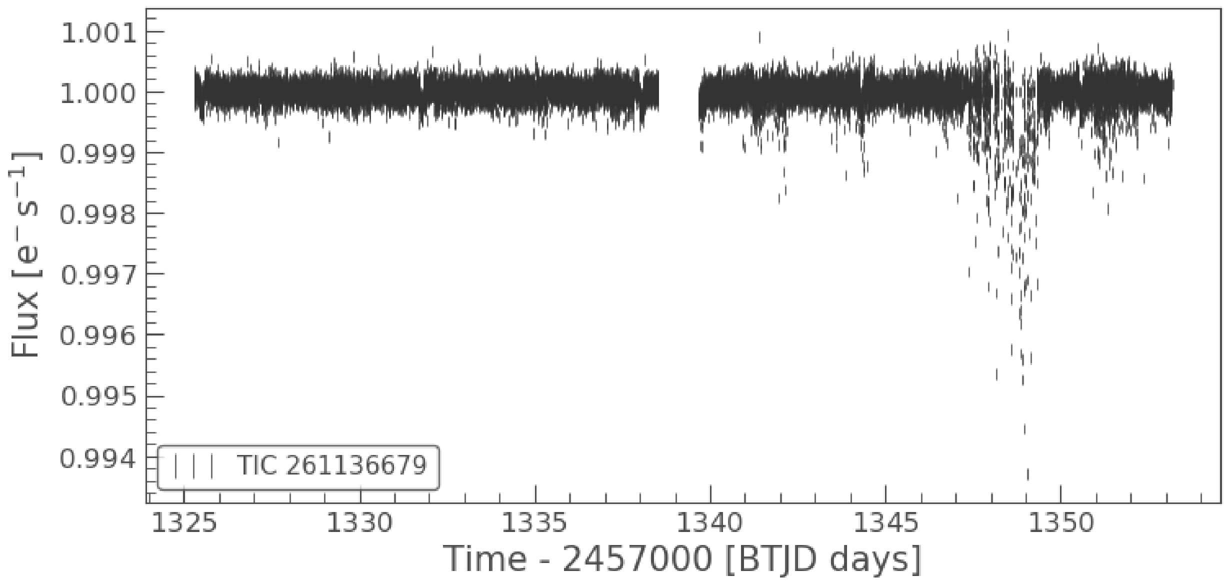
```
aperture_mask = tpf.create_threshold_mask(threshold=10)
lc = tpf.to_lightcurve(aperture_mask=aperture_mask)
```

```
lc.scatter();
```



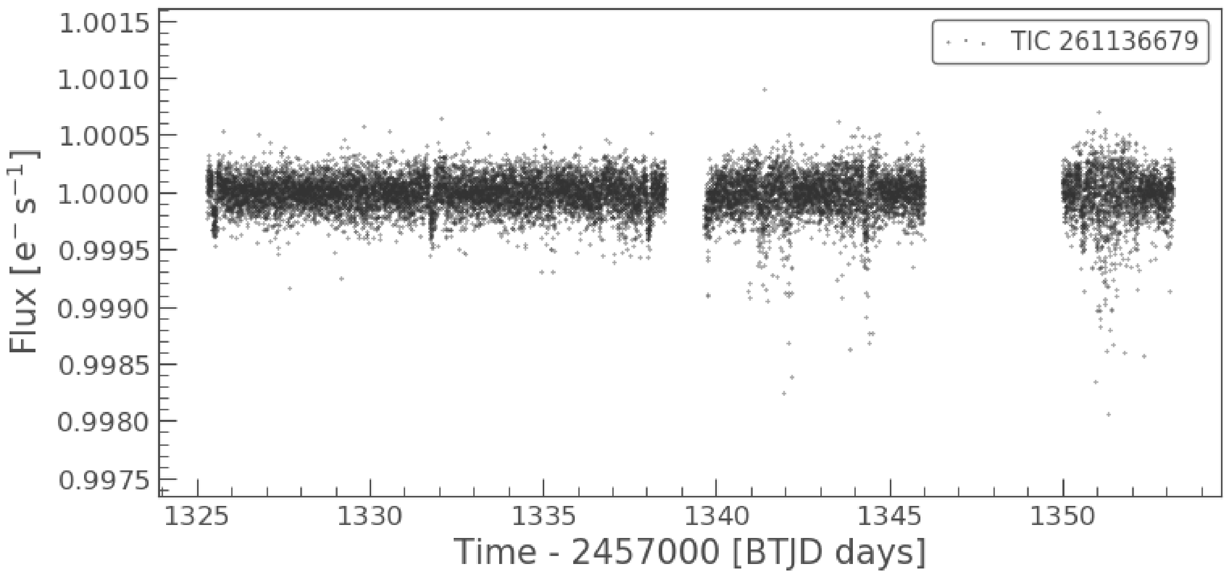
In [ ]:

```
flat_lc = lc.flatten(window_length=1001)
flat_lc.errorbar();
```



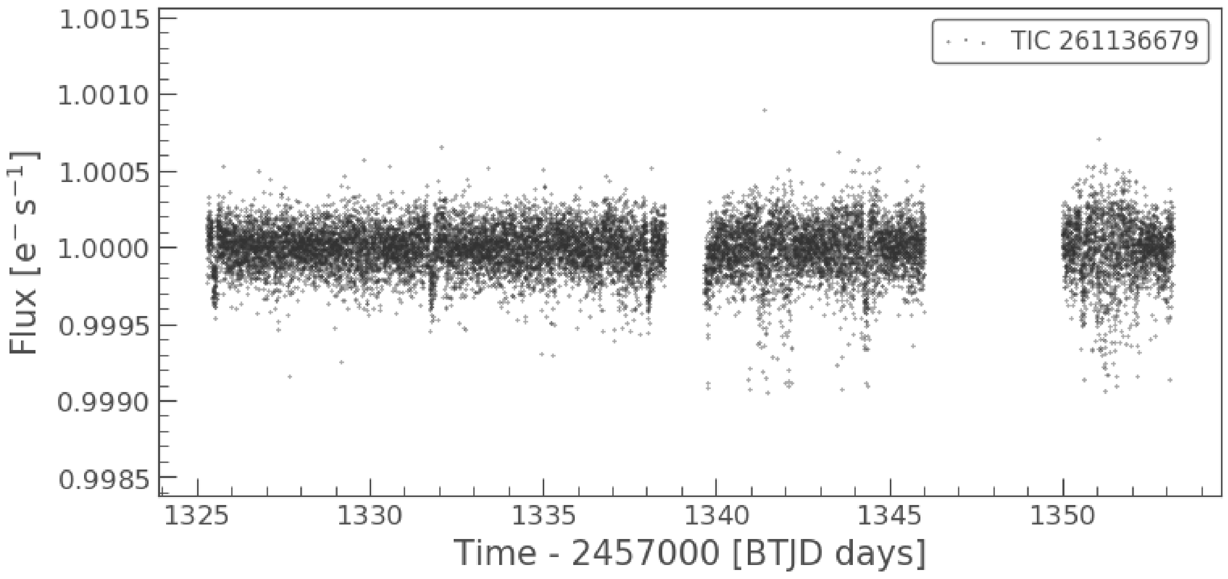
In [ ]:

```
mask = (flat_lc.time < 1346) | (flat_lc.time > 1350)
masked_lc = flat_lc[mask]
masked_lc.scatter(s=0.1);
```



