2. QUERYING ASTRONOMICAL DATABASES USING *VO*−TOOLS

Astrolab Landessternwarte Königstuhl

Jochen Heidt

March 4, 2024

Abstract

The aim of this task is to get familiar with astronomical tools - mostly *Aladin* and *Topcat* - which can be used to query databases from different surveys as well as some data from single observations. These tools are an integrated part of the *Virtual Observatory (VO).* The aim of the VO is to allow the user easy access to astronomical data from archives. These can be either images or spectra from major observatories like HST or CHANDRA, huge databases e.g. from GAIA, or tables published in astronomical journals. The task is divided into 3 separate tutorials all of which use own science cases to explain the various applications of the programs.

1 Introduction

The growing amount of available astronomical data requires more and more space and new ways have to be found to access those data. Obviously this can not be solved by simply uploading all available databases as querying them for certain data would take days. So for all the data registers have to be written and the use of programming languages such as Structured Query Language (*SQL*) are inevitable in order to avoid an overload of the data servers. The *Virtual Observatory (VO)* tools offer access to those survey data without requiring any programming skills. There a large number of JAVA-applications available, all of which automatically connect to the Internet, are linked with each other, and all of which have different applications. The two most important of them - *Aladin* and *Topcat* - will be explored in detail in this task.

These tools are nowaday routinely used by professional astronomers, some of them (in particular Aladin) are an integral part of the instrument software on 8m class telescopes.

In this task, you will use the latest versions of Aladin (Aladin.jar, v12.060), Topcat (topcat-full.jar, v4.8-8) and Splat (splat-vo-3.15_1.jar, v3.15.1). Unless they are not already installed on your laptop, you can download them from the astrolab WEB at

https://www.lsw.uni-heidelberg.de/users/jheidt/praktikum/Astrolab_material/Task2.html

You can start the programs via:

java -jar Aladin.jar, **java -jar topcat-full.jar** and **java -jar splat-vo-3.15_1.jar**, respectively.

Note for MAC users: During the astrolab in Feburary 2024, some of the MAC users had issues with using SPLAT. Formally the jar-file should work on all platforms. See also `http://star-www.dur.ac.uk/~pdraper/splat/splat-vo/splat-vo.html`. You may need to download a MACOS version of TOPCAT at `https://www.star.bris.ac.uk/~mbt/topcat/`. The visual appearance between the TOPCAT GUI for Linux and MACOS apparently differs somewhat. For example, some tab/button names within TOPCAT mentioned in the script differ from those when using TOPCAT on MacOS. It is better to navigate within the software by looking for the corresponding icons rather than relying on the button names.

2 The tutorials

The three tutorials, which you will be using are entitled “Explore the Pleiades with Topcat and Aladin”, “Identifying brown dwarf candidates in 2MASS and SDSS” and “Exploring GAIA data with Topcat and STILTS”. These tutorials have been developed by astronomers actively working on the improvement of VO applications. In the following, a few words on each of them is given:

Explore the Pleiades with Topcat and Aladin: In this tutorial you will identify members of the Hyades kinematically - they are formed from the same matter and at the same time, will examine their color-magnitude diagram and will locate them on the sky as seen by 2MASS.
Identifying brown dwarf candidates in 2MASS and SDSS: Here you will try to find brown dwarf candidates by their colors. This will be achieved using a crosscorrelation of the SDSS and 2MASS sky surveys. In addition to the already known Topcat and Aladin tools, you will here learn how to use ADQL (Astronmical Data Query Language) to query TAP (Table Access Protocol) services. Eventually you will use Splat to inspect spectra of potential candidates.
NB: Eventually, you may need to replace 3./3600. in the box on page 7 of the tutorial by 88./3600. to find 8 objects.
Exploring GAIA data with Topcat and STILTS In the last tutorial you will learn how powerful GAIA is. Indeed since some years GAIA supersedes the HST (and the JWST) in terms of astronomical publications! The latest data release GAIA DR3 has more than 1.5 billion sources with excellent astrometry, photometry and (low-resolution) spectroscopy). Here you will explore the strong capabilities of Topcat in pretty much detail. You will first select a comoving cluster in the M4 region and examine their proper motion and determine their parallaxes, before you inspect the Hyades in 3-D. The 3-D examination of Hyades data is real fun! Of course, you can combine data from GAIA and the HST at the same time, this is done for the small cluster NGC 346 in subtopic three of this task. Ignore subtopic 4 of this tutorial. This is a redo of subtopic 1 on the M4 region using STILTS, a command-line language. This is somewhat boring and only for nerds loving this. Finish up this task rather with subtopic 5, where you will create a local HRD from GAIA data which will allow you to distinguish between different stellar populations hidden in it.

Explore the Pleiades with TOPCAT and Aladin

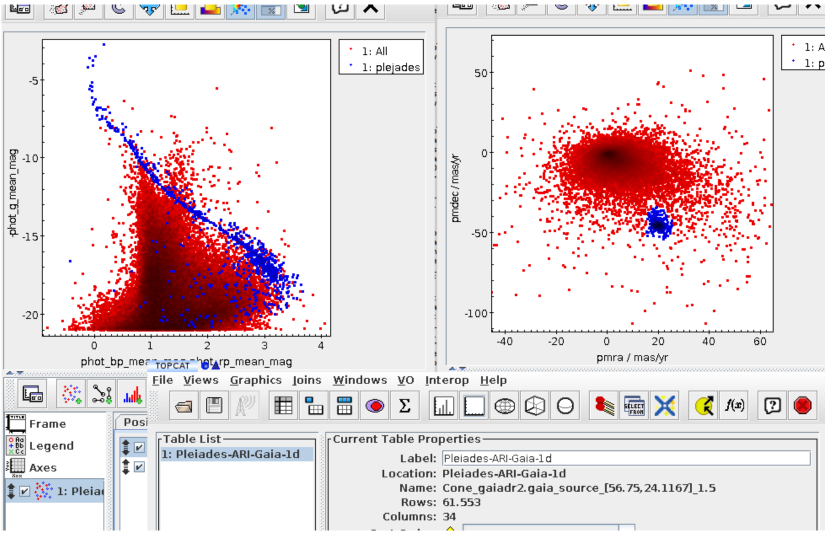


Table List

1: Pleiades-ARI-Gaia-1d

Current Table Properties

Label: Pleiades-ARI-Gaia-1d
Location: Pleiades-ARI-Gaia-1d
Name: Cone_gaiadr2.gaia_source_[56.75,24.1167]_1.5
Rows: 61.553
Columns: 34

Markus Demleitner and Hendrik Heinl

October 27, 2022

Abstract

As stars in open star clusters are formed from the same matter at the same time and have only a small velocity dispersion, it is relatively easy to identify them from the background stars. In this tutorial, we will identify the members of the Pleiades kinematically and get an idea of our success by examining a color-magnitude diagram using TOPCAT, Aladin, and a few VO services.

Software: TOPCAT Version 4.8, Aladin Version 11.0

1 Starting out

In this tutorial we use TOPCAT for data access and plotting and Aladin for image access. If you do not have both installed on your machine already, install them using the following links:
Aladin: `http://aladin.u-strasbg.fr/`
TOPCAT: `http://www.star.bris.ac.uk/~mbt/topcat/`

2 Accessing Gaia data and plotting

▷ **1** *Getting candidate members from Gaia* – To obtain a data set we need to find a catalog providing proper motions, positions and color filter bands. These days of course a good choice for this is the Gaia catalog, from which we will download a subset of objects in the region of the Pleiades.

