EXODUS

EPIC XMM-Newton Outburst Detector Ultimate System

Institute	IRAP, Toulouse, France
Author	Maitrayee Gupta (2022)
Contact	maitrayee.gupta@irap.omp.eu, natalie.webb@irap.omp.eu
Version	1.0, August 26, 2022
Source code	https://github.com/maitrayeegupta/EXOD

Abstract

The purpose of EXODUS is to detect faint rapid transients in XMM-Newton data, that may not have been detected automatically using the XMM-Newton pipeline. It works for all EPIC detectors and all read-out modes. The technique used in EXODUS is to break a long observation into short time windows and compare the number of counts in regions of the detector for each time window, to the average number of counts per region. EXODUS uses three sub-routines: the first one calculates the counts per region per window and determines the variability by comparing to the average rate for that region; the second one provides an image showing the level of variability on the whole detector; and a third, performs post-processing and collates the results.
This document provides both user and technical documentation for EXODUS.
If you use EXODUS for your research, please acknowledge it by citing Gupta et al (to be submitted).
EXODUS is built upon two projects: the original project "Variabilitectron", which was created by Damien Wojtowicz, and "EXOD" which was created by Pastor-Marazuela et al. (2020).

Contents

1 Introduction

This document contains the user guides for both EXOD by Pastor-Marazuela et al. (2020) and the EXODUS by Gupta et al. (in prep), combined for ease of readability.

XMM-Newton's automatic pipeline extracts time series and provides results from variability tests for detections with total counts > 100 (Webb et al. 2020). Fainter sources are not analysed for variability within an observation.
4XMM-DR12 [1] , the latest version of the catalogue, is the largest catalogue of X-ray sources detected using a single X-ray observatory to date, containing 12210 observations (Webb et al. 2020). 939270 detections have been made in these data, but some very faint sources that are only bright for a maximum of a few seconds may fail to be detected.

The main objective of EXODUS is to detect fast transients among XMM-Newton EPIC observations, covering timescales from less than a second to a few minutes. Sources variable on such timescales could include gamma-ray bursts at high redshift, type-I X-ray bursts in distant galaxies and possible X-ray counterpart to FRBs for example.

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-DR12/4XMM_DR12.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`
- `numpy`
- `matplotlib`
- `pandas`
- `astroquery`
- `astropy`
- `multiprocessing`
- `os`
- `scipy`
- `skimage`
- `csv`
- `pylab`
- `re`

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

3 The algorithm

3.1 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 "box-size" 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 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 good time interval (GTI) file which is produced in the data filtering step explained in section 5.1.
5. The good time ratio, defined as the time belonging to the GTI of the considered time window divided by the duration of the time window, is computed for each time window.

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

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

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 1.

