Tutorial pulsation search

1 - Introduction

This is a tutorial for pulsating sources search in the XMM-Newton data. The code described below heavily uses the Stingray library. A basic tutorial for this package is given here. Some of the code is directly taken from there.

a - Requirements

The list of python packages needed are given in requirements.txt. It is recommended to follow the same directory architecture.

2 - Directory Architecture

On the bell server at IRAP, the working directory and important files are organized as follows:

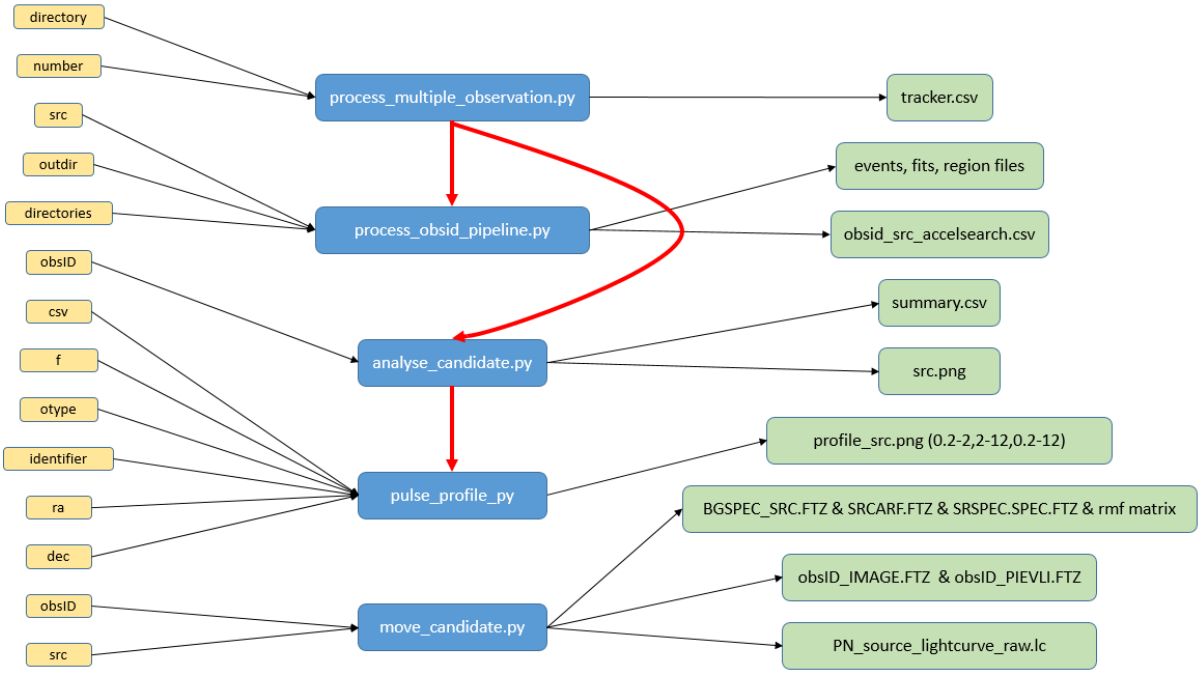
- home
  - amaury
    - code
      - analyse_candidate.py
      - move_candidate.py
      - process_multiple_observation.py
      - pulse_profil.py
      - process_obsid_pipeline.py
      - 4XMM_DR13cat_v1.0.fits
- mnt
  - Data
    - Amaury
      - candidates
        - obsID

            - src
        - Processed_data
            - obsID
            - Tracker.csv
    - xmmcat
        - 4XMM_data
            - DR13_incr_data
                - obsID
                    - odf
                    - product

Some of the paths written in the python files might be personalized and need to be changed for each user.
The code folder contains all of the useful code for this project.
Once an observation is analyzed, all of the outputs are placed in the Processed_data folder. If a particular source is interesting, it can be copied to the candidates folder. Doing so will also download and get useful files for further analysis (Spectral fitting for instance)

3 - Code architecture

The different python files are interdependent and work as follows :



The yellow boxes correspond to the arguments of each file, the blue bowes are the Python files and the green boxes are their outputs.

a - process_multiple_observation.py

This function allows the user to process a *number* of observations in a *directory*. It goes through the observation folders in *directory* and checks if it has been processed before (by checking the tracker.csv file) and if the observation mode is not PrimeSmallWindow (this can be modified in the function check_obs_mode to the user convenience).
If the observation has not been processed before and the observation mode was not PrimeSmallWindow, then it calls the file process_obsid_pipeline.py. It updates the tracker file once the observation is processed.

When the observation has been processed (acceleration search runned for each good enough source), it calls the analyse_candidate.py file.

b - process_obsid_pipeline.py

Code written by Matteo Bachetti (https://github.com/matteobachetti/xmm_pipeline) and modified by Amaury Perrocheau. Only good enough sources are processed (filtered with get_sum_flag function) (Note : the file 4XMM_DR13cat_v1.0.fits is needed). It extracts the events from the main fits file and creates event (ev.nc) region (.reg) and fits files (.fits).
It runs different versions of the acceleration search algorithm and stores the results in csv files corresponding to the version used.
The *outdir* argument corresponds to the directory where all the produced files will be put.
The *directories* argument corresponds to the directory where the PPS files are stored (mnt/xmmcat/4XMM_data/DR13_incr_data/obsID/product).
The *src* argument can be used if one wants to process only certain sources from an observation.

c - analyse_candidate.py

This code finds all the csv files produced by the processing pipeline and apply the ‘Quite fast folding algorithm” around the most powerful candidate for each source. It also computes the EF and Zn statistics for the best candidates and calls for the pulse profile if a Zn statistic is above some detection level.
The argument *obsID* is the path to processed observation (usually mnt/Data/Amaury/Processed_data/obsID). The output of this python file will be in the same folder.
The code was parallelized so that every source of an observation is analyzed at the same time (see par_analysis function).
Once a pulsating source is confirmed with the Zn statistic, a query to simbad if there are any sources already observed within a radius of 10 arcsec. It is looking for X-ray sources but it can be modified in the query_simbad function.
The output is a plot of the accelsearch, EF/Zn search results. A summary file is also written for each observation containing information on the detected pulsations.

d - pulse_profile.py

Plot the pulse profile of a given source in soft X-ray, hard X-ray and whole band. using the information found by the analyse_candidate.py code.
The arguments *csv, identifier, ra, dec, otype, f* can be used if one does not want to run the whole analysis just to plot the pulse profiles.

e - move candidate.py

Moves the interesting candidates from the processed data folder to the candidates folder. Doing so also downloads and copies useful files for spectral analysis (like the arf and rmf matrices) and further analysis (raw lightcurve).