CREATING AND ANALYSING MULTI-DIMENSIONAL DATA IN ASTRO-WISE ENVIRONMENT
A TESTCASE

Edwin A. Valentijn and Kor Begeman **30 Nov 2006**

One of the objectives of the AstroWise information system is:

to integrate the "user environment" for the creation, calibration and massaging of images with the environment used for deriving and analysing the catalogued parameters of the objects on these images, so called *Sourcelists*.

- For the images all the META data is stored in a distributed database, including pointers to a scalable storage network containing the files with the actual image data.
- This way, the database together with a separate fileserver can operate 100's Tbyte of image data.
- Typically the user queries the META data of the image, whose lineage is fully maintained by the system and the database.
- However for the sourcelists, next to the standard database storage of META data and lineage to images, the parameter values of the astronomical sources are fully ingested into the database. For OmegaCAM, sourcelists are expected to be created at a rate of 30.000 - 100.000 sources per half hour of observing, thus leading to Terabytes of sourcelist data during the years of operation.
- AstroWise is using Oracle Partitioning on the positional htm index to store, index and rapidly resolve queries to these large amounts of data.
- An own developed Associate tool allows to crossmatch rapidly different sourcelists on the basis of position on the sky (cone search), where it is up to the user to device the track of research and the filtering of the data, depending on the nature of the study, which can range from searching for moving targets, objects with variable light intensity, special colours or drop-outs.
- The AstroWise Associate tool works fully in the database and creates database references lists, so-called *Associatelists*, which contain cross-references to matched sources in up to a nearly unlimited number of different input *Sourcelists*. Also the *AssociateLists* have META data in the database with full data lineage up to the raw data taken at the telescope.
- The AstroWise objective is to provide the user with a research environment in which he can run small *Python* scripts, typically containing a handfull of lines, so-called *5 Lines Scripts* (5LS), to perform the various search, quality control, analysis and display steps.

Here, as a proof of concept, we explore the 5LS concept for analysing multi-dimensional catalogue data including quality control and whether the present implementation of AstroWise tools and documentation facilitates the end user to intelligently and efficiently use the system without having to bother about the high-level software engineering which is behind it.

We use the observational data of Kuijken et al who observed with the WFI@2.2meter telescope a single field centered on the CEN-A galaxy (in the R band) for in total 12 nights spread over a period of 4 month. At each night about 9 images of the field were taken (one night with 21 and few nights with 1, 3 or 4 images). P. Heraudeau calibrated and co-added these images in the AstroWise system and produced one *CoaddedRegriddedFrame* for each night, which is the starting point of our exploration.
The following **set of 5LS**, which in practice are supposed to be entered by the end-user, describe **completely how from the 12 *CoaddedRegriddedFrame*s 25.000 light curves** are obtained for ~25.000 astrononomical objects visible in the field. Display and further analysis and filtering can from there on being done in the Python environment. Some intermediate steps for quality control and fine tuning of parameter values are included .

First the **Sourcelists are created** from the CoaddedRegriddedFrames and inspected .
In this 5LScript Sextractor is run and Sourcelists are put into the database. Some Sextractor parameters are fine tuned according our needs.

```
"""
STEP 1: to make source lists for the  12 co-adds of each period
            - filenames of co-add were communicated by P. Heraudeau, but can also be retrieved from a db
query
"""
#
f  =["Sci-PHERAUDEAU-WFI-------#844---Coadd---Sci-53783.5591543.fits",\
"Sci-PHERAUDEAU-WFI-------#844---Coadd---Sci-53783.6695808.fits",\
"Sci-PHERAUDEAU-WFI-------#844---Coadd---Sci-53783.6662407.fits",\
"Sci-PHERAUDEAU-WFI-------#844---Coadd---Sci-53783.6699597.fits",\
"Sci-PHERAUDEAU-WFI-------#844---Coadd---Sci-53785.0100383.fits",\
"Sci-PHERAUDEAU-WFI-------#844---Coadd---Sci-53783.6844365.fits",\
"Sci-PHERAUDEAU-WFI-------#844---Coadd---Sci-53783.6878318.fits",\
"Sci-PHERAUDEAU-WFI-------#844---Coadd---Sci-53783.6884276.fits",\
"Sci-PHERAUDEAU-WFI-------#844---Coadd---Sci-53783.6876677.fits",\
"Sci-PHERAUDEAU-WFI-------#844---Coadd---Sci-53784.9582404.fits",\
"Sci-PHERAUDEAU-WFI-------#844---Coadd---Sci-53784.9216719.fits",\
"Sci-PHERAUDEAU-WFI-------#844---Coadd---Sci-53784.9209829.fits"]
#
#
for name in f:
  sl = SourceList()
  q = CoaddedRegriddedFrame.filename==name
  sl.frame = q[0]
  sl.frame.retrieve()
  sl.frame.weight.retrieve()                    # apply weight maps
  sl.name = 'CenA-ev-v4-'+str(f.index(name)+1)
  sl.sexconf.DETECT_THRESH = 5.0            # 8.0
  sl.sexconf.BACK_FILTERSIZE=1               # default 3
  sl.sexconf.PHOT_APERTURES=10               # default 5
  sl.sexconf.BACKPHOTO_TYPE='LOCAL'    # GLOBAL is default
  sl.sexparam = ['MAG_AUTO','MAGERR_AUTO','FLUXRADIUS']
  sl.make()
  sl.commit()
  sl.sources.make_skycat()                      # create sourcelist input files for Skycat display
```

