Jak získat data z TESS pomocí Pythonu

(inspirováno přednáškou O.Pejchy na 52. konferenci SPHE)

Python – včetně jeho interaktivní verze IPython a potřebných knihoven – již máme nainstalován, takže si můžeme ukázat, jak s knihovnou lightkurve pracovat.

Spustíme IPython. V příkazovém řádku postupně načteme grafickou knihovnu matplotlib pro interaktivní použití a knihovnu lightkurve:

```
Python 3.8.6 (default, Sep 25 2020, 00:00:00)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.12.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: %matplotlib

Using matplotlib backend: Qt5Agg

In [2]: import lightkurve as lk
```

Nyní se zeptáme databáze TESS, zda-li má nějaká měření, například proměnné hvězdy V0487 Cam:

```
In [3]: search_results = lk.search_tesscut('V0487 Cam')

In [4]: search_results

Out[4]:
SearchResult containing 3 data products.

#   observation   target_name productFilename distance
--- ------------- ----------- --------------- --------
 0 TESS Sector 19   V0487 Cam       TESSCut      0.0
 1 TESS Sector 20   V0487 Cam       TESSCut      0.0
 2 TESS Sector 26   V0487 Cam       TESSCut      0.0
```

Výpisem proměnné search_results jsme zjistili, že z TESS můžeme získat až 3 řady pozorování. Může se stát, že váš objekt (například CzeV proměnné) nebude TESS znát:

```
In [5]: search_results = lk.search_tesscut('CzeV 1416 Cam')

Could not resolve CzeV 1416 Cam to a sky position.
```

Není třeba klesat na mysli, místo katalogového jména lze zadat jeho souřadnice:

```
In [8]: search_results = lk.search_tesscut('07:53:07.163 +78:50:09.96')

In [9]: search_results
```

Out[9]:
SearchResult containing 3 data products.

#	observation	target_name	productFilename	distance
0	TESS Sector 19	07:53:07.163 +78:50:09.96	TESSCut	0.0
1	TESS Sector 20	07:53:07.163 +78:50:09.96	TESSCut	0.0
2	TESS Sector 26	07:53:07.163 +78:50:09.96	TESSCut	0.0

Pro objekt CzeV 1416 Cam má TESS až 3 záznamy měření.

Vraťme se však k V0487 Cam. Příkazem

In [5]: search_results = lk.search_tesscut('V0487 Cam')

jsme zjistili, že TESS pozoroval V0487 Cam celkem třikrát. Nyní je potřeba data stáhnout. To lze udělat příkazem

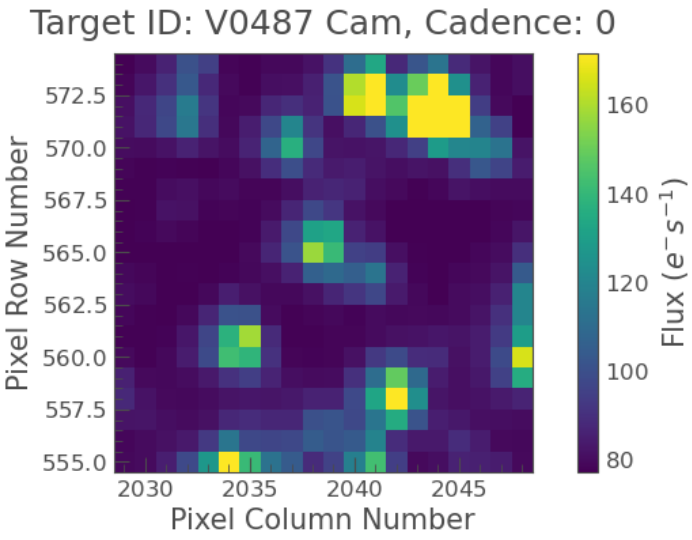
In [6]: tpfs = search_results.download_all(cutout_size=20,quality_bitmask='hardest')

Stáhli jsme z TESS všechna pozorování ve formátu výřezu 20x20 pixelů v okolí objektu/zadané pozice a omezili jsme se na nejvyšší kvalitu měření (quality_bitmask = 'hardest'). Kamery TESS mají rozlišení 21 úhlových vteřin na pixel, tj. námi stažené snímky mají zorné pole asi 0.12 x 0.12 stupně. Z TESS lze stáhnout maximální výřez 99 x 99 pixelů (zorné pole přibližně 0.58° x 0.58°).

