Astroinformatics I
Graded Practice 3

José B. Batista M.

1. **Repeat the plots from graded practice 1, but now with Python using the light curve filesfrom practice 2. For the plots, take into account how to make them more readable.**

   The Python script snippet provided below (saved as `practice_3.py`) generates raw light curve plots for each light curve file present in the `lc_data_folder` directory. The `get_lc_data` function is used to load and preprocess the TESS light curve files into an Astropy `TimeSeries` object. This function reads a light curve file, parses its columns (`TIME`, `PDCSAP_FLUX`, and `PDCSAP_FLUX_ERR`), and converts times to JD in TDB scale. It also converts each BJD timestamp to a UTC calendar date string, which is included as a new column, to be used. in the plots. The `plot_raw_lc` function is responsible for creating these plots. It uses `matplotlib` to generate scatter plots with error bars for flux values against BJD Epoch. To enhance readability and accessibility, the script utilizes different markers and a colorblind-fiendly palette for data points corresponding to unique observation dates (taken from the date string column). A legend is included to distinguish the dates, and axis labels and a title are provided for clarity. The plots are saved as PDF files in the `Plots` directory.

```
 1  import os, itertools, numpy as np, matplotlib.pyplot as plt, pandas
        as pd
 2  from astropy.time import Time
 3  from astropy.timeseries import TimeSeries, LombScargle
 4  from astropy import units as u
 5
 6  lc_data_folder = 'Practices/Practice_3/LC_Files'
 7
 8  def get_lc_data(filename):
 9      '''
10      Load and preprocess a TESS light curve file into an Astropy
        TimeSeries object.
11
12      This function reads a light curve file from the specified folder,
         parses its
13      columns (BTJD time, PDCSAP_FLUX, and flux error), and converts
        times to JD
14      in TDB scale. It also converts each BJD timestamp to a UTC
        calendar date
15      string, which is included as a new column.
16
17      Parameters
18      ----------
19      filename : str
20          The name of the light curve file (relative to `lc_data_folder
        `).
21
22
```

```
23        Returns
24        -------
25        lc_data : astropy.timeseries.TimeSeries
26            A TimeSeries object with columns:
27            - time (BJD, TDB)
28            - flux (in e-/s)
29            - flux_err (in e-/s)
30            - date (UTC calendar date string in 'YYYY-MM-DD' format)
31
32        Raises
33        ------
34        FileNotFoundError
35            If the specified file does not exist in the data folder.
36        Exception
37            For any other parsing or conversion errors.
38        '''
39        try:
40            lc_file = os.path.join(lc_data_folder, filename)
41            lc_df = pd.read_csv(lc_file, sep=' ', skiprows = 1, names=['
    TIME',
42                               'PDCSAP_FLUX', 'PDCSAP_FLUX_ERR'])
43            lc_df = lc_df.dropna()
44            btjd = lc_df['TIME'].astype(float)
45            flux = lc_df['PDCSAP_FLUX'].astype(float)
46            flux_error = lc_df['PDCSAP_FLUX_ERR'].astype(float)
47            bjd = btjd + 2457000.0