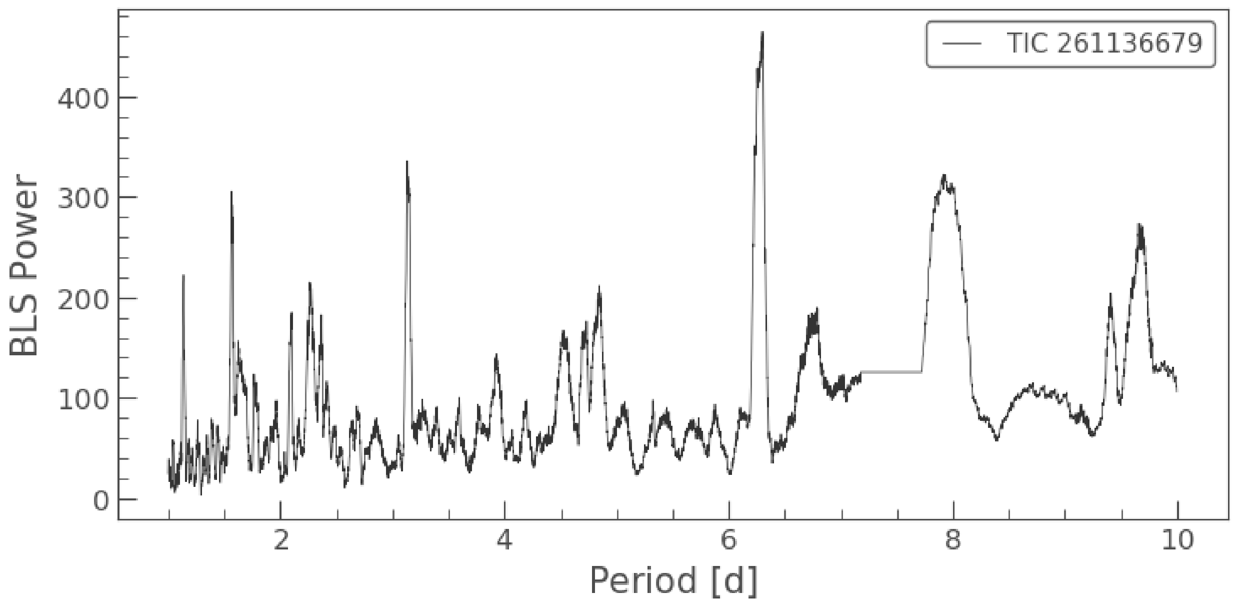
In [ ]:

```
clipped_lc = masked_lc.remove_outliers(sigma=6)
clipped_lc.scatter(s=0.1);
```



In [ ]:

```
import numpy as np
periodogram = clipped_lc.to_periodogram(method="bls", period=np.arange(1,
10, 0.001))
periodogram.plot();
```



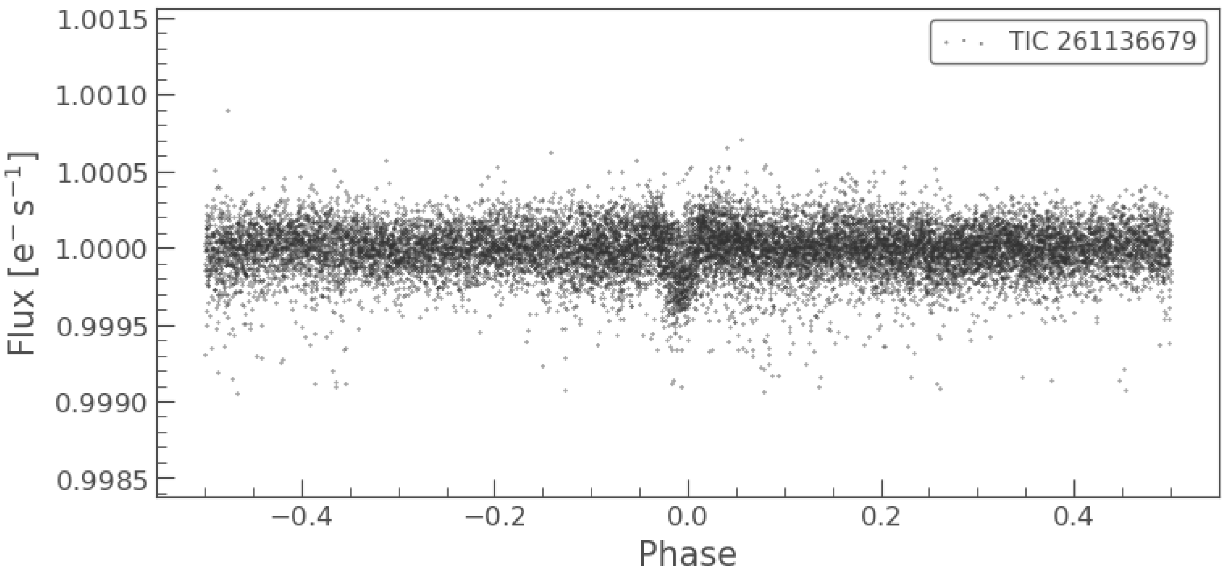
In [ ]:

```
best_fit_period = periodogram.period_at_max_power
print('Best fit period: {:.3f}'.format(best_fit_period))
```

```
Best fit period: 6.300 d
```

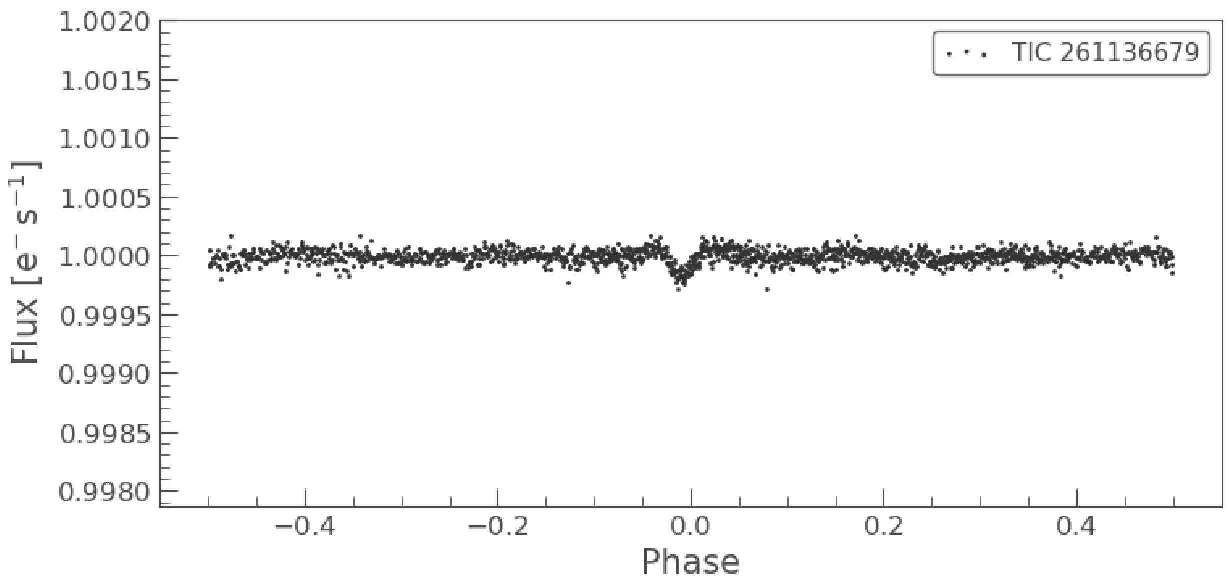
In [ ]:

```
folded_lc = clipped_lc.fold(period=6.300, t0=1325.504)
folded_lc.scatter(s=0.1);
```



