

Graded Practice 3

Astroinformatics I

Name: José Luis Ricra Mayorca
Date: Jun 30, 2025

Problem 1

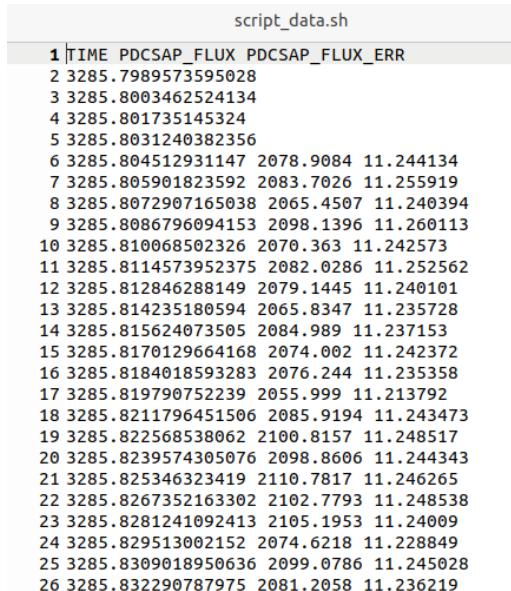
In Practice 1, 15 light curves were downloaded. In Practice 2, one of these light curves was selected, the commas were replaced with white spaces, the format was changed from CSV to LC, and finally, the columns for time, flux, and flux error were extracted. The first step to plot the 15 light curves again was to repeat the previous procedure for all 15 light curves. To speed up this process, a script called "script_data.sh" was written in Shell Script language (based on Tutorials 2 and 4, Hernitschek [2025a,b]), whose content is shown in Figure 1:



```
script_data.sh
1 #!/usr/bin/sh
2
3
4 for file in data*.csv; do
5     newname="data${file#data}"
6     newname="${newname%.csv}.lc"
7     cp "$file" "$newname"
8 done
9
10 for file in data*.lc; do
11     output="col_$(basename "$file")"
12     awk -F',' 'BEGIN {OFS=","} NR==1 || NF >= 9 {print $1, $8, $9}' "$file" > "$output"
13     echo "Archivo procesado: $output"
14 done
15
16 for file in col_data*.lc; do
17     sed 's/,/ /g' "$file" > "${file%.lc}_flux.lc"
18 done
```

Figure 1: Script to extract light curve data.

Figure 2 shows an example of one of the 15 files obtained after executing the "script_data.sh" script:



TIME	PDCSAP_FLUX	PDCSAP_FLUX_ERR
2 3285.7989573595028		
3 3285.8003462524134		
4 3285.801735145324		
5 3285.8031240382356		
6 3285.804512931147	2078.9084	11.244134
7 3285.805901823592	2083.7026	11.255919
8 3285.8072907165038	2065.4507	11.240394
9 3285.8086796094153	2098.1396	11.260113
10 3285.810068502326	2070.363	11.242573
11 3285.8114573952375	2082.0286	11.252562
12 3285.812846288149	2079.1445	11.240101
13 3285.814235180594	2065.8347	11.235728
14 3285.815624073505	2084.989	11.237153
15 3285.8170129664168	2074.002	11.242372
16 3285.8184018593283	2076.244	11.235358
17 3285.819790752239	2055.999	11.213792
18 3285.8211796451506	2085.9194	11.243473
19 3285.822568538062	2100.8157	11.248517
20 3285.8239574305076	2098.8606	11.244343
21 3285.825346323419	2110.7817	11.246265
22 3285.8267352163302	2102.7793	11.248538
23 3285.8281241092413	2105.1953	11.24009
24 3285.829513002152	2074.6218	11.228849
25 3285.8309018950636	2099.0786	11.245028
26 3285.832290787975	2081.2058	11.236219

Figure 2: One of the obtained files.

Then, a script called ‘plot.py‘ was written in the Python language to plot the 15 light curves. To create this script, the recommendations from the class ”Python Data Exploration and Visualization” [Hernitschek, 2025c] and Tutorial 6 [Hernitschek, 2025d] were taken into account. The script is shown in Figure 3.

