The Swift Reduction Package - GW - Users' Manual

*by Stefano Covino, 07 Jan 2016, v. 0**.2.0.***

Background
The Swift Reduction Package (hereafter SRP) is a packet of tools supposed to make everyday astronomer's life easier.
For any specific comment the main documentation for **SRP** is the reference source. Here we refer to a sub-package, **SRPAstro.GW**, devoted to the management of big photometric catalogues with the goal to identify transient sources. The scenario is mainly that related to the search for electromagnetic counterparts of gravitational wave events, but of course other applications are possible. The package should run with any **python 3.x** version.

Installation
If you are just updating **SRPAstro.GW** you just need to download the package from the PyPI archive with:

pip install -U SRPAstro.GW https://www.dropbox.com/s/6egaktz17zgkr38/SRPAstro.GW-0.1.0.tar.gz?dl=0 (possibly with superuser permissions)

If you, instead, are installing **SRPAstro.GW** for the first time or maybe you are upgrading to a new **Python** release, it is likely you need to install many different libraries **SRP** and the related sub-packges relies on.

In principle the command:

pip install -U SRPAstro.GW https://www.dropbox.com/s/6egaktz17zgkr38/SRPAstro.GW-0.1.0.tar.gz?dl=0

should again do the job. You might also consider to install the package in a virtual python environment if you do not want to interfere with the system python installation.

However, some of the required libraries can (will) require more concerned actions to allow their installation. In essentially all cases, browsing the web you can quickly find the solution to any problem.

An alternative and strongly advised procedure is to install one of the available open-source self-contained scientific python installations as the Anaconda distribution. Most of the required libraries would then available with no further efforts and **SRP** is installed smoothly (the Ureka project also deserves consideration).

Command description and demo session:

As a general rule, **SRPAstro.GW** works comparing photometry obtained in two (or more epochs) and also getting information about publicly available catalogs.

- **SRPGWAdaptSelect**

  This command allows one to apply an adaptive filter to select variable objects basing on the photometry available for two epochs. At present the only algorithm implemented is just the possibility to select objects varying more than n*σ beyond the standard deviation of the distribution down by common objects in two photometric catalogues.

- **SRPGWAnalyis**

  This command allows one to derive aperture photometry for sources in the matched catalogues (three tools are available) or to draw simple stamps of the sources for an easy visual check. It requires that the original **FITS** files are available.

- **SRPGWCalc**

  The catalogs are converted to tables and it is possible to compute operations on the table columns with this command.

- **SRPGWFITSStamp**

  Simple tool to generate FITS stamps centered at a given position.

- **SRPGWImportCats**

  This command converts photometric catalogues produced by various tools to tables to be managed by the various commands of the package.

- **SRPGWImportParSet**

  In order to know which column refers to what one might need to produce a file listing the columns to be imported and their name. This command generates a “typical” file to be modified basing on specific need. This is not always required since these pieces of information can often be available as headers of the photometric files.

- **SRPGWMatch**

  This command takes two different catalogues in input and generates three outputs, a list of “matched" objects, a list of “disappeared” objects and a list of the “appeared” objects. It requires that the entry catalogues have sky coordinate information.

- **SRPGWQuery**

  This command allows one to add information to the photometric catalogues. At present they are date (MJD from the original **FITS** files) and the possible matching entries in the Initial GAIA catalogue and in SIMBAD. The two last queries require, of course, an internet connection.

- **SRPGWSelect**

  This command allows one to apply selections on the input photometric tables.

- **SRPGWStat**

  With this command one can derive some simple statistics (mode, mean, median, etc.) on columns of the input tables.

- **SRPGWTabExtract**

This commands performs different tasks always related to the characterization of sources that are considered of interest after that the various possible selections and analyses are applied.

- **SRPGWTabPlot**
  This is a simple plotting tool for columns in the tables used for the analyses.
- **SRPGWTabSearch**
  This command allows one to search for objects with given coordinates in any table.
- **SRPGWTabViewer**
  This command allows one to visualize the input tables or convert them to HTML.
- **SRPGWVersion**
  This command is just to know which is the version of current **SRPAstro.GW** installation.

However, rather than a long and detailed description of what each commands actually can do, an example of a typical analysis session is probably more fruitful.