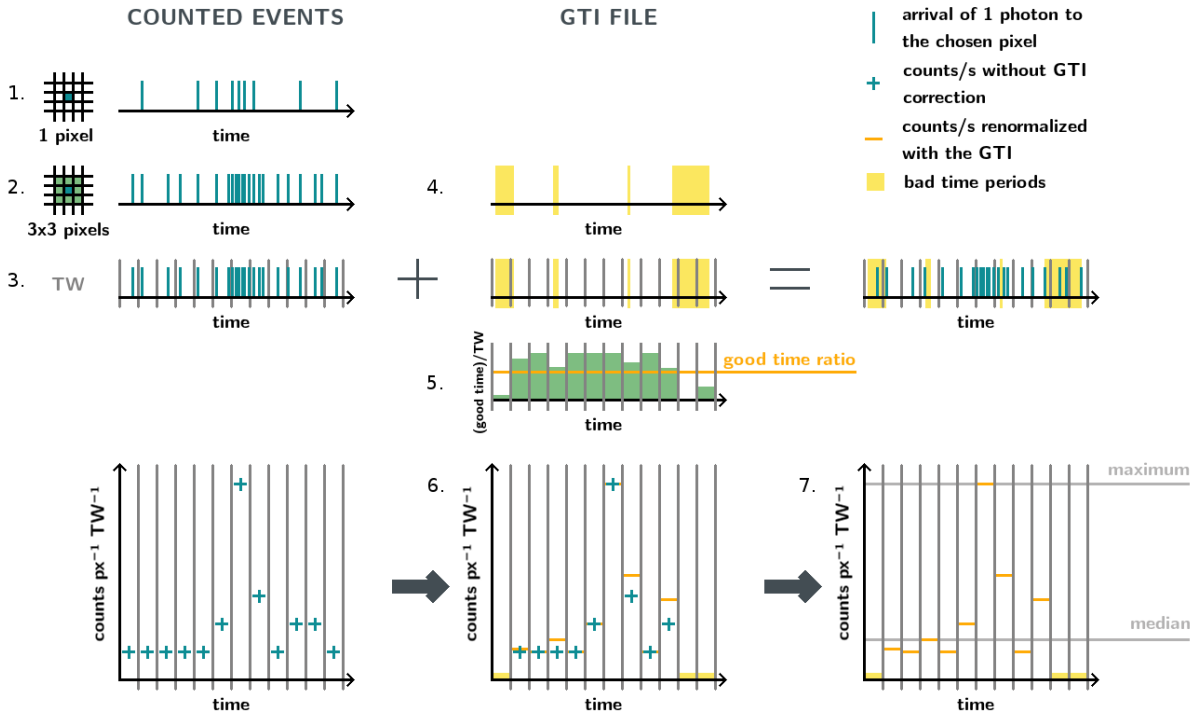

Figure 1: Diagram of the variability computation with the different stages of the algorithm.

3.2 Variable source 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.

A more details explanation of this algorithm can be found in Pastor-Marazuela et al. (2020).

4 Using EXODUS

The following section explains the use each of these three sub-programs of EXODUS, describes the parameters that can be varied by the user and explains the output
The source code can be found in the *scripts* folder in `https://github.com/maitrayeegupta/EXOD`. In order to run the program one additionally needs to have the XMM source catalog. All additional data such as XMM-Newton EPIC event lists as well as any additional tables can be generated using the code in this release.

4.1 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 EPIC event list.
`-obs`	OBSID of the observation.
`-out`	Path to the folder where the output files will be stored.

The optional parameters are the following:

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.

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 (5.7 ms for PN and 0.3s for MOS) 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 ds9.
--novar	-nv	boolean	False	Skip variability computation if already done for the same time window.

4.2 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.
  - `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 ordered by arrival time from the 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 within the sliding box at each position.
  - `__add_to_detected_areas` Function summing the variability values within the sliding box at each position.
  - `variable_areas_detection` Function detecting variable areas and writing them 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.

4.3 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 XMM 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 already detected using the automatic XMM-Newton pipeline but that have not been flagged as variable.

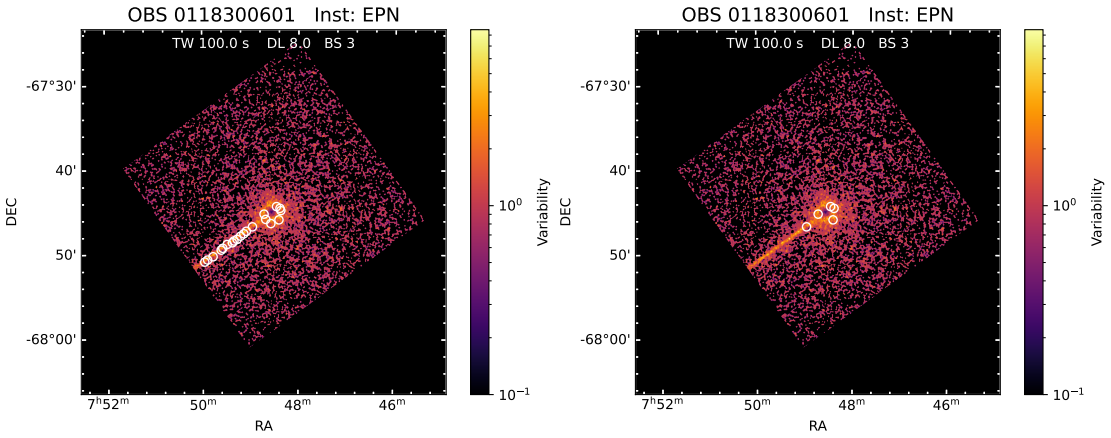

Figure 2: Removal of the readout streak. The colourbar represents the variability ($\mathcal{V}$) described in section 3.1

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

- `generate_bright_sources_file.py`
  We define bright sources to have an EPIC band 8 flux greater than 10^{-11} $ergcm^{-2}s^{-1}$. The following code generates a list of bright sources from the XMM catalogue. The generated bright_sources.csv table is used by remove_bright_sources.py.

