**Processing Document**

**Identification of GRBs (Non-Poissonian Peaks)**

Priyanka Shahane

8th May 2018

The software is to identify non- Poissonian events from the clean event file of CZTI obtained for each orbit within 3 different time binnings (0.1s, 1s, 10s) and three different energy ranges (less than 50 keV, 50 - 100 keV and greater than 100 keV).

**Prerequisites :**

1. Python2.7
2. CZTIpipeline

Identification of non-Poissonian peaks is performed in three stages.

**Stage 1:**

Stage one involves execution of several tasks of the pipeline.It will generate quad clean event file for further processing. **This stage is not required for Veto code as we are using mkf for that**

**Code To be Run -**

gen\_grb\_ip\_new.sh

**Modules Involved -**

cztdatasel,cztpixclean,cztevtclean.

**Input files -**

bc.evt,bc\_livetime.fits and path where the output files are stored

**Output file -**

quad\_clean.evt with det\_threshold=1000 and pix\_threshold=100

**Incase of livetime correction code quad\_livetime will also required.**

**How to Run -**

./gen\_grb\_ip\_new.sh <level2\_bc.evt> <level2\_bc\_livetime.fits>

<Path of directory where you want to store output files>

**e.g.**

./gen\_grb\_ip\_new.sh AS1A04\_199T01\_9000002040\_13792cztM0\_level2\_bc.evt

AS1A04\_199T01\_9000002040\_13792cztM0\_level2\_bc\_livetime.fits /home/cztipoc/test\_orbit

**Stage 2:**

Running this code generates output fits file contains Peak times.

**Code To be Run -**

peak\_serch\_V1.py, peak\_search\_livetime\_correction\_V2.py,

Peak\_search\_veto\_V1.py, Peak\_search\_veto\_detrend\_V2.py

**Input files -**

**Incase of (** peak\_serch\_V1.py**) -**

quad\_clean.evt ,sigma value(3,5,7)

**Incase of (**peak\_search\_livetime\_correction\_V2.py**) -**

quad\_clean.evt , quad\_livetime.fits, sigma value(3,5,7)

**Incase of (** Peak\_search\_veto\_V1.py,**) -**

MKF, sigma value(3,5,7)

**Output file -**

fits file

**How to Run -**

Python2.7 peak\_search\_V1.py <quad\_clean.evt> <output\_fits><sigma value>

e.g.

python2.7 peak\_search\_V1.py AS1G06\_029T01\_9000000964cztM0\_\_07040\_quad\_clean.evt AS1G06\_029T01\_9000000964cztM0\_\_07040\_peak.fits --sigma 3

**For livetime correction code:**

Python2.7 peak\_search\_livetime\_correction\_V2.py <quad\_clean.evt> <quad\_livetime.fits> <output fits file> <sigma value>

e.g.

python2.7 peak\_search\_livetime\_correction\_V1.py AS1G06\_029T01\_9000000964cztM0\_\_07040\_quad\_clean.evt AS1G06\_029T01\_9000000964cztM0\_07040\_quad\_livetime.fits

AS1G06\_029T01\_9000000964cztM0\_\_07040\_peak.fits --sigma 3

**For Veto code:**

Python2.7 Peak\_search\_veto\_V1.py <.mkf> <output fits file> <sigma value>

e.g.

Python2.7 peak\_search\_veto\_V1.py AS1G06\_029T01\_9000000964\_07040czt\_level2.mkf

AS1G06\_029T01\_9000000964cztM0\_\_07040\_peak.fits --sigma 3

**Stage 3:**

Plotting each event from output fits file, it will plot the trigger Time event in 4 quadrants with +100 sec and 100 sec around that event from quad\_clean.evt.For plotting veto detected events we have to use MKF file instead of quad\_clean.evt.

**Code To be Run :**

Plotting.py,plotting\_veto\_lc.py

**Input files :** quad\_clean.evt,.mkf(for veto code),trigger\_time(time from output fits file)

**Output file :** .png

**How to Run -**

python2.7 plotting.py <quad\_clean.evt> --tmark <peak time from output fits file>

**e.g.**

python2.7 plotting.py AS1G06\_029T01\_9000000964cztM0\_\_07040\_quad\_clean.evt --tmark 222198554.03

**For veto code**

python2.7 plotting.py <.mkf> --tmark <peak time from output fits file>

**e.g.**

python2.7 plotting.py AS1G06\_029T01\_9000000964\_07040czt\_level2.mkf --tmark 222198554.03

**Releases :**