48            bjd_epoch = Time(bjd, format = 'jd', scale = 'tdb')
49            dates_utc = bjd_epoch.utc.datetime
50            date_strs = [date.strftime('%Y-%m-%d') for date in dates_utc]
51            lc_df['DATE'] = date_strs
52            lc_data = TimeSeries(time=bjd_epoch,
53                               data={'flux': flux * u.electron/u.s,
54                                     'flux_err': flux_error * u.
    electron/u.s,
55                                     'date': lc_df['DATE'].values})
56            return lc_data
57        except FileNotFoundError:
58            print(f'Error: File not found at {filename}')
59            exit()
60        except Exception as e:
61            print(f'An error occurred while loading the file: {e}')
62            exit()
63
64  def plot_raw_lc(filename, lc_data):
65        '''
66        Plot the raw light curve with flux grouped by observation date.
67
68        This function generates a scatter plot of a light curve with
    error bars, grouping points
69        by date using distinct marker/colour combinations. The figure is
    saved to a PDF file.
70
71        Parameters
72        ----------
73        filename : str
74            Name of the light curve file being processed. Used for the
```
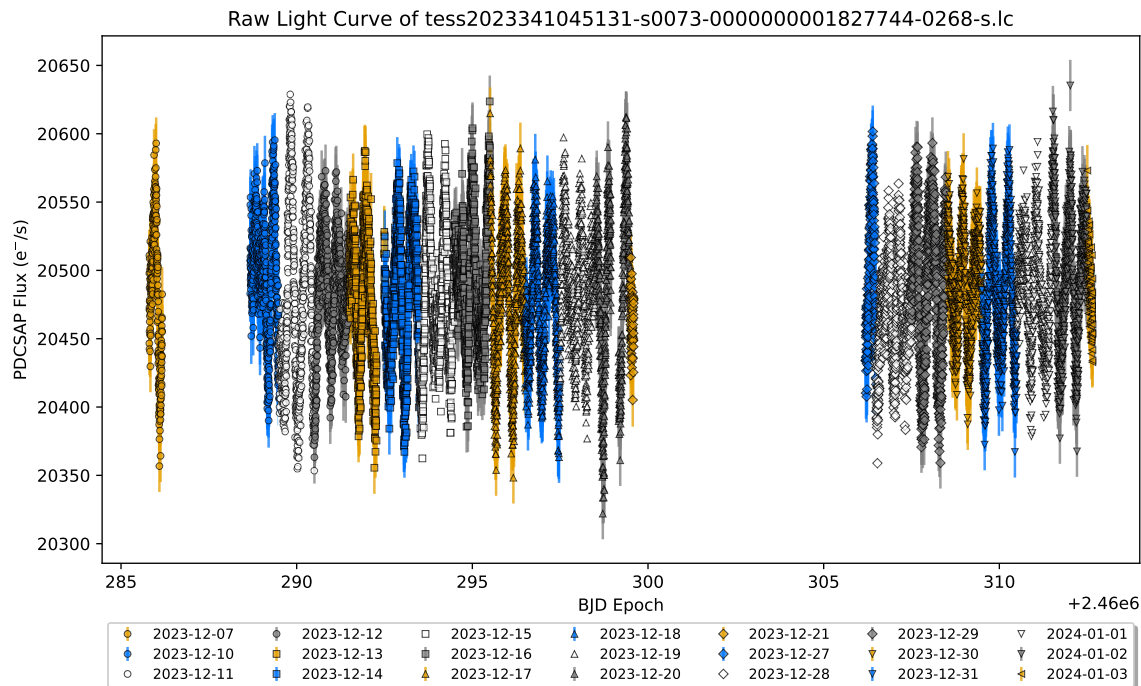
```
              plot title and output filename.
 75      lc_data : astropy.timeseries.TimeSeries
 76          Light curve time series object containing 'time', 'flux', '
         flux_err', and 'date' columns.
 77
 78      Notes
 79      -----
 80      - The plot legend is arranged in multiple columns below the plot
         to avoid overlapping data.
 81      - The output plot is saved as a PDF in the `Practices/Practice_3/
         Plots/` directory.
 82      '''
 83      #plt.rc('xtick', labelsize='x-small')
 84      #plt.rc('ytick', labelsize='x-small')
 85      plt.figure(figsize=(10, 6))
 86      colors = ['#E69F00', '#0077FF', 'w','gray']
 87      shapes = ['o', 's', '^', 'D', 'v', '<', '>', 'P', '*', 'h', 'd',
         'p']
 88      markers = list(itertools.product(shapes, colors))
 89      unique_dates = np.unique(lc_data['date'])
 90      num_labels = len(unique_dates)
 91      n_cols = int(np.ceil(num_labels/3))
 92      for i, date in enumerate(unique_dates):
 93          mask = lc_data['date'] == date
 94          marker = markers[i % len(markers)]
 95          shape, color = marker
 96          plt.errorbar(lc_data.time[mask].value, lc_data['flux'][mask].
         value,
 97                       yerr=lc_data['flux_err'][mask].value, fmt=shape,
 98                       color=color, ms=4, markeredgecolor='k',
         markeredgewidth=0.5,
 99                       alpha=0.75, label=f'{date}')
100      plt.xlabel('BJD Epoch')
101      plt.ylabel(r'PDCSAP Flux (e$^{{-}}$/s)')
102      plt.title(f'Lightcurve of {filename}')
103      plt.legend(fontsize='small', ncol=n_cols, loc='upper center',
104               bbox_to_anchor=(0.5, -0.1), fancybox=True, shadow=True
         )
105      plt.tight_layout()
106      plt.savefig(f'Practices/Practice_3/Plots/{filename}_raw.pdf',
107               format='pdf')
108      plt.close()
109
110 for filename in os.listdir(lc_data_folder):
111      lc_data = get_lc_data(filename)
112      plot_raw_lc(filename, lc_data)
```

