EXODUS

EPIC XMM-Newton Outburst Detector Ultimate System

Institute	IRAP, Toulouse, France
Author	Maitrayee Gupta (2022)
Author V1:	Inés Pastor Marazuela(2019)
Contact	maitrayee.gupta@irap.omp.eu, natalie.webb@irap.omp.eu
Version	1.0, July 27, 2022
Source code	https://github.com/maitrayeegupta/EXOD

Abstract

The purpose of EXODUS is detecting faint rapid transients in the XMM-Newton observations. It works with all EPIC detectors with all read-out modes.
The technique used in EXODUS counts the number of photons received during time windows slicing an observation and then compares those counts.
EXODUS comes into three sub-programmes: a first one calculating the variability and detecting the variable areas for a given observation cleaned from high background; a second one rendering an image of the computed variability; and the third, post-processing which collate the results.
This document provides both a user and technical documentation on EXODUS.
If you use EXODUS for your research, please acknowledge it by citing Gupta & Webb (in prep). Original projects "Variabilitectron" created by Damien Wojtowicz and "EXOD" was created by Pastor-Marazuela et al. (2020).

Contents

1 Introduction

XMM-Newton's automatic pipeline provides results from variability tests that are applied to sources where the number of counts is > 500 in total (PN, MOS1 and MOS2) (Watson et al. 2009). However, faint sources presenting rapid variations might not emit enough photons and could go unnoticed by the pipeline.
4XMM-DR11 [1] , the last version of the catalogue, is the largest catalogue of X-ray sources detected using a single X-ray observatory to date, containing 12712 observations (Rosen et al. 2016). This large amount of observations could contain many of these sources that would provide us with useful data to constrain known phenomena or possibly hide interesting sources that have never been observed before.

The main objective of this algorithm is to detect fast transients among XMM-Newton's EPIC-pn observations, covering timescales from less than a second to a few minutes. These timescales mainly include gamma-ray bursts at high redshift, type-I X-ray bursts in distant galaxies and the X-ray counterpart to FRBs.

EXODUS is a set of Python scripts consisting of three main programmes:

1. A detector that computes the variability in the whole field of view of an EPIC observation from the filtered events file. The detector has two levels of detection:
   (a) The computation of the variability of each pixel of the detector by binning the observation into time windows.
   (b) The detection of variable sources with the sliding boxes technique.
2. A renderer that will produce an output image of the variability of the observation in the World Coordinate System (WCS).
3. Post-processing which collate the results for multiple OBSIDs and cross match it with the SIMBAD [2] database and classify the objects.

[1] http://xmmssc.irap.omp.eu/Catalogue/4XMM-DR11/4XMM_DR11.html
[2] http://simbad.cds.unistra.fr/simbad/

2 Requirements

2.1 Hardware requirements

Depending on the way the programme is used, one should have at least 32 GB of RAM to run it.

2.2 Software requirements

EXODUS is a set of Python scripts, therefore, Python3 is needed. The programme had been tested with Python v3.8, on Linux.

The external libraries required for the renderer are the following:

- `argparse`
- `csv`
- `scipy`
- `multiprocessing`
- `astroquery`
- `numpy`
- `pylab`
- `skimage`
- `os`
- `astropy`
- `matplotlib`
- `re`

Additionally we require the HEASoft release [3] and Science Analysis System (SAS) tasks[4] in order to process the data.

[3] https://heasarc.gsfc.nasa.gov/docs/software/lheasoft/

[4] https://xmm-tools.cosmos.esa.int/external/xmm_user_support/documentation/sas_usg/USG/

3 Using EXODUS

EXODUS can primarily be divided into three sub-programmes: a detector performing variability calculation and detection; a renderer generating images illustrating the detection output; and post-processing scripts which collate the data and cross match . The following section explains the use each of these two scripts, enumerates the parameters offered to tweak the behaviour of the programme and enlightens the content of their outputs.
The source code can be found in the *scripts* folder in `https://github.com/maitrayeegupta/EXOD`.

3.1 Detector

`detector.py` is the main variability detection programme. It calls the functions defined in other python scripts. It returns the number of counts per pixel and per time window, the variability of each pixel, the time windows that were used as well as the detected variable areas and variable sources.
The detector is run with the following command:

```
python3 detector.py -path <obs_path> -obs <OBSID> -out <output_path> [options...]
```

The required arguments are listed below:

`-path`	Path to the clean observation file.
`-obs`	OBSID of the observation.
`-out`	Path to the folder where the output files will be stored.

The optional parameters are the following:

Table 1: Detector parameters

Parameter `--parameter`	`-par`	Accepted values	Default value	Function
`--box-size`	`-bs`	[2..64]	3	Length in pixels of the detection box. Limited by the width of EPIC CCD cameras.
`--detection-level`	`-dl`	$\mathbb{R}^+$	10.0	Level above which an area is considered as being variable.
`--time-window`	`-tw`	$\mathbb{R}^+$	100.0	Duration of the time bins to count the events. It will give the minimal timescale of the variability. Minimal value given by the time resolution of the instrument (73.3 ms) and maximal value limited by the duration of the observation.
`--good-time-ratio`	`-gtr`	[0;1]	1.0	Critical (good time)/TW above which the time window will be taken into account.
`--instrument`	`-inst`	[PN,M1,M2]	PN	EPIC detector.
`--max-threads-allowed`	`-mta`	$\mathbb{N}^*$	12	The maximal number of cores where the programme is allowed to run in parallel during the variability computation. It should be at most the maximal number of threads your computer can run simultaneously.
`--creator`	`-creator`	string	user	User creating the variability files.
`--render`	`-r`	boolean	False	Plot variability output and produce pdf.
`--ds9`	`-ds9`	boolean	False	Plot variability output in emerging ds9 window.
`--novar`	`-nv`	boolean	False	Skip variability computation if already done for the same time window.