For generic searches of the type "objects within x degrees of a position on the sky", the VO cone search protocol is straightforward. To use it, in TOPCAT click on VO → Cone Search. In the Cone Search window enter `gaia dr3` as Keywords and click on Find Services. This will perform a search over the VO registry for this keyword and the results will show up in the field below.

There are multiple instances of Gaia in the VO. Have a look in the service metadata window and see that next to different services providing full catalogues of different Gaia data releases, you also find services from Vizier. In principle you could use any of the services providing Gaia DR3 data and they should return the same result. In this tutorial we selected the on with the short name `DR3 lite Cone` – just do a lift click. In the field Object Name enter `Pleiades` and then click on Resolve right of it to determine the position around which to search (name resolution uses Sesame, a service of CDS) The full Pleiades cluster has a large coverage on the sky. To dodge some problems we might run into with resource limits on the service side, we will settle for the cluster core for now and use a Radius of `1.5` degrees. With OK, submit the query to the Cone Search service. After a few seconds, TOPCAT has loaded a few 10s of thousands of stars.

▷ **2** *Plotting proper motions* – The data set we obtained from the last step will not only contain members of the Pleiades, but also other objects. To define a subset with good candidates of members of the Pleiades, let us exploit the fact that all members of an open clusters share their proper motion (with a fairly small velocity dispersion) . This would be evident as an overdensity in a plot of the proper motion components.

Go to Graphics → Plane Plot. The plot controls are in the bottom part of the plotting window. As X axis, choose `pmra`, as Y axis choose `pmdec`. Zoom into the center for the point cloud a bit (e.g., by using your mouse wheel). You should fairly soon see an overdensity centered somewhere around 0.5 e-5, -1.5e-5 deg /yr.

▷ **3** *Plotting and drawing a subset* – This fairly certainly correspondes to the Pleiades. TOPCAT has the notion of *subsets*, i.e., parts of whole datasets defined in various ways. One way to define such a subset is by freehand selection, which is appropriate here.

To define a subset in this way, use Subsets → Draw Subset region. Now draw a region around the overdensity in PM space. When finished with that, go to Subsets → Finish Drawing region. In the window popping up in response, in New Subset Name enter something like "Pleiades candidates" (it doesn't really matter), and click on Add Subset. You will now see the subset in a different colour on the plot than the other objects.

In each plot of this dataset, you can now restrict display to only this subset in the Subsets tab in the lower part of the plot window. For this plot, that is somewhat pointless, but wait on.

▷ **4** *Defining a colour* – A plot in which the subset makes a lot of sense is a colour-magnitude diagram (CMD), where a colour, as is custom in astronomy, simply is the difference of magnitudes in two different bands.

There are several different ways you can introduce such colors (or other derived quantities) into TOPCAT tables. One is giving them directly in the plotting window. So, go back to the tab Position and as X: axis enter `phot_g_mean_mag-phot_rp_mean_mag` and as Y: axis enter `-phot_g_mean_mag`. Again go to the subsets tab in your plot and compare the CMDs of the whole dataset and of your Pleiades subset.

▷ **5** *Adding columns* – Sometimes derived quantities like colours can be reused in several places. In that case, you will want to add a column containing the results of an algebraic expression. To do that, in TOPCAT's main window go to views → Columns info. In the new window we click on Columns → New Synthetic Column. Here you can enter the parameters of our new column.

As an example, create a column called `g/rp`, defined by the Expression `phot_g_mean_mag-phot_rp_mean_mag`. With Index, control where the new column shall appear; use 3 here to follow this example. Click on OK to add the new column to the data. Return to TOPCAT's main window and go to Views → Table data to have a look at the data. In the third position you now can see the newly added column. Don't be confused that some data records don't have entries in this columns – that's simply because the same data records won't have entries at the columns jmag and hmag. Keep the table data window open because we will need it later.

▷ **6** *Some words about metadata* – In the previous steps we told you what to enter in the fields of the plotting window or when we added a column. But whenever you retrieve data, you want to take a look at the metadata hopefully provided in the same process. Within the VO, standards to provide and access metadata are crucial. Let's have a look at the metadata of our sample. In TOPCAT's main window go to Views → Column Info. Here you not only find the column names, but also descriptions, units and data formats. You can also edit the metadata by simply double clicking into the according field, which you should do right away for the column you just created.

3 SAMP

▷ 7 *Data exchange between VO tools* – One of the VO's creeds is that no one big, monolithic client program will serve all the VO user's needs. To support a situation in which many (relatively) small tools cooperate on behalf of the user, the VO defines SAMP (Simple Application Messaging Protocol), which allows exchanging tables, positions, and much more between different applications.

We will use it to send the table data from TOPCAT to Aladin. For this, start Aladin and in the Survey menu click on 2MASS. Then return to the TOPCAT main window. In the bottom right you see the SAMP status. In the Clients field you should see three symbols: one for the SAMP hub, one for TOPCAT and one for Aladin. This means that these applications are ready to receive messages via the SAMP interface. We want to use this to sent our Pleiades subset to Aladin.

A little way above the SAMP status row you will find Row Subset. Click there, and in the menu popping up select pleiades candidates (or whatever name you set for the subset). Then go to Interop → Send table to → Aladin. The subset will be loaded in Aladin now. For further funcionality we need to activate SAMP Actions in TOPCAT. This we do at Views → Activation Actions. In the new opened window you can configure TOPCAT's SAMP Actions. Here you need to check *Send Sky Coordinates*. You may notice that TOPCAT already guessed which table rows contain the coordinates. This is possible due to the provided metadata (the buzzword here is *UCDs*).

Now go to Aladin and you will see the objects displayed on the 2MASS images. You can zoom in, and out with your mous wheel. Select some of the objects and see that those objects are highlighted in TOPCAT's table data window.

Use this feature to at least look at some of the outliers in the Pleiades' CMD. Perhaps some of them are actually binary stars?

3.1 Going on

Can you determine the parallax of the Pleiades according to Gaia? To get some perspective on this, check out Melis et al (2014).

4 References

Taylor, M. http://www.star.bris.ac.uk/~mbt/topcat/#docs

Melis, C., Reid, M. J., Mioduszewski, A. J., Stauffer, J. R., & Bower, G. C. 2014, Science, 345, 1029 2014Sci...345.1029M.

Identifying Brown Dwarf candidates in 2MASS and SDSS

Hendrik Heinl and Markus Demleitner

September 19, 2023

Abstract

Brown dwarfs are faint objects with a mass below 0.8 solar masses. In their stellar core, hydrogen fusion never starts, and therefore their luminosity is very low. Even those brown dwarfs which are close to the solar system are faint objects, hard to detect and to identify as such. One method to identify brown dwarf candidates is using surveys as 2MASS and SDSS which provide colour photometry. In this use case we will identify brown dwarf candidates by applying the method introduced by Zhang et.al. (2010). We use the VO tools TOPCAT and Aladin, and especially ADQL to access data provided by TAP services for 2MASS and SDSS. We crossmatch our results with SIMBAD to weed out known brown dwarfs. Eventually we run a global discovery (using the VO protocol SSAP) for a spectrum for one of the objects not hitherto in SIMBAD.

