STONKS Documentation

Erwan Quintin*, Natalie Webb, Laurent Michel

May 2025

This is the documentation for the *Search for Transient Object in New observations using Known Sources* (STONKS) software, a Python script used to compare new XMM-Newton observations to pre-combined X-ray archives, in order to assess, characterize and alert about any transient behaviour from a source in the new observation's field of view.

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

1 Requirements

There are a handful of requirements to fulfill in order to run STONKS yourself.

- Python >3.6
- Most of the used pre-existing Python packages are standard in astrophysics:
  - Numpy
  - Astropy
  - Matplotlib
  - os
  - scipy
  - Tqdm (used for progress bars)
  - webbrowser (used for clickable lightcurve)

  We also use the RapidXMM package, which is contained in the 'api.py' script.
- The archival catalogues data, which is all contained in a "Data" folder at the same level as the Python scripts. Currently no permanent hosting solution for this file has been found.

2 Code structure & Workflow

The STONKS scripts, available on the GitHub, consist of several different scripts, the core of the pipeline consisting of the two following codes.

2.1 LoadSpecificMasterSources.py

This first script is the backbone of the STONKS package, and is loaded by the two other scripts detailed afterwards. Its role is to load the archival X-ray catalog in the form of custom-made Objects that correspond to two classes, `Source` and `MasterSource`. The `Source` objects correspond to a single source from a given X-ray catalog, which can mean a single or several detections, each with a range of properties. These `Source` objects were then matched, catalog by catalog, in order to assess their long-term evolutions. The precise matching process is described in Quintin et al., 2024. Once these associations are performed, `Source` objects that correspond to a single, astrophysical object are regrouped under a single `MasterSource` object; if a `Source` has no match in other catalogs, it will be attributed its own `MasterSource` object. This method results in about 1 million `MasterSources`, loaded in a dictionary that has the ID of each `MasterSource` as key and the object itself as value.

*erwan.quintin.astro@gmail.com

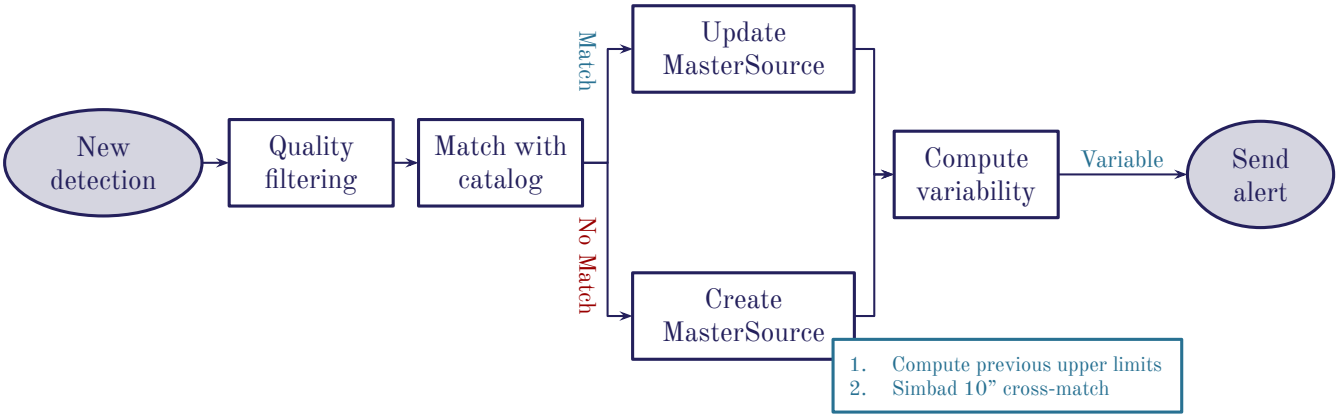

Figure 1: Workflow of the STONKS pipeline.

2.2 STONKS_PreComputed_Position_alert.py

This script is the main content of the STONKS software, and is used for the aforementioned use case of a comparison between a new XMM-Newton observation and archival data.

This module uses the archival X-ray catalog loaded through LoadSpecificMasterSource.py, compares the sources around the target position to the new detection at a given flux level, and returns the long-term lightcurve of the associated MasterSource object and an alert if it is variable.

The precise workflow of the script is described in Figure 1. For each new detection, we create a new Source object with the corresponding properties; it is then matched to the known catalog, using the 3σ position error. Any unique match is directly accepted; any ambiguous matching is resolved using a Bayesian framework. The possibles cases for multiple matches and their solution are as follow:

1. If one of the multiple MasterSource candidates contains an XMM-Newton source, then this MasterSource is accepted;
2. Else, we compute the match score for each match, given by formula 1.
   - If the highest score is at least 5 times above the next highest score, we select the best MasterSource candidate, and consider this a successful match;
   - Else, the matches are considered ambiguous.

The score of each match is given by the formula

$$B = \frac{2}{\sigma_1^2 + \sigma_2^2} \exp\left(-\frac{\psi^2}{2(\sigma_1^2 + \sigma_2^2)}\right) \tag{1}$$

where σ_i is the 1σ positional error of each source and ψ is the angular distance between them; see Quintin et al. 2024, for more details about this method.

In the case of an ambiguous match, the script stops here, as we cannot be certain about any level of variability of the putative MasterSource. In the case of a successful match, we update the previously known MasterSource with this information. In the case of an absolute lack of match, we create a new MasterSource containing only the new detection; we also compute the pre-existing upper limits at this position if they exist, through the RapidXMM framework, and retrieve the Optical/UV counterpart in the OM - UVOT catalogs if you decided to load them.

In both cases, we then compute the variability of the MasterSource, and trigger a transient alert in the case this flux variability is above a threshold of 5.

3 Usage

The standard use case for STONKS is through the website (https://xcatdb.unistra.fr/stonks/). There, you need to upload either an EPIC source list (*EP*OBSMLI*.FIT) or a tar ball containing both the EPIC source list and its corresponding EPIC image. It will return to the user an output tar ball containing all the alerts in the form of PDF and JSON files. This website calls the two previously mentioned scripts, through a Flask app.

If you want to run it yourself, you need to first get the underlying data by contacting us. Then, put the input tar ball in the Data/MasterSources folder, and edit the STONKS/python/launcher.py input names to use your file. You should then be able to run launcher.py, the output being stored in a session file in the Sessions folder.