After each optional parameter, the desired value (<val>) must be given as --parameter <val> or the shorter version -par <val>. If a boolean argument is not given, it will be false by default. When provided as -par or --parameter, it becomes true.

3.2 Libraries

In this section I present the complimentary scripts that are used for the implementation of EXODUS. These scripts are not runnable, they are used as libraries for both the renderer and the detector.

- renderer.py
  This script contains the functions that are used for plotting the output variability files.
  - render_variability Function producing output pdf plots from the computed variability.
  - render_variability_all Function producing output pdf plots from the variability computed with four different sets of parameters.
  - ds9_renderer Function opening a ds9 window with the output variability and detected sources.
- file_names.py
  This script defines the name of the output files that will be stored in <output_folder>. It also contains the path to where heasoft and xmmsas are installed.
- file_utils.py
  It contains functions complimentary to the variability computation: opening and closing files and geometrical transformations to define the coordinates of the variable sources. The functions defined in the script are the following:
  - open_files Function opening files and writing their legend.
  - close_files Function closing all files.
  - read_from_file Function returning the content of a file.
  - read_tws_from_file Reads the list of time windows from its file.
  - Source (class) Datastructure providing easy storage for detected sources.
  - read_sources_from_file Reads the source from their file.
  - ccd_config Provides PN CCD configuration to arrange the variability data.
  - data_transformation Performs geometrical transformations to convert from source raw coordinates to sky coordinates.
- fits_extractor.py
  This script extracts the data from the filtered events file and the GTI file with the following functions:
  - extraction_photons Function extracting the events list ordered by arrival time from its FITS events file.
  - extraction_info Function extracting some useful information from the header of the events file.

- `extraction_deleted_periods` Function extracting the list of periods removed by high background filtering from the GTI file.
- `fits_writer` Function writing the variability and detected sources to a fits file

- `variability_utils.py`
  The functions that compute the variability and detect the variable areas are defined in this script and are the following:

  - `variability_computation` Function implementing the variability calculation.
  - `box_computations` Function summing the variability values of a box.
  - `__add_to_detected_areas` Function summing the variability values into a box.
  - `variable_areas_detection` Function detecting variable areas into a variability matrix.
  - `variable_sources_position` Function computing the coordinates of the detected variable sources.
  - `remove_readout_streak` Function to detect and remove false variable sources which are part of a read out streak.

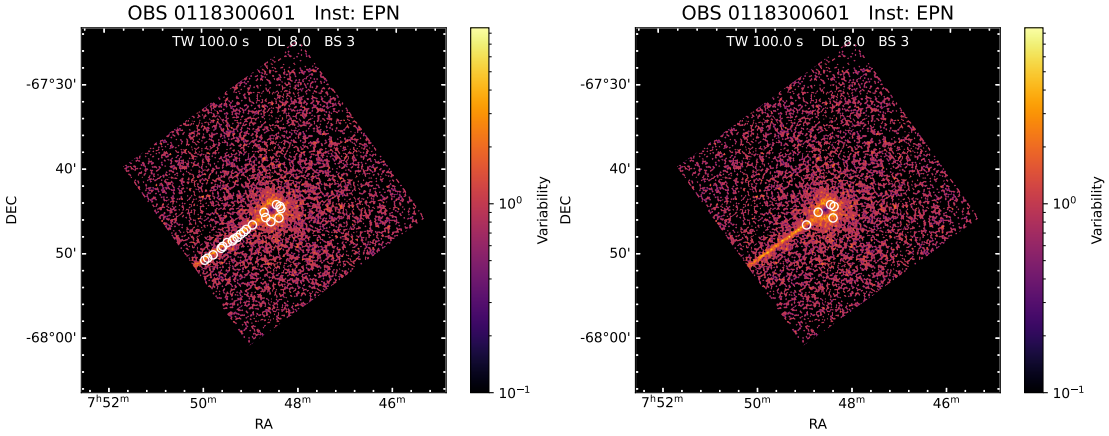

Figure 1: Removal of the readout streak

3.3 Post Processing

The post processing scripts are useful while running EXODUS on a large batch of OBSIDs.

- `match_with_xmm_catalog.py`
  This script matches the variable source with the 4XMM-DR11 catalogue. This helps us to determine if the variable source is already part of the source catalogue or if we have identified a new source. In addition this helps us to identify sources though already present in the catalogue have not been detected as variable as part of the XMM variability pipeline.

  - `path` Path where 4XMM-DR11 catalogue fits file is located.

- `remove_bright_sources.py`
  We define bright sources as ones with the EPIC combined band 8 flux is greater than 10^{-11} $ergcm^{-2}s^{-1}$. We eliminate any sources which lie within 1 arcmin of the bright source. The

list of bright sources from the 4XMM-DR11 catalogue can be found in bright_sources.fits table.

  - `path` Path where bright_sources.fits is located.

- `match_with_simbad.py`
  This script makes use of the `astroquery.simbad` utilities to query the SIMBAD database to find the closest matches to the detected variable sources. The result table also reports the matching distance which can be used to elimiate false matches.

- `simbad_subclass.py`
  The script uses the results of SIMBAD associations result and classifies the objects into one of five classes viz. CompactObject, Star, AGN, Galaxy and Unspecified. One can always edit the dictionary in this file to combine sub-classes or add additional classes based on what kind of objects one is interested in.

