Advanced Lab Course

Photometry of Star Clusters

Part II: Data Reduction & Analysis

Argelander-Institut für Astronomie
Universität Bonn

30th July 2023

written by
Jakob den Brok, Alberto Doria, Andreas Küpper, Xun Shi,
Alex Tudorica & Jan Luca van den Busch

Contents

1 Overview

Here in *Part II: Data Reduction & Analysis* we give a guide on how to perform the data reduction using `Python`. The steps or the tools required to perform the tasks are given in detail in the sections that follow. For the task you can either use your own computer (see apendix for how to install the required software) or you can use the student computers at AIfA.

This script is organized as follows:

- Section 2 is about the data reduction of your images and how you extract the information you need from the images.
- Section 3 is the main scientific part where you will be guided through the analysis of your data..
- The appendix will help you with the installation of software packages which you will use during this project. This is only needed if you decide to do the data reduction and analysis on your own computer.

Please read all sections (except the appendices) carefully **before** you start.

2 Data Reduction

For reducing the data and extracting a catalogue of sources out of the pictures, a number of steps are necessary and a variety of scripts and programmes will be used. In this section we will introduce the tools you need for creating a color-magnitude diagram, this will be done in the order in which you will have to use them. If you are using your own computer or Laptop, check the appendix for details on how to download and install the software packages. If you are using the students computers, then all the software is already installed and ready to use.

2.1 SKYCAT/DS9

You can check your own images and compare them with the virtual-telescope data by using Skycat or DS9. For the first one type

```
> skycat
```

in a terminal window. Open any image with the menu File/Open and select the the file you want to open. To adjust the view of the image push the button Auto Set Cut Levels first and then go to the View/Colors menu. In the new window you can choose various color scalings, color maps and intensity functions. Pick for example Linear, heat and gamma for a good representation of your star cluster (Fig. 1). With Skycat you can also access on-line catalogues like DSS. Simply choose an appropriate catalogue from the Data-Servers menu.

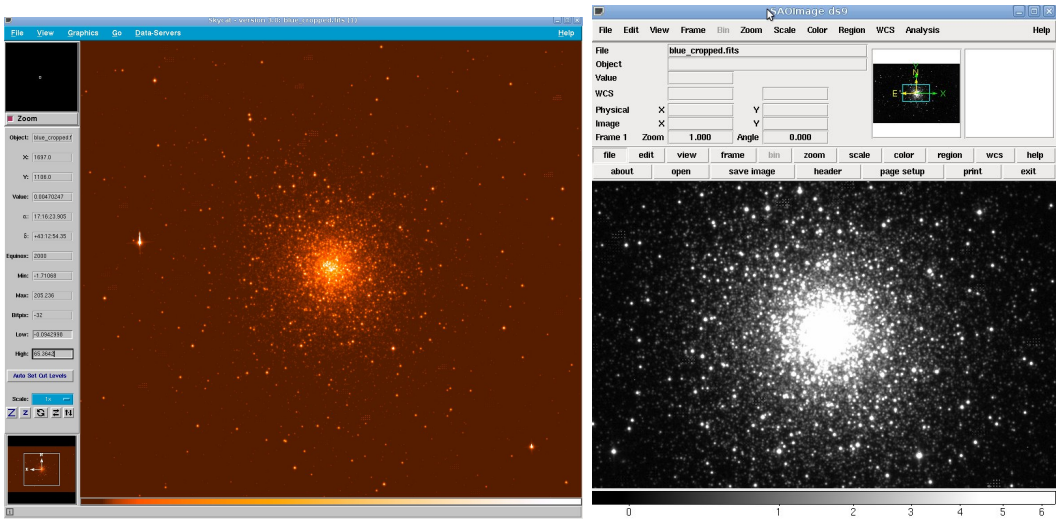

Figure 1: Main windows of Skycat (left) and DS9 (right).

DS9 is similar to Skycat but offers slightly different functionality. To start DS9 type

```
> ds9
```

in a terminal window and the DS9 window will show up (Fig. 1). As in Skycat, open any file with the File/Open menu. The scaling of the view can be adjusted via the Scale and the Color menu. Choose, for example, Zscale and Grey. You can now easily adjust the scaling by holding the right mouse button and moving the cursor

up and down or left and right. To visualize the star cluster clearly it is suggested to use the zscale in the color tab. If you want to access on-line catalogues, you can find them in menu at `Analysis > Image servers`.

2.2 Data Reduction using `Python`

In the following sections we will give you hints and a description of which commands you will have to use for a successful data reduction. We recommend to use `Jupyter Notebook` to do these operations. But of course this is just optional.

1) Open Images

To open an image and save the data in an `np.ndarray` called `data`, just run the following lines:

```
from astropy.io import fits
data, header = fits.getdata('path_to_image', header = True, ext = 0)
```

2) Operations on the Data

The lines above generated `data` and `header`. `data` contains the electronic counts of each pixel. This is thus a matrix and we can do simple operations such as adding, subtracting, dividing, taking the mean or the standard deviation. An example of how to apply these operations is seen in the lines bellow:

```
diff = data1 - data2
```

In the following way you can create a median image. Store for example each 'data variable' in a list or create a 3D array:

```
import numpy as np
image_list = [data1, data2]
median_image = np.median(image_list, axis = 0)
```

3) Saving the Result

We can save the resulting array again in a new FITS file. For that we use the following line of code. This will save the result into a file called `out_image_path`, which you can open for example in DS9 to check if everything went well.

```
fits.writeto('out_image_path', median_image, header = header,
             overwrite = True, output_verify='ignore')
```

These lines of `Python` will help you to perform the dark subtraction and flat-field correction.