V proměnné tpfs (Target Pixel Files) jsou uloženy požadované výřezy. Protože TESS pozorovala objekt vícekrát (3x), je proměnná tpfs **pole**. První pozorování je uloženo v prvku tpfs[0], druhé v tpfs[1] a tak dále. Před dalším zpracováním doporučuji data prohlédnout. Občas se totiž stane, že objekt je zachycen na kraji zorného pole, taková data bude nutno možná vyřadit. Výřezy stažených snímků si zobrazíme pomocí příkazu plot:

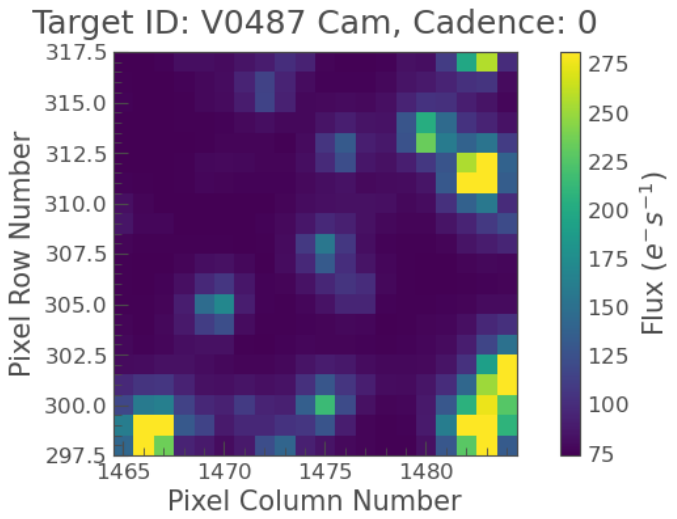
In [7]: tpfs[0].plot()

Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff756e939d0>



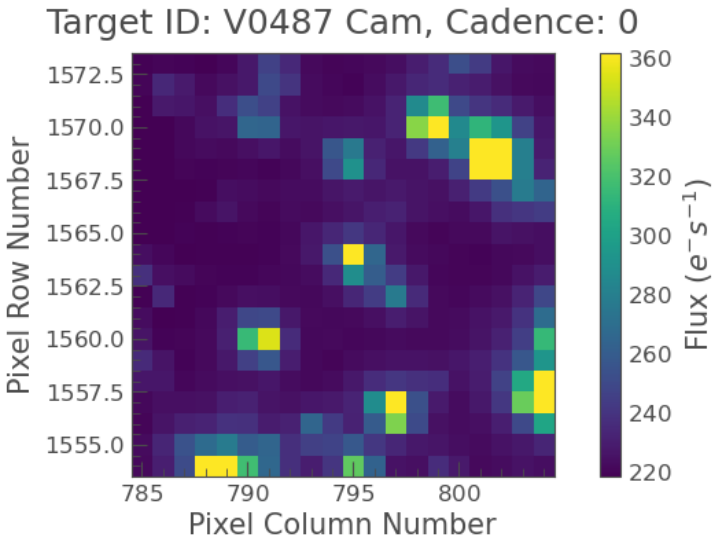
In [8]: tpfs[1].plot()

Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff72f973340>



In [9]: tpfs[2].plot()

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff731173b50>



Z obrázků je patrné, že při žádném měření se objekt nenacházel na kraji zorného pole (4096x4096 pixelů). Abychom získali světelnou křivku, je potřeba vyčíst světelný tok z objektu pomocí aperturní masky a odečíst pozadí. Totéž můžeme učinit i pro vybranou srovnávací hvězdu a určit změnu jasnosti hvězdy v magnitudách.