The first thing to do is to choose which data are to be imported. What we need is a simple text file with one SE*xtractor* (or any other tool, actually) catalog per line.
We assume to have these data accessible in our data tree. For instance, my case, data are at the path: */Volumes/data/GW_GRB*. Then, in the directories *2015-09-16* and *2015-09-24* we have, for instance, all the data for epochs 1 and 4 for our observations (what these observations actually are is not important now).
In this simulation we work only on data of pointing #50, and the information about the catalogs to be imported are in the two text files *catlistep1_50.dat* and *catlistep4_50.dat* that contain the two following lines, respectively:
/Volumes/data/GW_GRB/2015-09-16/G184098_VST_r_e01_p50.cat

and

/Volumes/data/GW_GRB/2015-09-24/G184098_VST_r_e04_p50.cat

Of course you can list all the catalog file you need. I have used standard UNIX tool to generate these files (**find**, **ls**, etc.) but one can use any tool he/she likes.

Then we actually import these data in tables suited to our needs. The commands are:

SRPGWImportCats -v -i catlistep1_50.dat -o Ep1_p50.dat
Processing file: /Volumes/data/GW_GRB/2015-09-16/G184098_VST_r_e01_p50.cat
Sorting table...
Source name...
Saving...
Table Ep1_p50.dat with 121232 entries saved.

SRPGWImportCats -v -i catlistep4_50.dat -o Ep4_p50.dat

Processing file: /Volumes/data/GW_GRB/2015-09-24/G184098_VST_r_e04_p50.cat
Sorting table...
Source name...
Saving...
Table Ep4_p50.dat with 47133 entries saved.

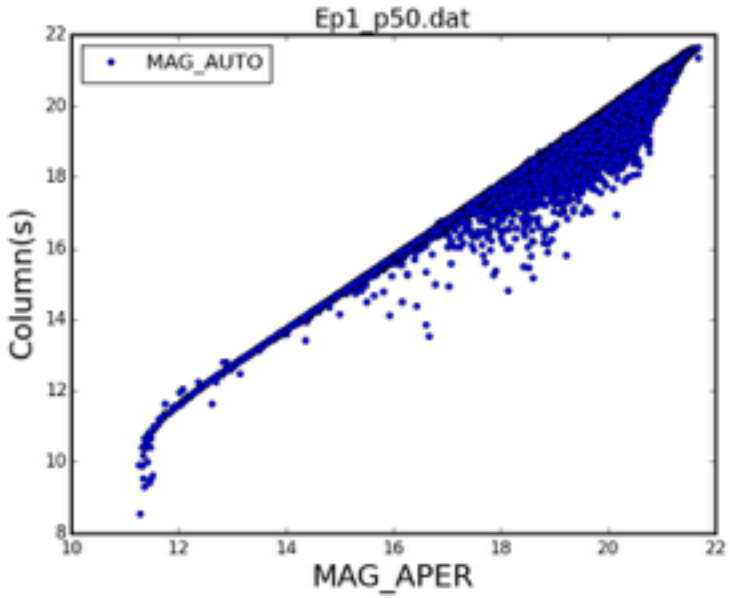
The generated tables are again regular text files, they follow the so-called "*enhanced csv*" format, that essentially means you can add to the text file meta-data as column names, etc. You can anyway read, modify, write them as any other text file.

Given that our tables are just text files you can also visualize and analyze their content with many different tools. If you are lazy you can just see the table within your browser with, for instance:

SRPGWTabViewer –v –i Ep1_p50.dat -b
Reading table Ep1_p50.dat

It is also possible to produce simple plots basing on the table columns with, for instance:

SRPGWTabPlot –v –i Ep1_p50.dat –p MAG_APER MAG_AUTO
Reading table Ep1_p50.dat



You can also perform selections on the table columns. For instance in order to keep short the execution time of this "demo" analysis, we can choose to work only on with the brightest objects:

SRPGWSelect -v -i Ep1_p50.dat -o Ep1_p50_mag19.dat -s "MAG_APER < 19"
Reading table Ep1_p50.dat
Selecting...
Table Ep1_p50_mag19.dat with 25808 entries saved.

SRPGWSelect -v -i Ep4_p50.dat -o Ep4_p50_mag19.dat -s "MAG_APER < 19"
Reading table Ep4_p50.dat
Selecting...
Table Ep4_p50_mag19.dat with 15435 entries saved.