How to Proceed: Create a `Python` script that creates a master dark and a normalized flat field using the lines of code above. Then you need to dark subtract and flat field correct every individual raw science frame separately. To be able to add the frames together, we first need to do the image registration (see next section). When you are done, you should have the same amount of reduced science frames as you had raw science frames. Check the reduced images using DS9. They should show a homogenious background (as oposed to the flat field, that was bright in the middle and dimmer at the edges).

2.3 Image Registration

To continue further, it is advised to co-add the reduced science frames together. Due to dithering or bad tracking of the telescope, the images might not perfectly be aligned and so we cannot just simply add the data arrays. Use the following lines of code to align your science frames. Then you can generate the master reduced science frame by taking the median for every pixel (like we created the master dark in the previous section). Assume you have two images, `data1` and `data2`, then you can get the aligned image:

```
import scipy.ndimage as sciim
from image_registration import chi2_shift

#find the offsets
xoff, yoff, exoff, eyoff = chi2_shift(data2, data1, return_error=True)

#interpolate image data2 to match data1
corrected_image = sciim.interpolation.shift(data2, (yoff, xoff))
```

Use as reference image always the same image (e.g. if you decide using your first science frame of the B band, then use this image also when registering the V and R band images).

To create the final master reduced science frame you simply have to take the median of the list of aligned images:

```
list_images = [data1, corrected_image2, corrected_image3]
master_reduced = np.nanmedian(list_images, axis = 0)
```

2.4 Creating a Weight Map

In order to determine the weights, we can just use the flat field. We do not normalize it by the mean, but by the maximum, that way we are guaranteed to get values between 0 and 1. For that we need the following lines of code:

```
data, header = fits.getdata('path_to_master_flat', header = True, ext = 0)
weight = data/np.max(data)
fits.writeto('out_weight_path', weight, header = header overwrite = True,
             output_verify="ignore")
```

2.5 Cosmic Ray Subtraction

You need to think about whether you need to do a cosmic ray subtraction or not. If you have less than three raw science frames per filter, or you created the master reduced science frame by co-adding (instead of taking the median) then you have to perform the cosmic ray subtraction. There are two options on how to perform the cosmic ray subtraction. You can either use `astroscrappy` or the `Cosmics.py` script.

1) `astroscrappy`
When you decided to use `astroscrappy` use the following lines of code:

```
from astroscrappy import detect_cosmics
from astropy.io import fits
data, header = fits.getdata('path_to_image', header = True, ext = 0)
mask, clean = detect_cosmics(data, inmask=None, sigclip=4.0,
                             sigfrac=0.3, objlim=5.0, gain=1.15,
                             readnoise=6.5, satlevel=65536,
                             niter=4, sepmed=True,
                             cleantype='meanmask', fsmode = 'median',
                             psfmodel='gauss', psffwhm=2.5,
                             psfsize= 7,
                             psfk=None, psfbeta=4.765, verbose=False)
fits.writeto('path_to_image_cr.fits', clean, header = header,
             overwrite = True,  output_verify='ignore')
```

The `detect_cosmics` takes a lot of parameters as input. Figure out yourselfs by reading the documentation which values you do need to enter. The values given are just placeholders. Usually the information is given in the header or you can read in the instrument manual, like the CCD read out noise or saturation level. The function is described in detail here: `https://astroscrappy.readthedocs.io/en/latest/api/astroscrappy.detect_cosmics.html#astroscrappy.detect_cosmics`.

2) `Cosmics.py`

To download the `Cosmics.py` script, go to `https://github.com/ebellm/pyraf-dbsp/blob/master/cosmics.py`. If you are unsure how to download the file, ask your tutor to show you how to download data from GitHub. The use the following python code:

```
import cosmics
data, header = fits.getdata('path_to_image', header = True, ext = 0)
c = cosmics.cosmicsimage(data, gain =1.15, readnoise=0.0, sigclip=5.0,
                         sigfrac=0.3, objlim=5.0, satlevel=65535)
fits.writeto('path_to_image_cr.fits', c.cleanarray, header = header,
             overwrite = True, output_verify='ignore')
```

Again, read the docs and check the instrument manual and/or FITS header for the proper values of the parameters.

2.6 Background Subtraction

For the background subtraction we can use the package `photutils`. A very detailed description can be found in `https://photutils.readthedocs.io/en/stable/background.html`. The procedure is the following: 1) You estimate the background for every science frame. 2) You subtract the background image from the science frame. 3) Check the resulted image using e.g. DS9.(Hint: you can use the image statistics funtion in ds9 to plot a histogram of the average count value in a given region devoid of any sources to check if the background subtraction is done successfully.) The following code will help you performing these steps:

```
from astropy .io import fits
from astropy.stats import sigma_clipped_stats, SigmaClip
from photutils.segmentation import detect_threshold, detect_sources
from photutils.utils import circular_footprint
```

```
data, header = fits.getdata('path_to_image', header = True, ext = 0)

sigma_clip = SigmaClip(sigma=3.0, maxiters=10)
threshold = detect_threshold(data, nsigma=2.0, sigma_clip=sigma_clip)
segment_img = detect_sources(data, threshold, npixels=10)
footprint = circular_footprint(radius=10)
mask = segment_img.make_source_mask(footprint=footprint)
mean, median, std = sigma_clipped_stats(data, sigma=3.0, mask=mask)

backsub_data = data - mean
```

This task is not trivial and you will have to play with the input parameters a little bit. Make sure you check the resulting image with DS9 to make sure this step has been performed correctly.