File: /home/jose/practica_3/plot.py

Page 1 of 1

```
import matplotlib.pyplot as plt
import numpy as np
import glob

archivos = sorted(glob.glob("col_data*_flux.lc"))

for archivo in archivos:
    try:

        with open(archivo, 'r') as f:
            lineas = f.readlines()

        # Eliminar encabezado y filtrar solo líneas con 3 columnas válidas
        datos = []
        for linea in lineas[1:]:
            partes = linea.strip().split()
            if len(partes) == 3:
                try:
                    tiempo = float(partes[0])
                    flujo = float(partes[1])
                    error = float(partes[2])
                    datos.append([tiempo, flujo, error])
                except ValueError:
                    continue

        # Verificar si hay datos válidos
        if not datos:
            print(f"Error")
            continue

        datos = np.array(datos)

        tiempo = datos[:, 0]
        flujo = datos[:, 1]
        error = datos[:, 2]

        fig, ax = plt.subplots(figsize=(8, 5))

        ax.errorbar(tiempo, flujo, yerr=error, fmt='o', markersize=3, capsized=0,
                    ecolor='gray', elinewidth=1, label='Flux')

        ax.set_title(f'Flux vs Time - {archivo}')
        ax.set_xlabel('BJD - 2457000.0 (d)')
        ax.set_ylabel('PDCSAP_FLUX (e-/s)')

        ax.legend()
        plt.tight_layout()

        nombre_pdf = archivo.replace(".lc", ".pdf")
        plt.savefig(nombre_pdf)
        plt.close()
        print(f"Gráfico guardado como: {nombre_pdf}")

    except Exception as e:
        print(f"")
```

Figure 3: Python script for plotting.

Figure 4 shows all the plots obtained:

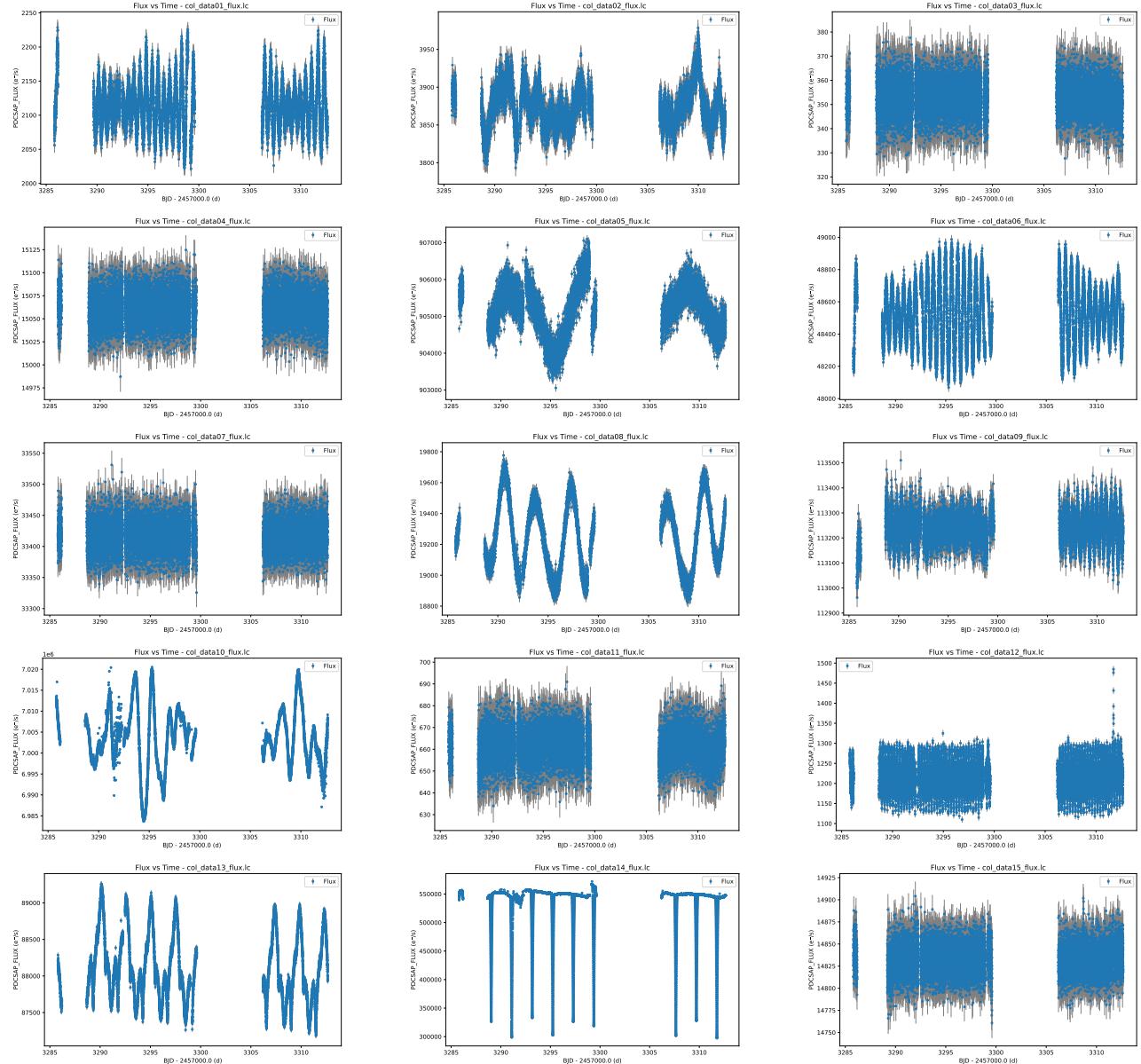


Figure 4: Set of 15 generated plots.

Problem 2

To detect outliers, it was necessary to establish a condition to identify *outliers* in the flux data series. This condition was the following:

$$\text{outliers} = |f_i - \tilde{f}| > 4\sigma$$

Where:

- f_i is the individual flux value at instant i ,
- \tilde{f} is the median of all flux values,
- σ is the standard deviation of the flux,
- $|\cdot|$ denotes the absolute value.

A point is considered an *outlier* if its value deviates more than 4 times the standard deviation from the median. This condition was implemented in a script called ‘outliers.py’ (see Figure 5).

File: /home/jose/practica_3/outliers.py Page 1 of 1

```
import matplotlib.pyplot as plt
import numpy as np
import glob

archivos = sorted(glob.glob("col_data*_.lc"))

for archivo in archivos:
    try:
        with open(archivo, 'r') as f:
            lineas = f.readlines()

        datos = []
        for linea in lineas[1:]:
            partes = linea.strip().split()
            if len(partes) == 3:
                try:
                    tiempo = float(partes[0])
                    flujo = float(partes[1])
                    error = float(partes[2])
                    datos.append([tiempo, flujo, error])
                except ValueError:
                    continue

        if not datos:
            print(f"Error")
            continue

        datos = np.array(datos)

        tiempo = datos[:, 0]
        flujo = datos[:, 1]
        error = datos[:, 2]

        # Detección de outliers
        mediana_flujo = np.median(flujo)
        sigma = np.std(flujo)
        outliers = np.abs(flujo - mediana_flujo) > 4 * sigma

        plt.figure(figsize=(8, 5))

        plt.errorbar(tiempo[~outliers], flujo[~outliers], yerr=error[~outliers],
                     fmt='o', markersize=3, capsizes=0,
                     ecolor='gray', elinewidth=1, color='blue', label='Flux')

        plt.errorbar(tiempo[outliers], flujo[outliers], yerr=error[outliers],
                     fmt='o', markersize=4, capsizes=0,
                     ecolor='red', elinewidth=1, color='red', label='Outliers')

        plt.title(f'Flux vs Time - {archivo}')
        plt.xlabel('BJD - 2457000.0 (d)')
        plt.ylabel('PDCSAP_FLUX (e-/s)')
        plt.legend()
        plt.tight_layout()

        nombre_pdf = archivo.replace(".lc", ".pdf")
        plt.savefig(nombre_pdf)
        plt.close()

        print(f"Gráfico guardado como: {nombre_pdf}")

    except Exception as e:
        print(f"")
```

Figure 5: Script to detect *outliers*.

Figure 6 shows the *outliers* detected in each light curve (in red). Note that in those light curves that do not exhibit horizontal symmetry, the outlier detection criterion does not perform well.