In [ ]:

```
binned_lc = folded_lc.bin(binsize=10)
binned_lc.scatter();
```



In [ ]:

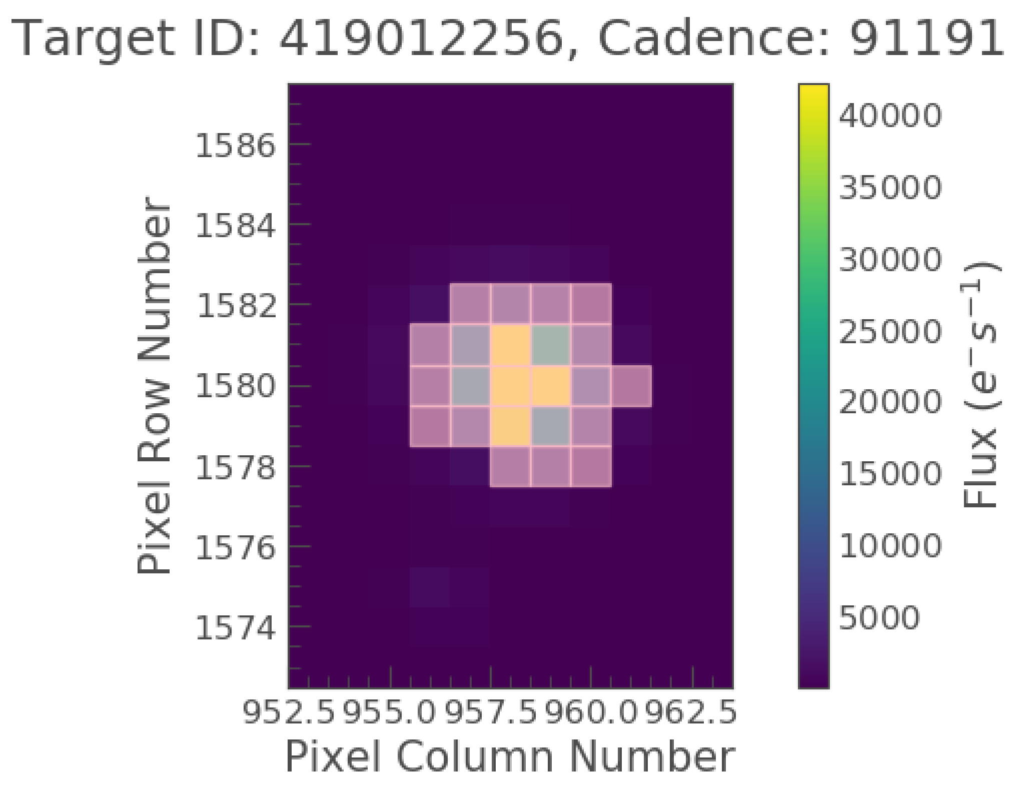
```
from lightkurve import TessTargetPixelFile
import lightkurve as lk
tpf = 
TessTargetPixelFile("/home/anton/Documents/Jupyter/tess2018234235059-s000
2-0000000419012256-0121-s_tp.fits")
tpf
```

Out[ ]:

```
TessTargetPixelFile(TICID: 419012256)
```

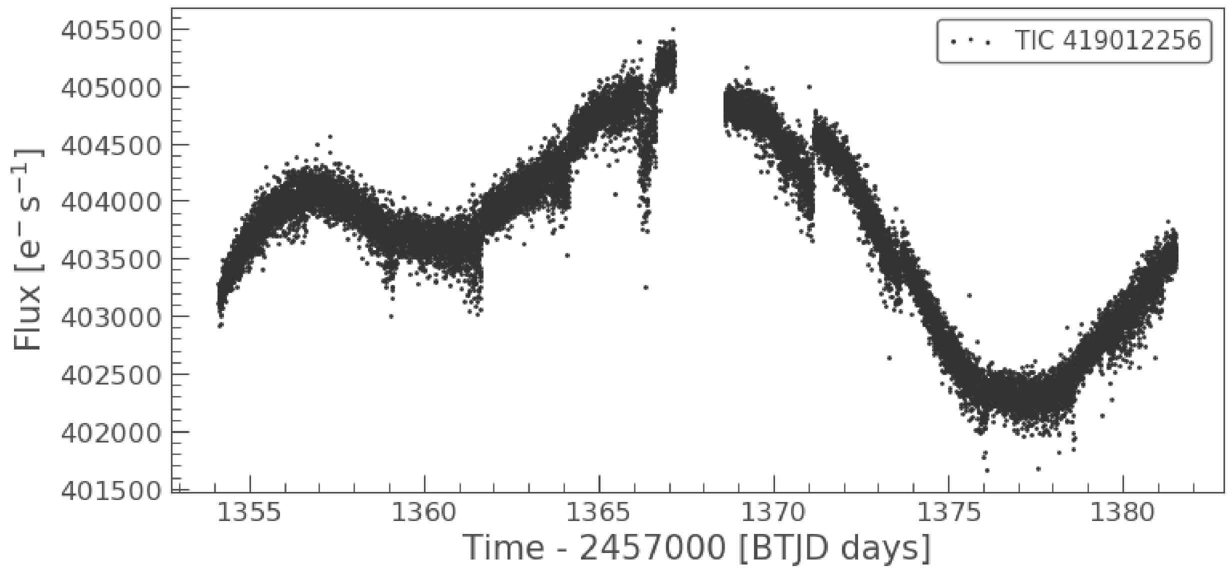
In [ ]:

```
tpf.plot(aperture_mask=tpf.pipeline_mask);
```



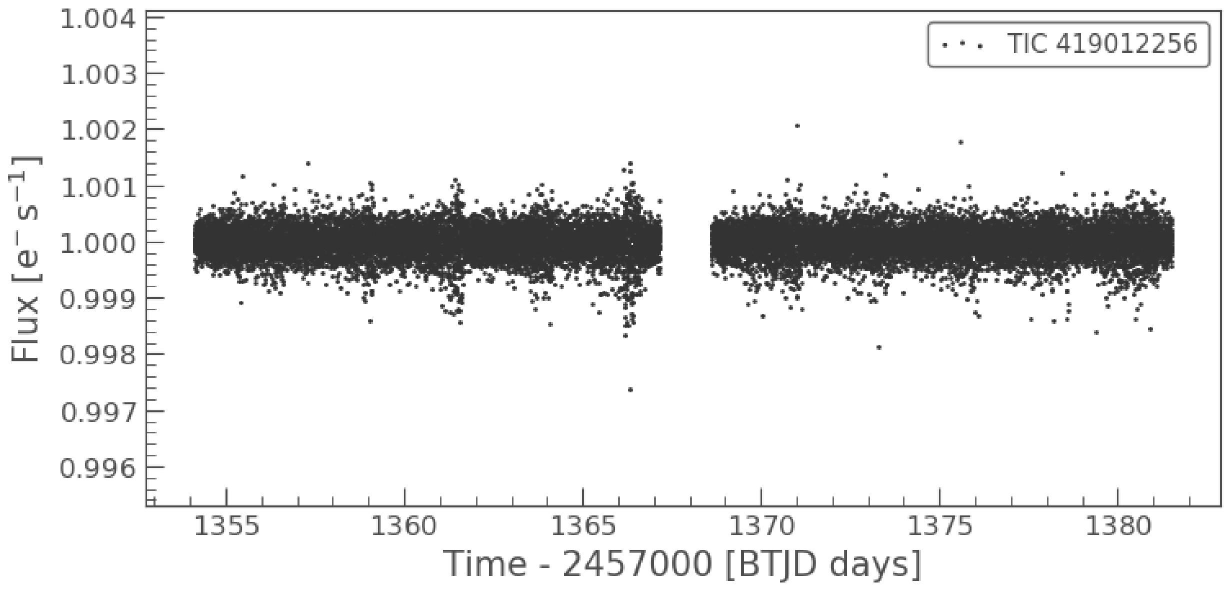
In [ ]:

lc = tpf.to_lightcurve()
lc.scatter();