2.7 Astrometric Calibration

Astrometric calibration matches the image co-ordinates to the sky co-ordinates. This can be done using **Astrometry.net.** This is an online tool that performs the astrometric calibration and is probably the most simple and straight forward way to do. Go to `http://nova.astrometry.net/upload`. There you can upload a single image or you can collect the images into a tarball (`.tar, .gz`) and upload. Then you can let astrometry.net do the rest of the work and eventually download the resulting FITS file with astrometric solution. If you are doing it online, you must be aware that the uploaded images are public, if you are not logged in. What astrometry.net does it that is searches for objects in your field and cross-matches them with a catalogue of stars to find the coordinates of your field.
The method described above can have trouble finding the right solution when only providing a field with only few source or a very crowded field.

2.8 Photometric Calibration

In order to perform the photometric calibration, we need to have an observation of a standard star. The science frame of this standard star needs to be dark subtracted and flat-field corrected. For a detailed description of the photometric correction, you can follow this link: `https://slittlefair.staff.shef.ac.uk/teaching/phy241/lectures/l07/`. This step can be performed using the `photutils` package in `Python` by applying aperture photometry. Follow the description given in `https://photutils.readthedocs.io/en/stable/aperture.html#performing-aperture-photometry`. Alternatively, the tutor can provide you with the right photometry correction in case you had trouble observing a standard star.

No Standard Star was observed: If no standard star was observed, we cannot perform this step. A standard star needs the requirement that it's brightness should not vary and so we cannot simply chose another random star in our field of view, as we might have picked a variable star. To proceed, we will select a number (20-25) of stars in our image and take the mean of the magnitudes for each filter. That way

we are sure to mitigate variability effects. To do that however, we need to first run `SExtractor` (see section 2.11). So skip this step and follow the instruction there.

2.9 Cropping a FITS image

In order for source extractor to properly select stars from the cluster we have to crop the science frame. Otherwise source extractor select too many stars that are not in the cluster and thus the Hertzsprung Russell diagram can not be properly made. For that, use this `Python`-package: `https://docs.astropy.org/en/stable/nddata/utils.html`. This packages makes sure that when you crop the image, the astrometric solution get's also "cropped". Read the documentation carefully or ask your tutor for help. Check however first, if you really need to crop the image. If the extension of the cluster is larger than our field of view, then we do not have to crop the image of course.

2.10 Summary

Because of the many steps, it is easy to loose track of what we need to do. Here a short overview of the individual steps:

1. **Dark Subtraction & Flat Field Correction**
   → **Use:** `Python` (`astropy, scipy, numpy`)
   → **Result:** A single dark & flat corrected image for every filter.
2. **Weight Maps**
   → **Use:** `Python` (`astropy, numpy`)
   → **Result:** A weight map for every filter.
3. **Cosmic Ray Subtraction**
   → **Use:** `Python` (`astroscrappy` or `Cosmics.py`)
   → **Result:** Reduced Science Images are now free of cosmic rays
4. **Background Subtraction**
   → **Use:** `Python` (`photutils`)
   → **Result:** Reduced Science Images corrected for background light
5. **Astrometric Solution**
   → **Use:** astrometry.net
   → **Result:** Coordinate information in header
6. **Photometric Solution**
   → **Use:** `Python` (`photutils`)
   → **Result:** Reduced Science Images with correct flux/magnitude
7. **Cropping Image**
   → **Use:** `Python` (`astropy`)
   → **Result:** Reduced Science Images with reduced field of view

After you have finished these steps, we can go over to extracting the sources from the image using `Sextractor`.

2.11 SExtractor

The final extraction of the sources will be done using SExtractor[1], a program that builds a catalogue of objects from an astronomical image. Although it is particularly oriented towards reduction of large scale galaxy-survey data, it can perform reasonably well on moderately crowded star fields. Thus, it may lead to bad results in the centers of globular clusters, but will suffice for our purposes. You can use a homemader script if working on the students computers at AIfA, or you can just run source extractor yourself.