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

- `remove_bright_sources.py`
  We eliminate any sources which lie within 1 arcmin of the bright source. The list of bright sources from the XMM catalogue can be found in bright_sources.csv table. The

  - `path` Path where bright_sources.csv 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 eliminate false matches (if the distance between the match and X ray source is very large).

- `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
```

Table 2: Files generated by EXODUS

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 variable source positions compatible with ds9.	
`log.txt`	Provides a log of the output generated whilst the script was running.	
`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.	

4.4 Output files

Figure 3 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 white circle the position of the detected variable source.

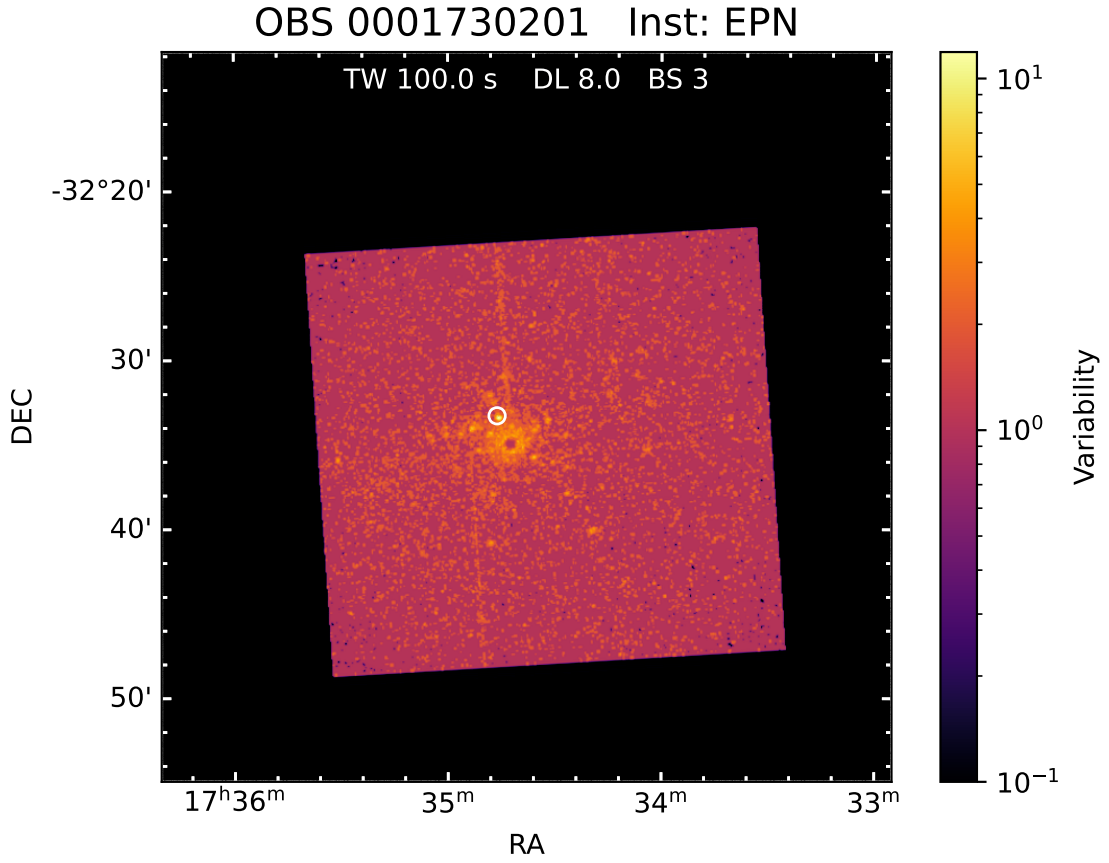

Figure 3: `sources.pdf` of the observation 0001730201 computed with the default values.

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 EXODUS 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) (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 for PN and PATTERN>12 for MOS. 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} for PN and 0.4 counts s^{-1} for MOS, 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.

Table 3: lightcurve.sh parameters

Parameter		Default	Function
`--source-id`	`-id`	1	Source identifier, given in the first column of `detected_variable_sources.csv`.
`--folder`	`-f`	/mnt/data/Maitrayee/data	Path to the folder containing the observations.
`--scripts`	`-s`	/mnt/data/Maitrayee/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>_<BS>_<GTR>`.

- `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.

- `run_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:

  This script calls the following scripts: `download_observation.sh`, `filtering.sh`, `lightcurve.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).

Table 4: run_exodus.sh parameters

Parameter	Default	Function
`-o`	No default	Path to the file containing the obsids.
`-l`	false	generate lightcurves for variable sources.
`-i`	false	run in iterative detection level mode.

- `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. The output file is called `exodus_results.csv`. An example output is shown in Table 5.

The columns in the table are described below:

- OBS_ID - XMM OBSID
- CLASS - Either Triple (**T**) (same source found in PN, M1 and M2), Double (**D**) (same source matched between PN and M1, or PN or M2, or M1 and M2), or Single (**S**).
- PN_REGION_NUMBER - A unique ID assigned to a variable source in the PN field of view.