Aperturní masku vytvoříme příkazem

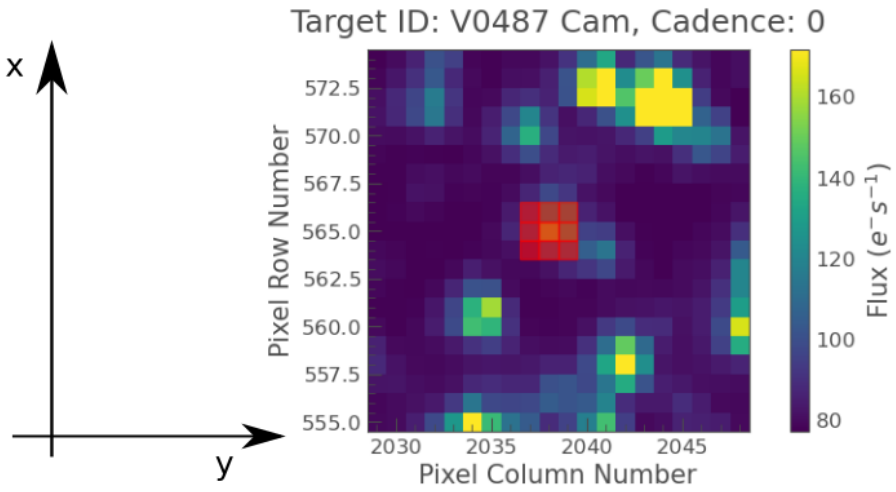
In [10]: target_mask = tpfs[0].create_threshold_mask(threshold = 150, reference_pixel = 'center')

Maska je pole, jehož prvky mají hodnoty reálných čísel, které určují průsvitnost: 0 -neprůsvitná, 1-100% průsvitná. Maska v proměnné target_mask má zatím všude hodnoty nula, tj. jakoby žádný objekt na snímku nebyl. Potřebujeme tedy definovat prvky masky/pole, které umožní vyčíst

světelný tok z objektu včetně pozadí. Počátek souřadnic je v levém dolním rohu, x-ová souřadnice je svislá, y-ová vodorovná. Pro náš objekt vychází maska

In [11]: target_mask [9:12,8:11] = 1

Zobrazit si ji můžeme příkazem (maska bude vykreslena červeně)

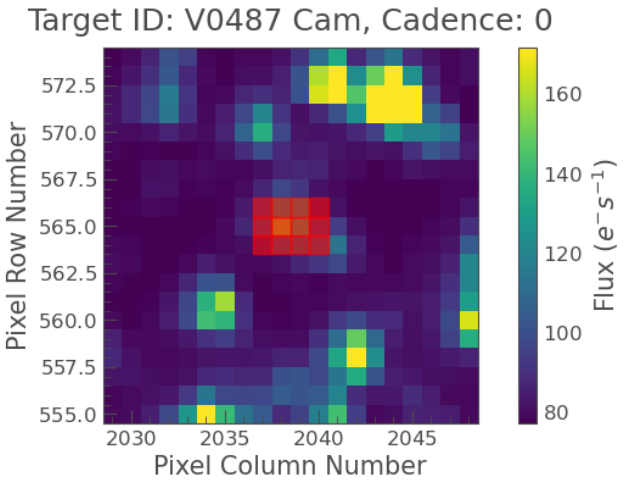


In [12]: tpfs[0].plot(aperture_mask = target_mask, mask_color = 'r')

Pokud budete masku postupně „ladit“, je nutné mít v patrnosti, že je potřeba nastavit hodnoty 0 i 1. Například posloupnost příkazů

target_mask [9:12,9:12] = 1
target_mask [9:12,8:11] = 1

nevytvoří požadovanou masku, ale masku, kde jednotkové prvky budou mít prvky s indexy [9:12,8:12]. Lepší je použít příkazy target_mask [0:19,0:19] = 0; target_mask [9:12,8:11] = 1, kdy nejdříve celá maska vynuluje.



Pro odečtení pozadí budeme potřebovat znát počet pixelů v pozadí s nenulovou hodnotou. Ten zjistíme příkazem

In [13]: n_target_pixels = target_mask.sum()

Výpis hodnoty v proměnné n_target_pixels ukáže hodnotu 9 (máme masku 3x3 pixely)

In [14]: n_target_pixels
Out[14]: 9

Světelný tok odpovídající aperturní masce vyčteme ze snímků příkazem

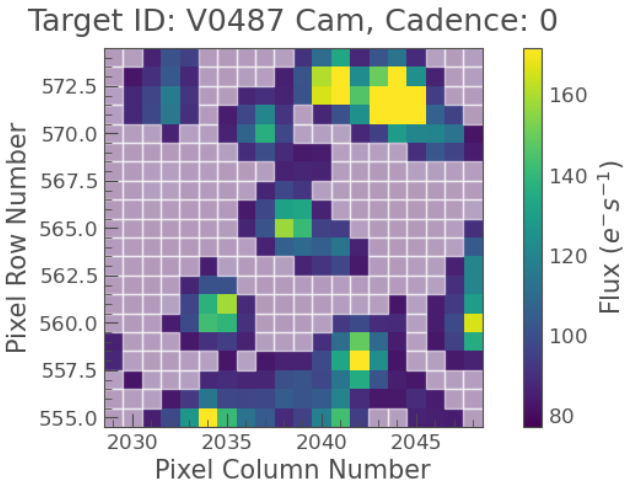
In [15]: target_lc = tpfs[0].to_lightcurve(aperture_mask = target_mask)