2.11.1 Working on your own computer

i) Run Source Extractor on your image. For that we need to generate the dafault configuration file (if you are working on the student computers, use `sextractor` or `source-extractor` instead of `sex` at the beginning):

Code Listing 1: Create Default Configure File

```
$ sex -d > default.conf
$ sex -dp > default.param
```

ii) Most of the parameters in `default.conf` can be left at their default values (however, try to under- stand them). You should modify only the parameters listed below:

- `PARAMETERS_NAME  default.param`
- `DETECT_MINAREA  10`
- `DETECT_THRESH  2.5`
- `ANALYSIS_THRESH  2.5`
- `FILTER_NAME  /opt/local/share/sextractor/default.conv`
- `STARNNW_NAME  /opt/local/share/sextractor/default.nnw`

Note: on your computer the path to `default.conv` and `default.nnw` could be different. But you can also use the files provided by our tutor.
The `default.param` file lists all the parameters that you want to be saved in the output catalog. You should output at least the following parameters (check the documentation for possible other output parameters):

- `NUMBER`
- `X_WORLD`
- `Y_WORLD`
- `MAG_AUTO`
- `MAGERR_AUTO`

[1] `http://www.astromatic.net/software/sextractor`

iii) First we run source extractor on the B filter image:

Code Listing 2: Run SExtractor 1

```
$ sex -c default.conf input_image.fits -CATALOG_NAME output_catalog.cat
       -MAG_ZEROPOINT xxx  -WEIGHT_TYPE MAP_WEIGHT
       -CHECKIMAGE_NAME check_image.fits -WEIGHT_IMAGE weight_image.fits
```

The magnitude zeropoint is the magnitude of an object that produces only one count per second. You find this value using the following formula:

$$m = -2.5 \log_{10}(n_{\rm ct}/t_{\rm exp}) + m_0$$

where m is the magnitude of your standard star, $n_{\rm ct}$ are the number of counts of your standard star observation, $t_{\rm exp}$ is the exposure time (in seconds) and m_0 is the zeropoint magnitude. If you did not observe a standard star, just leave this parameter out or set it to 0.

iv) Next run SExtractor on the V band image in the so-called 'dual-image mode'. In this mode SExtractor will first identify objects on the B band image, and then extract the flux of the same objects on the V band image.

Code Listing 3: Run SExtractor 2

```
$ sex -c default.conf ref_image.fits,input_image.fits
      -CATALOG_NAME output_catalog.cat -MAG_ZEROPOINT xxx
      -WEIGHT_TYPE MAP_WEIGHT - CHECKIMAGE_NAME check_image.fits
      -WEIGHT_IMAGE weight_image.fits
```

Check whether you have the same number of objects as in the B filter image catalog and if the sources are in the same order.

In principle, we could do the same excercise for the R band image, but for the analysis part, we just need the B and V filter band images.

Task: Run Source Extractor several times and change in the `default.conf` file the parameters `DETECT_MINAREA` and `DETECT_THRESH`. Choose one of the pairs (5,5), (10,10), (15,40) and (20,100). You can play with the values and compare the resulting numbers of detections. The larger values will give you less false detections but also much less sources. You have to decide, which detections you will base your analysis on.

2.11.2 Computer at AIfA

If you are working on the students computer at AIfA, you can extract the sources with a small home-made shell script that makes use of the SExtractor routines. Therefore, create a folder, e.g.
`../2014.02.29/M92_CATALOGUE/`
and copy the following files into this directory:

- `get_catalogue.sh`,

- get_catalogue.assoc,
- get_catalogue.makessc,
- the cropped images,
- the cropped weights.

Run the script by typing the following command in a terminal window:

Code Listing 4: Run Source Extractor

```
$ ./get_catalogue.sh PATH B_NAME V_NAME R_NAME
```

where PATH is the full path of the current folder, B_NAME is the name of the blue cropped image, V_NAME is the green cropped image and R_NAME is the third, in case you have made observations in a third filter. Otherwise simply repeat argument V_NAME, in this way you will get an "R" column in the catalogue which is redundant, since it repeats the magnitude of the V-band.

The script will create source catalogues for each filter and cross reference the three catalogues. Only sources which get detected in all three filters will be written to the resulting ascii catalogue which will be located in
../2014.02.29/M92_CATALOGUE/result/result_ascii.cat
where the columns are:

1. right ascension (J2000) in degrees,
2. declination (J2000) in degrees,
3. B magnitude,
4. B magnitude error,
5. V magnitude,
6. V magnitude error,
7. R magnitude,
8. R magnitude error.

You additionally get catalogues for each filter and the combined catalogue in an enhanced file format (LDAC) comparable to the .fits format for images, which you don't need here any more, but which you should keep in mind in case you ever need to deal with more sophisticated catalogues.

You can quickly count the number of detected sources by typing

Code Listing 5: Count Detections

```
$ wc -l result_ascii.cat
```

within the result directory. The number of detections depends on the parameters DETECT_MINAREA and DETECT_THRESH. You can set those parameters by editing the script with a text editor or by simply adding the two numbers to the command line argument:

Code Listing 6: Run Source Extractor Advanced

```
$ ./get_catalogue.sh PATH B_NAME V_NAME R_NAME DETECT_MINAREA DETECT_THRESH
```

where you should choose one of the pairs (5,5), (10,10), (15,40) or (20,100). You can play with the values and compare the resulting numbers of detections. The larger values will give you less false detections but also much less sources. You have to decide, which detections you will base your analysis on.