- PN_RA - Right ascension of the variable source in the PN field of view.
- PN_DEC - Declination of the variable source in the PN field of view.
- M1_REGION_NUMBER - A unique ID assigned to a variables ource in the M1 field of view.
- M1_RA - Right ascension of the variable source in the M1 field of view.
- M1_DEC - Declination of the variable source in the M1 field of view.
- M2_REGION_NUMBER - A unique ID assigned to a variables ource in the M2 field of view.
- M2_RA - Right ascension of the variable source in the M2 field of view.
- M2_DEC - Declination of the variable source in the M2 field of view.
- PN_M1_SEP - Distance between the the PN and M1 sources in arcseconds.
- PN_M2_SEP- Distance between the the PN and M2 sources in arcseconds.
- M1_M2_SEP- Distance between the the M1 and M2 sources in arcseconds.
- EXOD_RA - Centroid of the PN, M1 and M2 RA positions.
- EXOD_DEC - Centroid of the PN, M1 and M2 DEC positions.
- XMM_RA - Right ascension of the closest XMM source in the XMM catalogue (within 30 arcsec).
- XMM_DEC- Declination of the closest XMM source in the XMM catalogue (within 30 arcsec).
- EXOD_XMM_SEP - Separation between the source from the XMM catalogue and the EXODUS position.

– XMM_SRCID - Source ID of the source from the XMM catalogue.
– XMM_EP8_FLUX - The EPIC combined band 8 flux from the XMM catalogue.
– XMM_PIPELINE_VARIABLE - One of three categories, Yes_and_variable or Yes_and_not_varia or No_match
– SIMBAD_match_sep_1 - Distance between the variable source and closest association to the variable source within the SIMBAD database.
– SIMBAD_match_name_1 - Closest association to the variable source within the SIMBAD database (arcsec).
– SIMBAD_match_obj_type_1 - SIMBAD object type.
– SIMBAD_match_sep_2 - Distance between the variable source and second closest association to the variable source within the SIMBAD database (arcsec).
– SIMBAD_match_name_2 - Second closest association to the variable source within the SIMBAD database.
– SIMBAD_match_obj_type_2 - SIMBAD object type.
– SIMBAD_match_sep_3 - Distance between the variable source and third closest association to the variable source within the SIMBAD database (arcsec).
– SIMBAD_match_name_3 - Third closest association to the variable source within the SIMBAD database.
– SIMBAD_match_obj_type_3 - SIMBAD object type.
– SIMBAD_CLASS - one of five classes viz. CompactObject, Star, AGN, Galaxy and Unspecified.

6 Tutorial

In this section, we give an example of how to use EXODUS, 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 EXODUS 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. The environment needs to be setup to point to the right path, below we show the ones we used on our machine and need to be modified accordingly.