One of the resulting light curve plots is shown below (all plots for this practice, as well
as the Python script, are available in a repository).

Raw Light Curve of tess2023341045131-s0073-0000000001827744-0268-s.lc

Legend:
- ◆ 2023-12-07
- ● 2023-12-10
- ○ 2023-12-11
- ◐ 2023-12-12
- ■ 2023-12-13
- ■ 2023-12-14
- □ 2023-12-15
- ■ 2023-12-16
- ▲ 2023-12-17
- ▲ 2023-12-18
- △ 2023-12-19
- ▲ 2023-12-20
- ◆ 2023-12-21
- ◆ 2023-12-27
- ◇ 2023-12-28
- ◆ 2023-12-29
- ▽ 2023-12-30
- ▼ 2023-12-31
- ▽ 2024-01-01
- ▼ 2024-01-02
- ◀ 2024-01-03

2. When you make the plots, can you identify outliers? Highlight them. Try writing code to identify at least the most extreme outliers.

The Python script snipped provided below identifies and highlights outliers in the raw light curve plots. The `identify_outliers` function performs a robust sigma-clipping method to detect outliers. For each unique observing date, it calculates the median and median absolute deviation (mad) of the flux values. Points deviating more than a specified threshold (defaulting to 3) times the robust sigma estimate (`1.4826*mad`) are flagged as outliers. These identified outliers are then plotted with a distinct black `'x'` marker on the raw light curve plots, making them clearly visible.

```
1  def identify_outliers(flux, dates, threshold=3):
2      '''
3      Identify outliers in a light curve using robust sigma-clipping
       per observing
4      date.
5
6      For each unique date, this function computes the median and MAD (
       median
7      absolute deviation) of the flux values, then flags as outliers
       all points
8      deviating more than `threshold` times the robust sigma estimate.
9
10     Parameters
11     ----------
12     flux : ndarray
13         Array of flux values (e.g., in e-/s).
14     dates : array-like of str
15         Array of corresponding unique date strings in 'YYYY-MM-DD'
       format, one
16         per flux value.
17     threshold : float, optional
```

```
18          Number of robust standard deviations (sigma) to use for
       outlier rejection.
19          Default is 3.
20
21      Returns
22      -------
23      outlier_mask : ndarray of bool
24          Boolean array of the same length as `flux`, where `True`
       marks an outlier.
25      '''
26      outlier_mask = np.zeros(len(flux), dtype=bool)
27      for date in dates:
28          mask = dates == date
29          flux_day = flux[mask]
30          median = np.median(flux_day)
31          mad = np.median(np.abs(flux_day - median))
32          sigma = 1.4826 * mad
33          outliers = np.abs(flux_day - median) > threshold * sigma
34          outlier_mask[mask] = outliers
35      return outlier_mask
36
37  # ---- UPDATED PLOT FUNCTION -----
38
39  def plot_raw_lc(filename, lc_data, threshold=3):
40      '''
41      Plot the raw light curve with flux grouped by observation date
       and annotated outliers.
42
43      This function generates a scatter plot of a light curve with
       error bars, grouping points