2.11.3 Photometric Correction without a standard star.

If you do not have a standard star observation we will apply the following technique to perform the photometric calibration. Open the catalog and chose 20 stars for every filter from that catalog (it does not matter which threshold you used for the catalog, you just pick one of the thresholds).

```
 1 #   1 Ra_B                Right ascension of barycenter (J2000)          [deg]
 2 #   2 Dec_B               Declination of barycenter (J2000)              [deg]
 3 #   3 MAGAUTO_B           Kron-like elliptical aperture magnitude        [mag]
 4 #   4 MAGERR_AUTO_B       RMS error for AUTO magnitude                   [mag]
 5 #   5 MAGAUTO_V           Kron-like elliptical aperture magnitude        [mag]
 6 #   6 MAGERR_AUTO_V       RMS error for AUTO magnitude                   [mag]
 7 #   7 MAGAUTO_R           Kron-like elliptical aperture magnitude        [mag]
 8 #   8 MAGERR_AUTO_R       RMS error for AUTO magnitude                   [mag]
 9 40.56065101 42.88051542 6.11067 0.000296648 5.70197 0.000222006 5.70197 0.000222006
10 40.56845998 42.88003573 7.28115 0.000810245 6.73460 0.000534074 6.73460 0.000534074
11 40.41527067 42.87906767 6.07487 0.000302097 5.69599 0.000232379 5.69599 0.000232379
12 40.55999698 42.86645116 4.84259 9.78641e-05 4.54896 8.14283e-05 4.54896 8.14283e-05
13 40.46166200 42.86045086 7.29508 0.000845191 6.75271 0.000559257 6.75271 0.000559257
14 40.55881855 42.85615432 8.26699 0.00188921 7.41657 0.000941269 7.41657 0.000941269
15 40.60962285 42.85491587 9.91393 0.00824595 9.24433 0.00485296 9.24433 0.00485296
16 40.68684603 42.84556014 8.68943 0.00270854 7.87682 0.00139732 7.87682 0.00139732
17 40.52340083 42.85069288 8.46524 0.00222012 7.77144 0.00127780 7.77144 0.00127780
18 40.50139942 42.84999145 10.5172 0.0136903 9.83412 0.00795761 9.83412 0.00795761
19 40.54385625 42.84369413 9.61048 0.00657193 8.90483 0.00374141 8.90483 0.00374141
20 40.54704328 42.84341418 10.6299 0.0164606 9.73571 0.00787695 9.73571 0.00787695
```

Figure 2: Example of how the output for the sources from all three sources using `Sextractor` should look like when you worked on your own computer

For the stars that you selected you take the coordinates and go to simbad, where you can search for the star by coordinates (`http://simbad.u-strasbg.fr/simbad/sim-fcoo`). Be sure to remove the "e+01" and move the comma by one place when entering the cooridinates.

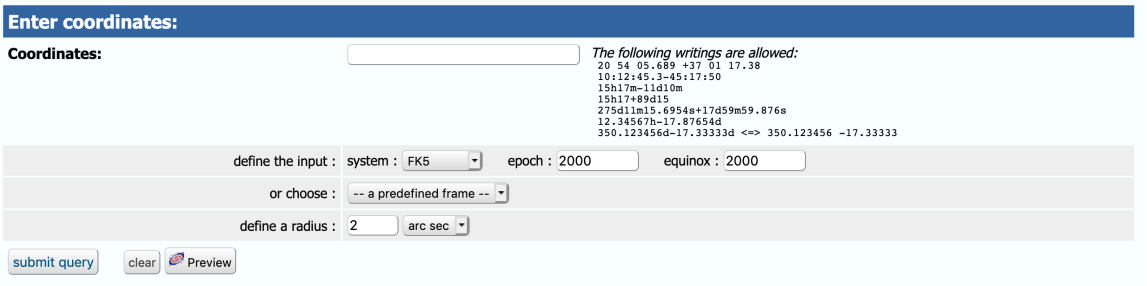

Figure 3: Go to `http://simbad.u-strasbg.fr/simbad/sim-fcoo` and search by coordinates. Be sure to remove the "e+01" and move the comma by one place when entering the cooridinates. Also, select the *define a radius* to 2 **arc sec**.

Once you have selected your sample of 20-25 stars, take the mean value of `MAG_AUTO` of the stars that you selected and the mean value of the magnitude that you found on Simbad. The difference is the offset that you need to account for when plotting the data.

3 Analysis

After extracting the catalogue from the observational data you can begin the analysis. Aim of the analysis will be to determine the most basic parameters of the clusters, the distance, the age and metallicity. We recommend to do the analysis with PYTHON since it is a very convenient tool for such tasks as plotting data from many different files, however if you have any other preference, feel free to use your favourite plotting tool.

3.1 Calibration

Begin your analysis by drawing a first color-magnitude diagram. We recommend using `Jupyter Notebook`. Plot the color index B-V versus the V magnitude with using the following lines of code:

```
import numpy as np
import matplotlib.pyplot as plt

# import the result_ascii.cat file
result = np.genfromtxt("result_ascii.cat")

#Prepare the data
B_V = result[:,2]-result[:,3]
V = result[:,3]

#Plot the data
plt.figure()
plt.scatter(B_V, V)
plt.show()
```

If you have not yet performed the photometric correction, you can do it manually. For that use the standard star that you observed or use the factor provided by your tutor. To correct it manually, you can apply the following when preparing the data:

```
#Prepare the data
B_V = (result[:,2]+0.3)-(result[:,3]+0.1)
V = (result[:,2]+0.1)
```

where the numbers 0.3 and 0.1 are the corrections. Now you can draw a calibrated CMD.