The post processing scripts can be run as follows.

```
python3 $SCRIPTS/match_with_xmm_catalog.py -path /mnt/data/Maitrayee/EXOD/
python3 $SCRIPTS/remove_bright_sources.py -path /mnt/data/Maitrayee/EXOD/
python3 $SCRIPTS/match_with_simbad.py
python3 $SCRIPTS/simbad_subclass.py
```

3.4 Output files

Table 2: Files generated by EXOD

File	Contents	Structure
`variable_sources.csv`	Properties of the detected variable sources.	Source ID; CCD number; RAWX on CCD; RAWY on CCD; Physical X; Physical Y; Physical radius; Right Ascension; Declination; Radius in arc seconds
`variability_file.fits`	Fits file of the computed variability. Can be opened with ds9.	Primary Image file with the projected on-sky variability and Binary file with the detected sources coordinates.
`ds9_variable_sources.reg`	Region file with the sources position compatible with ds9.	
`log.txt`	Contains informations about the execution of the detector.	
`variability.pdf`	Image showing variability of the pixels. The darker colors represent a lower variability and the lighter colors represent a higher variability with a logarithmic colorscale.	
`sources.pdf`	Image showing the variability and the location of the detected sources, represented by a white circle.	

Figure 2 presents the output of the renderer: the variability chart of the analysed observation. The X-axis represents the Right Ascension (RA) and the Y-axis represents the Declination (Dec) of the field of view. The color code corresponds to the variability of each pixel and the green circle the position of the detected variable source.

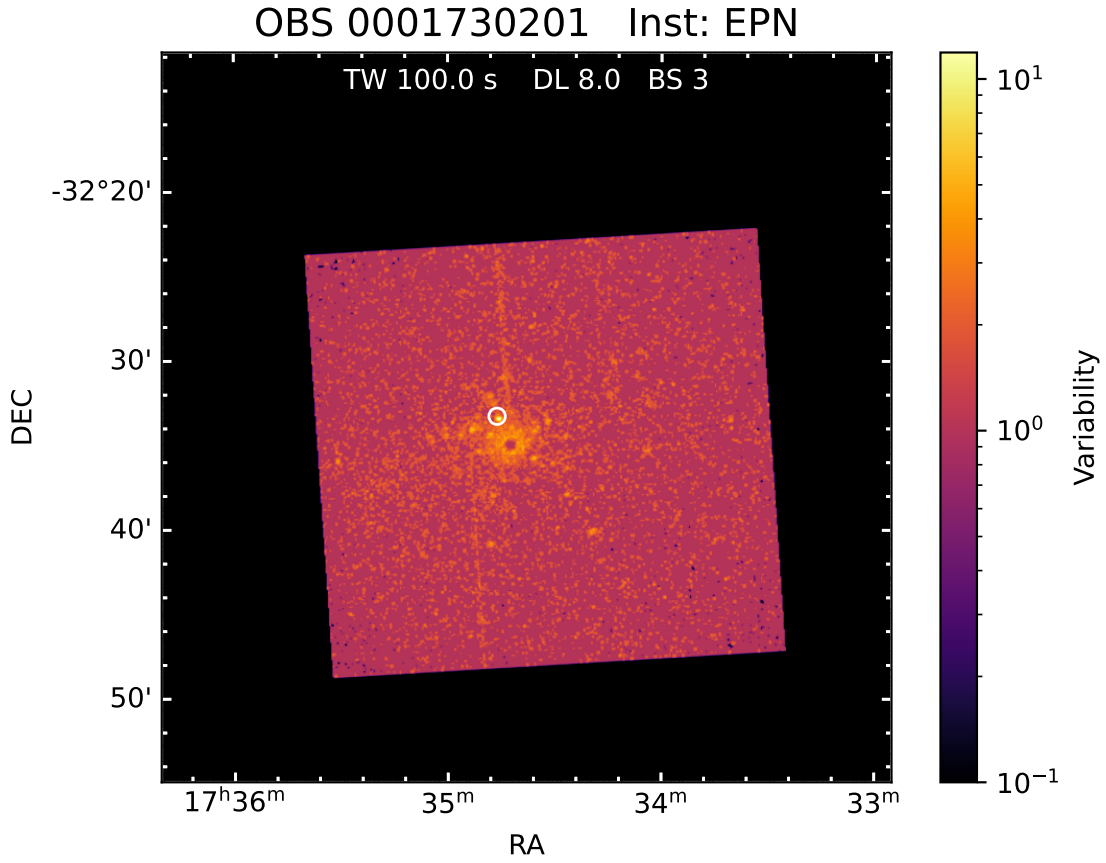

Figure 2: `sources.pdf` of the observation 0001730201 computed with the default values.

4 The algorithm

4.1 Variability computation

The steps to compute the variability of the filtered events are described below.

1. The time and pixel arrival of every detected event of the observation are extracted from the events file.
2. Photons detected in a 3×3 square around the central pixel are added together to ensure that only real sources are detected and to increase the signal-to-noise ratio.
3. The counted events are stored in a three-dimensional matrix that contains the number of photons that have been detected per pixel and per time window.
   - The first dimension is the RAWX pixel of the EPIC-pn camera that detected the photon.
   - The second dimension is the RAWY pixel of the camera that detected the photon.
   - The third dimension is the time window which stores the arrival time of the photon.

   This is the equivalent of creating a lightcurve for every single photon of the detector.
4. The good time intervals are extracted from the GTI file.
5. The good time ratio, defined as the time belonging to the GTI (good time) of the considered time window divided by the duration of the time window, is computed for each time window.

6. The number of counts per time window will be divided by the good time ratio. Only those time windows with the good time ratio above a critical value of the good time ratio will be considered. This will normalize the number of photons that have been detected during a time window that has been shortened by the bad time periods.