Software: TOPCAT Version 4.8-8, Aladin Version 12.0, SPLAT, TAP, ADQL

1 Starting out

We will use the VO tools TOPCAT, Aladin and Splat-VO. These software packages be found here (or perhaps in the repositories of your distribution):

Aladin: `http://aladin.u-strasbg.fr/`
TOPCAT: `http://www.star.bris.ac.uk/~mbt/topcat/`
Splat-VO: `http://www.g-vo.org/pmwiki/About/SPLAT`

2 ADQL and TAP services

ADQL (Astronomical Data Query Language) is the language used in the VO to query remote TAP services. It is based on SQL and can be learned easily. We will use ADQL to post queries to different TAP services and especially to let the remote services do some of the work, so we receive only the data we are interested in and we do not have to download whole catalogs.

To use TAP (Table Access Protocol) services, one needs a piece of software (a “client”) speaking that protocol. In this use case we use TOPCAT as a TAP client.

Be aware that this is not an introduction to ADQL. Though the steps will be understandable without any ADQL skills, we recommend to have a longer look the GAVO ADQL course at `http://docs.g-vo.org/adql/html/` .

▷ **1** *Searching TAP services in the VO registry* – A good starting point for our search for brown dwarfs is 2MASS, because it provides astrometry as well as magnitudes in infrared filter bands. To find an appropriate TAP service for 2MASS (or really, anything in the VO), there is the VO Registry. TOPCAT has interfaces to that Registry, so you don’t have to to bother with the details. Just go to VO → Table Access Protocol. In the TAP Query window at Keywords, enter `2MASS`. In the line below, check Description (you may need to adjust the window width to see all options). Then click on Submit Query (that’s querying the Registry behind the scenes).

In the pane below the search field you can now see a list of TAP services related to 2MASS. When searching for services, you may need to spend some time to find the one best suited for your needs. In our case, we already have a good guess that the GAVO Data Center TAP service will be the right choice because it provides us with more data we will need in a later step. So in this list, we mark the line by clicking once. Then click Use Service at the bottom of the window.

Now we have to select the table we want to query. On the left side you see all tables which the GAVO data center TAP service provides. You can either scroll down until you find twomass.data, or instead use the search field above the list. Note that you get a first glance of the metadata of this table in the field on the right. Here you find a good overview when you mine for data. We will take a closer look at metadata a few steps ahead.

▷ **2** *Submitting a first ADQL Query: cone selection* – Within the VO the term *cone search* references to queries for “things” in an area of the sky defined by a position and a radius around it. You may have performed cone searches already with TOPCAT using a dedicated VO protocol for that. Here is a different approach using TAP and ADQL to give you an idea how these work. Luckily TOPCAT comes with examples for the most common ADQL queries – plus sometimes queries the TAP service mantainers assume to be common or at least useful.

In the lower window at ADQL Text click on Examples → Cone Selection. You see an example ADQL query appearing in the field:

```
SELECT TOP 1000
       *
FROM twomass.data
WHERE DISTANCE(raj2000, dej2000, 126.248, 1.02) < 0.05
```

Let's go through it step by step. Each ADQL query starts with SELECT followed by specifications of "what" to select, what to do with the selected records and how to return the results. TOP 1000 means no more than 1000 data records will be returned, regardless of how many actually match. Note that this limit does not depend on any order, though you might think that "top" implies that there might be a "bottom" too. The database will just return the first 1000 results coming in.

With * we select all columns of matching data records. If you just want certain columns from a table to be returned, you could specify these here. We will see in a later step how to do this. FROM twomass.data specifies the table of the TAP service we want to query. The next lines are the query conditions. In our example we use the ADQL built in functions that define a cone search. The WHERE clause extracts those data records that match. In our case these criteria shall be sources in a cone around a certain position.

The DISTANCE function computes the distance between two geometries, but in case of two points, it just takes four arguments: ra and dec of each. Eventually we add the maximum distance of 0.05 degrees.

Clicking OK will start the cone search and within a few seconds we retrieve the first 1000 data records. Return to the TOPCAT main window and play around with this data (like, do some plots) if you like to.

▷ **3** *Metadata* – As mentioned above you can have a glance at the metadata of a TAP service in TOPCATs TAP window. For our hunt of brown dwarfs, we basically need colour indexes and therefore search the broadband photometry. In the TAP window we click on Table → Table columns. Here we have an overview of the table metadata and get the names of the columns which we need later to specify our ADQL query. We are especially interested in J, H and K magnitudes, which we see are named hmag, jmag and kmag. Knowing the column names, we now can modify the ADQL query accordingly.

▷ **4** *Searching 2MASS with ADQL* – Now we want to search 2MASS for those objects, which are good candidates for brown dwarfs. Because brown dwarfs are faint objects, we estimate they have a J magnitude of 15.3 or higher (yes, this is a bit of a didactic cheat – accept it for now).

Also, brown dwarfs are very red sources. Hence, we want to search for sources with a colour index of

$$jmag - kmag > 0.8.$$

Running the query over the whole 2MASS catalog would take a long time so for thie exercise we don't search over the whole sky but limit the search to a 2 degrees cone instead. Finally, we change the search cone coordinates.

All this we can do with ADQL by modifying the query as following:

```
SELECT *
FROM twomass.data
WHERE DISTANCE(raj2000, dej2000,127.0000, 1.2000) < 2.0
AND jmag > 15.3
AND jmag-kmag > 0.8
```

Before we submit the query, we have to set a maximum so we receive all the matching data records. To do so look at Service Capabilities → Max Rows and select `max` in the drop down menu. We sent the query by clicking on OK and the result should be returned in a few seconds.

▷ **5** *Crossmatching with SDSS* – With the J, H, and K broadband magnitudes from 2MASS we found first candidates for brown dwarfs. To further week out non-brown dwarfs we now apply criteria derived from Zhang et al. (2010). Therefore, we need magnitudes in SDSS i, r, and z bands. In the Topcat TAP window we now enter sdss, again check Description and submit the query. While it doesn't really matter, for this exercise choose GAVO's service again.

We could now do a similar search as for 2MASS and then compare the two tables locally. But we rather use a more elegant method and let the remote TAP service do some work for us. As long as both tables are on the very same service, we can merge the two queries into a single one and obtain the result at once; when that is not the case, on most services you can upload local tables and use them remotely for the duration of your query (we will see later how that works).

You will have noted that we start with `WITH tm AS` This is called Common Table Expressions (CTE) which makes it bit easier to understand complex queries, and also gives you a bit of control over the order at which the database will perform the parts of the queries. We also use several `SELECT` statements, which may look a bit confusing, but it helps structuring the queries: keep in mind that the result of a `SELECT`-statement is always a table and we can perform all table related functions on this table. You can see this in the part of the cone selection on the table twomass.data. Using the alias `AS tm` we can refer to this result when we join it with the table sdssdr16.main. Also note that we use CTEs for defining two tables. The actual CTE wrapper followed by a SELECT statement is quite simple:

```
WITH
  tm AS
    ( SELECT [...] ),
  sdss AS
    ( SELECT [...] )

SELECT [...]
```

It may help to apply natural language here, then it’s obvious that this is defining a list named tables, which can be referred to. We can also use one of these results to perform select statements on them for a different table. And that’s the big advantage: by defining these ”dependencies” we force the database to perform parts of the query in a specific order. Please be aware that with such power comes responsibility: in the one case this may help you to significantly improve the performance, but in another case you may mess it up. It’s strongly recommended to have a look into the ADQL course (linked below) to get a better understanding of how ADQL works.