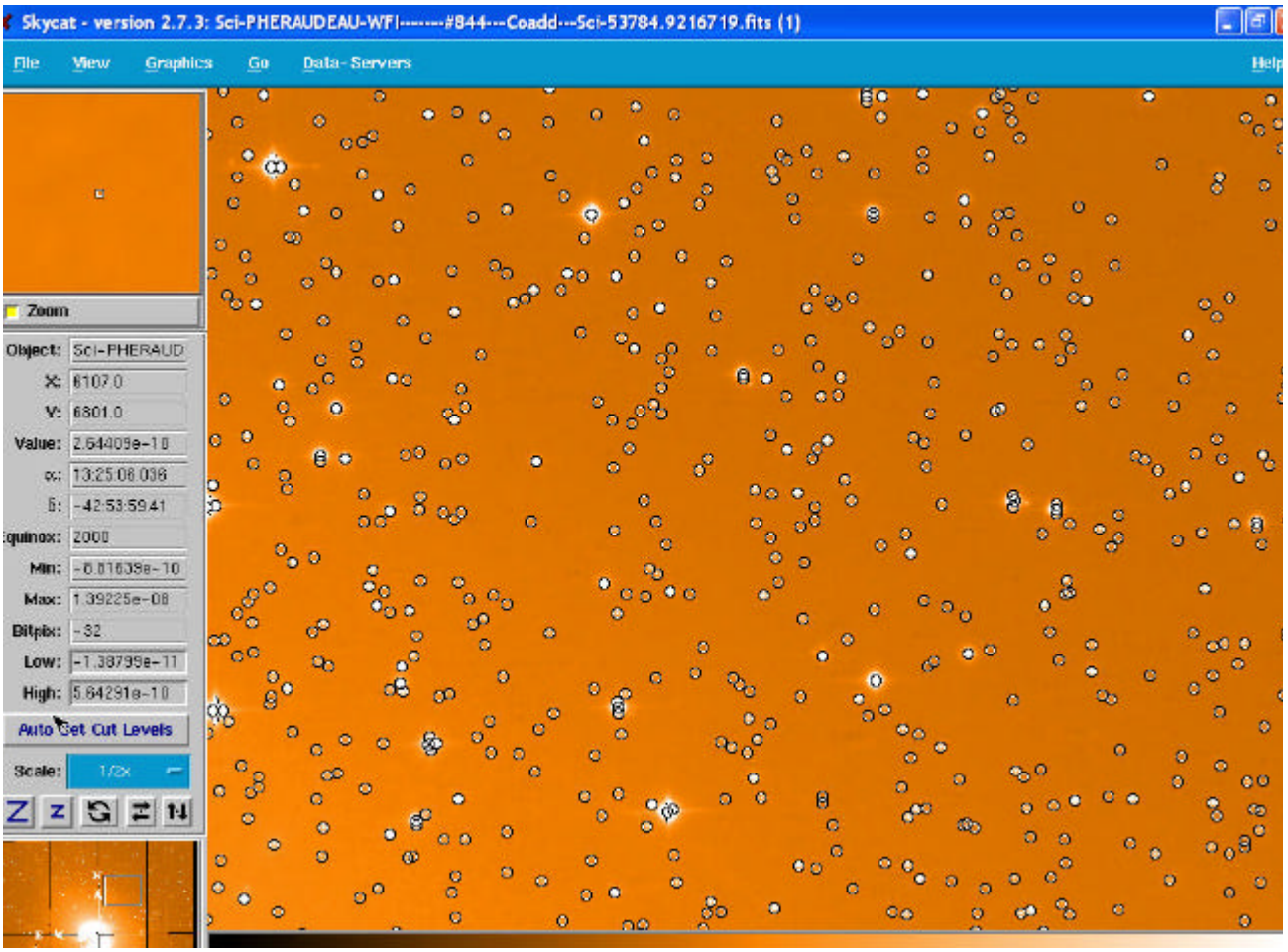
The above script is entered in the Awe prompt. Alternatively the user can obtain a Sourcelist target in the AstroWise/portal web interface, by means of a few mouse clicks.