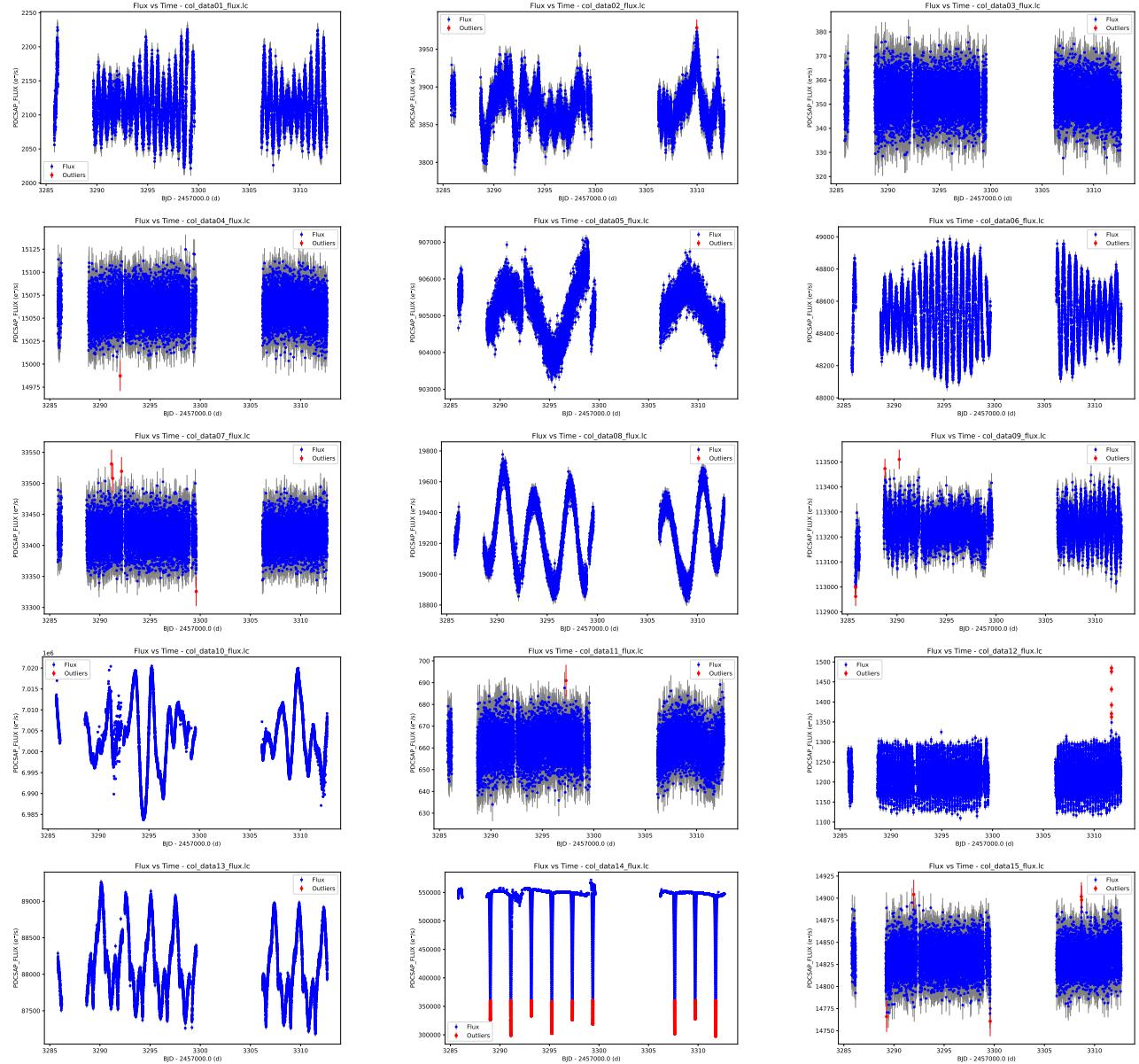


Figure 6: *Outliers* detected.

Problem 3

A script called ‘stat.py’ was written in the Python language to calculate important statistics for each light curve. These statistics are: minimum value, maximum value, amplitude, mean, median, and standard deviation. Figure 7 shows the code, and Figure 8 shows the results obtained.