To link the rows from the two tables (2MASS and SDSS), we use the JOIN... ON construct in ADQL to merge the data. We use the DISTANCE function to compute the distance between our selected points and the the points in 2MASS and also in the x-match with SDSS. We keep the distance very small at 1”.

Note that we moved the conditions of brightness and the colour cuts to another SELECT-statement. Thus, we tell the databse explicitly to first perform the cone selection in 2MASS, then x-match with SDSS, and in the last step check the colours in sdss. The reason for this is performance: if we put all of the query in a single SELECT-statement, the database will have a guess on which solution might be the fastest way and it concludes that starting with the SDSS colour cuts seems the best approach – and of course this computation over more than 500 million sources will take much longer time than performing the cone selection first on 2MASS first, then perform the x-match with SDSS and compute the colours last on much fewer sources. Take another look at the query and you will notice how we use CTEs to force the database to recognize this order.

Eventually in the SELECT statement outside the CTE we perform the colour cut and make a selection on the columns we want and use AS to rename the columns in a way that we see from which catalougue they origin.

The criteria we use to perform the SDSS colour cuts are taken from Zhang et al. This is the full query:

```
WITH tm AS
  ( SELECT TOP 20000 * FROM twomass.data
     WHERE
       DISTANCE ( raj2000, dej2000, 127.0000, 1.2000) < 2.0
       AND jmag > 15.3
       AND jmag-kmag > 0.8
  ),

tmXsd AS
  ( SELECT TOP 500000 *
    FROM tm
    JOIN sdssdr16.main as sdss
```

```
      ON DISTANCE ( sdss.ra, sdss.dec,
                    tm.raj2000, tm.dej2000)< 1/3600.
  )

SELECT
  obj_id AS sdss_id,
  mainid AS tm_id,

  ra AS sdss_ra,
  dec AS sdss_dec,
  raj2000 AS tm_raj2000,
  dej2000 AS tm_dej2000,

  jmag AS tm_jmag,
  hmag AS tm_hmag,
  kmag AS tm_kmag,
  u AS sdss_u,
  g AS sdss_g,
  r AS sdss_r,
  i AS sdss_i,
  z AS sdss_z

FROM tmXsd
WHERE i - z BETWEEN 1.5 AND 2
  AND r - i BETWEEN 1.5 AND 4.5
```

After submitting the query we receive a table of hopefully good candidates for brown dwarfs, so let's take a look at the data. You can immediately see that for some 2MASS identifiers there are two records in our result set. This is due to the TAP search which returned all matches around a source from the 2MASS data, and not the single best match.

To see what is behind these duplications, look at how the objects look like on the sky. Thankfully the VO provides us with the perfect tool for that: Aladin.

▷ **6** *Resolving suspicious results with Aladin and TOPCAT* – In this step we want to see the images in 2MASS and SDSS of our brown dwarf candidates, with a particular view to the duplicated SDSS counterparts for 2MASS objects. For this we use the SAMP protocol to send data from TOPCAT to Aladin. We start Aladin and select 2MASS as a background. Zoom in to a field of view of an arcminute or so.

Then we go back to the TOPCAT main window and check the table with our candidates from the 2MASS-SDSS x-match. We use SAMP to send the hole table to Aladin: Interop → Send table to → Aladin. We then click in Current Table Properties → Activation Action → Transmit Coordinates. That tells

TOPCAT to tell all other SAMP clients the sky position of a row (or a point in a plot) when you click on it. Aladin, for instance will then automatically jump to that position, showing the vicinity of that position.

Still in Topcat we open the table view and click on any table row. In Aladin we can see the position change. If we compare the images from our duplicated data records, we see that the sources point to the same object in the catalog. We can switch between 2MASS and SDSS images in Aladin and see the same effect. This is simply because we received all matching data records from our crossmatch with SDSS and in the SDSS catalog the different observations to a single object are not merged (probably – at least one object might actually be a bona fide double star). Anyway, let us clean up our little table so the duplicate matches go away.

We solve this in TOPCAT. In the main window we mark the table and click on Joins → Internal match. In the new window at Algorithm select Exact Value, and at Table we chose our current table and as Matched Value column select tm_id. Eventually as Action check Eliminate All But First of Each Group. Of course you should think twice before blindly throwing away records in a real science project. For the current discovery purpose, it's just fine since we don't lose any positions. Now we reduced our brown dwarfs candidates table to 11 rows and it's about time to check the accuracy of our method – and if perhaps we've discovered new brown dwarfs.

▷ **7** *TAP upload to SIMBAD* – Now we want to know how many of our candidates are already confirmed brown dwarfs. A good location to check for this of course is SIMBAD. We again go to the TOPCAT registry search window an now search for Simbad. From the few options listed we select the Simbad service.

We select the table `public.basic`. This time we don't want to get all data from SIMBAD but only the column that contains the object type. Checking the service metadata we find out that we will need the column otype_txt. The special feature of the query we want to perform is the TAP upload though. TAP lets you upload local data and treat it the same way as any other table on the remote service. We will use this to perform a crossmatch between our brown dwarf candidates and the SIMBAD database.

In the topcat TAP window click on Examples → Upload → Upload Join. This will generate a good base for the query, which we will have to adjust a little.

You may notice that the `JOIN` command is different to the one we used above: instead of referring to a table name, it's using `TAP_UPLOAD.tX`, where X will be a number in your case. This is simply the table number from the topcat main window and thus the specific number depends on what you did during the current topcat session. Adjust it so it points to the right table number, which it likely already does. The other adjustment is concerning the `JOINT` command: here we put a `RIGHT OUTER` in front. This is due to the behaviour of `JOINT`: a normal ("inner") JOIN will only return records with matches in both tables. In this case you would only receive rows *with* matches in Simbad.

But it this step we actually are interested in keeping those of our local records that do not match records in SIMBAD: They could be good candidates for further research, because apparently they have not been identified as brown dwarfs in the published literature yet. RIGH OUTER JOIN roughly translates to "Join table1 with table2 where the following conditions are met. Where the conditions are not met, fill any columns from table1 with NULLs." Analogously there is the construct LEFT OUTER JOIN implemented in ADQL.

Make sure your query looks like this (don't forget to adjust the table number!):

```
SELECT TOP 1000
       tc.*, db.otype_txt
FROM basic AS db
RIGHT OUTER JOIN TAP_UPLOAD.t1 AS tc
  ON 1=CONTAINS(POINT('ICRS', db.ra, db.dec),
                CIRCLE('ICRS', tc.sdss_ra, tc.sdss_dec, 5./3600.)
                    )
```

Now let's look at our output table: we have an additional column otype_txt in which we find the object type. You see that some of our objects are listed in SIMBAD as brown dwarfs (BD*). The other objects may be candidates worth further research, especially the one that seemed a candidate for a double star system.

▷ 8 *Obtaining spectra with Splat-VO and SSAP* –

We now want to use Splat-VO and the Simple Spectrum Access Protocol (SSAP) to search for spectra that might give us further hints of what the object might be. We open Splat and in there open the SSAP window: File → SSAP_. In the left part of the SSAP window we see the server selection. Here- one can preselect services according to the spectra you are interested in. For us, it's fine to not further constrain the waveband and to check all services by clicking on select all.

As search parameters we will give the position of one of our candidates from the topcat table: RA: 126.918696, Dec -0.146868 for the position and 1 arcminute as Radius. We submit the query by clicking on the green button SEND QUERY. Splat-VO now searches on all SSAP services that we selected for spectra within the range of 1 arcminute of our given position.