The resultant Sourcelists can be queried like:

```
slh = (SourceList.name.like('CenA-ev-v4*'))
len(slh)
12
for k in slh:
      print k.SLID,k.name, k.number_of_sources, len(k.frame.regridded_frames)/8
9967 CenA-ev-v4-9 28476 9
9960 CenA-ev-v4-2 27281 9
9966 CenA-ev-v4-8 27027 9
9959 CenA-ev-v4-1 15390 3
9961 CenA-ev-v4-3 36580 4
9963 CenA-ev-v4-5 21791 1
9962 CenA-ev-v4-4 32617 9
9968 CenA-ev-v4-10 29811 21
9965 CenA-ev-v4-7 29612 9
9964 CenA-ev-v4-6 28256 8
9957 CenA-ev-v4-11 25107 12
9958 CenA-ev-v4-12 34948 9
```

The resultant Sourcelists were displayed on top of the CoaddedRegriddedFrames (retrieved in the user

workdirectory in this 5LS) using SkyCAT_transfer files for quality control and fine-tuning of Sextractor parameters. Below an example of a small cut-out of one of the frames is displayed. Obviously, the area around some of the brightest stars are not very well threathed and produce some spurious sources. This can be filtered out in later steps (but improving this during source extraction by masking is desirable).



In the following 5LS **one AssociateList is made on two Sourcelists**. Matching is done within a search cone with diameter 3 arcsec.

```
"""""""
STEP 2 Create AssociateList of two input coadds to do Quality control
          - here db query is done by directly referring to some Sourcelist-identifyers created in STEP 1
"""""""
Al = AssociateList()
Al.input_lists.append(SourceList.SLID ==9957)[0])
Al.input_lists.append(SourceList.SLID ==9858)[0])
Al.set_search_distance(3.0)
Al.make()
Al.commit()
```

Note data lineage from AssociateList to Sourcelist (and beyond) is maintained, e.g the 5LS.:

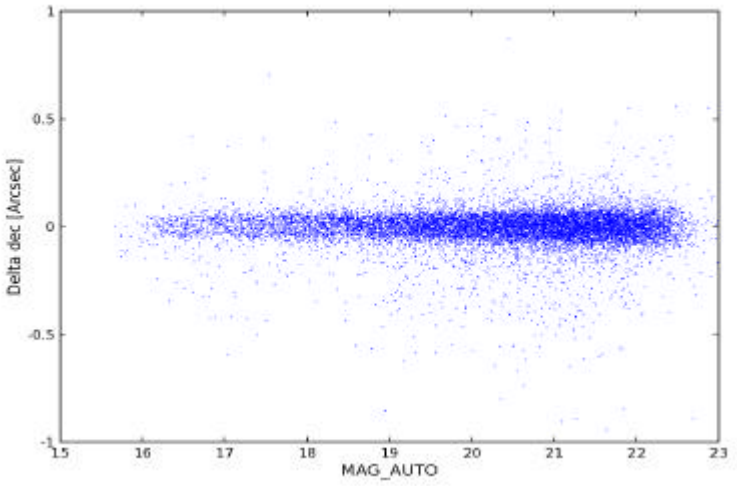
```
print "ALID   Sourcelists"
for k in Al.sourcelists:
    print Al.ALID, k.name, k.SLID
```