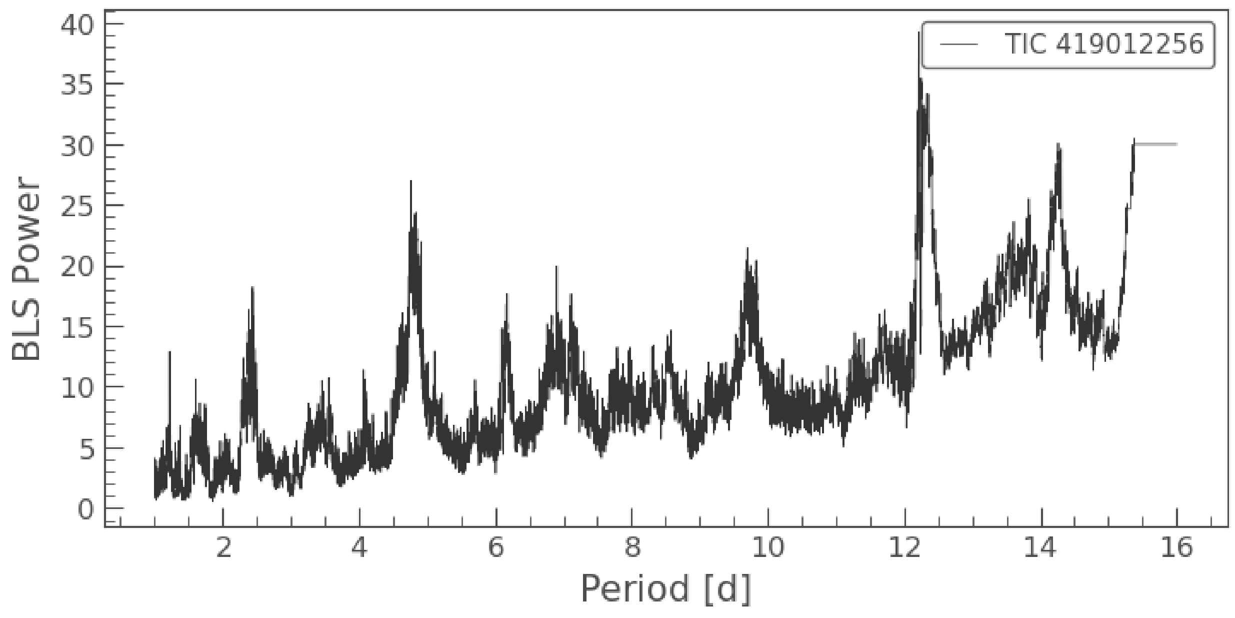
In [ ]:

flat_lc = lc.flatten()
flat_lc.scatter(s=1);



In [ ]:

```
import numpy as np
periodogram = flat_lc.to_periodogram(method="bls", period=np.arange(1, 16, 0.001))
periodogram.plot();
```



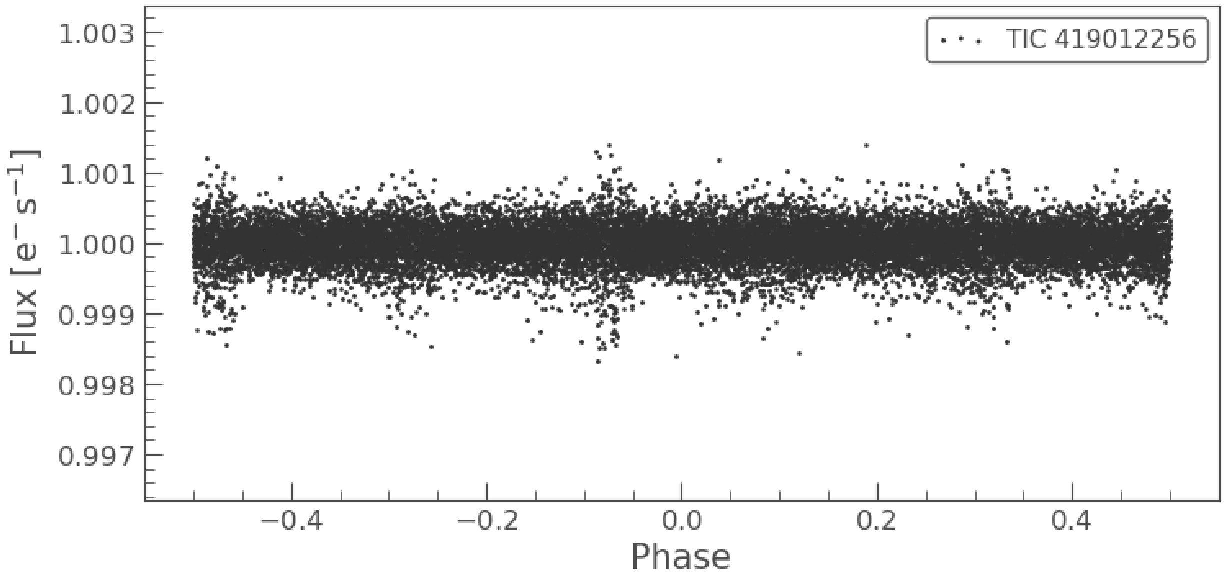
In [ ]:

```
best_fit_period = periodogram.period_at_max_power
print('Best fit period: {:.3f}'.format(best_fit_period))
```

```
Best fit period: 12.223 d
```

In [ ]:

```
folded_lc = clipped_lc.fold(period=12.223, t0=1355)
folded_lc.scatter(s=1);
```



In [ ]:

```
binned_lc = folded_lc.bin(binsize=10)
binned_lc.scatter();
```

Finding exoplanets with Python + Lightkurve

In [ ]:

```
pip install lightkurve
```

In [ ]:

```
from lightkurve import search_targetpixelfile from lightkurve import
TessTargetPixelFile import lightkurve as lk import numpy as np
```