After a few moments we will receive results from three services and we can select and download spectra. Thanks to the metadata provided by the services we can figure out which spectra are of interest for us. Unfortunately, this is not always the case: please complain to the operators (you can find contact information in the registry) if you find missing metadata – every bug fixed improves the VO for those using it after you.

3 References

Demleitner, M. http://docs.g-vo.org/adql/html/

Fernique, P. http://aladin.u-strasbg.fr/java/AladinManual6.pdf

Ortiz, I., Lusted, J., Dowler, P. et al. http://www.ivoa.net/documents/REC/ADQL/ADQL-20081030.pdf

Taylor, M. http://www.star.bris.ac.uk/~mbt/topcat/#docs

Zhang, Z. H., Pinfield, D. J., Day-Jones, A. C., et al. 2010, MNRAS, 404, 1817, 2010MNRAS.404.1817Z

Tutorial: Exploring Gaia data with TOPCAT and STILTS

Mark Taylor,
University of Bristol,
m.b.taylor@bristol.ac.uk

TOPCAT: `http://www.starlink.ac.uk/topcat/` *(version 4.7 or later recommended)*

STILTS: `http://www.starlink.ac.uk/stilts/` *(version 3.2 or later recommended)*

Mailing list: `topcat-user@jiscmail.ac.uk`

Version: `2021-02-03    Revision 4057778 (https://github.com/mbtaylor/tctuto)`

Contents

1 Cluster identification #1: Messier 4 in proper motion space 2
1.1 Acquire Gaia data in the M4 region . . . 2
1.2 Identify comoving cluster . . . 2
1.3 Manage Subsets . . . 3
1.4 Examine cluster members . . . 3
1.5 Determine parallax . . . 3

2 Cluster identification #2: Hyades in 3-D velocity space 4
2.1 Locate Gaia TAP service . . . 4
2.2 Explore the TAP service . . . 4
2.3 Acquire astrometric data . . . 4
2.4 Calculate 3-d velocity components . . . 5
2.5 Identify Hyades graphically in 3-d velocity space . . . 5
2.6 View positions . . . 6
2.7 Colour-magnitude diagram . . . 6

3 Match Gaia and HST observations 7
3.1 Acquire HST observations . . . 7
3.2 Crossmatch with Gaia . . . 7
3.3 Visualise the crossmatch . . . 8
3.4 Investigate and identify matches . . . 8
3.5 Alternative crossmatch: use local files . . . 9

4 Messier 4 in proper motion space using STILTS 10
4.1 Get STILTS running . . . 10
4.2 Acquire data from Cone Search service . . . 10
4.3 Manipulate the downloaded table using `tpipe` . . . 10
4.4 Plot using STILTS . . . 11

5 Local Herzsprung-Russell Diagram 12
5.1 Acquire data from TAP service . . . 12
5.2 Plot HRD . . . 12
5.3 Exclude astrometrically suspect sources . . . 13
5.4 Exclude photometrically suspect sources . . . 13
5.5 Explore the HRD . . . 13

This tutorial uses data from Gaia Early Data Release 3 (EDR3) [1] to lead you through some of the capabilities of TOPCAT and STILTS. For best results, you should have the manuals to hand: `http://www.starlink.ac.uk/topcat/sun253/` and `http://www.starlink.ac.uk/stilts/sun256/`.

1 Cluster identification #1: Messier 4 in proper motion space

In this example we will determine the mean parallax of the stars in the globular cluster Messier 4 (M4, or NGC 6121).

1.1 Acquire Gaia data in the M4 region

1. Start TOPCAT.

2. Open the **VO|Cone Search** window (i.e. use the **Cone Search** submenu of the **VO** menu in the main topcat window).

3. Fill in **Keywords**: "`gaia edr3`", and hit **Find Services**.

4. There are a few options, that should mostly give similar results. The one with Short Name "`eDR3 lite Cone`" is a good choice. Select it by clicking on its row, and the partial URL of the service appears in the **Cone URL** field.

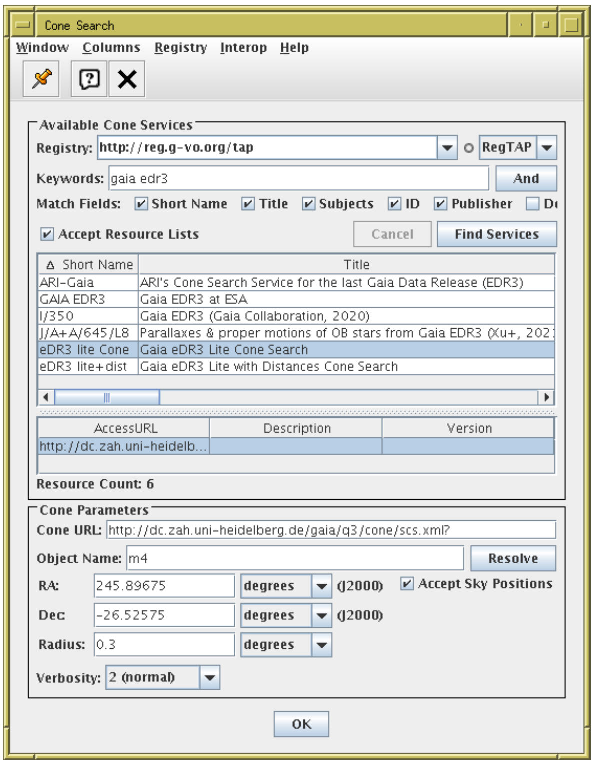
5. Fill in **Object Name**: "`M4`", then hit **Resolve** to fill in sky position fields.

6. **Radius**: "`0.3`" (degrees)

7. Hit **OK**; new table is loaded into topcat main control window with about 50 000 rows. If the download is too slow, cancel and try with **Radius**: "`0.1`", which retrieves about 20 000 rows, and maybe **Verbosity**: "`1`", which reduces the number of columns requested.

8. Use the **Graphics|Sky Plot** window to see the positions on the sky.

9. Play with the plot. Note overdense regions are coloured darker. Use the options in the **Form** tab to change marker size, colour and shading. Practice navigation: use mouse drag and wheel (or CTRL-drag). Click the little ? button at bottom left for navigation help; note navigation details are different for different plot types.



Cone Search

Window Columns Registry Interop Help

Available Cone Services

Registry: http://reg.g-vo.org/tap ○ RegTAP

Keywords: gaia edr3 And

Match Fields: ☑ Short Name ☑ Title ☑ Subjects ☑ ID ☑ Publisher ☐ De

☑ Accept Resource Lists Cancel Find Services

Short Name	Title
ARI-Gaia	ARI's Cone Search Service for the last Gaia Data Release (EDR3)
GAIA EDR3	Gaia EDR3 at ESA
I/350	Gaia EDR3 (Gaia Collaboration, 2020)
J/A+A/645/L8	Parallaxes & proper motions of OB stars from Gaia EDR3 (Xu+, 202
eDR3 lite Cone	Gaia eDR3 Lite Cone Search
eDR3 lite+dist	Gaia eDR3 Lite with Distances Cone Search

AccessURL	Description	Version
http://dc.zah.uni-heidelb...		

Resource Count: 6

Cone Parameters

Cone URL: http://dc.zah.uni-heidelberg.de/gaia/q3/cone/scs.xml?

