LYR

Python Likelihood Ratio algorithm v1.0

A. Peca

June 19, 2019

Contents

1 Scientific background

1.1 Motivation

When you make science using surveys, the identification of the sources' counterparts in different wavelengths is not trivial.

1.2 The method

Source matching between different catalogs can be computed using the **Likelihood-Ratio** (LR) technique (e.g. Zamorani et al. 1999; Ciliegi et al. 2003; Brusa et al. 2007; Luo et al. 2010). The LR procedure, described firstly by Sutherland and Saunders 1992, which has been shown to be valid also in case of multiple candidates:

$$LR = \frac{q(m)f(r)}{n(m)} \tag{1}$$

where $q(m)$ is the expected magnitude distribution of counterparts, $f(r)$ is the probability distribution function of of the angular separation[1], and $n(m)$ is the surface density of background objects with magnitude m.

We assume that $f(r)$ follows a Gaussian distribution (e.g. Zamorani et al. 1999):

$$f(r) = \frac{1}{2\pi\sigma^2}\exp\left(\frac{-r^2}{2\sigma^2}\right) \tag{2}$$

with the standard deviation $\sigma = \sqrt{\sigma_o^2 + \sigma_X^2}$, where σ_o and σ_X are the 1σ positional errors of the ONIR (Optical and Near-Infrared) and X-ray sources, respectively. **LYR** can compute the 1σ positional errors in different ways, as shown in the parameters file described in (3.2). It is possibile to use positional errors for every single source, a static error for all sources (also $\sigma = 0$) or it is possibile to estimate the mean σ error from catalogs and use it for all the sample. This is valid for both the input/output catalogs.

The magnitude-dependent surface density of the background sources, $n(m)$

$$LR = \frac{q(m)f(r)}{n(m)}f(\Delta F) \tag{3}$$

[1] $r = \sqrt{\Delta RA^2 + \Delta DEC^2}$, where ΔRA and ΔDEC are the positional offsets of the candidate counterparts sources position.

2 Installation and requirements

LYR code is written in a **Python 3.x** environment. If you need to use Python 2.x you may change little code strings and you may check for specific packages.

2.1 Required packages

LYR code is written using the following packages:

- `numpy` v1.15;
- `scipy` v1.1;
- `sklearn` v0.20;
- `matplotlib` v3.0;
- `os`;
- `time`;
- `mpldatacursors` (https://github.com/joferkington/mpldatacursor/).

The last package is used by **LYR** only for graphic interaction. If you don't want to use it, you simply put the cursors variable to false (see section below).

3 Settings

3.1 Input/Output catalogs

LYR code needs two catalogs in which it calculates the likelihood ratio with (1) or (3). It doesn't matter in which wavelengths the catalogs were observed, even if the two catalogs are in the same band or they are simulated catalogs.

The different values of the two catalogs must be specified in the `LYR_input_parameters.py` file with the numpy module `genfromtxt`. The user must specify the paths' catalogs and the column number for each quantity, like:

```
file_input = "/folder1/folder2/input_cat.txt"
data_input = np.genfromtxt(file_input)
ID_input = np.array(data_input[:,0])
ra_input = np.array(data_input[:,1])
dec_input = np.array(data_input[:,2])
```

In this case **LYR** loads the `file_input` catalogs, where ID, RA and DEC of each sources are columns number 0, 1 and 2, respectively.

3.2 Parameters

LYR needs some settings parameters, which are located in the file `LYR_input_parameters.py`:

1. **Searching regions**:

   - **r_fr**: searching radius (in arcseconds) centered on the output sources in which the code creates the *f(r)* distribution in (3);
   - **r_in_tm**: searching radius (in arcseconds) for counterparts distribution around output sources;
   - **r_lr**: searching radius (in arcseconds) in which he code calculates LR (1);
   - **r_min_nm** and **r_max_nm**: inner and outer radius (in arcseconds) of the searching annulus in which the code searches for the background distribution $n(m)$ in (3).

2. **Paths and output file names**:

   - **path_LR**: path of **LYR** folder, in string format;

- **path_output**: path of output files in string format. If the folder doesn't exist yet, **LYR** will create it;
- **path_images**: path of output images in string format. If the folder doesn't exist yet, **LYR** will create it.
- **filename_LR**: filename of LR, reliability and all parameters of (3);
- **filename_outinfo**: txt file with some output information (Q value, non-matched input/output sources, computational time).

3. **catalogs**:

   - **sigma_out_finder**: string format, is used to spicify σ_{output} in the $f(r)$ term of 3. `'all'` means using every sigle source associated errors, `'1sigma'` means that **LYR** will calculate the mean error of the whole output catalogs;
   - **input_cat_type**: way to use σ_{input} in the $f(r)$ term of 3. Integer format, `0` means that no positional errors will be used ($\sigma_{input} = 0$), `1` is for usual every single source errors and with `2` the code calculates a mean sigma distribution (ΔRA and ΔDEC) between two input catalogs previously matched;
   - **noxy**: integer or float, is the value for non detection in the input/output catalogs (default -99.);
   - **nomag**: the same of **noxy** but for input magnitudes or fluxes;
   - **delta_f2**: boolean `True` or `False`, for using (3) or (1), respectively.
   - **delta_f2_fluxerr**: integer value, if 0 a static flux error is applied on every input source, if 1 the errors of every input source is used. Note that input flux errors are used only if `delta_f2` is set to `True`.