3.2 Extinction

Extinction by interstellar dust makes stars appear dimmer than they are. This can be corrected for by adding a constant factor to each magnitude. In principle these factors can be determined through a color-color diagram in which two color indices are plotted versus each other. But therefore you need reliable images in three colors and sophisticated analyses. This is hardly possible with the data taken from Bonn due to light pollution, thus we will stick to values listed in the literature. Often the extinction factor is smaller than the achievable accuracy in magnitudes which can be achieved from Bonn. Thus, assume an extinction factor E(B-V) of zero, if not stated otherwise by your tutor.

3.3 Distance Determination

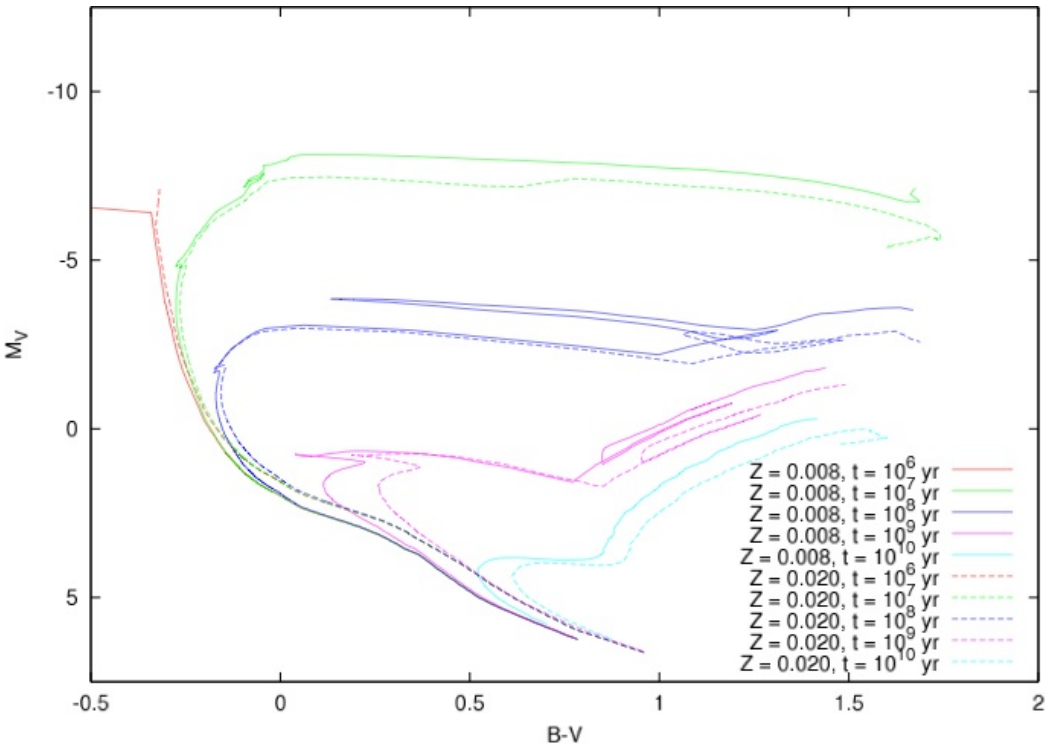

Figure 4: Isochrones for high and low metallicity stellar populations. The main sequence of young clusters is almost independent of metallicity for $B - V < 0$. For old clusters (older than 10^9 yr) the horizontal branch is independently of metallicity located at +0.5.

The distance to star clusters can be determined in different ways, e.g. with variable stars or supernovae. With our CMD we can also derive the distance since there are stars in each cluster with well defined absolute magnitudes. By comparing the observed magnitudes of those stars with their theoretical absolute magnitudes we can evaluate the distance modulus (eq. 1) of the cluster. But the stellar populations of open clusters and globular clusters can differ significantly, such that not every method can be applied to every cluster.

3.3.1 Main-Sequence Fitting

From Fig. 4 we can see that open clusters of young age have a well defined main sequence for high mass stars, which does not vary significantly with metallicity. Based on this part of the main sequence we can estimate the distance without regard to the age and metallicity of the cluster. Therefore take the youngest isochrone of one of the isochrone directories, e.g.
`isochrones/geneva/008/iso_c008_0600.UBVRIJHKLM`
and plot column 9 (B-V) versus column 7 (absolute V magnitude) into your open cluster CMD, i.e.

Apply a constant shift to the V magnitudes such that the high mass part of the main sequence fits the observations. From this shift you can calculate the distance d by using

$$d = 10^{0.2(m-M+5)}\text{pc}. \tag{1}$$

3.3.2 Position of the Horizontal Branch

Old globular clusters do not show an extended main sequence (Fig. 4). But, as stellar populations evolve, massive stars move onto the horizontal branch where they burn helium in the core. The absolute magnitude at which they shine in this stage is well known and can be used to determine the distance to old star clusters.

M_V of horizontal branch stars is approximately +0.5 mag. Thus, identify the horizontal branch in your CMD, shift your V magnitudes accordingly and apply equation 1.

3.4 Isochrone Fitting

After we have produced a well calibrated CMD we can fit a theoretical isochrone to the data and see which parameters reproduce the cluster best. For this task we use the Geneva database of isochrones (Lejeune & Schaerer, 2001), named after the university of origin. Note that these isochrones are based on simplified assumptions and that they obey an intrinsic uncertainty. There are other databases of isochrones, coming, e.g., from Padova (Nasi et al., 2008), which in some cases differ substantially from the Geneva models. For simplicity we stick to the Geneva database here.