44      by date using distinct marker/colour combinations. Outliers are
       identified using
45      sigma-clipping and plotted with a distinct marker. The figure is
       saved to a PDF file.
46
47      Parameters
48      ----------
49      filename : str
50          Name of the light curve file being processed. Used for the
       plot title and output filename.
51      lc_data : astropy.timeseries.TimeSeries
52          Light curve time series object containing 'time', 'flux', '
       flux_err', and 'date' columns.
53      threshold : float, optional
54          Sigma threshold for outlier rejection (default is 3).
55
56      Notes
57      -----
58      - Outliers are identified per date using the `identify_outliers`
       function with a default
59        threshold of σ3.
60      - The plot legend is arranged in multiple columns below the plot
       to avoid overlapping data.
61      - The output plot is saved as a PDF in the `Practices/Practice_3/
       Plots/` directory.
62      '''
63      #plt.rc('xtick', labelsize='x-small')
```

```
64        #plt.rc('ytick', labelsize='x-small')
65        plt.figure(figsize=(10, 6))
66        colors = ['#E69F00', '#0077FF', 'w','gray']
67        shapes = ['o', 's', '^', 'D', 'v', '<', '>', 'P', '*', 'h', 'd',
          'p']
68        markers = list(itertools.product(shapes, colors))
69        unique_dates = np.unique(lc_data['date'])
70        flux = lc_data['flux'].value
71        dates = lc_data['date']
72        outlier_mask = identify_outliers(flux, dates, threshold)
73        num_labels = len(unique_dates)
74        n_cols = int(np.ceil(num_labels/3))
75        for i, date in enumerate(unique_dates):
76            mask = lc_data['date'] == date
77            marker = markers[i % len(markers)]
78            shape, color = marker
79            plt.errorbar(lc_data.time[mask].value, lc_data['flux'][mask].
      value,
80                        yerr=lc_data['flux_err'][mask].value, fmt=shape,
81                        color=color, ms=4, markeredgecolor='k',
      markeredgewidth=0.5,
82                        alpha=0.75, label=f'{date}')
83        plt.scatter(lc_data.time[outlier_mask].value,
84                    lc_data['flux'][outlier_mask].value, marker='x',
      color='k', s=25,
85                    label = fr'{threshold}$\sigma$ Outliers', zorder=10)
86        plt.xlabel('BJD Epoch')
87        plt.ylabel(r'PDCSAP Flux (e$^{{-}}$/s)')
88        plt.title(f'Raw Light Curve of {filename}')
89        plt.legend(fontsize='small', ncol=n_cols, loc='upper center',
90                    bbox_to_anchor=(0.5, -0.1), fancybox=True, shadow=True
      )
91        plt.tight_layout()
92        plt.savefig(f'Practices/Practice_3/Plots/{filename}_raw.pdf',
93                    format='pdf')
94        plt.close()
95
96 outliers_sigma = 3
97 for filename in os.listdir(lc_data_folder):
98     lc_data = get_lc_data(filename)
99     plot_raw_lc(filename, lc_data, outliers_sigma)
```

The updated light curve plot with the highlighted outliers for the same example is shown in the figure below.