7. The variability $\mathcal{V}$ of each pixel is given by:

$$\mathcal{V} = \begin{cases} max(\mathcal{C}_{max} - \tilde{\mathcal{C}}, |\mathcal{C}_{min} - \tilde{\mathcal{C}}|)/\tilde{\mathcal{C}} & \text{if } \tilde{\mathcal{C}} \neq 0 \\ \mathcal{C}_{max} & \text{if } \tilde{\mathcal{C}} = 0 \end{cases}$$

Where $\mathcal{C}$ are the counts per pixel per time window, $\mathcal{C}_{max}$ and $\mathcal{C}_{min}$ are respectively the maximal and minimal number of counts of the considered pixel and $\tilde{\mathcal{C}}$ is the median number of counts over the time windows for the considered pixel.
The expression $\mathcal{C}_{max} - \tilde{\mathcal{C}}$ targets sources presenting outbursts, while $|\mathcal{C}_{min} - \tilde{\mathcal{C}}|$ points to those sources with a period of lower flux. Considering the maximum between the two allows the detection of a wider variety of phenomena.
The division by the median $\tilde{\mathcal{C}}$ will give the variability relative to the flux.

The variability computation algorithm can be visualised in Figure 3.

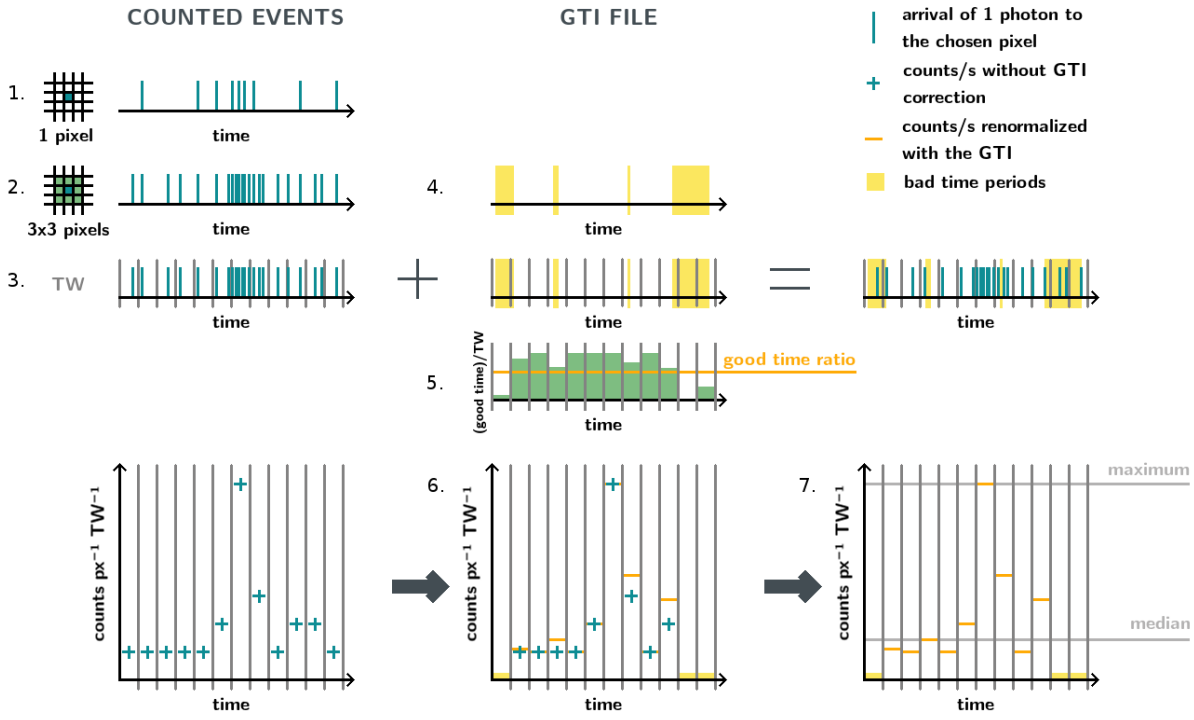

Figure 3: Diagram of the variability computation with the different stages of the algorithm.

4.2 Variable sources detection

The variable areas are detected with the sliding boxes technique. A box of size $|b|^2$ pixels will move through all of the pixels of the observation.

1. The median of the variability of the pixels of the detector, $\tilde{\mathcal{V}}$, is computed.

2. Once the position of the box is defined, the variability of each pixel in the box is summed.

3. When the variability of the box $\mathcal{V}_{box}$ is above a chosen threshold, the pixels of the box are considered as a variable area. The value of this threshold is given by the following expression:

$$\mathcal{V}_{box} > DL \times |b|^2 \times \tilde{\mathcal{V}}$$

Where DL is the detection level, b the length of the box in pixels and $\tilde{\mathcal{V}}$ the median value of the variability.

4. Once this is done, the position of the detection box is shifted by one pixel and the process is repeated.

5. When two consecutive boxes are variable, the pixels of both boxes are joined into a single variable area.

The variable sources are located at the barycenter of the variable areas. The position of the (X,Y) coordinates will be the mean value of the position of the pixels belonging to the variable areas.

5 Data analysis

EXODUS is also provided with a set of scripts that allow one to perform the whole data analysis of the observations. These scripts include the following steps:

- Download the required files
- Filter the events files
- Apply EXOD to a set of observations
- Automatically generate the lightcurves of the detected sources and compute the χ^2 and Kolmogorov-Smirnov probabilities of constancy, P(χ^2) and P(KS) respectively.

Additionally to the EXODUS python scripts, it makes use of XMM-Newton's Science Analysis System (SAS) version 16.1.0 (SOC 2018)[5]. We used the Xronos sub-package[6] of HEASOFT version 6.22.1 (Blackburn 1995).

5.1 Analysing a set of observations

The *bash* folder contains some scripts that can be used to perform the whole variability analysis of one or a whole set of observations.