The available set of isochrones covers metallicities (in solar abundances) of $Z = \{0.001, 0.004, 0.008, 0.020, 0.040, 0.1\}$ and ages of 10^3 yr to 15 Gyr in a reasonable step width. The data is stored in the folder `geneva`. This folder contains subfolders for each metallicity and in every subfolder are separate files for each age. For example the file
`geneva/001/iso_c001_0950.UBVRIJHKLM`
contains the isochrone for $Z = 0.001$ and for an age of $10^{9.5}$ yr. The columns in the files are always the same. Use column 9 for B-V and column 7 for the absolute V magnitude. For each cluster go through the data set and compare the theoretical isochrones with your data. For young clusters you should stick to high metallicities whereas for old clusters you should apply low metallicities. You are free to use any fitting method you want, but make sure you perform a thorough analysis and account for all errors as well as document it in the report.

When you have a good fit, you can save the figure using the following lines of code:

```
plt.figure()
plt.scatter(B_V, V)
plt.plot(isochrone[:,8],isochrone[:,6])
plt.savefig("isochrone_c001_0950.pdf", bbox_inches = "tight")
plt.show()
```

Our aim is it to understand anything in our CMD, since there should be an explanation for any position of any data point in the graph. Therefore, answer the following questions:

- What age has the specific cluster?
- How large are the uncertainties in this determination? How can you estimate reasonable uncertainties?
- What influences the width of the main sequence, i.e. why is the main sequence not as thin as the isochrone predicts? Think about how the isochrone is produced, and what is assumed in this process.
- Can you identify stars that lie beyond the turn-off point, i.e. are bluer than the turn-off point but still appear to lie on the main sequence? How can this happen?
- How many stars can you identify in this way to appear abnormal in age/-color and what does it tell you about the stellar population of this cluster?

Do this for all clusters for which you have taken data and for each additional catalogue which your tutor hands out to you.

References

Lejeune T., Schaerer D., 2001, A&A, 366, 538

Nasi E., Bertelli G., Girardi L., Marigo P., 2008, MmSAI, 79, 738

A Software Requirements

Here we will describe the data reduction software requirements when using the programing language `Pyhton`. In nowadays astronomy, most of the routines and scripts – as long as we are not dealing with high–performance computing – are written in `Python`. That is why it is essential that you acquire the skills to use this language. Other software required are DS9, Source Extractor, SWARP and/or DeepSkyStacker. If you are using the computer in the computer room. If you want to work on your own computer, then follow the instructions below. The appendix is on large part taken from the "Brief Guide to Data Reduction" written by Luca Tortorelli at the ETH Zurich and updated.

A.1 Python

A.1.1 Installation

The procedure of the installation depends on the operating system of your computer.
⇒ **Linux/Mac**
If you are using Linux or Mac, `Python` should already be installed.
⇒ **Windows**
If you are using Windows on the other hand, you will have to download and setup Python yourself. To download, go to `https://www.python.org/downloads/windows/`. Here you can chose from a range of versions. I would suggest picking a version 3.7.*. I would not recommend Python 2, as it has already passed its end–of–life support, which means it is no longer fully maintained. When choosing the newest version on the other hand, it might be that some packaged are not yet fully up–to–date. A step–by–step guide on how to install can be found here: `https://www.howtogeek.com/197947/how-to-install-python-on-windows/`.

A.1.2 Virtual Environment

Using a virtual environment is strongly recommended[2], as different python application often use different packages that are not part of the standard library. Sometimes they need a specific version of that module and with a virtual environment, you can better control and manage the package versions that you need for your script. There are two ways: Either you set up the environment yourself, or you install Anaconda. Especially for MacOS users, we recommend that us create the virtual environment yourself.

⇒ **MacOS** We need to install homebrew first. Go to `https://brew.sh/`. Past the shown line into your terminal and press Enter. Homebrew is a package manager. For Mac, there exists also another package manager: `MacPort`. When using both package manager on the same computer, one needs to be careful. I recommend using Homebrew (as it is a little bit more safe if you don't know what you're doing, because it

[2]We recommend it, but it is not required! If you do not manage to set up the virtual environment after an hour, just skip it or install Anaconda. In general, should you have any trouble with the installation, you can ask your tutors or use the student computers at AIfA.

doesn't need root access).

1) Install `Python` using Homebrew by entering the following line into your terminal (without the $ sign):

Code Listing 7: Brew install

```
$ brew install python3
```

2) Check, where `Python` is located. It should be located in `/usr/local/bin`. You can see the path by running:

Code Listing 8: Python path

```
$ which python3
```

3) Next we need to install the setuptools and then we can install virtualenv using the following lines:

Code Listing 9: Installing Setuptools

```
$ pip3 install -U pip setuptools
$ pip3 install virtualenv
```

Now everything should be set up to create virtual environment.

4) In a final step, we need to set our `~/.profile` file such that the language. This can be done with the following lines in the terminal

Code Listing 10: Changing Profile File

```
$ cat ~/.profile
$ export LC_ALL=en_US.UTF-8
$ export LANG=en_US.UTF-8
$ source ~/.profile
```

Now everything should be setup to generate a virtual environment.