In the following 5LS we do some **Quality control** by comparing data from sourcelists taken at different nights.
The specified parameters of the matched sourcelists are retrieved from the database and are put in a Python structure r, which is a dictionary of lists. The r structure can from here on be used for further analysis using

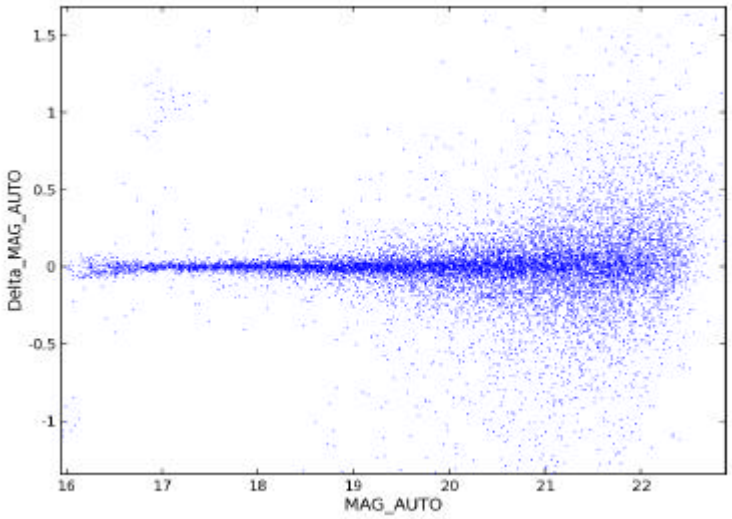
Python, also outside the AstroWise environment.

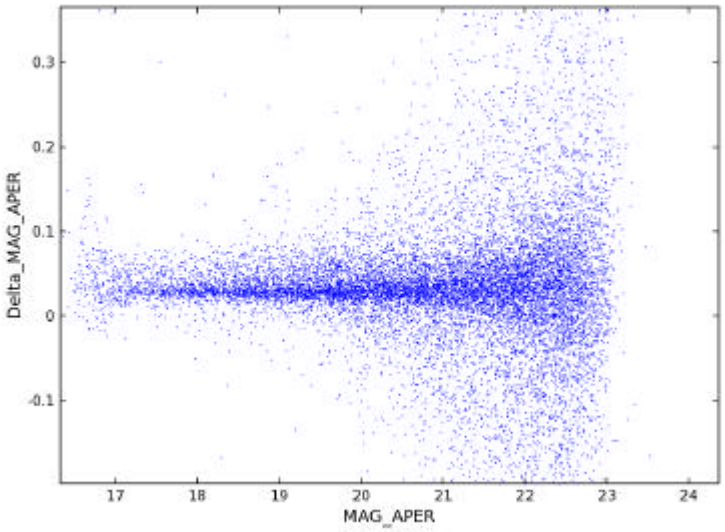
```
"""
STEP 3 to retrieve associations from AL CenA and  plot deltas for some favourite parameters
      - Used Associatelist ALID created in STEP  2
"""
Al = (AssociateList.ALID ==1431)[0]
arlist =  ['RA', 'DEC', 'MAG_ISO', 'MAG_AUTO', 'MAG_APER']
r = Al.associates.get_data(arlist,mask=3,mode='ALL')
x = []
y = []
for aid in r.keys():
    x.append( r[aid][0][5])                    #  2 = RA, 3 = DEC, 4 = MAG_ISO, 5 = MAG_AUTO, 6= MAG_APER
    y.append(r[aid][0][3] - r[aid][1][3])
pylab.plot(x,y,'.')
```