- `download_observation.sh`
  This script downloads the files required for the variability computation and lightcurve generation. More information about the data download can be found in the XMM-Newton Science Archive[7]

  ```
  download_observation.sh <FOLDER> <OBSID>
  ```

- `filtering.sh`
  This script filters the PIEVLI file in an energy range between 0.5 and 12 keV, and it removes high background rates, bad pixels and PATTERN>4. This is done by calling some SAS commands. The path to where the SAS is installed is defined in the `preliminaries` function and might need to be changed.
  The default rate for the creation of the GTI file is 0.5 counts s^{-1}, but this can be modified with the optional `<RATE>` argument.
  The script is run with the following command:

  ```
      bash filtering.sh -obs <OBSID> --rate [RATE] --instrument [INSTRUMENT] \
  --folder [FOLDER] --scripts [SCRIPTS]
  ```

  Where `<OBSID>` is the observation identifier; `[FOLDER]` is the path to where the observation is stored (there is a default folder); `[SCRIPTS]` is the path to the EXODUS sctripts (there is a default folder); `[INSTRUMENT]` is by default PN; `[RATE]` is the threshold rate for the GTI generation with 0.5 as default value. If a non numerical value or no value is given, the rate of the observation will be plotted so that the user can choose a GTI rate.
  This function generates a clean events file `$FOLDER/$OBSID/PN_clean.fits`, a GTI file `$FOLDER/$OBSID/PN_gti.fits`, and an image file `$FOLDER/$OBSID/PN_image.fits`.

[5] https://xmm-tools.cosmos.esa.int/external/xmm_user_support/documentation/sas_usg/USG/
[6] https://heasarc.gsfc.nasa.gov/ftools/xronos.html
[7] http://nxsa.esac.esa.int/nxsa-web/#aio

- `lightcurve.sh`
  This script generates lightcurves automatically from the position of the variable source detected with `detector.py`, and it computes P(χ^2) and P(KS). It uses a set of SAS and FTOOLS Xronos tasks. The lightcurves are plotted with `lcurve.py`, described below.
  The script is run with the following command:

```
bash lightcurve.sh -obs <OBSID> [options]
```

  `<OBSID>` is the observation identifier, and the options are described in the table below.

Parameter		Default	Function
`--source-id`	`-id`	1	Source identifier, given in the first column of `detected_variable_sources.csv`.
`--folder`	`-f`	/mnt/data/Ines/data	Path to the folder containing the observations.
`--scripts`	`-s`	/mnt/data/Ines/EXOD	Path to the folder where the scripts are contained.
`--detection-level`	`-dl`	8	Detection level.
`--time-window`	`-tw`	100	Time window.
`--good-time-ratio`	`-gtr`	1.0	Good time ratio.
`--box-size`	`-bs`	3	Box size.
`--instrument`	`-i`	PN	Instrument in use.

  This script produces fits files with the .lc extension for the source, the backgrounds and the correlated lightcurves. The extraction regions are stored in the *lc.log file, and the lightcurve is stored as a pdf file. All of the outputs contain the name of the source. The probability of constancy of the source is stored in an output file `$FOLDER/sources_variability_<DL>_<TW>`

- `exodus.sh`
  This script performs the whole variability analysis for a list of observations. It performs the variability detection on all 3 EPIC detectors. This script can also generate the lightcurves for all detected variable sources (by default it does not). Additionally this script can be run in two modes, iterative detection mode on or off. If this mode is enabled, the code steps though detection levels starting from the largest to the lowest till a detection is made. This is useful to detect the most variable sources in an observation. The parameters of the script are the following:

Parameter		Default	Function
`--obs_id_file`	`-obs`	No default	Path to the file containing the obsids.
`--lc_enable`	`-lc`	false	generate lighcurves for variable sources.
`--iterative_det_level`	`-it`	false	run in iterative detection level mode.

  This script calls the following scripts: `download_observation.sh`, `filtering.sh`, `lightcurve.sh`, `parallel.sh`, `detector.py` and `renderer.py` as well as the post processing scripts.
  It creates a an output directory for each OBSID, and each of those directories contains a series of output files. The most important are the following:

  - `sources_variability_<DL>_<TW>_<BS>_<GTR>` : text file containing the name and observation of the detected sources as well as their P(χ^2) and P(KS).
  - `variability_observations_<DL>_<TW>_<BS>_<GTR>.pdf` : pdf containing the `sources.pdf` of each observation.
  - `lightcurves_<DL>_<TW>_<BS>_<GTR>.pdf` : pdf containing the lightcurves of all the detected sources.

In addition it also produces a collated table of all the variable sources detected in the set of OBSIDs which have been cross matched with the SIMBAD database as well as the classified based on the object type. An example output is shown in Table ??.

- `parallel.sh`
  This script is used to run processes in parallel. The processes are read line by line from the file where they have been written. The script is used as follows:

```
bash parallel.sh <file> <CPUS>
```

  `<file>` is the file containing the commands and `<CPUS>` the number of CPUs that will be used in parallel.

6 Tutorial

In this section, we give an example of how to use EXOD, including downloading the files, filtering them, computing the variability and generating the lightcurves automatically. We give the command lines that should be launched from the terminal.

6.1 Preliminaries

The EXOD repository has to be cloned to the machine where the variability analysis will be performed. The first thing after cloning the repository from github is to modify the paths to the HEADAS, SAS and CCF installations. These values are defined in `file_names.py` and called by different scripts, they are given by

```
export HEADAS=/sasbuild/local/sasbld03n/GNU_CC_CXX_9.2.0/headas/x86_64-pc-linux-gnu-libc2.27
export LD_LIBRARY_PATH=/sasbuild/local/sasbld03n/GNU_CC_CXX_9.2.0/qt-x11-free/lib/
export SAS_DIR=/home/filippos/SASscreeningDR11/
export SAS_CCFPATH=/home/filippos/sasbuild/ccf/pub
source /home/filippos/SASscreeningDR11/sas-setup.sh
```

Unless you have the exact same version of HEADAS and SAS installed in the exact same path, the nviroment needs to be setup to point to the right path.