Dalším krokem je definice masky pozadí. Tu vytvoříme příkazem

In [16]: bg_mask = ~tpfs[0].create_threshold_mask(threshold = 0.001, reference_pixel = **None**)

Vykreslit si ji můžeme opět příkazem plot (maska bude vykreslena bíle),

In [17]: tpfs[0].plot(aperture_mask = bg_mask, mask_color = 'w')



Počet pixelů, které tvoří masku pozadí, určíme příkazem

In [18]: n_bg_pixels = bg_mask.sum()

In [19]: n_bg_pixels

Out[19]: 200

Abychom mohli odečíst pozadí, které přispívá do světelné křivky, je potřeba určit průměrnou hodnotu pozadí na jeden pixel,

In [20]: bg_lc_per_pixel = tpfs[0].to_lightcurve(aperture_mask = bg_mask)/n_bg_pixels

Hodnota pozadí v aperturní masce je pak dána součinem počtu pixelů v ní a průměrné hodnoty pozadí,

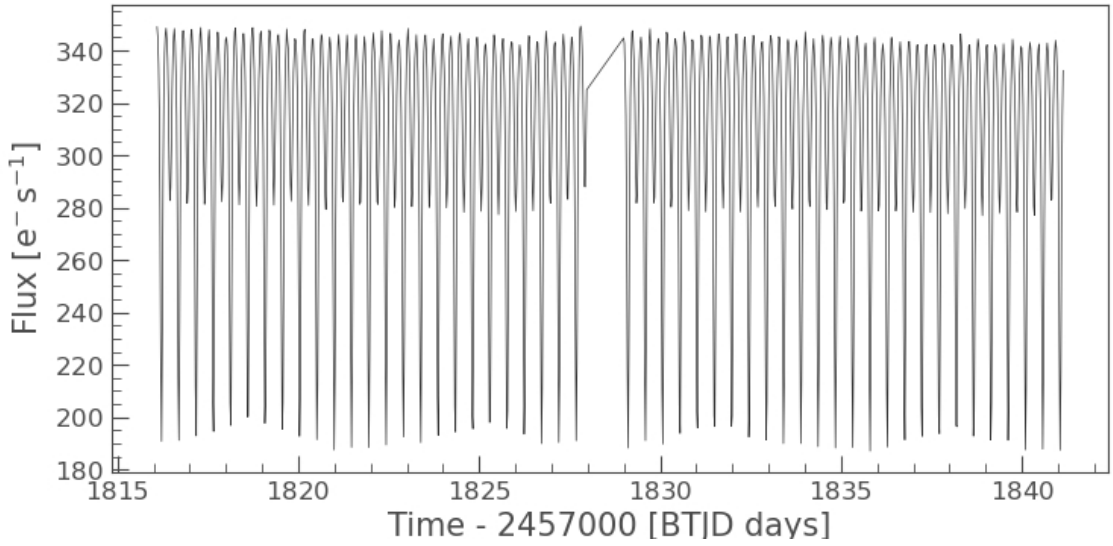
In [21]: bg_estimate_lc = bg_lc_per_pixel * n_target_pixels