Object Name: m4 Resolve

RA: 245.89675 degrees (J2000) ☑ Accept Sky Positions

Dec: -26.52575 degrees (J2000)

Radius: 0.3 degrees

Verbosity: 2 (normal)

OK

1.2 Identify comoving cluster

1. Plot sources in proper motion space:
   Graphics|Plane Plot window,
   X: "`pmra`", **Y**: "`pmdec`"

2. Note overdensity far from (0,0); use mouse to navigate

3. Graphically select this comoving cluster as new Subset:
   Subsets|Draw Subset Region button,
   drag mouse around cluster, hit button again

4. A **New Subset** dialogue pops up: fill in **New Subset Name**: "`comoving`", **Add Subset**

5. Look in **Subsets** tab of plot window; turn **All** and **comoving** subsets off/on

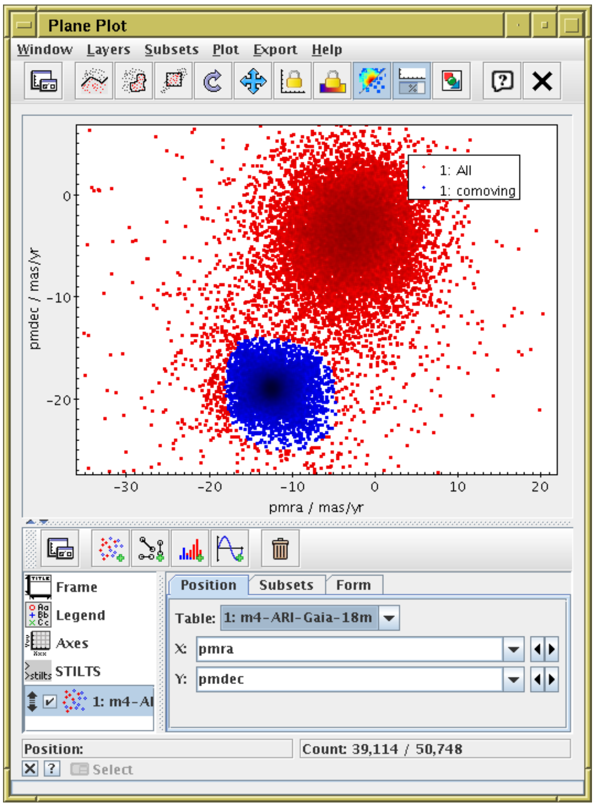


Table: 1: m4-ARI-Gaia-18m

X: pmra

Y: pmdec

Position:

Count: 39,114 / 50,748

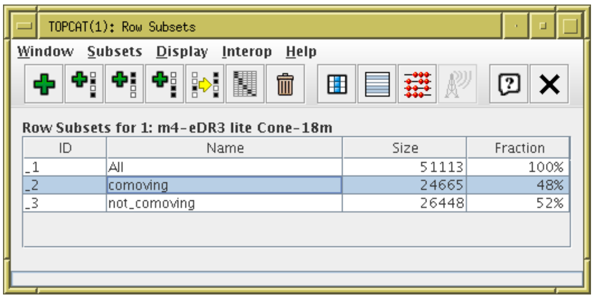
1.3 Manage Subsets

1. Open the **Views|Row Subsets** window

2. See the new **comoving** subset

3. Create an additional subset that contains the background objects, i.e. those not in the comoving set: click to select the **comoving** row, then select the **Subsets|Invert Subset** action.

4. A new subset **not_comoving** will appear; if you like you can rename it "**background**" by double-clicking in the **Name** field



TOPCAT(1): Row Subsets

Row Subsets for 1: m4-eDR3 lite Cone-18m

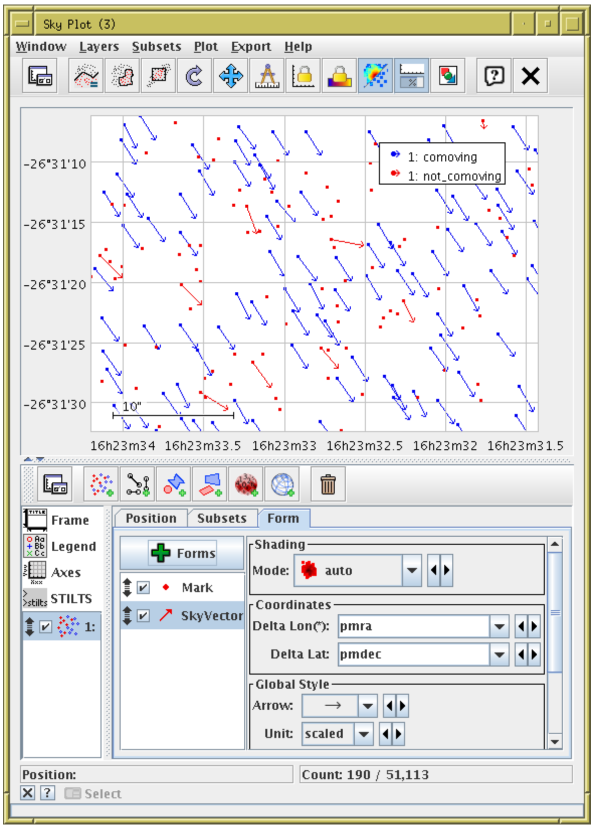
ID	Name	Size	Fraction
_1	All	51113	100%
_2	comoving	24665	48%
_3	not_comoving	26448	52%

1.4 Examine cluster members

1. Go back to the **Graphics|Sky Plot** from section 1.1 (or open a new one)

2. In **Subsets** tab turn different subsets on/off using checkboxes

3. Plot the proper motions. In the **Form** tab, use the **Forms** menu and select the **Add SkyVector** item, with **Delta Lon(*)**: "`pmra`", **Delta Lat**: "`pmdec`". The arrows will initially be much too long (units of degrees); you will have to set **Unit**: "`scaled`" (auto-scaling), and optionally adjust to taste with the **Scale** slider. Zoom in so you can see some individual objects. All the cluster objects have similar proper motions, non-cluster ones have various directions, or none (no measured P.M.).



Sky Plot (3)

Shading
Mode: auto

Coordinates
Delta Lon(*): pmra
Delta Lat: pmdec

Global Style
Arrow: →
Unit: scaled

Position:
Count: 190 / 51,113

1.5 Determine parallax

1. Plot histogram of parallaxes:
Graphics|Histogram Plot,
X: "`parallax`"

2. In **Subsets** tab, make sure only subsets **comoving** and **not_comoving**, not **All**, are plotted

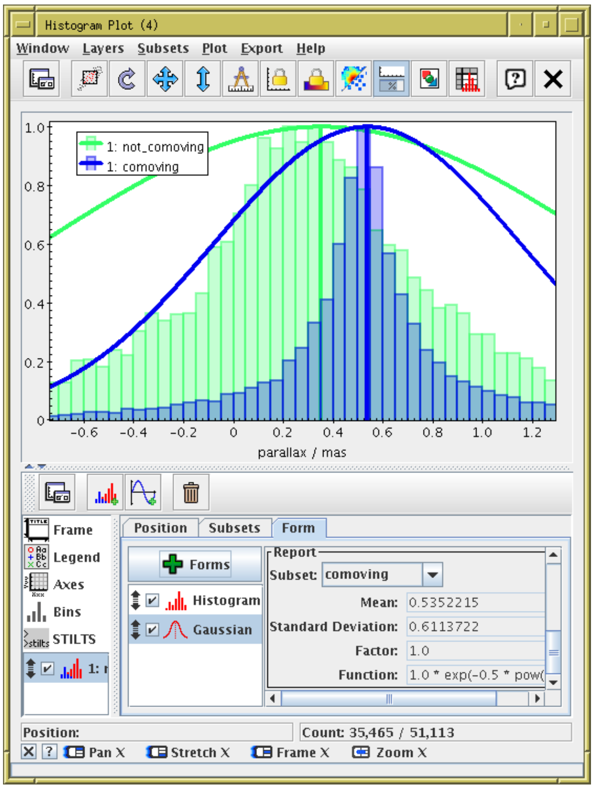
3. Normalise histograms to the same height: click on the **Bins** control in the left-hand panel, select the **General** tab, and set **Normalise** to **maximum**. Return to **histogram layer** control

4. Zoom in horizontally (mouse below X axis, use drag and wheel/CTRL-drag) to read off the comoving peak/average parallax. Note the comoving (cluster) distribution is different from the background sample.