6.2 One observation

We are first going to set some useful variables that we will use in multiple occasions. For this tutorial, we will be analysing observation 0831790701. First we define some variables:

```
obs=0831790701
FOLDER=/mnt/data/Maitrayee/EXOD/data
SCRIPTS=/mnt/data/Maitrayee/EXOD/scripts
```

We now proceed to the download of the observation, which can take from a few seconds to ~ 2 min, depending on the observation and your machine.

```
bash $SCRIPTS/download_observation.sh $FOLDER $obs PN MOS1 MOS2
```

After the download, we filter the observation. In this example, the threshold rate for the GTI generation will be the default (0.5 for PN and 0.4 for MOS). The - flag can be used to run the script on either the PN or the M1 or M2 observation.

```
bash $SCRIPTS/filtering.sh -f $FOLDER -o $obs -s $SCRIPTS -i PN
```

This should generate the following files in the `$FOLDER/$obs` folder: `PN_clean.fits`, `PN_gti.fits`, `PN_rate.fits`, `PN_image.fits` `ccf.cif`.
With these files, we are now ready to compute the variability of our observation! In this step we apply `detector.py` to the observation. See Table 1 for an explanation of the parameters used.

```
python3 -W"ignore" $SCRIPTS/detector.py -path $FOLDER/$obs \
        -bs 3 -dl 8 -tw 100 -gtr 1.0 -mta 16 --render -inst PN
```

This should generate the files listed in Table 2 in the `$FOLDER/8_100_3_1.0` folder, and a ds9 window showing the variability and any detected sources should pop up. Excitingly, in Fig. 4 we can see that one variable source was detected. The same process can be repeated on the MOS1 and MOS2 observations too. We will now proceed to the extraction of the lightcurve for the object that was detected.

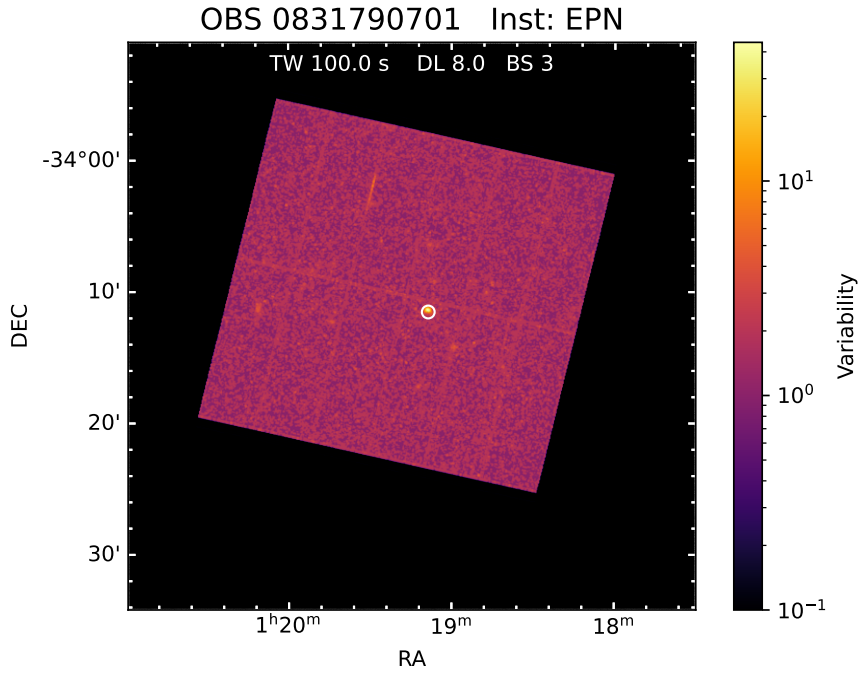

Figure 4: `sources.pdf` of the observation 0652250701 computed with the values of the tutorial. One variable source is detected at the position of the white circle.

```
bash  $SCRIPTS/lightcurve.sh <path_obs> <path_scripts> <instument> <id> \
<DL> <TW> <GTR> <BoxSize> <output_log>

for example:
bash $SCRIPTS/lightcurve.sh $FOLDER $SCRIPTS PN 1 8 100 1.0 3 LC_PN
```

With `src=J011908-341133` in this case, the files `$src_lc_0.074_bgd.lc`, `$src_lc_0.074_src.lc`, `$src_lc_100_bgd.lc`, `$src_lc_100_src.lc`, `$src_lccorr_100.lc`, `$src_region.txt` and `$src_lc_100.pdf` should have been generated in the `$FOLDER/$obs/lcurve_100_PN_1` folder. The pdf of this lightcurve is shown in Fig. 5.

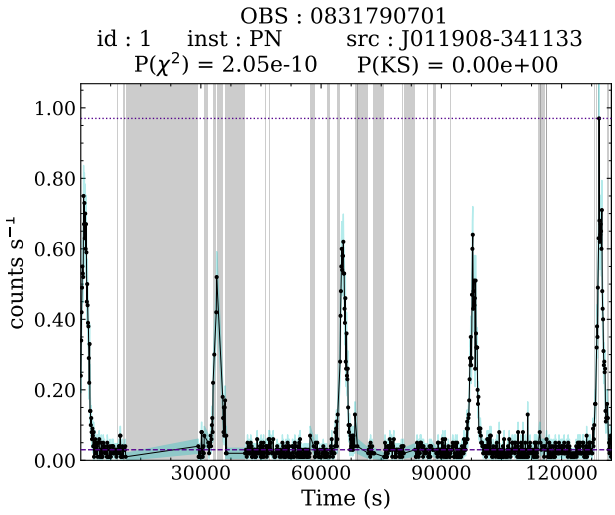

Figure 5: Lightcurve of the detected source in observation 0831790701 plotted with `lcurve.py`. Also shown are the computed χ^2 and Kolmogorov-Smirnov probabilities of constancy, P(χ^2) and P(KS) respectively.