Let's look at a star we know has a planet

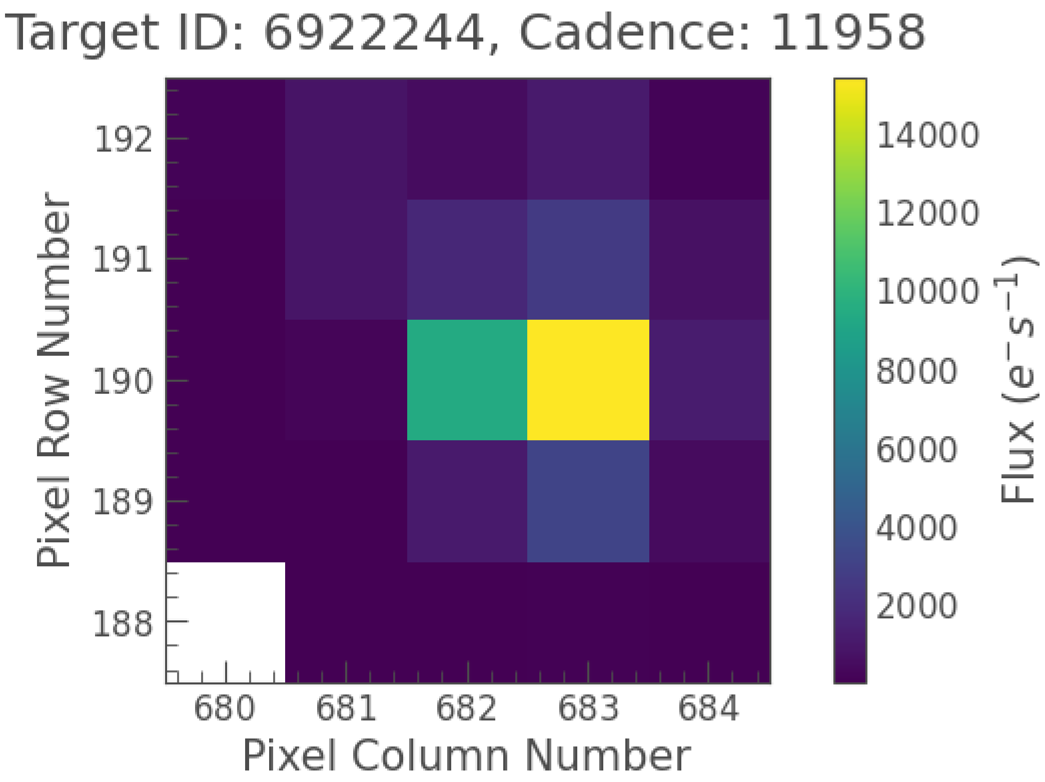
If we can spot a blob periodically transiting between the star and us, chances are it's an exoplanet

In [ ]:

```
# Download the pixelfile for a given star # A quarter means a quarter of a year pixelFile = search_targetpixelfile('KIC 6922244', author="Kepler", cadence="long", quarter=4).download() # Show a single snapshot pixelFile.plot(frame=42)
```

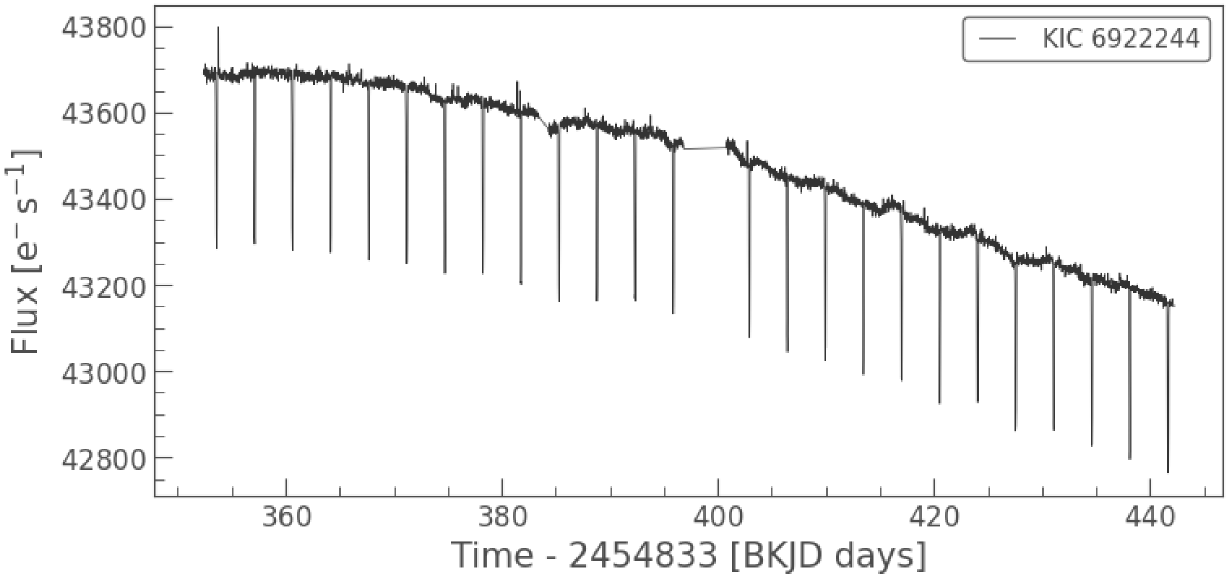
Out[ ]:

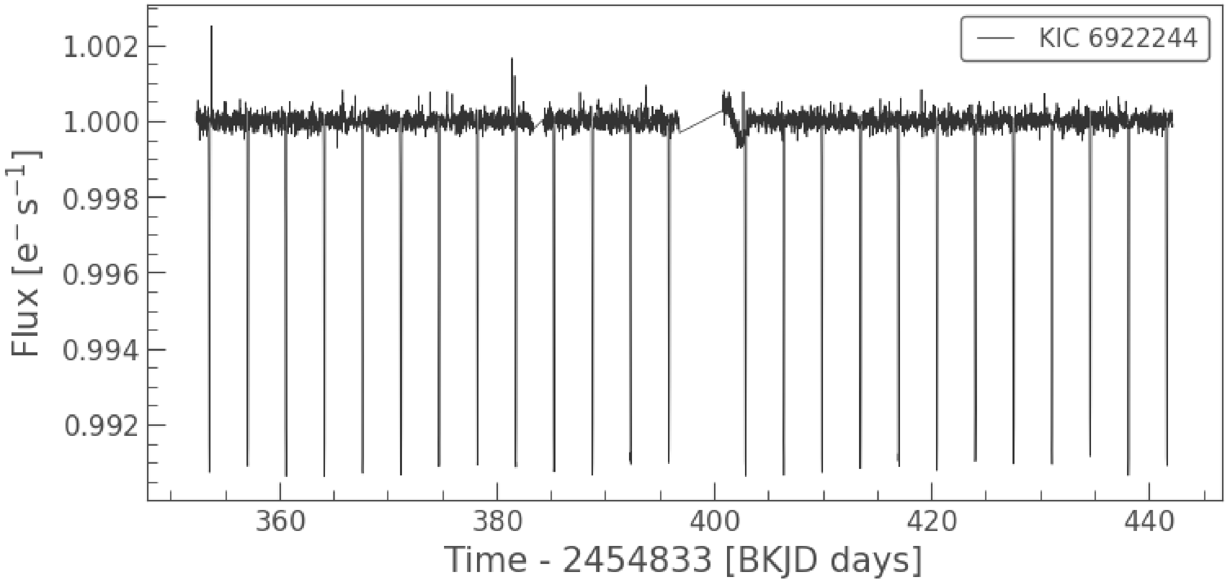
```
<matplotlib.axes._subplots.AxesSubplot at 0x7faf26d246d0>
```