Nyní již můžeme odečíst odhad pozadí od světelné křivky v proměnné lc_target

In [22]: corr_lc = target_lc - bg_estimate_lc.flux

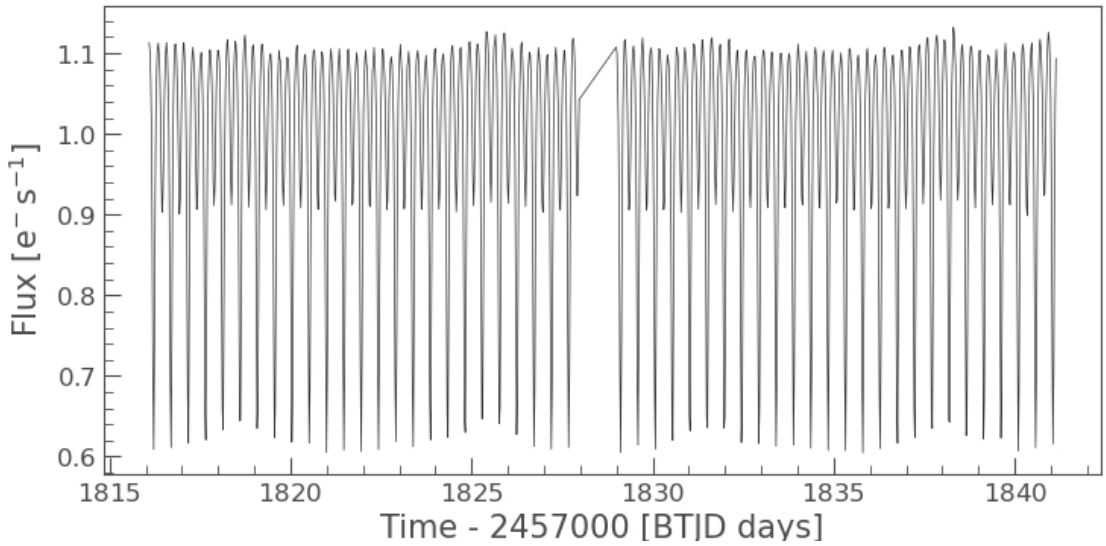
a zobrazit si ji příkazem

In [23]: corr_lc.plot()



Na světelné křivce je vidět trend, pokles toku v čase. Ten je možné eliminovat pomocí funkce flatten, která využívá Savitzkého-Golayova filtru:

In [24]: corr_lc.flatten(101).plot()



Hodnota parametru 101 odpovídá kadenci dat, jak je uvedeno v manuálu ke knihovně lightkurve. Světelnou křivku si můžeme uložit pro pozdější zpracování do textového souboru:

In [25]: corr_lc.flatten(101).to_csv("V0487Cam_0.csv")

Soubor bude uložen do aktuálního adresáře, který zjistíme příkazem **pwd**. Pokud chceme ukládat csv soubory jinam, buď zadáme cestu k souboru i s jeho jménem

corr_lc.flatten(101).to_csv("C:\moje\data\TESS\V0487Cam_0.csv")

nebo pomocí příkazu **cd** změníme aktuální adresář:

cd ‘C:\moje\data\TESS‘.

Podíváme-li se však do vzniklého souboru, zjistíme, že hodnoty času nejsou v JD, ale v JD mínus nějaké číslo (v našem případě 2457000). Tuto korekci lze zjistit z hlavičky dat z TESS, kde se vyskytují parametry BJDREFI (celočíselná část) a BJDREFF desetinná část této hodnoty. "Správný" čas tak určíme přičtením těchto parametrů k hodnotám ve světelné křivce:

```
In [26]: corr_lc.time = corr_lc.time+tpfs[0].get_keyword('BJDREFI')
+tpfs[0].get_keyword('BJDREFF')
```

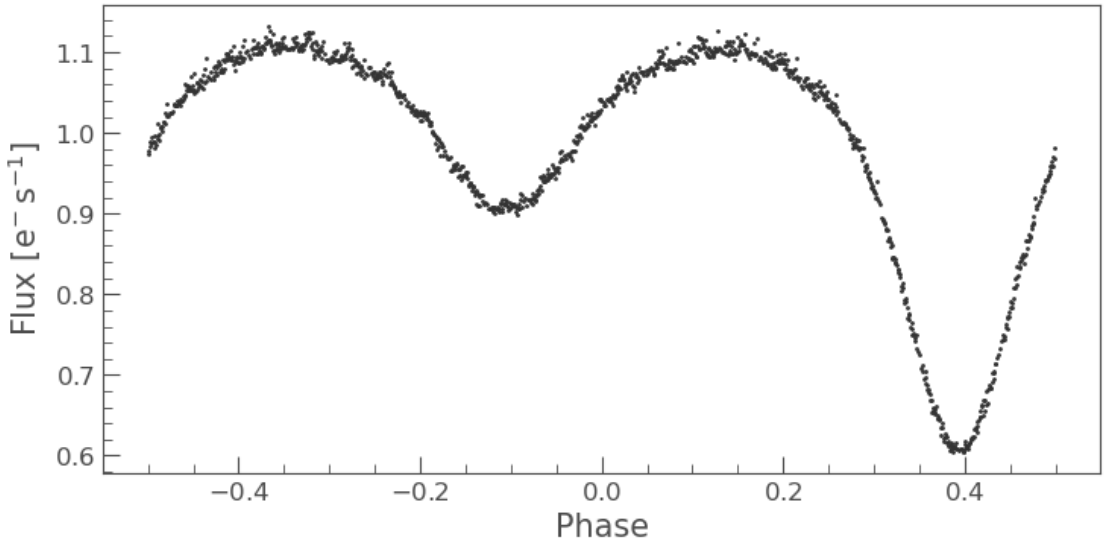
Nyní již příkaz uložení zapíše správné hodnoty JD.

```
In [27]: corr_lc.flatten(101).to_csv("V0487Cam_0.csv")
```