Of course you can apply any selection you need even iteratively.
I mention in passing that in any case photometric catalogs obtained in two different nights cannot typically be equally deep and some selection at the faint end of the deepest is often required to make the comparison meaningful. The case we are considering is exactly one of these.

So, once you have starting catalogs you are happy of, we can match them with:

SRPGWMatch -v -f Ep1_p50_mag19.dat -s Ep4_p50_mag19.dat -r 0.5 -o Ep1-4_p50_mag19_matched.dat Ep1-4_p50_mag19_disapp.dat Ep1-4_p50_mag19_app.dat
Reading table Ep1_p50_mag19.dat
Reading table Ep4_p50_mag19.dat
Tables contain 25808 and 15435 entries, respectively.
Matching...
Saving results...
15088 matches found.
10720 entries disappeared.
347 entries appeared.

The chosen matching radius here is "*0.5 arcsec*". We have three different tables in output. The first is the table with "*matched*" objects, the second is the one with "*disappeared*" objects (entries in the first table and no more in the second) and the third is the one with "*appeared*" objects (entries in the second table with no correspondence in the first). The matching of big tables is not an easy task, and it can be rather time-consuming. So, in a real run with millions of objects, this could be a good time for a coffee (or one more, at least).

Now one of the problems of the specific dataset we are using (not necessarily of other datasets) is that there definitely are objects populating the *appeared/disappeared* lists simply because they fall in unlucky positions on the detectors, i.e. close to the borders or to the inter-chip gaps, etc. The VST pipeline provides information for at least partially evaluating these cases by means of the *weight* maps. Providing that the *FITS weight* maps are available at the same path of the catalogs you can add *weight* information to the generated tables with:

SRPGWQuery -v -i Ep1-4_p50_mag19_matched.dat -o Ep1-4_p50_mag19_matched.dat -w
Reading table Ep1-4_p50_mag19_matched.dat
Weight queries...
First epoch...
0% 100%
[##############################] | ETA[sec]: 0.000
Total time elapsed: 5.006 sec
Second epoch...
0% 100%
[##############################] | ETA[sec]: 0.000
Total time elapsed: 5.565 sec
Table Ep1-4_p50_mag19_matched.dat with 15088 entries saved.

And the same for the other two produced tables:

SRPGWQuery -v -i Ep1-4_p50_mag19_disapp.dat -o Ep1-4_p50_mag19_disapp.dat -w
Reading table Ep1-4_p50_mag19_disapp.dat
Weight queries...
First epoch...
0% 100%
[##############################] | ETA[sec]: 0.000
Total time elapsed: 2.893 sec
Second epoch...
0% 100%
[##############################] | ETA[sec]: 0.000
Total time elapsed: 3.469 sec
Table Ep1-4_p50_mag19_disapp.dat with 10720 entries saved.

SRPGWQuery -v -i Ep1-4_p50_mag19_app.dat -o Ep1-4_p50_mag19_app.dat -w
Reading table Ep1-4_p50_mag19_app.dat
Weight queries...
First epoch...
0% 100%
[##############################] | ETA[sec]: 0.000
Total time elapsed: 0.183 sec
Second epoch...
0% 100%
[##############################] | ETA[sec]: 0.000
Total time elapsed: 0.146 sec
Table Ep1-4_p50_mag19_app.dat with 347 entries saved.

Choosing the right *weight* values to select just properly exposed sources is not an easy task since the "*weight*" is a rather complex function of exposure, average, etc. With, for instance:

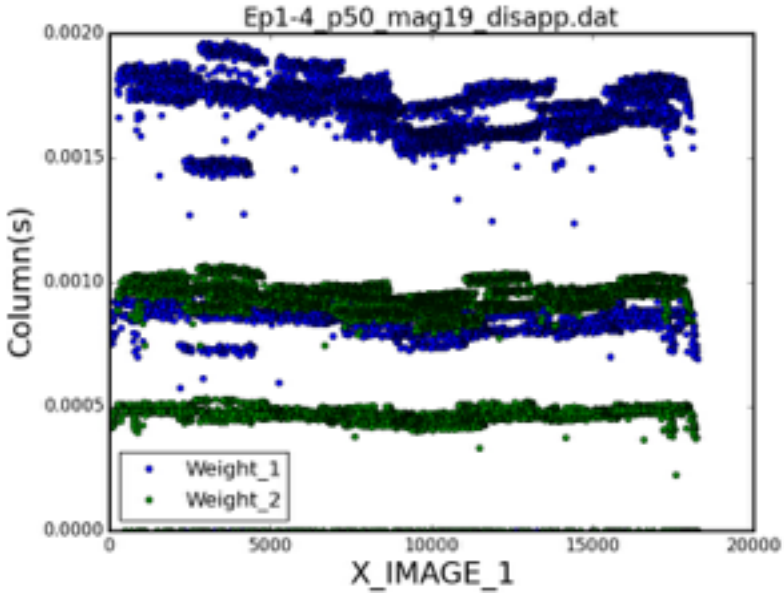
SRPGWStat -v -i Ep1-4_p50_mag19_disapp.dat -c Weight_1
Reading table Ep1-4_p50_mag19_disapp.dat
Mean: 0.00148552
Stddev: 0.000403082
Median: 0.00167227
Max: 0.00196533
Min: 0.0
Count: 10720

We see that the "*weights*" are distributed between *0* and *~0.002*. It is probably better to visually check the "*weight*" distribution to identify a suitable threshold:

SRPGWTabPlot -v -i Ep1-4_p50_mag19_disapp.dat -p X_IMAGE_1 Weight_1 Weight_2
Reading table Ep1-4_p50_mag19_disapp.dat

Weight_1 and *Weight_2* refer to the first and second epoch, respectively. And *X_IMAG_1* is the X position on the chip. A conservative limit for filtering badly exposed sources out is around 0.0003. Of course a more aggressive strategies are also possible. Then:

SRPGWSelect -v -i Ep1-4_p50_mag19_matched.dat -s "(Weight_1 >= 0.0003) & (Weight_2 >= 0.0003)" -o Ep1-4_p50_mag19_matched_w.dat
Reading table Ep1-4_p50_mag19_matched.dat
Selecting...
Table Ep1-4_p50_mag19_matched_w.dat with 15073 entries saved.



SRPGWSelect -v -i Ep1-4_p50_mag19_disapp.dat -s "(Weight_1 >= 0.0003) & (Weight_2 >= 0.0003)" -o Ep1-4_p50_mag19_disapp_w.dat
Reading table Ep1-4_p50_mag19_disapp.dat
Selecting...
Table Ep1-4_p50_mag19_disapp_w.dat with 10428 entries saved.

SRPGWSelect -v -i Ep1-4_p50_mag19_app.dat -s "(Weight_1 >= 0.0003) & (Weight_2 >= 0.0003)" -o Ep1-4_p50_mag19_app_w.dat
Reading table Ep1-4_p50_mag19_app.dat
Selecting...
Table Ep1-4_p50_mag19_app_w.dat with 145 entries saved.

Now, while on average the input Extractor magnitudes are fine, there are good reasons to re-compute them at least for having magnitudes for the objects not matched between the two catalogues, i.e. present in just one list. It is then possible to derive aperture photometry with the following commands. Again it is a time consuming task, and for large tables this is the right time for lunch or dinner!

SRPGWAnalysis -v -i Ep1-4_p50_mag19_disapp_w.dat -o Ep1-4_p50_mag19_disapp_wm.dat -a
Reading table Ep1-4_p50_mag19_matched_w.dat
Native photometric tool enabled.
Aperture photometry...
First epoch...
0% 100%
[##############################] | ETA[sec]: 0.000
Total time elapsed: 101.419 sec
Second epoch...
0% 100%
[##############################] | ETA[sec]: 0.000
Total time elapsed: 134.230 sec
Table Ep1-4_p50_mag19_matched_wm.dat with 15073 entries saved.

SRPGWAnalysis -v -i Ep1-4_p50_mag19_disapp_w.dat -o Ep1-4_p50_mag19_disapp_wm.dat -a
Reading table Ep1-4_p50_mag19_disapp_w.dat
Native photometric tool enabled.
Aperture photometry...
First epoch...
0% 100%
[##############################] | ETA[sec]: 0.000
Total time elapsed: 78.196 sec
Second epoch...
0% 100%
[##############################] | ETA[sec]: 0.000

Total time elapsed: 71.589 sec
Table Ep1-4_p50_mag19_disapp_wm.dat with 10428 entries saved.