5. For a more accurate result, fit Gaussian to data: **Forms** menu in **Form** tab, **Gaussian** option. Then scroll to bottom of Gaussian layer description in **Form** tab, select **Subset**: "`comoving`", and read off parallax **Mean** and **S.D.**

6. Invert mean parallax get distance to cluster (1000/parallax in mas = distance in parsec).

7. To do this without plotting, you can read off the mean and S.D. for parallax in the Σ **Views|Column Statistics** window for the **comoving** subset.



Histogram Plot (4)

Report
Subset: comoving
Mean: 0.5352215
Standard Deviation: 0.6113722
Factor: 1.0
Function: 1.0 * exp(-0.5 * pow(

Position:
Count: 35,465 / 51,113

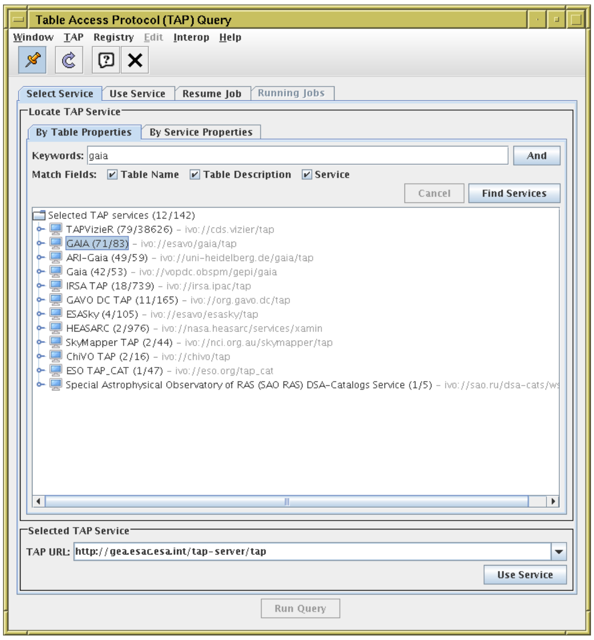
Note: careful when inverting parallaxes!
In general $r = 1/\varpi$ is *not reliable* because of errors. It's OK here because we are averaging over many measurements with smallish errors. Rule of thumb for single measurements: if $\varpi/\sigma_\varpi > 5$ it's probably OK. See Luri et al. 2018 [2] for full discussion.

2 Cluster identification #2: Hyades in 3-D velocity space

This example locates the Hyades open cluster in 3-dimensional velocity space, using Gaia's proper motion and radial velocity observations. We can't start this time by making a positional query (cone search), since the Hyades is very delocalised on the sky, because it's so close, so a cone would contain way too many sources. So we need to make a more sophisticated query using TAP.

2.1 Locate Gaia TAP service

1. Open the TAP window:
   VO|Table Access Protocol (TAP) Query
2. Fill in **Keywords**: "gaia" and hit **Find Services** button
3. There are several services with Gaia data in various forms; **GAIA** (ESA) or **ARI-Gaia** (Heidelberg) are good choices. The service URL appears in the field at the bottom of the window.
4. Hit the **Use Service** button at the bottom



Keywords: gaia

Match Fields: ☑ Table Name ☑ Table Description ☑ Service

Selected TAP services (12/142)
- TAPVizieR (79/38626) – ivo://cds.vizier/tap
- GAIA (71/83) – ivo://esavo/gaia/tap
- ARI-Gaia (49/59) – ivo://uni-heidelberg.de/gaia/tap
- Gaia (42/53) – ivo://vopdc.obspm/gepi/gaia
- IRSA TAP (18/739) – ivo://irsa.ipac/tap
- GAVO DC TAP (11/165) – ivo://org.gavo.dc/tap
- ESASky (4/105) – ivo://esavo/esasky/tap
- HEASARC (2/976) – ivo://nasa.heasarc/services/xamin
- SkyMapper TAP (2/44) – ivo://nci.org.au/skymapper/tap
- ChiVO TAP (2/16) – ivo://chivo/tap
- ESO TAP_CAT (1/47) – ivo://eso.org/tap_cat
- Special Astrophysical Observatory of RAS (SAO RAS) DSA-Catalogs Service (1/5) – ivo://sao.ru/dsa-cats/ws

TAP URL: http://gea.esac.esa.int/tap-server/tap

2.2 Explore the TAP service

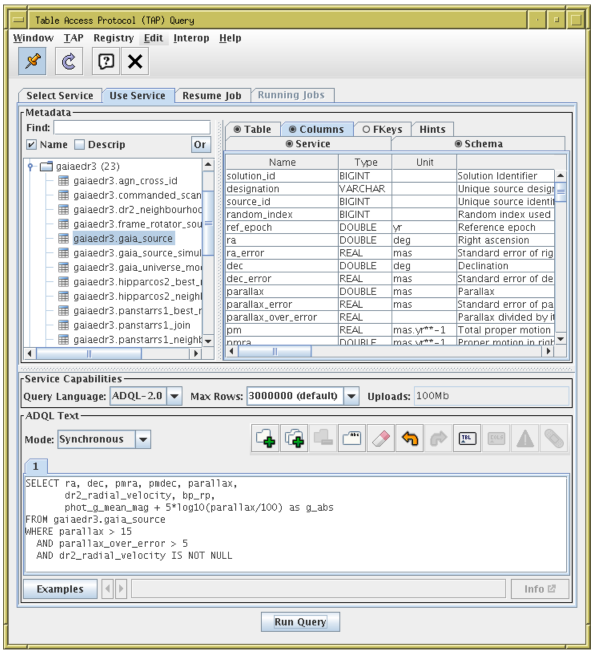
Use the TAP window to explore the tables that are present and their metadata.