Raw Light Curve of tess2023341045131-s0073-0000000001827744-0268-s.lc
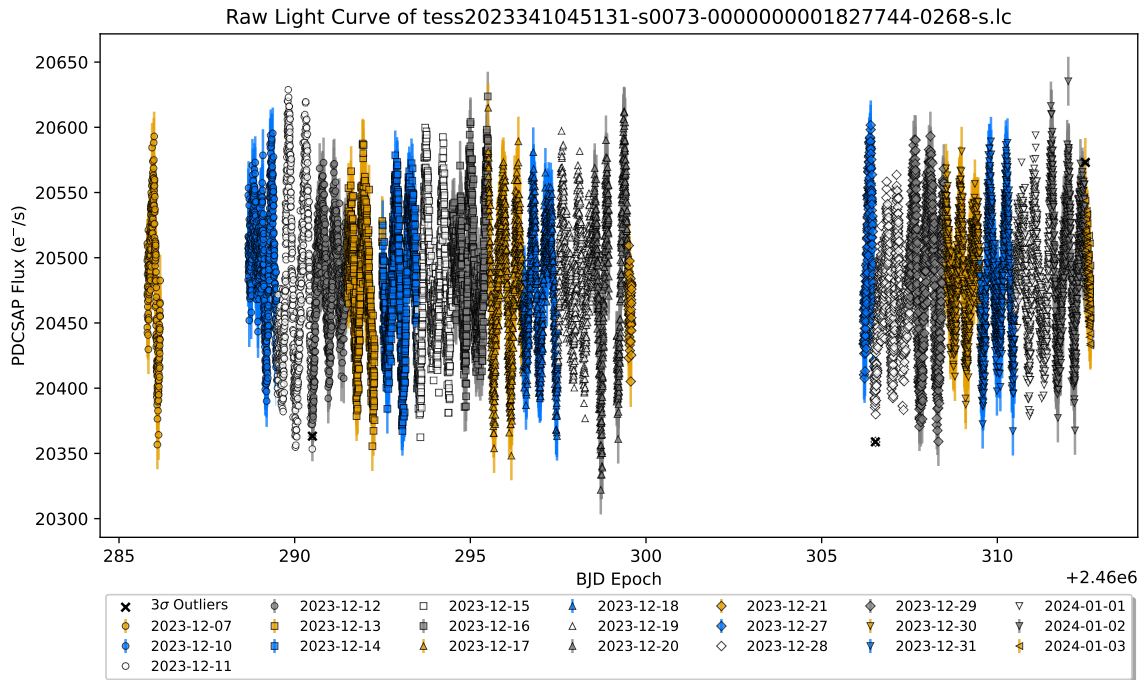
3. Think about basic statistics to describe light curves, such as amplitudes, and implementat least two of them.

The Python script snippet provided below mplements basic statistics to describe light curves, specifically focusing on the phase, period and amplitude. The `fold_lc` function analyzes and plots a phase-folded light curve and its Lomb-Scargle periodogram. It first prepares the data by masking out outliers using the `identify_outliers` function with the specified threshold. It then computes the Lomb-Scargle periodogram to find the best period within a given range (`min_period` to `max_period`). The light curve is then phase-folded using this best period. Finally, the amplitude is estimated from the folded light curve as the difference between the 97.5th and 2.5th percentiles of the flux values. The function generates a two-subplot figure, displaying both the phase-folded light curve and the Lomb-Scargle periodogram, along with annotations for the period, amplitude, and an estimated initial BJD epoch.

```
1  def fold_lc(filename, lc_data, threshold=3, min_period=0.1,
       max_period=10):
2      """
3      Analyse and plot a phase-folded light curve and its Lomb-Scargle
       periodogram.
4
5      Parameters
6      ----------
7      lc_data : astropy.timeseries.TimeSeries
8          Light curve data with 'time', 'flux', 'flux_err', and 'date'
       columns.
9      filename : str
10         Filename for plot titles.
11     threshold : float, optional
12         Sigma threshold for outlier rejection (default is 3).
13     min_period : float, optional
```