SRPGWAnalysis -v -i Ep1-4_p50_mag19_app_w.dat -o Ep1-4_p50_mag19_app_-wm.dat -a
Reading table Ep1-4_p50_mag19_app_w.dat
Native photometric tool enabled.
Aperture photometry...
First epoch...
0% 100%
[##############################] | ETA[sec]: 0.000
Total time elapsed: 0.884 sec
Second epoch...
0% 100%
[##############################] | ETA[sec]: 0.000
Total time elapsed: 1.188 sec
Table Ep1-4_p50_mag19_app_wm.dat with 145 entries saved.

After that we can compute the magnitude differences between the two epochs:

SRPGWCalc -v -i Ep1-4_p50_mag19_matched_wm.dat -c ":DSRPMAG = SRPMAG_1 - SRPMAG_2" -o Ep1-4_p50_mag19_matched_wm.dat
Reading table Ep1-4_p50_mag19_matched_wm.dat
Table Ep1-4_p50_mag19_matched_wm.dat with 15073 entries saved.

***SRPMAG_1* and *SRPMAG_2* are the columns with the magnitudes computed in the previous steps.** The parser for possible instructions in the package is very simple, it is possible that in a future I will substitute it with something more serious, at present please do not forget to add blank spaces around the column names in order they are correctly processed. If you want instead create a non-existing column you need to indicate it with "*:*" as first character, e.g.: "*:MYCOL = MAG_APER + 1.0*". After that our table will include the new column named "*MYCOL*".

In order to compensate for likely differences in the photometric zero-points of the two epochs we can draw some statistics with the command:

SRPGWStat -v -i Ep1-4_p50_mag19_matched_wm.dat -c DSRPMAG
Reading table Ep1-4_p50_mag19_matched_wm.dat
Mean: -0.382263367883
Stddev: 0.0331372071284
Median: -0.3813835805
Max: 0.3643328507
Min: -1.1075366755

Count: 15073

And we realize that, taking the first epoch as reference, we have to add *-0.382* (the median) to the magnitudes of the second epoch. So:

SRPGWCalc -v -i Ep1-4_p50_mag19_matched_wm.dat -o Ep1-4_p50_mag19_matched_wm.dat -c "SRPMAG_2 = SRPMAG_2 - 0.382"
Reading table Ep1-4_p50_mag19_matched_wm.dat
Table Ep1-4_p50_mag19_matched_wm.dat with 15073 entries saved.

And, after that, repeat the command:

SRPGWCalc -v -i Ep1-4_p50_mag19_matched_wm.dat -c ":DSRPMAG = SRPMAG_1 - SRPMAG_2" -o Ep1-4_p50_mag19_matched_wm.dat
Reading table Ep1-4_p50_mag19_matched_wm.dat
Table Ep1-4_p50_mag19_matched_wm.dat with 15073 entries saved.

And finally we have correctly aligned instrumental magnitudes between the two epochs (check!).
Now the same for the *disappeared/appeared* objects:

SRPGWCalc -v -i Ep1-4_p50_mag19_disapp_wm.dat -o Ep1-4_p50_mag19_disapp_wm.dat -c "SRPMAG_2 = SRPMAG_2 - 0.382"
Reading table Ep1-4_p50_mag19_disapp_wm.dat
Table Ep1-4_p50_mag19_disapp_wm.dat with 10428 entries saved.

SRPGWCalc -v -i Ep1-4_p50_mag19_disapp_wm.dat -c ":DSRPMAG = SRPMAG_1 - SRPMAG_2" -o Ep1-4_p50_mag19_disapp_wm.dat
Reading table Ep1-4_p50_mag19_disapp_wm.dat
Table Ep1-4_p50_mag19_disapp_wm.dat with 10428 entries saved.

And:

SRPGWCalc -v -i Ep1-4_p50_mag19_app_wm.dat -o Ep1-4_p50_mag19_app_wm.dat -c "SRPMAG_1 = SRPMAG_1 - 0.382"
Reading table Ep1-4_p50_mag19_app_wm.dat
Table Ep1-4_p50_mag19_app_wm.dat with 145 entries saved.