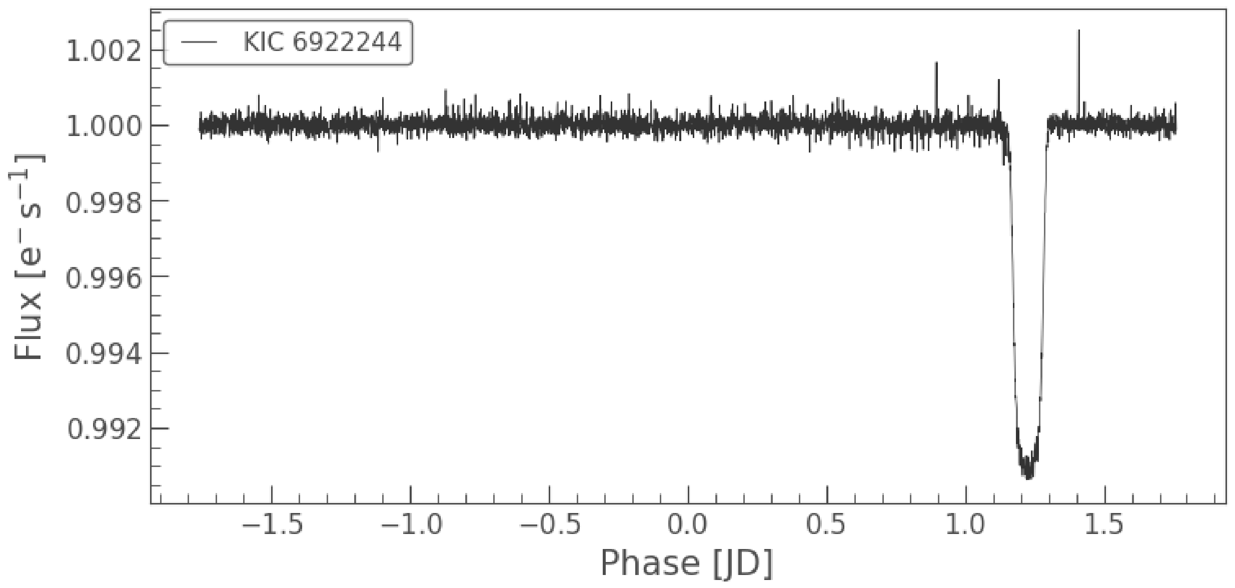


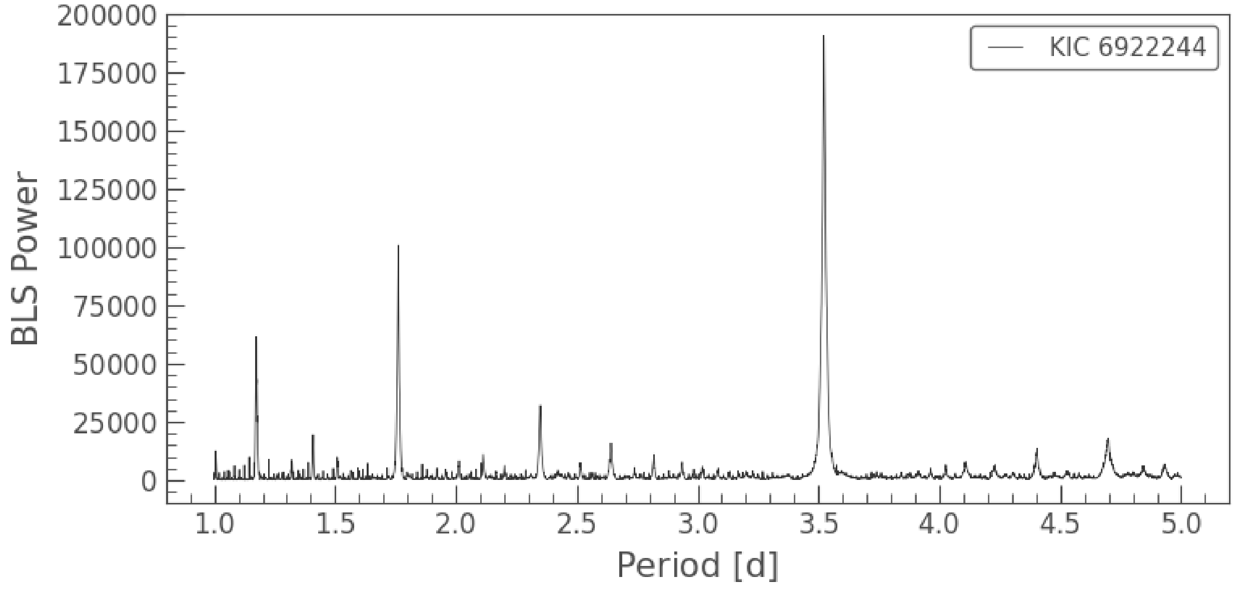
In [ ]:

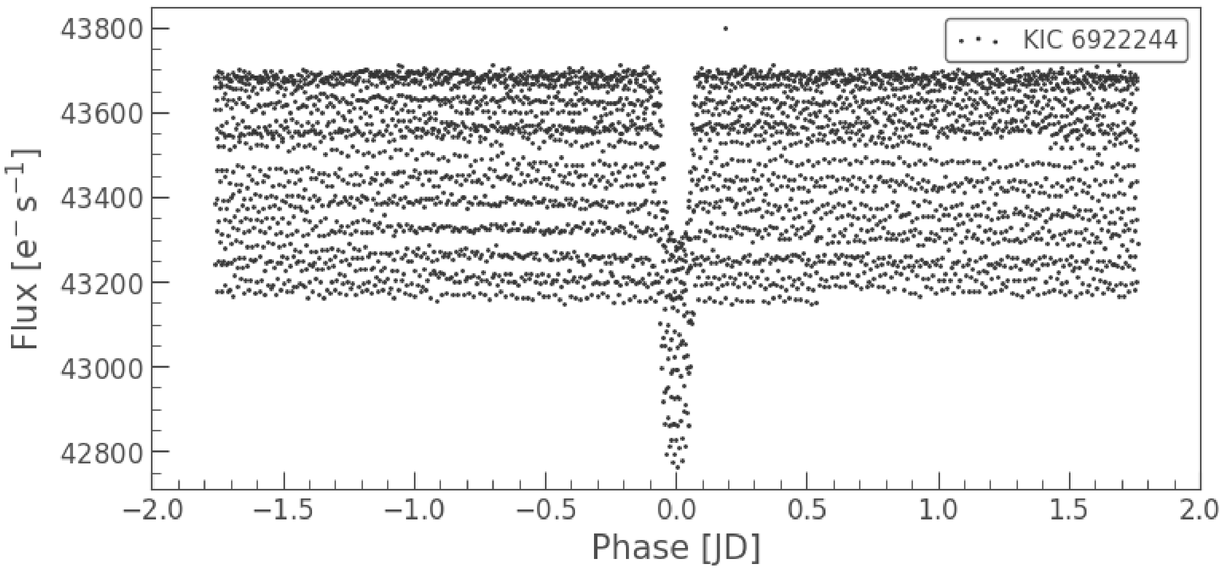
```
# We'll combine the individual frames into a lightcurve # Aperture masks make the image look better for analysis lc = pixelFile.to_lightcurve(aperture_mask=pixelFile.pipeline_mask) lc.plot() # We may find it easier to spot the pattern if we flatten the curve flat_lc = lc.flatten() flat_lc.plot() # Phase-fold the light curve to verify that the period and transit time # correspond to the transit signal # This puts the frequency spikes on top of each other if we get the period right folded_lc = flat_lc.fold(period=3.5225) folded_lc.plot() # How to discover the correct period? # Use a periodogram to show all the repetitive patterns in usr graph # Gives us the most likely candidate period = np.linspace(1, 5, 10000) # BLS = Box Least Squares bls = lc.to_periodogram(method='bls', period=period, frequency_factor=500) bls.plot() # Period value corresponding to the highest peak in the periodogram planet_x_period = bls.period_at_max_power planet_x_to = bls.transit_time_at_max_power planet_x_dur = bls.duration_at_max_power # Folding can yield a lot of information about the planet
# The depth can tell us about the size, etc ax = lc.fold(period=planet_x_period,
epoch_time=planet_x_t0).scatter() ax.set_xlim(-2,2) print(planet_x_period)
print(planet_x_t0) print(planet_x_dur)
3.522652265226523 d
353.601324850352 85
0.1 d
```