6.3 Group of observations

The individual step shown above have been fully automated to produce a collated report of all variable sources given a set of OBSIDs.

```
bash run_exodus.sh -obs_id_file obs_ids.txt \
-lc_enable false -iterative_det_level false
```

The obs_ids.txt is a file containing a list of OBSIDs of the observations we want to analyse. In this example we pick three of them 0831790701 0001730201 and 0002970201.

```
SCRIPTS=/mnt/data/Maitrayee/EXOD/scripts
FOLDER=/mnt/data/Maitrayee/EXOD/data
```

The output of the renderer, `variability_observations_8_100_1.0_3.pdf`, is shown in Fig. 6, and the lightcurves of the detected sources `lightcurves_8_100_1.0_3.pdf` are plotted in Fig. 7.

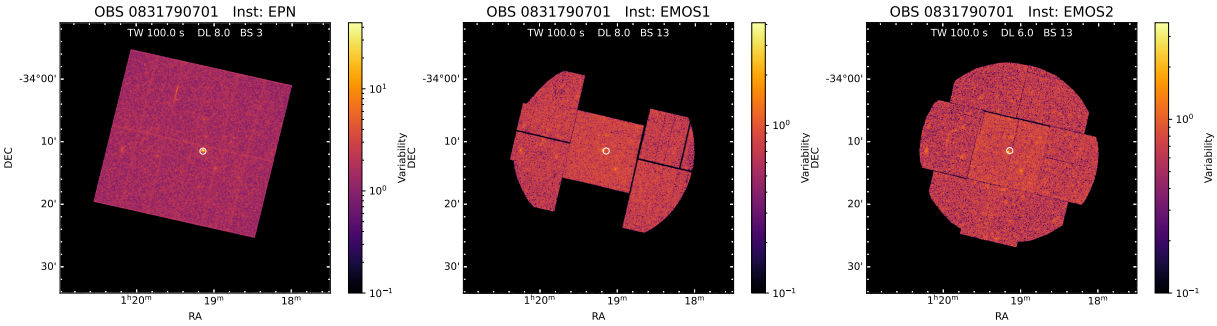

Figure 6: `sources.pdf` of the observation 0652250701 computed with the values of the tutorial. One variable source is detected at the position of the white circle.

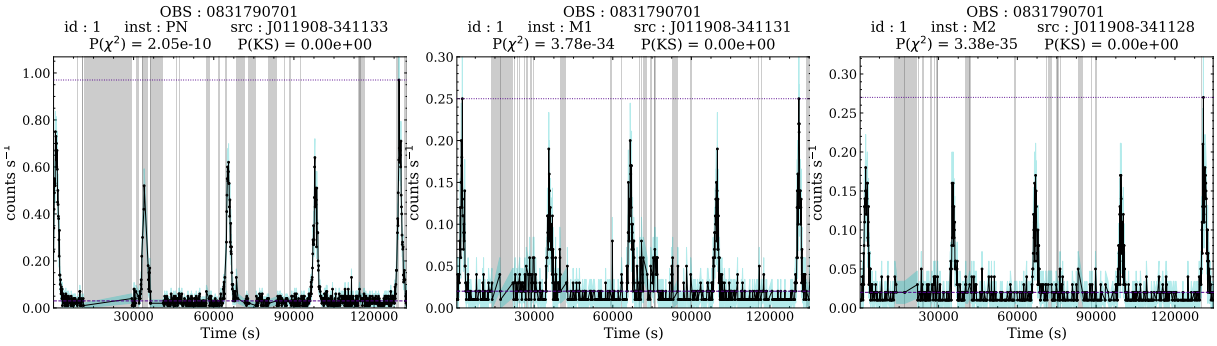

Figure 7: `sources.pdf` of the observation 0652250701 computed with the values of the tutorial. One variable source is detected at the position of the white circle.

An example output table is shown in Table ??