⇒ **Linux**

Check if you can submit `Python` in the command line by typing `python3`. If you cannot submit this, the you need to install `Python` using the following line in your terminal (if you use Ubuntu):

Code Listing 11: Installing python

```
$ sudo apt install python3
```

1) Check to make sure that you have pip installed by typing:

Code Listing 12: Installing python

```
$ pip3 --help
```

Here `--` are two `-` after each other.

2) If pip3 is not found, we install it using the following lines of code:

Code Listing 13: Installing Pip

```
$ sudo apt install python3-pip
```

3) Next we can install virtualenv by running the following line in the terminal:

Code Listing 14: Installing Virtualenv

```
$ sudo pip3 install virtualenv
```

4) Finally we need to change the ∼/.bashrc file:

Code Listing 15: Changing Profile File

```
$ cat ∼/.bashrc
$ export LC_ALL=en_US.UTF-8
$ export LANG=en_US.UTF-8
$ source ∼/.bashrc
```

Activating a Virtual Environment on MacOS or Linux
1) We are now ready to activate the virtual environment. To do this, the commands are the same for MacOS and Linux. Create a directory. For example, let's make the folder `Data_Reduction` in the terminal:

Code Listing 16: Creating Virtualenv

```
$ mkdir Data_Reduction
$ cd Data_Reduction
$ python3 -m venv venv
```

The term `mkdir` stands for "make directory". With the term `cd` we "change directory". With the last line, we have created a virtual environment. When typing `ls venv`, it listst the folders that are in the directory `venv`. Here `bin, include` and `lib` should be listed.
2) Next we have to activate the virtual environment. This we can do in the following way:

Code Listing 17: Activating Virtualenv

```
$ cd venv
$ source /bin/activate
```

3) When you type `which python3`, you should now see the path to the bin folder instead to the previous path (so it should now say `venv/bin/python3` or so). Now you can install packages using pip

Code Listing 18: Pip Install

```
$ pip3 install <package_name>
```

where of course you change the `<package_name>` with the actual name of the package you want to install.

⇒ **Windows**

If you have installed `Python` from the link provided at the beginning (`https://www.python.org/downloads/windows/`), then pip should already be installed on your computer. To check, type `pip --help` into the command prompt. If the command is not found, we need to install it manually from `https://pip.pypa.io/en/stable/`

installing/#do-i-need-to-install-pip. Save it on your desktop and navigate in the command prompt to your Desktop and execute the get-pip.py file:

Code Listing 19: Install Python

```
$ cd Dektop
$ python get-pip.py
```

1) Go to the working directory (e.g. Data_Reduction) where you want the virtual environment to be installed. Type:

Code Listing 20: Create Virtual Environment

```
$ virtualenv env
```

2) This has created a batchfile which has the extension ".bat". To activate the virtual environment, we run the script in the following way:

Code Listing 21: Activate Environment

```
$ C: \Users\'Username'\venv\Scripts\activate.bat
```

3) To install a new python package, you just have to type: `pip install <package>`

A.1.3 Anaconda

Anaconda is a simple package manager for Python. You can download it for all operating systems from https://www.anaconda.com. A very detailed installation guide can be found here: https://docs.anaconda.com/anaconda/install/. Here you also find a nice description of how to activate the virtual environment, written by better experts than me.

A.1.4 List of Required Python Packages

For the data reduction with Python requires the following packages. Install them using `pip3 install <package>` after you activated the virtual environment.

- **numpy**: This is the most basic package which is used in almost every script.
- **scipy**: Package with many helpful analysis tools.
- **Astropy**: The basic package needed when handling astronomical data.
- **photutils**: Package needed for the flux calibration.
- **astroscrappy**: Package that can deal with cosmic ray subtraction.
- **astroalign**: Package needed for astrometric correction.
- **pyfits**: Needed for Cosmics.py
- **Juypter Notebook (Optional)**: Nice interactive environment for Python.

A.2 DS9

SAOImage DS) is a software program that can be used to display FITS images. To download it, go to `http://ds9.si.edu/site/Download.html`. A very limited and simple guide is given in `http://ds9.si.edu/doc/user/gui/index.html`.

A.3 Source Extractor

Source Extractor is a software that catalogues sources from an image. Again, the installation differs from operating system to operating system.
⇒**MacOS**
The installation can be done using Homebrew or Macports. Do not forget to deactivate the virtual environment before you start the installation via the terminal. Using Homebrew, enter the following line:

Code Listing 22: Install Source Extractor

```
$ brew install homebrew/science/sextractor
or
$ brew install brewsci/science/sextractor
```

Because the packages change sometimes, you have to see which works for you. In case you want to use Macports, I assume you have already installed it. If not, go to `https://www.macports.org/install.php` and follow the instructions. Again, be cautious when using both package managers side by side. Then enter in the terminal:

Code Listing 23: MacPort install Source Extractor

```
$ sudo port install sextractor
```

⇒**Linux**
For Linux, simply type:

Code Listing 24: Linux install Source Extractor

```
$ apt-get sextractor (Debian)
$ dnf sextractor (Fedora)
```

(Chose the one according to your Linux system). Alternatively you can go to `https://sextractor.readthedocs.io/en/latest/Installing.html` and follow the description there.
⇒**Windows**
On Windows, you will have a hard time with Source Extractor. On Windows 10, it might work in the Unix terminal.

How to use Source Extractor:
An introduction of how to use Source Extractor can be found here:`https://sextractor.readthedocs.io/en/latest/Using.html`