KIC 6922244
Flux [e^{-}s^{-1}]
1.002
1.000
0.998
0.996
0.994
0.992
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
Phase [JD]


KIC 6922244
BLS Power
200000
175000
150000
125000
100000
75000
50000
25000
0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Period [d]

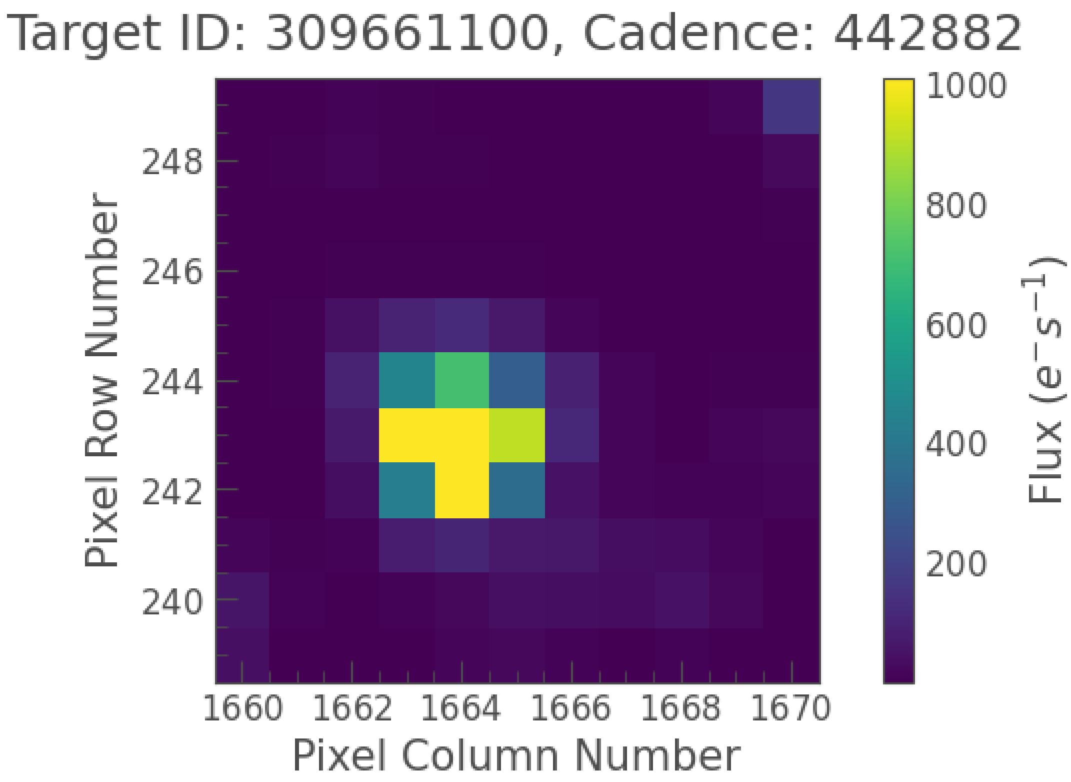


Selecting a star

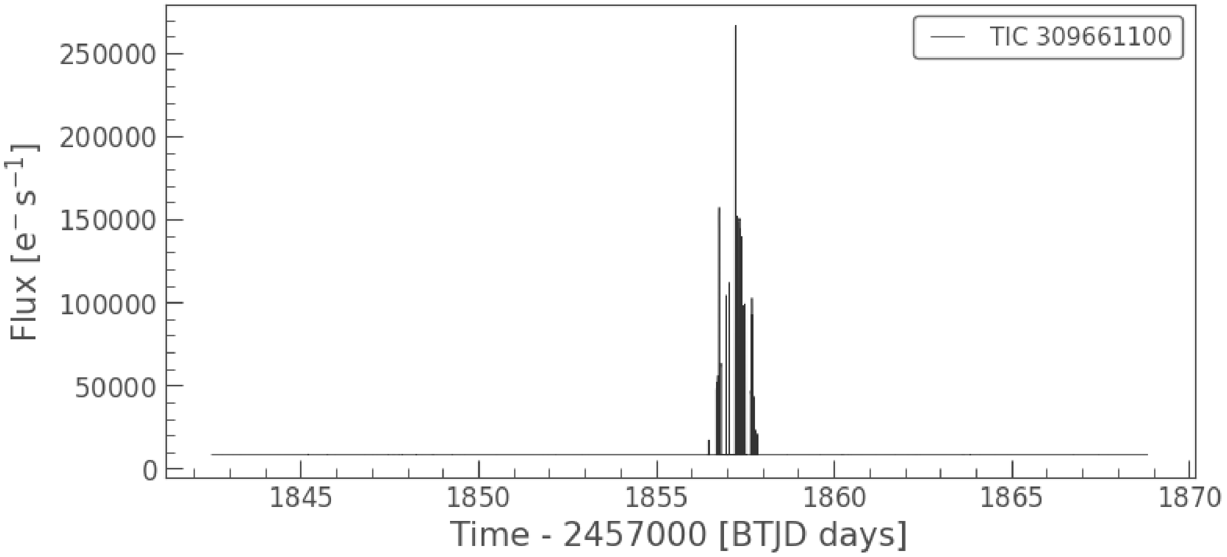
Analyze the star

```
tpf = TessTargetPixelFile("/TESS/MAST_2022-01-20T1747/TESS/tess2019357164649-s0020-0000000309661100-0165-s/tess2019357164649-s0020-0000000309661100-0165-s_tp.fits") # Show a single snapshot tpf.plot(frame=42) # Plot the lightcurve lc = tpf.to_lightcurve(aperture_mask=tpf.pipeline_mask) lc.plot() # Flatten it flat_lc = lc.flatten() flat_lc.plot() # Try and find the period of the most prominent orbiting object period = np.linspace(1, 5, 10000) bls = lc.to_periodogram(method='bls', period=period, frequency_factor=500) bls.plot() planet_x_period = bls.period_at_max_power planet_x_to = bls.transit_time_at_max_power planet_x_dur = bls.duration_at_max_power # Phase-fold the ligthcurve based on the discovered period at max power ax = lc.fold(period=planet_x_period, epoch_time=planet_x_to).scatter() ax.set_xlim(-3,3)
```