BAT NAME	z	$\log L_{14-195}$	$F_{1.4}$	$F_{\nu_{W3}}$	$\log R$	Radio class	Optical type	$\log M_{BH}/M_{\odot}$	L_{bol}	λ_E	L_{14-195}/L_{bol}	Γ_{14-195}	E_c
		erg s^{-1}	erg s^{-1}cm^{-2}Hz-1	erg s^{-1}cm^{-2}Hz-1					erg s^{-1}				
SWIFT J0021.2-1909	0.0957	44.6	1.20E-23	2.26E-25	1.73	RL	2	9.16	45.1	-2.2	0.31	1.7	
SWIFT J0042.9-2332	0.0221	43.8	4.22E-25	1.17E-24	-0.44	RQ	1	8.6	44.5	-2.25	0.2	1.76	
SWIFT J0109.0+1320	0.0597	44.5	1.37E-22	3.72E-25	2.57	RL	2	8.48	44.8	-1.74	0.44	1.97	
SWIFT J0134.1-3625	0.0298	44.3	5.09E-23	9.93E-25	1.71	RL	2	8.64	44.6	-2.1	0.49	1.07	
SWIFT J0215.6-1301	0.148	44.9	4.68E-23	1.48E-25	2.5	RL	2	8.81	45.3	-1.63	0.41	1.87	
SWIFT J0235.3-2934	0.06	44.2	2.47E-24	1.97E-25	1.1	RL	1	9.02	44.6	-2.55	0.45	1.68	
SWIFT J0249.1+2627	0.0582	44.5	2.22E-25	6.87E-25	-0.49	RQ	2	9.15	45.1	-2.17	0.26	1.78	
SWIFT J0255.2-0011	0.0289	44.4	1.54E-24	1.65E-24	-0.03	RQ	2	9.25	44.8	-2.51	0.34	1.69	
SWIFT J0300.0-1048	0.0326	43.6	1.75E-25	3.98E-25	-0.36	RQ	1	8.71	44.3	-2.48	0.2	1.98	
SWIFT J0407.4+0339	0.0884	44.8	5.68E-23	7.59E-26	2.87	RL	2	8.61	44.5	-2.2	2.08	1.29	
SWIFT J0418.3+3800	0.0493	44.8	1.53E-22	6.91E-25	2.35	RL	1	8.8	44.9	-1.96	0.75	1.76	
SWIFT J0429.6-2114	0.07	44.1	1.03E-25	4.32E-25	-0.62	RQ	1	8.82	45.1	-1.87	0.11	1.81	
SWIFT J0433.5-5846	0.103	44.4	2.50E-26	7.11E-26	-0.45	RQU	2	8.5	44.6	-1.98	0.65	1.45	
SWIFT J0445.0-2816	0.147	45	6.74E-23	1.16E-25	2.77	RL	2	9.11	45.2	-2.05	0.64	1.44	
SWIFT J0508.1+1727	0.0174	43.2	2.50E-26	9.37E-25	-1.57	RQU	1	8.59	44.1	-2.55	0.12	1.98	
SWIFT J0516.2-0009	0.0325	44.2	1.23E-25	2.20E-24	-1.25	RQ	1	8.72	45.1	-1.75	0.14	2.07	
SWIFT J0519.5-4545	0.0351	44	8.50E-22	6.23E-25	3.14	RL	2	8.7	44.6	-2.21	0.28	1.79	
SWIFT J0752.2+1937	0.117	44.7	2.49E-25	4.85E-25	-0.29	RQ	1	9.36	45.6	-1.89	0.12	1.28	
SWIFT J0830.1+4154	0.126	44.7	4.70E-26	1.40E-25	-0.47	RQ	1	8.56	45.1	-1.56	0.4	1.8	
SWIFT J0832.5+3703	0.0923	44.5	8.21E-26	2.79E-25	-0.53	RQ	1	9.13	45.1	-2.12	0.26	1.87	
SWIFT J0839.6-1213	0.198	45.5	1.86E-23	2.31E-25	1.91	RL	1	8.86	45.8	-1.21	0.5	2.03	
SWIFT J0918.5+1618	0.0295	43.8	6.10E-26	2.76E-24	-1.66	RQ	1	8.48	45.1	-1.5	0.05	2.08	
SWIFT J0947.7+0726	0.0864	44.6	7.64E-23	2.68E-25	2.45	RL	1	8.9	45	-1.96	0.39	1.77	
SWIFT J1031.9-1418	0.0851	44.8	1.43E-25	9.88E-25	-0.84	RQ	1	8.59	45.6	-1.1	0.15	1.86	
SWIFT J1052.8+1043	0.088	44.5	1.02E-25	3.24E-25	-0.5	RQ	2	9.04	45.1	-2.01	0.21	1.22	
SWIFT J1115.3+5423	0.0703	44.4	6.60E-26	2.65E-25	-0.6	RQ	2	8.58	44.8	-1.84	0.39	1.69	
SWIFT J1138.9+2529B	0.219	44.9	8.29E-26	1.21E-25	-0.16	RQ	2	8.87	45.6	-1.4	0.21	1.56	
SWIFT J1149.3+5307	0.0955	44.1	3.90E-26	6.19E-26	-0.2	RQ	1	8.61	44.5	-2.22	0.43	1.67	
SWIFT J1158.9+4234	0.0314	43.5	4.67E-25	8.05E-25	-0.24	RQ	2	8.6	44.6	-2.1	0.09	1.75	
SWIFT J1207.5+3355	0.079	44.2	4.90E-24	1.09E-25	1.65	RL	2	8.55	44.6	-2.08	0.44	1.5	
SWIFT J1211.3-3935	0.0609	44.3	2.40E-25	3.78E-25	-0.2	RQ	2	8.57	44.9	-1.81	0.25	1.67	
SWIFT J1222.4+7520	0.0703	44.3	9.21E-26	4.52E-25	-0.69	RQ	1	8.61	45.1	-1.64	0.18	2.05	
SWIFT J1238.9-2720	0.0242	44.3	7.36E-25	1.27E-24	-0.24	RQ	2	8.99	44.6	-2.53	0.5	1.53	
SWIFT J1300.1+1635	0.0798	44.3	1.14E-25	9.41E-25	-0.92	RQ	2	9.19	45.5	-1.78	0.06	1.41	
SWIFT J1309.2+1139	0.0252	44	2.50E-26	5.60E-25	-1.35	RQ	2	8.54	44.2	-2.4	0.53	1.21	
SWIFT J1315.8+4420	0.0355	43.7	1.43E-25	1.06E-24	-0.87	RQ	2	8.67	44.8	-1.94	0.08	1.99	

Table 3: Sample

References

Blackburn, J. K. 1995, in , 367

Pastor-Marazuela, I., Webb, N. A., Wojtowicz, D. T., & van Leeuwen, J. 2020, , 640, A124

Rosen, S. R., Webb, N. A., Watson, M. G., et al. 2016, Astronomy and Astrophysics, 590, A1

SOC, E. X.-N. 2018

Watson, M. G., Schröder, A. C., Fyfe, D., et al. 2009, Astronomy and Astrophysics, 493, 339