4. **Other parameters**:

   - **distrib_bins**: number of bins of all (3) distributions, must be an integer;
   - **plus_erf**: integer or float, if you want to multiply output errors with a static factor;
   - **iwp**: boolean value, if you want to view and interact with python output images at the end of the code;

- **cursors**: boolean value, if you want to interactive plot information about every single point in the LR vs Reliability final plot. It uses the `mpldatacursor` package;
- **save_images**: boolean value. Set `True` if you want **LYR** saves all images;
- **save_output**: boolean value. Set `True` if you want **LYR** saves two output txt files (**filename_LR** and **filename_outinfo**). Note that if you chose `False`, the *LR* vs reliability plot will neither be plotted and created;
- **add_title** & **add_string**: strings that are added on images plots and images names, respectively. If you don't want to add them, you simple set an empty string.

3.3 Run LYR:

When you have specified all parameters you simply digit:

```
python  LYR_main.py
```

4 Results in J1030

4.1 Positional errors

The 1σ positional errors of the X-ray sources were estimated with a semi-theoretical approach, following Puccetti et al. 2009, as explained in detail in (A).

A X-ray errors strategy

It is well known that X-ray source centroids errors calculated with *wavdetect* are too small and compared to real uncertainties of the detections (e.g., Luo et al. 2010; Puccetti et al. 2009; Hickox and Markevitch 2006). Also using *Acis Extract*, that should give more precise X-ray centroid, the uncertainties seem to be underestimated (Luo et al. 2010).

After several tests, we adopted a semi-theoretical approach. We used the solution proposed by Puccetti et al. 2009:

$$Pos_{Error} = \frac{PSF_{Radius}}{\sqrt{C_S}} \tag{4}$$

where C_S are the net, background subtracted, counts extracted with *wavdetect* or *Acis Extract*, and PSF_{Radius} is evaluated with the estimate for the 90% encircled energy radius (at $E = 1.5$ keV) given by Hickox and Markevitch 2006:

$$r_{90} \simeq 1'' + 10''(\theta/10')^2 \tag{5}$$

Having the net counts of the catalogs's sources, it is possibile to calculate positional errors (4) using the file `LYR_Xposerr.py`, which needs:

- **ra_center** and **dec_center**: the position of the center of the field within the catalogs is made, in degrees;

- file name and column numbers of the catalogs, as described above.

It creates a new folder named **new_err** with a new file **new_Xerr.txt** with new errors estimation. For launching the code you simply digit:

```
python  LYR_Xposerr.py
```

The only Python packages required are *numpy* and *os*.

References

[1] M. Brusa et al. "The XMM-Newton Wide-Field Survey in the COSMOS Field. III. Optical Identification and Multiwavelength Properties of a Large Sample of X-Ray-Selected Sources". In: *APJS* 172 (Sept. 2007), pp. 353–367. DOI: 10.1086/516575. eprint: astro-ph/0612358.

[2] P. Ciliegi et al. "A deep VLA survey at 6 cm in the Lockman Hole". In: *AAP* 398 (Feb. 2003), pp. 901–918. DOI: 10.1051/0004-6361:20021721. eprint: astro-ph/0211625.

[3] R. C. Hickox and M. Markevitch. "Absolute Measurement of the Unresolved Cosmic X-Ray Background in the 0.5-8 keV Band with Chandra". In: *APJ* 645 (July 2006), pp. 95–114. DOI: 10.1086/504070. eprint: astro-ph/0512542.

[4] B. Luo et al. "Identifications and Photometric Redshifts of the 2 Ms Chandra Deep Field-South Sources". In: *APJS* 187 (Apr. 2010), pp. 560–580. DOI: 10.1088/0067-0049/187/2/560. arXiv: 1002.3154.

[5] S. Puccetti et al. "The Chandra Survey of the COSMOS Field. II. Source Detection and Photometry". In: *APJS* 185 (Dec. 2009), pp. 586–601. DOI: 10.1088/0067-0049/185/2/586. arXiv: 0910.2617 [astro-ph.IM].

[6] W. Sutherland and W. Saunders. "On the likelihood ratio for source identification". In: *MNRAS* 259 (Dec. 1992), pp. 413–420. DOI: 10.1093/mnras/259.3.413.

[7] G. Zamorani et al. "The ROSAT deep survey. V. X-ray sources and optical identifications in the Marano field". In: *AAP* 346 (June 1999), pp. 731–752. eprint: astro-ph/9904335.