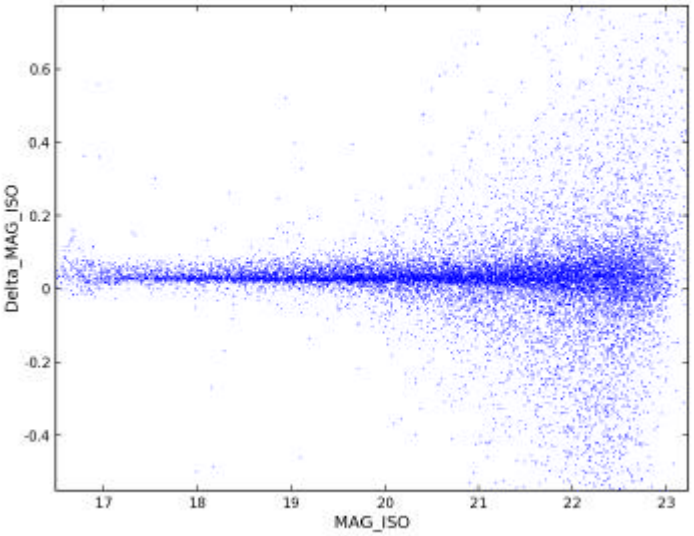
In the above example the difference of declination coordinates of objects observed in different co-adds (taken at different nights) are plotted versus the MAG_AUTO magnitude. Note, the very small positional differences, observational data is taken at different nights and are astrometrically calibrated independently, but with the same reference catalogue stars (USNO). Therefor, systematic errors vanish and the residues reflect the internal accuracy of the astrometric calibrations. About 4% of the objects have residues larger than 0.2 arcsec and most of them will be spurious objects, the remaining **96% of the objects have a rms positional difference of 0.05 arcsec.**



The following plot shows the magnitude differences of two different sourcelists (calibrated with the same zeropoint values) and also indicate the overall reproducabilty of magnitudes in different obervations. **For the brightest 2000 objects the rms magnitude diffrences are 0.03 mag;** the erros is likely dominated by the same zeropoints applies to all co-adds and the difficulties with the sky subtraction on an image with the target galaxy covering a very large fraction of the image. When using MAG_APER these numbers are 3578 objects with R<20 have rms 0.04 mag (similar number for MAG_ISO)







When happy about the quality of Sourcelist (STEP 1) and the quality of the Associated data (STEP 3) we **associate all Sourcelists to that of the first night** and create one big Associatelist. This is a slightly expanded script of STEP 2.
The very good internal positional accuraccy allows to set the cone search diameter to 0.4 arcsec (or even

lower), thus filtering out many artifacts.

```
"""
STEP 4 to Associate all lists;     associate all with one same master
       - use input names
"""
sl    =  (SourceList.name.like('CenA-ev-v4*'))
n = [str(i) for i in range(1,13)]
for i in range(0,len(sl)-1):
    Al = AssociateList()
    Al.associatelisttype=2       # master = sl[0]
    if i==0:
        A = sl[0]                    # first time: input sourcelist
    else:
        A = first                    # other: input previous associatelist
    Al.input_lists.append(A)
    Al.input_lists.append(sl[i+1])
    Al.set_search_distance(0.4)
    Al.OBJECT=name+n[i]
    Al.make()
    Al.commit()
    first =Al
```

9 minutes to run 11 times Associate, evaluating 25,000(start) - 100.000 (end) associations (including null associations).

Now **retrieve the parameters values for the matched sources** on Pythons structure r, which is a dictionary of lists. User can filter and display or apply whatever is available in his Python environment. This is an expanded version of the script in STEP 3.

```
"""
STEP 5 retrieve filter display data of all 12 sourcelists
   - plot light curve
   - use AssociateLIST identification of STEP 4 as input
"""
Al  = (AssociateList.ALID ==1534)[0]
x = range(len(Al.sourcelists))
for i in range(len(Al.sourcelists)):
      x[i] =Al.sourcelists[i].frame.observing_block.date_obs.absdate-732000
      print i,Al.sourcelists[i].name, "Absdate:", x[i], ' number regridded
frames:',len(Al.sourcelists[i].frame.regridded_frames)
##
##0 CenA-ev-v4-12 Absdate: 106  number regridded frames: 72
##1 CenA-ev-v4-9 Absdate: 96  number regridded frames: 72
##2 CenA-ev-v4-2 Absdate: 52  number regridded frames: 72
##3 CenA-ev-v4-8 Absdate: 79  number regridded frames: 72
##4 CenA-ev-v4-1 Absdate: 12  number regridded frames: 24
##5 CenA-ev-v4-3 Absdate: 54  number regridded frames: 32
##6 CenA-ev-v4-4 Absdate: 67  number regridded frames: 72
##7 CenA-ev-v4-10 Absdate: 98  number regridded frames: 168
##8 CenA-ev-v4-7 Absdate: 78  number regridded frames: 72
##9 CenA-ev-v4-6 Absdate: 76  number regridded frames: 64
##10 CenA-ev-v4-11 Absdate: 99  number regridded frames: 96
arlist =  ['RA', 'DEC', 'MAG_ISO', 'MAG_AUTO', 'MAG_APER']
r = Al.associates.get_data(arlist)
for aid in r.keys():
    y = [r[aid][i][5] for i in range(len(r[aid]))]
    if len(x) == len(y):
        z = zip(x,y)                        # sort obsdate for line plotting
        z.sort()
        x=[z[i][0] for i in range(len(y))]
        y=[z[i][1] for i in range(len(y))]
```