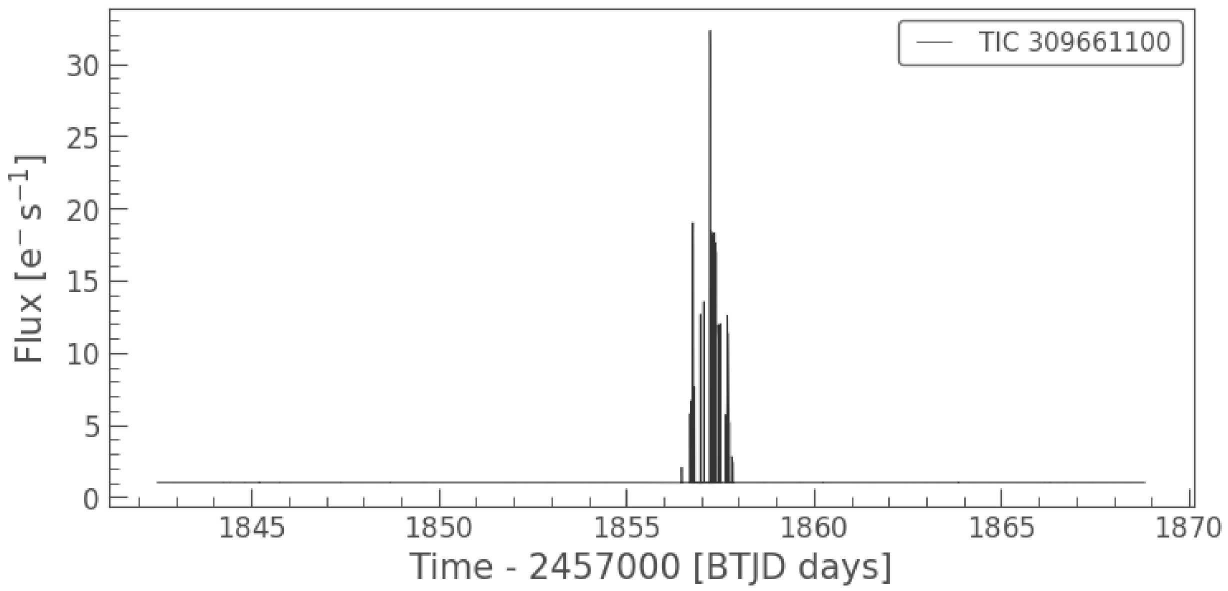
```
(-3.0, 3.0)
```

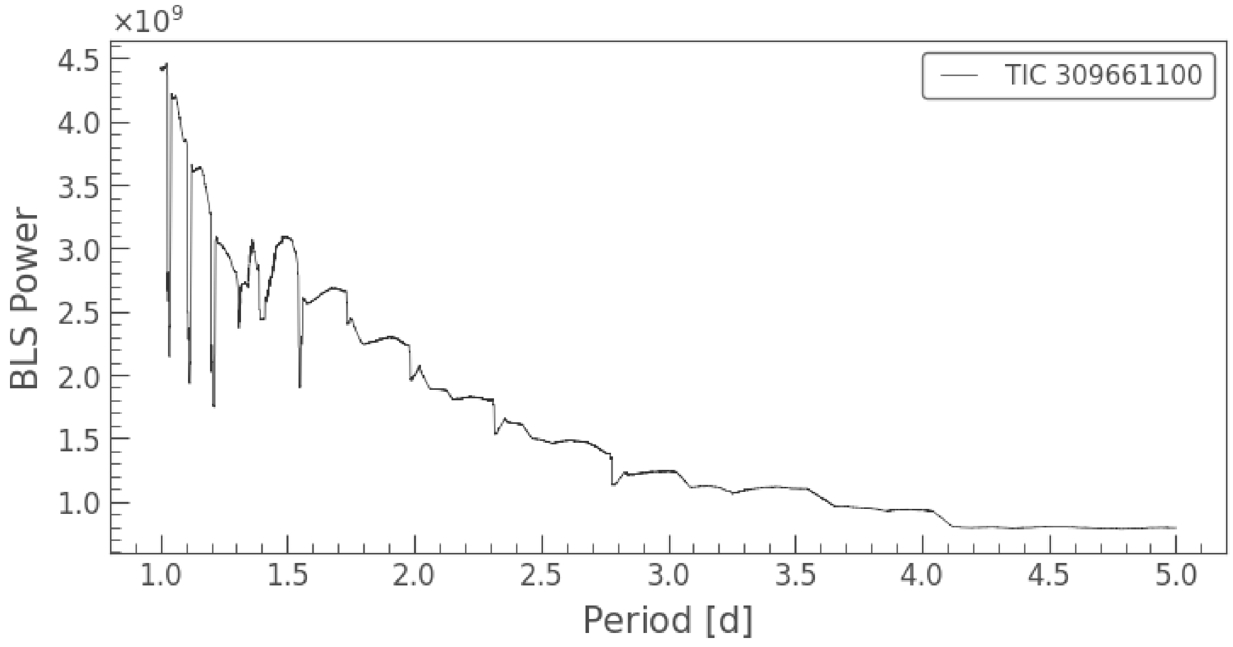


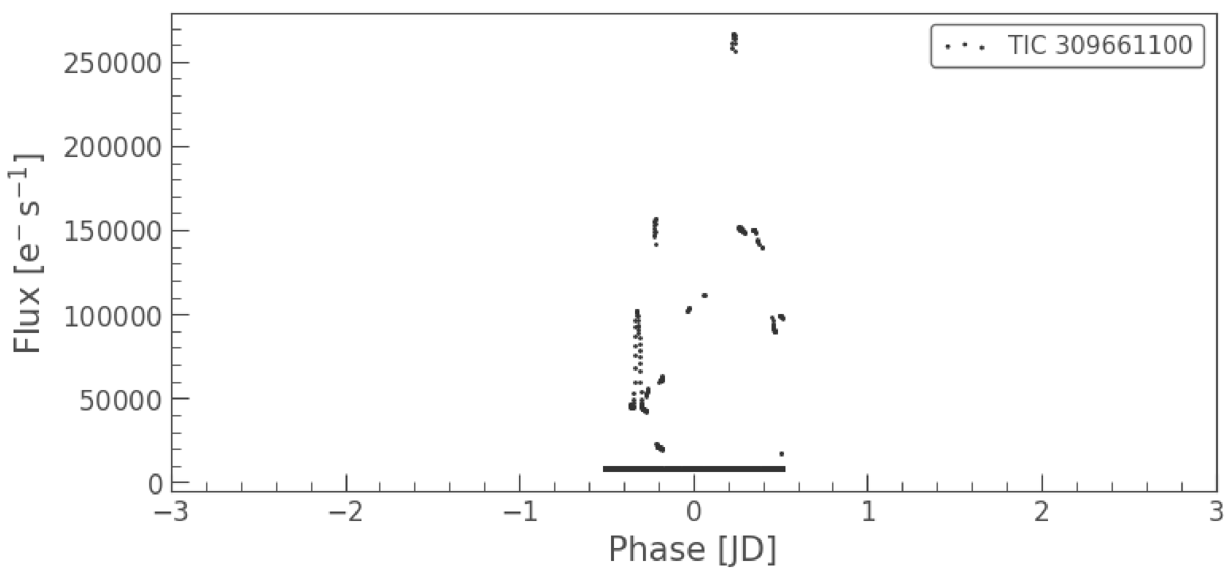
Target ID: 309661100, Cadence: 442882
Pixel Row Number
248
246
244
242
240
1660
1662
1664
1666
1668
1670
Pixel Column Number
1000
800
600
400
200
Flux (e^-s^{-1})



TIC 309661100
Flux [e^-s^{-1}]
250000
200000
150000
100000
50000
0
1845
1850
1855
1860
1865
1870
Time - 2457000 [BTJD days]


TIC 309661100
Flux [e⁻ s⁻¹]
Time - 2457000 [BTJD days]
0
5
10
15
20
25
30
1845
1850
1855
1860
1865
1870


×10⁹
TIC 309661100
BLS Power
Period [d]
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0



TIC 309661100
Flux [e$^-$ s^{-1}]
Phase [JD]
0
50000
100000
150000
200000
250000
−3
−2
−1
0
1
2
3