SRPGWCalc -v -i Ep1-4_p50_mag19_app_wm.dat -c ":DSRPMAG = SRPMAG_1 - SRPMAG_2" -o Ep1-4_p50_mag19_app_wm.dat
Reading table Ep1-4_p50_mag19_app_wm.dat
Table Ep1-4_p50_mag19_app_wm.dat with 145 entries saved.

Please note that for the *appeared* sources the columns with "_1" refer to the second epoch, i.e. the only one with these sources reported.

Now, in principle, we could add information to the table entries as GAIA or SIMBAD cross-identification, etc. However, these operations are again quite time-consuming since they query catalogs through the web and they can be execute profitably later.

We can, instead, select now only the “interesting" objects among those listed in the tables. As a matter of fact, we are not really interested in finding regular variable objects. We are trying to locate a transient characterized by a large variation or that even disappears during our monitoring. So we start selecting only the matched objects with magnitude variation larger than, say, *0.5 mag*:

SRPGWSelect -v -i Ep1-4_p50_mag19_matched_wm.dat -o Ep1-4_p50_mag19_-matched_wms.dat -s "numpy.abs(DSRPMAG) >= 0.5"
Reading table Ep1-4_p50_mag19_matched_wm.dat
Selecting...
Table Ep1-4_p50_mag19_matched_wms.dat with 7 entries saved.

There is also the possibility to apply a filtering dependent on the magnitude of the source, with *SRPGWAdaptSelect* (try!).

For the disappeared/appeared lists it is enough to ask that the magnitude difference is lower than *-0.5 mag*.

SRPGWSelect -v -i Ep1-4_p50_mag19_disapp_wm.dat -o Ep1-4_p50_mag19_disapp_wms.dat -s "DSRPMAG <= -0.5"
Reading table Ep1-4_p50_mag19_disapp_wm.dat
Selecting...
Table Ep1-4_p50_mag19_disapp_wms.dat with 34 entries saved.

and

SRPGWSelect -v -i Ep1-4_p50_mag19_app_wm.dat -o Ep1-4_p50_mag19_app_-wms.dat -s "DSRPMAG <= -0.5"
Reading table Ep1-4_p50_mag19_app_wm.dat
Selecting...
Table Ep1-4_p50_mag19_app_wms.dat with 4 entries saved.

Although in this specific example we are left with only a few sources, in principle we can use more information for further sections. For instance, source already catalogued in the Initial GAIA catalogue of more generically in SIMBAD are very unlikely of interest for us (at least at these magnitudes). So we can collect these pieces of information: Since this step, in case of large tables, can be quite time-consuming this then one more opportunity for a cup of coffee!

SRPGWQuery -v -i Ep1-4_p50_mag19_matched_wms.dat -o Ep1-4_p50_mag19_-matched_wmsg.dat -g 0.5

Reading table Ep1-4_p50_mag19_matched_wms.dat
GAIA queries...
0% 100%
[#######] l ETA[sec]: 0.000
Total time elapsed: 0.800 sec
Table Ep1-4_p50_mag19_matched_wmsg.dat with 7 entries saved.

SRPGWQuery -v -i Ep1-4_p50_mag19_disapp_wms.dat -o Ep1-4_p50_mag19_disapp_wmsg.dat -g 0.5
Reading table Ep1-4_p50_mag19_disapp_wms.dat
GAIA queries...
0% 100%
[##############################] l ETA[sec]: 0.000
Total time elapsed: 4.489 sec
Table Ep1-4_p50_mag19_disapp_wmsg.dat with 34 entries saved.

SRPGWQuery -v -i Ep1-4_p50_mag19_app_wms.dat -o Ep1-4_p50_mag19_app_wmsg.dat -g 0.5
Reading table Ep1-4_p50_mag19_app_wms.dat
GAIA queries...
0% 100%
[####] l ETA[sec]: 0.000
Total time elapsed: 6.223 sec
Table Ep1-4_p50_mag19_app_wmsg.dat with 4 entries saved.

And the same for SIMBAD:

SRPGWQuery -v -i Ep1-4_p50_mag19_matched_wmsg.dat -o Ep1-4_p50_mag19_matched_wmsgs.dat -s 2.0
Reading table Ep1-4_p50_mag19_matched_wmsg.dat
Simbad queries...
0% 100%
[#######] l ETA[sec]: 0.000
Total time elapsed: 1.247 sec
Table Ep1-4_p50_mag19_matched_wmsgs.dat with 7 entries saved.