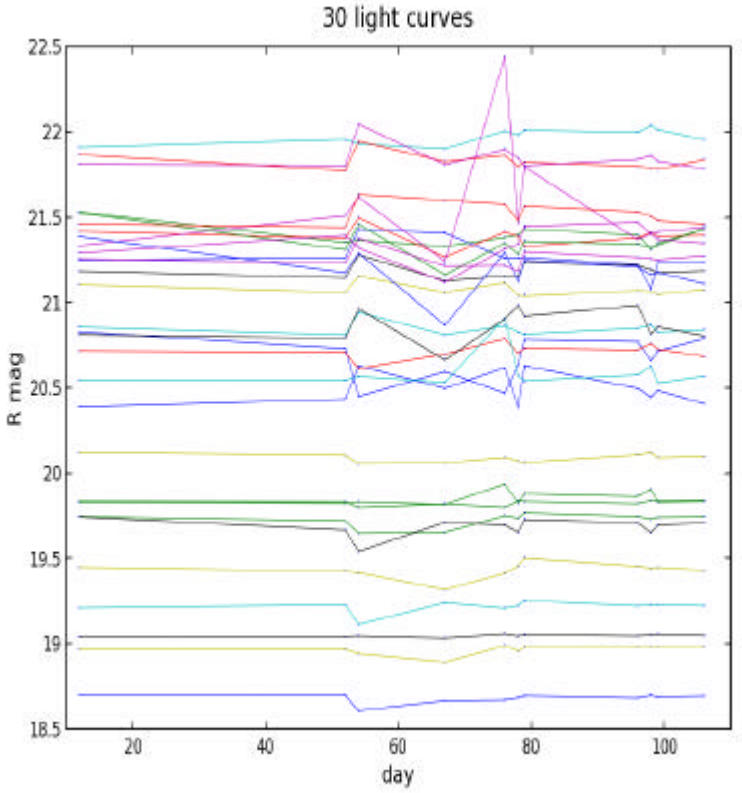
```
        pylab.plot(x,y,'.')
        pylab.plot(x,y)

# pickle.dump(r,'all')
# pickle.loads('all')
```

Note the statement to retrieve the observing date: Al.sourcelists[i].frame.observing_block.date_obs.absdate, which goes 5 levels deep in the object model, starting from the Al which is just a collection of pointers.
It might appear complicated to retrieve the observing date this way, but note that any data value that went into the derivation of Al can be assessed with the same technique. In practice, by playing with the dir() command one can quickly view and follow the dependencies in the object model.

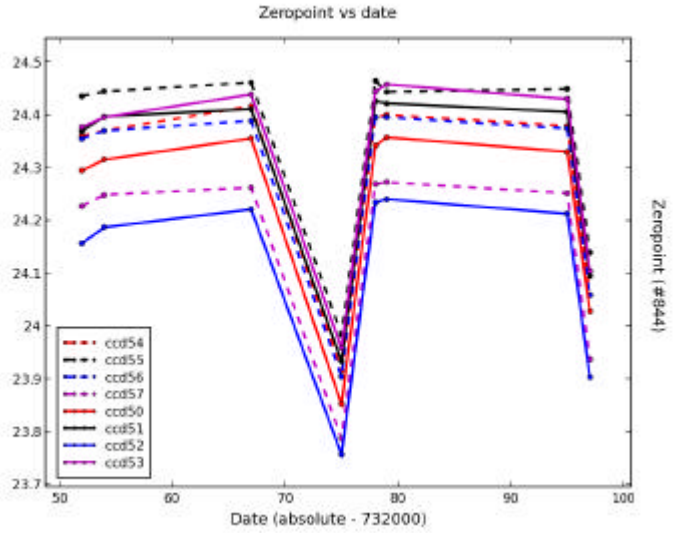
7312 objects were detected on all 12 Co-adds within matching search diameter of 0.4 arcsec