```
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
```

6.2 One observation

We are first going to set some useful variables that we will use on 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 the bandwidth of your connection.

```
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
```

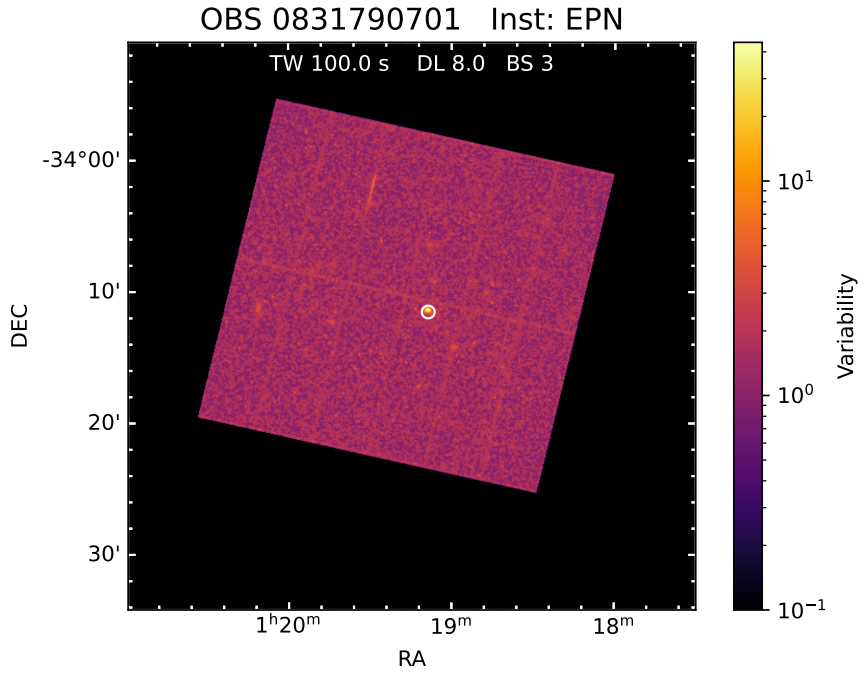

Figure 4: `sources.pdf` of the observation 0652250701 computed with the values of the tutorial. One variable source is detected at the position of the white circle.

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.

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

for example:
bash $SCRIPTS/lightcurve.sh $FOLDER $SCRIPTS -i PN -id 1 -dl 8 -tw 100 -gtr 1.0 -bs 3 -out I
```

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.

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. 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