SRPGWQuery -v -i Ep1-4_p50_mag19_disapp_wmsg.dat -o Ep1-4_p50_mag19_disapp_wmsgs.dat -s 2.0
Reading table Ep1-4_p50_mag19_disapp_wmsg.dat
Simbad queries...
0% 100%
[##############################] l ETA[sec]: 0.000
Total time elapsed: 2.822 sec
Table Ep1-4_p50_mag19_disapp_wmsgs.dat with 34 entries saved.

SRPGWQuery -v -i Ep1-4_p50_mag19_app_wmsg.dat -o Ep1-4_p50_mag19_app_wmsgs.dat -s 2.0
Reading table Ep1-4_p50_mag19_app_wmsg.dat
Simbad queries...
0% 100%
[####] | ETA[sec]: 0.000
Total time elapsed: 0.461 sec
Table Ep1-4_p50_mag19_app_wmsgs.dat with 4 entries saved.

We can now select only the sources with not identification in the two queries catalogued:

SRPGWSelect -v -i Ep1-4_p50_mag19_matched_wmsgs.dat -o Ep1-4_p50_mag19_matched_wmsgs.dat   -s "(GAIA == 'No') & (Simbad == 'No')"
Reading table Ep1-4_p50_mag19_matched_wmsgs.dat
Selecting...
Table Ep1-4_p50_mag19_matched_wmsgs.dat with 1 entries saved.

RPGWSelect -v -i Ep1-4_p50_mag19_disapp_wmsgs.dat -o Ep1-4_p50_mag19_disapp_wmsgs.dat -s "(GAIA == 'No') & (Simbad == 'No')"
Reading table Ep1-4_p50_mag19_disapp_wmsgs.dat
Selecting...
Table Ep1-4_p50_mag19_disapp_wmsgs.dat with 3 entries saved.

SRPGWSelect -v -i Ep1-4_p50_mag19_app_wmsgs.dat -o Ep1-4_p50_mag19_app_wmsgs.dat -s "(GAIA == 'No') & (Simbad == 'No')"
Reading table Ep1-4_p50_mag19_app_wmsgs.dat
Selecting...
Table Ep1-4_p50_mag19_app_wmsgs.dat with 2 entries saved.

Of course one can apply as many selections as possible but eventually we reach a point where a visual check is required. It is thus possible, provided the original FITS files are are the same path of the catalogs, to generate small stamps for each entry in the tables with:

SRPGWAnalysis -v -i Ep1-4_p50_mag19_matched_wmsgs.dat -o Ep1-4_p50_mag19_matched_wmsgsp.dat -p
Reading table Ep1-4_p50_mag19_matched_wmsgs.dat
Drawing pictures...
First epoch...
0% 100%
[#] | ETA[sec]: 0.000
Total time elapsed: 0.791 sec
Second epoch...
0% 100%
[#] | ETA[sec]: 0.000

Total time elapsed: 0.500 sec
Table Ep1-4_p50_mag19_matched_wmsgsp.dat with 1 entries saved.

SRPGWAnalysis -v -i Ep1-4_p50_mag19_disapp_wmsgs.dat -o Ep1-4_p50_mag19_disapp_wmsgsp.dat -p
Reading table Ep1-4_p50_mag19_disapp_wmsgs.dat
Drawing pictures...
First epoch...
0% 100%
[###] | ETA[sec]: 0.000
Total time elapsed: 1.456 sec
Second epoch...
0% 100%
[###] | ETA[sec]: 0.000
Total time elapsed: 1.267 sec
Table Ep1-4_p50_mag19_disapp_wmsgsp.dat with 3 entries saved.

RPGWAnalysis -v -i Ep1-4_p50_mag19_app_wmsgs.dat -o Ep1-4_p50_mag19_app_wmsgsp.dat -p
Reading table Ep1-4_p50_mag19_app_wmsgs.dat
Drawing pictures...
First epoch...
0% 100%
[##] | ETA[sec]: 0.000
Total time elapsed: 1.123 sec
Second epoch...
0% 100%
[##] | ETA[sec]: 0.000
Total time elapsed: 0.869 sec
Table Ep1-4_p50_mag19_app_wmsgsp.dat with 2 entries saved.