Je-li známa perioda proměnné a čas M0, je možné si nechat rovnou vykreslit fázovou křivku. Z katalogu VSX zjistíme, že V0487 Cam má periodu 0.47766 dne a M0 = 2451514.925,

```
In [28]: perioda = 0.47766; M0 = 2451514.925
```

```
In [29]: corr_lc.flatten(101).fold(perioda, t0 = M0).scatter()
```



Kroky 10 až 29 zopakujeme i pro další pozorování. Pouze nahradíme index v poli tpfs:
tpfs[0] -> tpfs[1] -> tpfs[2].

Stejný postup můžeme také aplikovat na srovnávací hvězdu a určit změnu jasnosti proměnné v magnitudách. V případě V0487 Cam používám srovnávací hvězdu s katalogovým označením UCAC4 841-009024 na souřadnicích 07:43:58.798 +78:03:31.32.

Pomocí již známých příkazů získáme data z TESS na zadané pozici (opět vyčítáme podsnímek 20x20 pixelů):

```
In [30]: search_results_cmp = lk.search_tesscut('07:43:58.798  +78:03:31.32')
```

```
In [31]: search_results_cmp
```

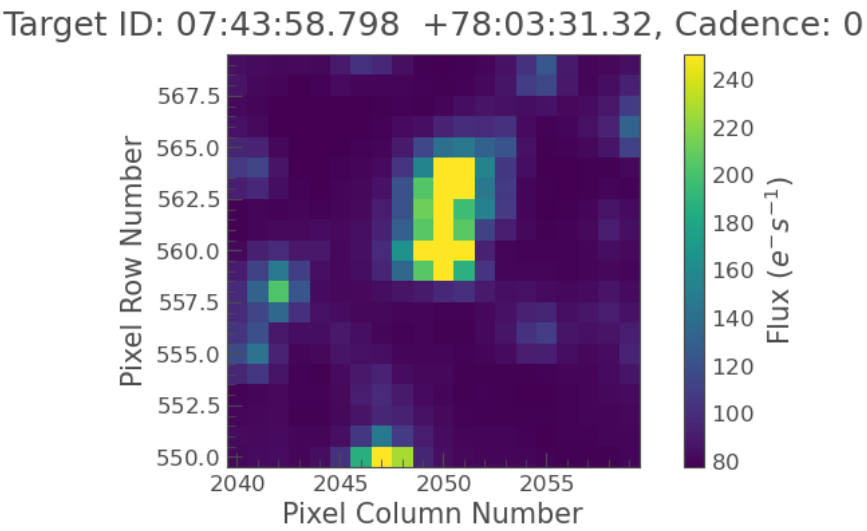
Out[31]:
SearchResult containing 3 data products.

#	observation	target_name	productFilename	distance
0	TESS Sector 19	07:43:58.798 +78:03:31.32	TESSCut	0.0
1	TESS Sector 20	07:43:58.798 +78:03:31.32	TESSCut	0.0
2	TESS Sector 26	07:43:58.798 +78:03:31.32	TESSCut	0.0

In [32]:
```
tpfs_cmp =
    search_results_cmp.download_all(cutout_size=20,quality_bitmask='hardest')
```

Pro konstrukci aperturní masky si jeden snímek z první série zobrazíme:

In [33]:
```
tpfs_cmp[0].plot()
```

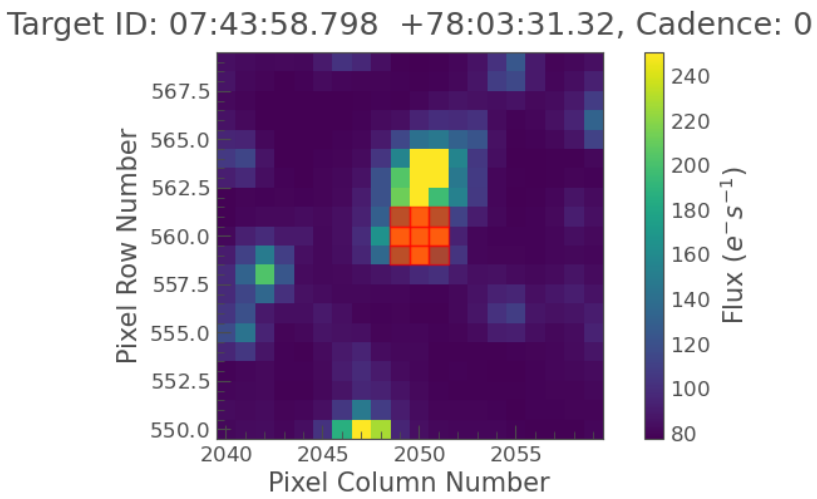


Masku vytvoříme takto:

In [34]:
```
target_mask_cmp = tpfs_cmp[0].create_threshold_mask(threshold = 150,
reference_pixel = 'center')
```

In [35]:
```
target_mask_cmp[9:12,9:12] = 1
```

In [36]: tpfs_cmp[0].plot(aperture_mask = target_mask_cmp, mask_color = 'r')



Počet pixelů v masce zjistíme příkazem

In [37]: n_target_cmp_pixels = target_mask_cmp.sum()

In [38]: n_target_cmp_pixels
Out[38]: 9

Nyní zopakujeme postup uvedený výše v krocích 15 až 27 i pro srovnávací hvězdu:

In [39]: target_cmp_lc = tpfs_cmp[0].to_lightcurve(aperture_mask = target_mask_cmp)

In [40]: bg_mask_cmp = ~tpfs_cmp[0].create_threshold_mask(threshold = 0.001, reference_pixel = **None**)

In [41]: n_bg_cmp_pixels = bg_mask_cmp.sum()

In [42]: bg_cmp_lc_per_pixel = tpfs_cmp[0].to_lightcurve(aperture_mask = bg_mask_cmp)/n_bg_cmp_pixels

In [43]: bg_estimate_cmp_lc = bg_cmp_lc_per_pixel * n_target_cmp_pixels

In [44]: corr_cmp_lc = target_cmp_lc - bg_estimate_cmp_lc.flux

In [72]: corr_cmp_lc.time = corr_cmp_lc.time+tpfs_cmp[0].get_keyword('BJDREFI') +tpfs_cmp[0].get_keyword('BJDREFF')

In [73]: corr_cmp_lc.flatten(101).to_csv("V0487Cam_cmp_0.csv")

Fotometrii, tj. porovnání jasnosti se srovnávací hvězdou, můžeme v Pythonu provést následovně:

Nejdříve si zkopírujeme korigované světelné křivky do pomocných proměnných ***(nesmíme použít de-trendované - normalizované - světlené křivky)***:

```
In [74]: lc_cmp = corr_cmp_lc

In [75]: lc = corr_lc
```

Nyní spočteme změnu jasnosti včetně std. odchylek změn jasnosti pomocí cyklu (je nutno načíst knihovnu numpy pro výpočet logaritmu a druhé odmocniny). K výpočtu použijeme známé vztahy

$$\Delta m_{V-C} = -2.5 \log_{10} \left(\frac{\Phi_V}{\Phi_C} \right)$$

$$\sigma_{\Delta m_{V-C}} = \sqrt{\left(\frac{2.5}{\ln(10)\Phi_V} \sigma_{\Phi_V} \right)^2 + \left(\frac{2.5}{\ln(10)\Phi_C} \sigma_{\Phi_C} \right)^2}$$

kde Φ_V a Φ_C označují světelné toky proměnné a srovnávací hvězdy a σ jejich směrodatné odchylky.

```
In [76]: import numpy as np

In [77]: for ii in range(len(corr_cmp_lc.time)):
   ...:     lc_cmp.flux[ii] = -2.5*np.log10(lc.flux[ii]/lc_cmp.flux[ii])
   ...:     lc_cmp.flux_err[ii] =
np.sqrt((2.5/np.log(10)/lc.flux[ii]*lc.flux_err[ii])**2+(2.5/np.log(10)/
corr_cmp_lc.flux[ii]*corr_cmp_lc.flux_err[ii])**2)
```