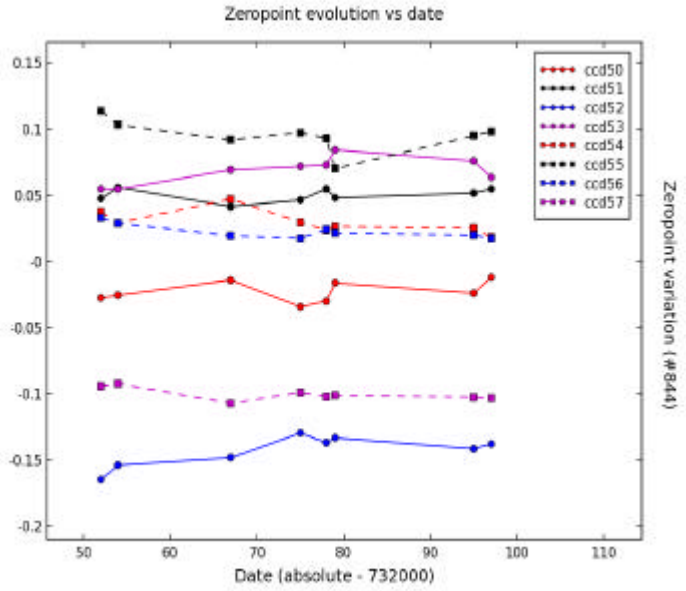
example with plot of light curve



Note the systematic errors appear to dominate the lightcurves.
At this point at becomes clear that the assumption of identical chip-to-chip varation of zeropoint over a long period of time is not justified.

STEP 6 derive individual zerpoints for each night with chip to chip variation using phot pipeline, described elsewhere
in 2 hours time for 12 nights the zeropoints were determined for 12 nights





STEP 7 **rederive co-adds with on the fly re-processing or even sourcelists using Target processor**
process parameters can be tuned

in practice the dataset was re-processed several times to assess the effect of sky background subtraction methods , using the process parameters settings menu in the Target processor.

Results of mean magnitude and dispersion of subsets of the same stars in the various co-adds for various reductions are:

#v6a
##0 CenA-ev-v6-a8 Absdate: 79 number regridded frames: 72
##1 CenA-ev-v6-a7 Absdate: 54 number regridded frames: 32
##2 CenA-ev-v6-a1 Absdate: 12 number regridded frames: 24
##3 CenA-ev-v6-a12 Absdate: 68 number regridded frames: 8
##4 CenA-ev-v6-a2 Absdate: 76 number regridded frames: 64
##5 CenA-ev-v6-a4 Absdate: 52 number regridded frames: 72
##6 CenA-ev-v6-a6 Absdate: 98 number regridded frames: 168
##7 CenA-ev-v6-a13 Absdate: 97 number regridded frames: 72
##8 CenA-ev-v6-a9 Absdate: 67 number regridded frames: 72
##9 CenA-ev-v6-a5 Absdate: 78 number regridded frames: 72
##10 CenA-ev-v6-a11 Absdate: 106 number regridded frames: 72
##11 CenA-ev-v6-a3 Absdate: 96 number regridded frames: 72

```
ssum = get_mean_and_stdev(sum)
print 'average over all', ssum[0], 'sd:', ssum[1]
```