File: /home/jose/practica_3/stat.py

Page 1 of 1

```
import numpy as np
import glob
import pandas as pd

files = sorted(glob.glob("col_data*_flux.lc"))

results = []

for file in files:
    try:
        with open(file, 'r') as f:
            lines = f.readlines()

        flux_values = []
        for line in lines[1:]:
            parts = line.strip().split()
            if len(parts) == 3:
                try:
                    flux = float(parts[1])
                    flux_values.append(flux)
                except ValueError:
                    continue

        if not flux_values:
            print(f"[WARNING] File {file} has no valid data.")
            continue

        flux_values = np.array(flux_values)

        # Compute statistics
        minimum = np.min(flux_values)
        maximum = np.max(flux_values)
        mean = np.mean(flux_values)
        median = np.median(flux_values)
        std_dev = np.std(flux_values)
        amplitude = maximum - minimum

        results.append({
            'File': file,
            'Min': minimum,
            'Max': maximum,
            'Amplitude': amplitude,
            'Mean': mean,
            'Median': median,
            'Std_Dev': std_dev
        })
    except Exception as e:
        print(f"[ERROR] Could not process {file}: {e}")

summary_table = pd.DataFrame(results)

print(summary_table)

summary_table.to_csv("flux_statistics_summary.csv", index=False)
print("\n[OK] Summary table saved as 'flux_statistics_summary.csv'")
```

Figure 7: Script to compute statistical values.

```

(jupyter3.10) jose@Skynet:~/practica_3$ python stat.py
      File      Min      Max      Amplitude      Mean      Median      Std_Dev
0  col_data01_flux.lc  2.021487e+03  2.228492e+03  207.00560  2.114210e+03  2.111316e+03  34.293567
1  col_data02_flux.lc  3.793244e+03  3.978416e+03  185.17160  3.874070e+03  3.872892e+03  25.869184
2  col_data03_flux.lc  3.276648e+02  3.776209e+02   49.95608  3.526658e+02  3.527686e+02  6.774859
3  col_data04_flux.lc  1.498717e+04  1.512473e+04  137.56200  1.506093e+04  1.506103e+04  16.306841
4  col_data05_flux.lc  9.030547e+05  9.071032e+05  4048.50000  9.051978e+05  9.052562e+05  635.937153
5  col_data06_flux.lc  4.806892e+04  4.898700e+04  918.07800  4.852376e+04  4.851923e+04  196.127280
6  col_data07_flux.lc  3.332578e+04  3.353123e+04  205.45300  3.341685e+04  3.341693e+04  22.693263
7  col_data08_flux.lc  1.882321e+04  1.977672e+04  953.50700  1.925262e+04  1.925009e+04  214.272458
8  col_data09_flux.lc  1.129617e+05  1.135102e+05  548.45000  1.132393e+05  1.132391e+05  58.445152
9  col_data10_flux.lc  6.983660e+06  7.020503e+06  36842.50000  7.003608e+06  7.003596e+06  7036.357146
10 col_data11_flux.lc  6.340154e+02  6.909569e+02   56.94145  6.615712e+02  6.616030e+02  7.241769
11 col_data12_flux.lc  1.110016e+03  1.484623e+03  374.60690  1.216415e+03  1.214280e+03  36.422395
12 col_data13_flux.lc  8.718454e+04  8.925826e+04  2073.72000  8.810616e+04  8.801679e+04  452.035406
13 col_data14_flux.lc  2.974727e+05  5.719849e+05  274512.22000  5.348370e+05  5.494011e+05  47143.514921
14 col_data15_flux.lc  1.476073e+04  1.490415e+04  143.42300  1.483277e+04  1.483277e+04  16.267743

[OK] Summary table saved as 'flux_statistics_summary.csv'
(jupyter3.10) jose@Skynet:~/practica_3$ 

```

Figure 8: Obtained statistics.

References

- N. Hernitschek. Tutorial 2. https://github.com/ninahernitschek/astroinformatica_I_2025_1/blob/main/tutorial2.pdf, 2025a. PDF available on GitHub.
- N. Hernitschek. Tutorial 4. https://github.com/ninahernitschek/astroinformatica_I_2025_1/blob/main/tutorial4.pdf, 2025b. PDF available on GitHub.
- N. Hernitschek. Python data exploration and visualization. https://github.com/ninahernitschek/astroinformatica_I_2025_1/blob/main/astroinformaticaI_lecture6_pythondataexplorationvisualization.pdf, 2025c. PDF available on GitHub.
- N. Hernitschek. Tutorial session 6: Python data exploration and visualization. https://github.com/ninahernitschek/astroinformatica_I_2025_1/blob/main/tutorial_6.ipynb, 2025d. PDF available on GitHub.