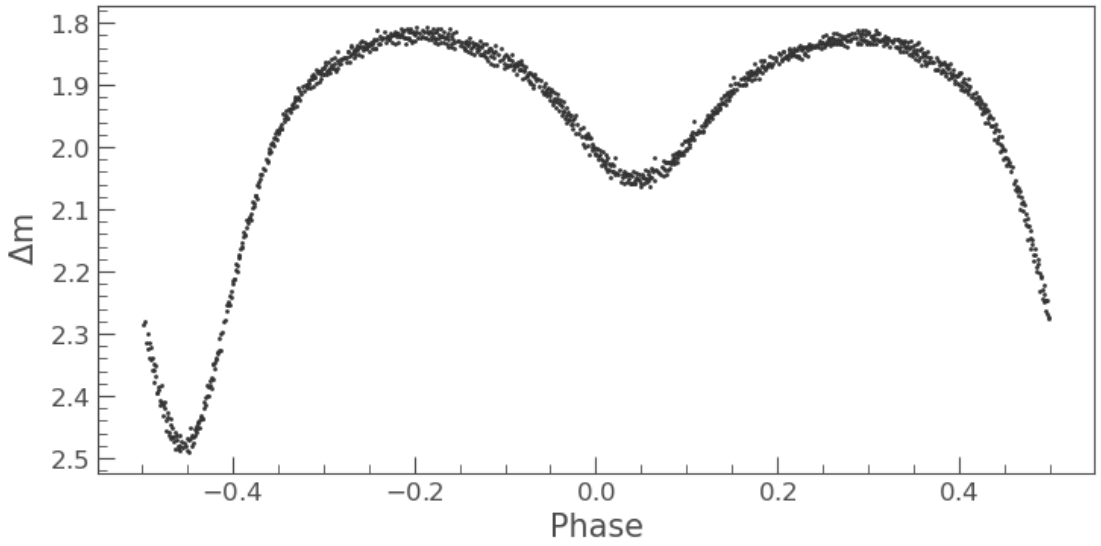
Aby se cyklus (for smyčka) provedla, je potřeba po zadání druhého řádku stisknout Alt+Enter (linux) nebo 2x Ctrl/Shift+Enter (WIndows). Python je tzv. interpreter, výpočet může pro větší pole trvat delší dobu.

Novou světelnou křivku obsahuje proměnná lc_cmp. Uložíme ji příkazem

```
In [78]: lc_cmp.to_csv('V0487Cam_fotom_0.csv')
```

Fázovou křivku si zobrazíme příkazem (.invert_yaxis() otočí smysl osy y):

In [79]: lc_cmp.fold(perioda, t0 = M0).scatter(ylabel='Δm').invert_yaxis()



Uložit si ji do souboru můžeme příkazy

In [80]: lc_cmp.fold(perioda, t0 = M0).to_csv('V0487Cam_fotom_0_faze.csv')

Podobně budeme postupovat i pro další série pozorování, jak bylo uvedeno straně 7.

Pro program SILICUPS můžeme světelnou křivku uložit následovně: Příkaz

In [81]: f = open('V0487Cam_TESS_silicups.dat','wt')

otevře soubor V0487Cam_TESS_silicups.dat pro zápis v textovém formátu. Příkazem f.write() zapíšeme hlavičku – příkaz vrací počet zapsaných znaků.

```
In [82]: f.write("JD              V-C             s(V-C)  \n")
Out[82]: 46
In [83]: f.write("Filter: I, JD: heliocentric\n")
Out[83]: 28
```

Data uložíme do souboru for cyklem přes všechna data v dané sérii:

```
In [84]: for ii in range(len(lc_cmp.time)):
   ...:     buf = "%.12f %.12f %.12f\n" % (lc_cmp.time[ii], lc_cmp.flux[ii], lc_cmp.flux_err[ii])
   ...:     f.write(buf)
```

Smyčku spustíme stiskem Alt+Enter (linux) nebo 2x Ctrl/Shift + Enter (windows) za příkazem f.write(buf). Nakonec soubor uzavřeme.

In [85]: f.close()

Ipython ukončíme příkazem exit:

In [86]: exit