##----------
v4 fi <0.13; **mean zeropoint, no skysubtraction ; aper 15px**
##average= 20.2245539758 266 objects
##average= 20.2238334785 266 objects
##average= 20.3004742099 266 objects
##average= 20.2990666726 266 objects
##average= 20.2835629327 266 objects
##average= 20.3061961769 266 objects
##average= 20.2131450481 266 objects
##average= 20.2869843648 266 objects
##average= 20.3787397442 266 objects
##average= 20.5904598236 266 objects
##average= 20.3140992473 266 objects
##average over all 20.3110105159 **sd: 0.104378348292**
##
##v5 fi<0.13: **Aper 15px, skysubtraction , individual Chip zeropoint**
##average= 20.0647759169 835 objects
##average= 20.2816570031 835 objects
##average= 20.4420485331 835 objects
##average= 20.2916087556 835 objects
##average= 20.2942621425 835 objects
##average= 20.2803295958 835 objects
##average= 20.2752370937 835 objects
##average= 20.3965377693 835 objects
##average= 20.3297143719 835 objects
##average= 20.2909508665 835 objects
##average= 20.0486000586 835 objects
##average over all 20.2723383734 **sd: 0.119171122152**
#
v 6a fi<0.13 **Aper 25 px; no skysubtraction, individual Chip zeropoint**
##average= 19.6990545135 2742 objects
##average= 19.6347413693 2742 objects
##average= **19.56**86281108 2742 objects
##average= **19.58**75206328 2742 objects
##average= **19.60**17207112 2742 objects
##average= **19.59**86159378 2742 objects
##average= 19.3244963402 2742 objects
##average= 19.7943698944 2742 objects
##average= **19.57**23472021 2742 objects
##average= 19.7491232603 2742 objects
##average= **19.57**36349476 2742 objects
##average= **19.57**08026594 2742 objects
##average over all 19.6016666667 sd: **0.115823795116**

Conclusion:
- variation in results is partly caused by non photometric nights (7 nights have dispersion <0.01-0.02)
- effect of sky substraction in Sextractor is playing up in the results at a large than desired level (0.02-0.03m)

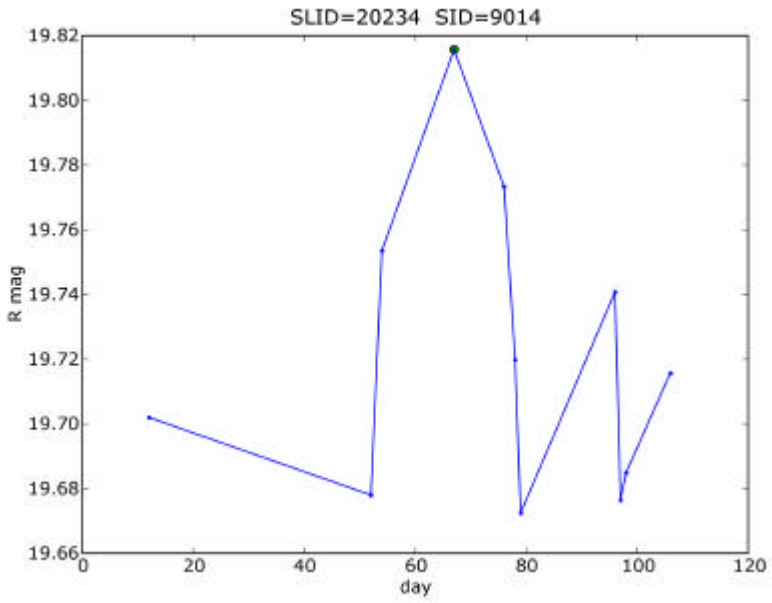
coadd-picture view

STEP 8 **redo STEPS**
1 SI ,
4 associate all SLs
5 plot light curves of selected EVENTS

A fast way to quickly look at selected objects at various images is to use the cutout server by adding the following lines

"eclipsing" stars
feed cutout server via ALD

```
imgdict = AI.associates.make_image_dict(ALD,mode='grid')
ic      = imgclient.imgclient(imgdict, wide_high=[150, 150])
ic.getzipfile()   # receive zipped cut-out fitsfiles
ic.getimg()       # make an HTML-page with image-links to png representations
pylab.plot(x,y,'.')
```



view cutouts in .html

linktable Sci-EVALENTYN-WFI------#844---C

Sci-EVALEN YN-WFI---# -Sci-53875 8286050	Sci-EVALEN YN-WFI---# -Sci-53881 9367447	Sci-EVALEN YN-WFI---# -Sci-53876 6224133	Sci-EVALEN YN-WFI---# -Sci-53876 4051544	Sci-EVALEN YN-WFI---# -Sci-53876 5305774	Sci-EVALEN YN-WFI---# -Sci-53874 5844669