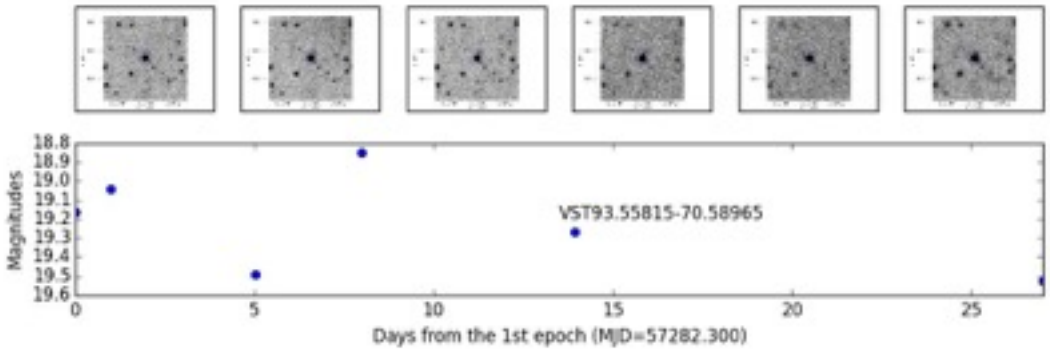
And, finally, if you want to check your candidates you can transform your tables to HTML tables that can be checked visually in an easy way:

SRPGWTabViewer -v -H -i Ep1-4_p50_mag19_matched_wmsgsp.dat
Reading table Ep1-4_p50_mag19_matched_wmsgsp.dat
Converting table to HTML...
Saving Ep1-4_p50_mag19_matched_wmsgsp.html

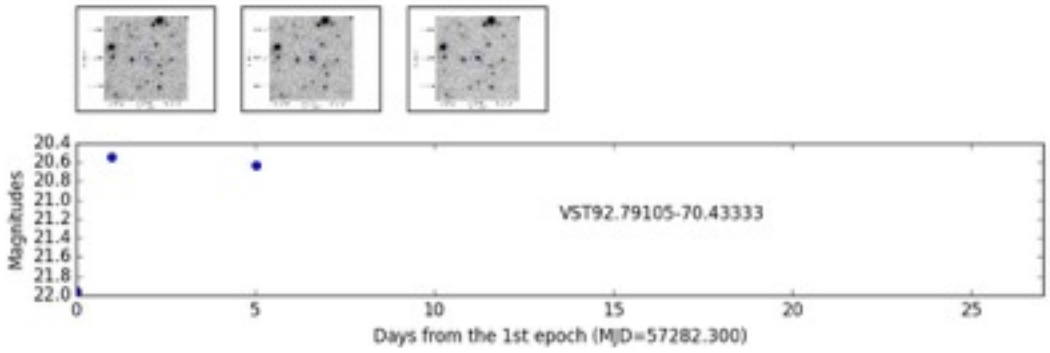
SRPGWTabViewer -v -H -i Ep1-4_p50_mag19_disapp_wmsgsp.dat
Reading table Ep1-4_p50_mag19_disapp_wmsgsp.dat
Converting table to HTML...
Saving Ep1-4_p50_mag19_disapp_wmsgsp.html

SRPGWTabViewer -v -H -i Ep1-4_p50_mag19_app_wmsgsp.dat
Reading table Ep1-4_p50_mag19_app_wmsgsp.dat
Converting table to HTML...
Saving Ep1-4_p50_mag19_app_wmsgsp.html

And it is easy to check that none of the selected candidates can survive the "eye-check".
A real analysis would imply more epochs, and that could allow one to draw a "light curve", as in the example below always in the #50 pointing, but fainter than the limits we have applied here:



It is likely an uncatalogued variable star, not really one of the objects we are looking for. However, this shows the possibilities of an analysis based on photometric catalogues. Unfortunately, although this is intrinsic in any observational techniques, going close to the detection limits make things more and more difficult. In the following example we have one more, faint, possible transient. Unfortunately, the decrease in quality of the observing epochs after the firsts, made it impossible to follow the photometric evolution of this interesting object:



Bugs, comments, etc.
Of course, as already stated, any contribution from anyone is welcome. In case you find bugs, have improvements to suggest, would like to contribute to the code, etc. Please send an e-mail to Stefano Covino, stefano.covino@brera.inaf.it. We can not promise to take into account all your comments, but we will anyway try to improve the package.