7

```python
14              Minimum period to search (days), default 0.1.
15      max_period : float, optional
16          Maximum period to search (days), default 10.
17      """
18      # Prepare data and mask out outliers
19      flux = lc_data['flux'].value
20      flux_err = lc_data['flux_err'].value
21      dates = lc_data['date']
22      time = lc_data.time.value
23      mask = ~identify_outliers(flux, dates, threshold)
24      time_clean = time[mask]
25      flux_clean = flux[mask]
26      flux_err_clean = flux_err[mask]
27      dates_clean = dates[mask]
28      bjd_0 = np.median(time_clean)
29      # Compute Lomb-Scargle periodogram
30      ls = LombScargle(time_clean, flux_clean, flux_err_clean)
31      frequency, power = ls.autopower(minimum_frequency=1/max_period,
32                                      maximum_frequency=1/min_period,
33                                      samples_per_peak = 25)
34      best_frequency = frequency[np.argmax(power)]
35      best_period = 1 / best_frequency
36      # Phase folding
37      phase = (time_clean % best_period) / best_period
38      # Estimate amplitude from folded light curve
39      amplitude = np.percentile(flux_clean, 97.5) - np.percentile(
        flux_clean, 2.5)
40      # Create figure with two subplots (folded LC + periodogram)
41      fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 8),
42                                      gridspec_kw={'height_ratios': [2,
        1]})
43      # Plot folded light curve
44      shapes = ['o', 's', '^', 'D', 'v', '<', '>', 'P', '*', 'h', 'd',
        'p']
45      colors = ['orange', "#0050FF", 'w', 'gray']
46      markers = list(itertools.product(shapes, colors))
47      unique_dates = np.unique(dates_clean)
48      n_cols = int(np.ceil(len(unique_dates) / 3))
49      for i, date in enumerate(unique_dates):
50          mask_date = dates_clean == date
51          shape, color = markers[i % len(markers)]
52          ax1.errorbar(phase[mask_date], flux_clean[mask_date],
53                      yerr=flux_err_clean[mask_date], fmt=shape, color
        =color,
54                      ms=4, alpha=0.75, markeredgecolor='k',
        markeredgewidth=0.5,
55                      label=date)
56      ax1.set_xlabel('Phase')
57      ax1.set_ylabel(r'PDCSAP Flux (e$^{{-}}$/s)')
58      ax1.set_title(f'Phase-Folded Light Curve of {filename}')
59      # Annotation above x-axis in the folded plot
60      info_text = rf'$BJD_0$: {bjd_0},$\quad$Period: {best_period:.5f}
        d,$\quad$Amplitude: $\sim${amplitude:.2f} e$^{{-}}$/s'
61      ax1.annotate(
62          info_text,
63          xy = (0.01, 0.01),
```

```
64          xycoords = 'axes fraction',
65          fontsize = 9,
66          color = 'black',
67      )
68      ax1.legend(fontsize='small', ncol=n_cols, loc='upper center',
69                  bbox_to_anchor=(0.5, -0.1), fancybox=True, shadow=True
        )
70      # Plot periodogram
71      ax2.plot(1/frequency, power, color="#0050FF")
72      max_power = np.max(power)
73      ax2.annotate(
74          f'Best Period: {best_period:.5f} d',
75          xy = (best_period, max_power),
76          xytext = (best_period + 0.15*(max_period - best_period), 0.8*
    max_power),
77          arrowprops = dict(shrink=0.1, width=1.5, headwidth=5,
    facecolor='orange',
78                            edgecolor = 'none'),
79          ha = 'center',
80          color = 'k',
81          fontsize = 9
82      )
83      ax2.set_xlabel('Period (days)')
84      ax2.set_ylabel('-LombScargle Power')
85      ax2.set_title('-LombScargle Periodogram')
86      plt.tight_layout()
87      plt.savefig(f'Practices/Practice_3/Plots/{filename}_folded.pdf',
88              format='pdf')
89      plt.close()
90
91 outliers_sigma = 3
92 for filename in os.listdir(lc_data_folder):
93     lc_data = get_lc_data(filename)
94     fold_lc(filename, lc_data, threshold=outliers_sigma)
```

The folded light curve plot with the periodogram for the same example is shown in the figure below.

It's important to note that not all light curves fold as well as the example shown. The periodogram of most of the light curves have spurious peaks, which could be due to several factors such as dropping NaN values from the data, gaps in the TESS observations, the signal not beign a perfect sinusoid, the presence of aliases due to the observation setup, etc. The interaction of these (and other) effects means that in practice there is no absolute guarantee that the highest peak corresponds to the best frequency, and results must be interpreted carefully.

Phase-Folded Light Curve of tess2023341045131-s0073-0000000001827744-0268-s.lc

BJD_0: 2460297.178891341,   Period: 0.43414 d,   Amplitude: ~180.74 e$^-$/s

Lomb–Scargle Periodogram

Best Period: 0.43414 d