bash run_exodus.sh -o obs_ids.txt -l true -i true
```

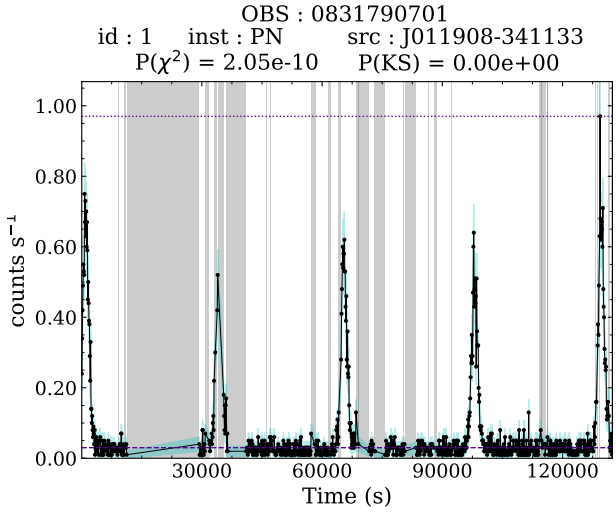

Figure 5: Lightcurve of the detected source in observation 0831790701 plotted with `lcurve.py`. Also shown are the computed χ^2 and Kolmogorov-Smirnov probabilities of constancy, P(χ^2) and P(KS) respectively.

The above script would run EXODUS on all three OBSIDs listed above. There would be 3 directories created in `$FOLDER` with the OBSID as the name and all the correspoding output files get deposited in the directories. In this example we are also running with lightcurve generation enabled, so the lightcurves for each of the detected variable sources are also produced.

The results of one of the OBSIDs 0831790701 is shown in Fig. 6, and the lightcurves of the detected sources are shown in Fig. 7.

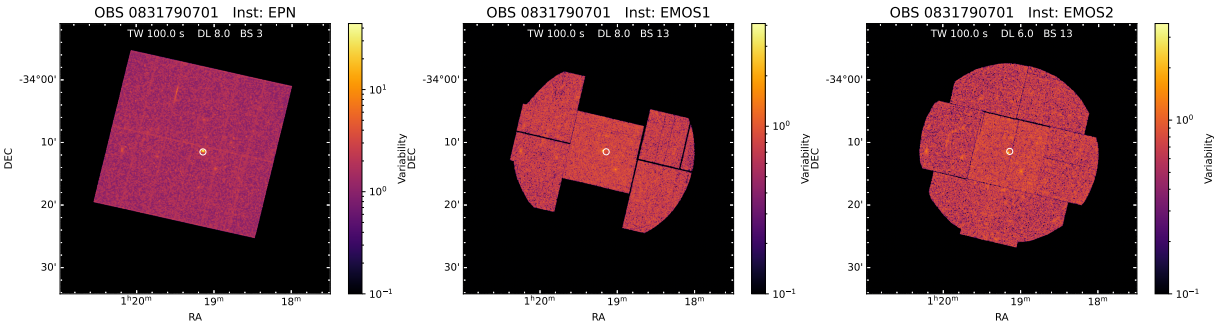

Figure 6: The PN, MOS1 and MOS2 `sources.pdf` of the observation 0831790701 computed with the values of the tutorial. One variable source is detected at the position of the white circle. As we can observe the same source has been identified using all three EPIC cameras.

Once all the objects are processed, the post processing scripts are run and produce an output file `exodus_results.csv` which is a detailed table containing collated results of the run. This file for the above run can be found in the examples directory on github.

A few selected columns from the example output table `exodus_results.csv` is shown in Table 5 (the complete table can be found in the can be found in the examples directory on github).

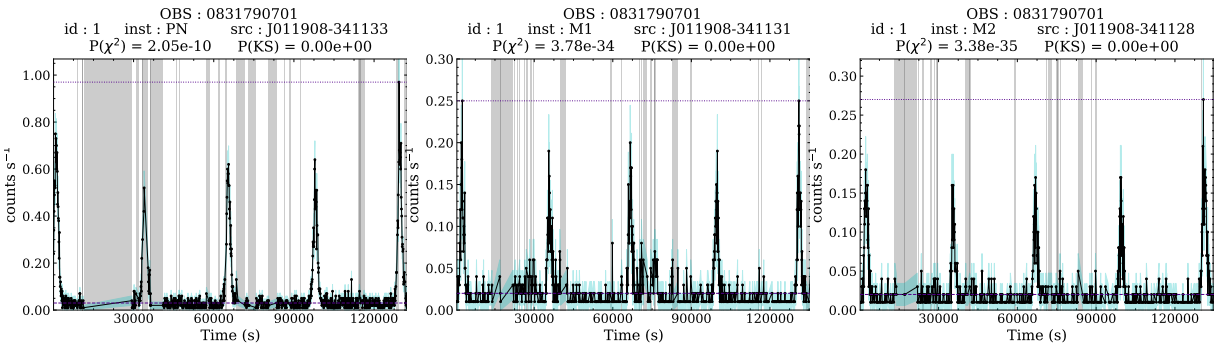

Figure 7: The light curves of the sources detected in Fig. 6. Also shown are the computed χ^2 and Kolmogorov-Smirnov probabilities of constancy, P(χ^2) and P(KS) respectively.

OBS_ID	class	PN_REG_NO.	M1_REG_NO.	M2_REG_NO.	EXOD_RA	EXOD_DEC	XMM_PPL_VAR	SIMBAD_sep	SIMBAD_name	SIMBAD_CLASS
XMM 0831790701	T	1	1	1	19.78	-34.19	Yes_and_variable	1.45	2MASX J01190869-3411305	AGN
XMM 0001730201	D	-	1	1	263.67	-32.58	Yes_and_variable	3.29	HD 159176	Star
XMM 0001730201	S	1	-	-	263.69	-32.55	No_match	5.003	2MASS J17344581-3233180	Star
XMM 0002970201	D	2	1	-	35.79	42.99	Yes_and_not_variable	8.73	UGC 1841	AGN
XMM 0002970201	D	-	2	1	35.66	43.03	Yes_and_not_variable	0.89	NAME 3C66A Cluster	Galaxy
XMM 0002970201	S	1	-	-	35.79	42.92	Yes_and_not_variable	9.58	TYC 2839-34-1	Star
XMM 0002970201	S	3	-	-	35.70	42.86	Yes_and_not_variable	NA	NA	NA
XMM 0002970201	S	-	1	-	35.79	42.99	Yes_and_not_variable	2.34	UGC 1841	AGN

Table 5: Sample

Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n°101004168, the XMM2ATHENA project.

References

Blackburn, J. K. 1995, in , 367. http://adsabs.harvard.edu/abs/1995ASPC...77..367B

Pastor-Marazuela, I., Webb, N. A., Wojtowicz, D. T., & van Leeuwen, J. 2020, , 640, A124, doi: 10.1051/0004-6361/201936869

SOC, E. X.-N. 2018. https://xmm-tools.cosmos.esa.int/external/xmm_user_support/documentation/sas_usg/USG/

Webb, N. A., Coriat, M., Traulsen, I., et al. 2020, , 641, A136, doi: 10.1051/0004-6361/201937353