1. Browse the table list on the left, The tables in the `gaiaedr3` schema are the ones with Gaia EDR3 data.
2. Select table `gaiaedr3.gaia_source` and look at **Table** and **Columns** tabs, to see information about available columns.
3. Look in the **Service** tab to see information about the service
4. Look in the **Hints** tab for a very basic ADQL cheat sheet
5. Type in to the bottom panel some very simple ADQL:
   "`SELECT TOP 10 ra, dec FROM gaiaedr3.gaia_source`"
6. Note that syntax errors (including partial or misspelt tables/columns) are highlighted in red.
7. Hit **Run Query** to run the query; if successful a new table is loaded

2.3 Acquire astrometric data

In the TAP window, execute this query:

```
SELECT ra, dec, pmra, pmdec, parallax,
       dr2_radial_velocity, bp_rp,
       phot_g_mean_mag + 5*log10(parallax/100) as g_abs
FROM gaiaedr3.gaia_source
WHERE parallax > 15
AND parallax_over_error > 5
AND dr2_radial_velocity IS NOT NULL
```



☑ Name ☐ Descrip

gaiaedr3 (23)
- gaiaedr3.agn_cross_id
- gaiaedr3.commanded_scan
- gaiaedr3.dr2_neighbourhoo
- gaiaedr3.frame_rotator_sou
- gaiaedr3.gaia_source
- gaiaedr3.gaia_source_simul
- gaiaedr3.gaia_universe_mo
- gaiaedr3.hipparcos2_best_
- gaiaedr3.hipparcos2_neighl
- gaiaedr3.panstarrs1_best_r
- gaiaedr3.panstarrs1_join
- gaiaedr3.panstarrs1_neight

Name	Type	Unit	
solution_id	BIGINT		Solution Identifier
designation	VARCHAR		Unique source desigr
source_id	BIGINT		Unique source identi
random_index	BIGINT		Random index used
ref_epoch	DOUBLE	yr	Reference epoch
ra	DOUBLE	deg	Right ascension
ra_error	REAL	mas	Standard error of rig
dec	DOUBLE	deg	Declination
dec_error	REAL	mas	Standard error of de
parallax	DOUBLE	mas	Parallax
parallax_error	REAL	mas	Standard error of pa
parallax_over_error	REAL		Parallax divided by i
pm	REAL	mas.yr**-1	Total proper motion
pmra	DOUBLE	mas.yr**-1	Proper motion in righ

Query Language: ADQL-2.0 Max Rows: 3000000 (default) Uploads: 100Mb

Mode: Synchronous

```
SELECT ra, dec, pmra, pmdec, parallax,
       dr2_radial_velocity, bp_rp,
       phot_g_mean_mag + 5*log10(parallax/100) as g_abs
FROM gaiaedr3.gaia_source
WHERE parallax > 15
  AND parallax_over_error > 5
  AND dr2_radial_velocity IS NOT NULL
```

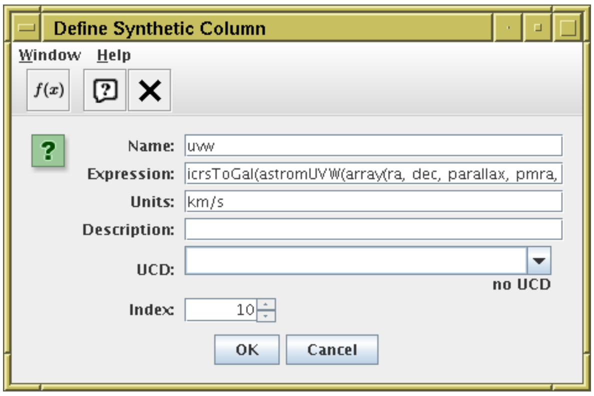
The result should contain about 25 000 rows.

The query is for all the nearby sources (nominally within 1000/15 ≈ 66 parsec) with observed radial velocities (only about 7 million DR2/EDR3 sources have RV) and good determinations of parallax. The fact that parallax error is ≤20% means that it's OK to invert parallax to calculate distance and absolute magnitude. We are retrieving all the basic astrometric parameters and some photometry.

The Hyades should be in there somewhere. Can we find them?

2.4 Calculate 3-d velocity components

We have the astrometric quantities measured by Gaia, which contain full phase space information, but need transformations to yield Cartesian position/velocity coordinates. TOPCAT's *expression language* can help.



Name: uvw
Expression: icrsToGal(astromUVW(array(ra, dec, parallax, pmra,
Units: km/s
Description:
UCD:
no UCD
Index: 10
OK
Cancel

1. Open the f(x) **Help|Available Functions** browser to see what functions TOPCAT provides (they are listed in the manual too).
2. Look under the **Gaia** entry to see astrometry-specific items
3. We will use the `astromUVW` and maybe `icrsToGal` and `astromXYZ` functions. The **Examples** items in the function documentation are useful; for use with the `gaia_source` catalogue, you can often just cut and paste, though note "`radial_velocity`" may need to be changed to "`dr2_radial_velocity`".
4. Open the **Views|Column Info** window and choose the ✚ **Columns|New Synthetic Column** action
5. Create a new column giving Cartesian velocity components: **Name**: "`uvw`", **Units**: "`km/s`" and for **Expression**:

   ```
   astromUVW(array(ra, dec, parallax, pmra, pmdec, dr2_radial_velocity))
   ```

   That calculates velocities along ICRS axes. If you want it in Galactic coordinates, wrap the whole expression in the `icrsToGal(...)` function.
6. Look at the new column in the **Views|Table Data** window (scroll all the way to the right). It is a 3-element array; you can access the array elements using expressions `uvw[0]`, `uvw[1]`, `uvw[2]`

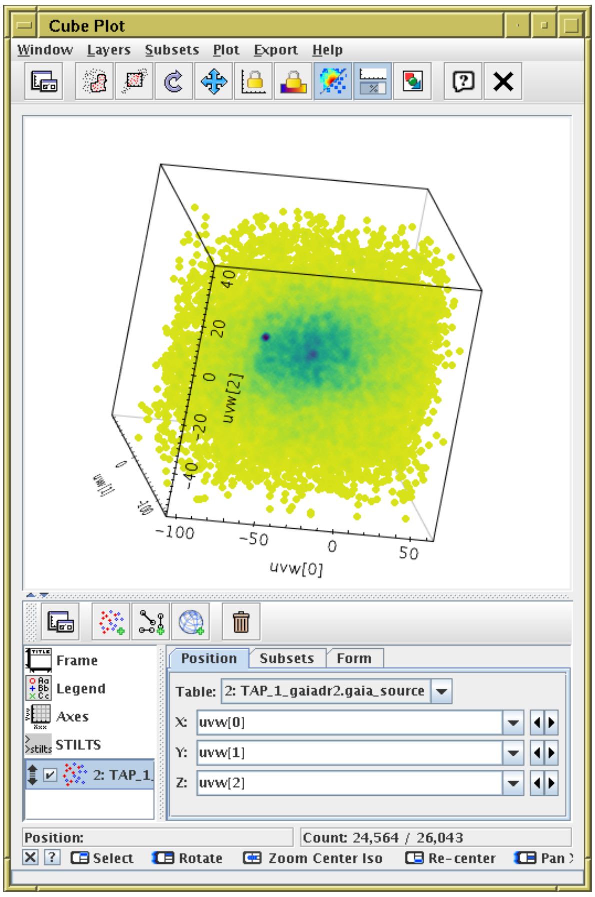


Table: 2: TAP_1_gaiadr2.gaia_source
X: uvw[0]
Y: uvw[1]
Z: uvw[2]
Position:
Count: 24,564 / 26,043

2.5 Identify Hyades graphically in 3-d velocity space

1. Plot points in 3-d space: **Graphics|Cube Plot**, **X**: "`uvw[0]`", **Y**: "`uvw[1]`", **Z**: "`uvw[2]`". Note, you can type in any expression for the plot coordinates, you don't have to just select from available columns. The `uvw` column itself doesn't appear in the selection list, since it's not a scalar.
2. Select **Mode**: "`density`" in the **Form** tab. You can change the colour map to taste using the **Density Shader** selector.
3. Now, navigate through the cube to find an overdensity. This takes a bit of practice, but it's fun once you work out how. Click the little ? button at bottom left for navigation help; the most useful actions are mouse wheel (2-fingered up/down drag on some trackpads) to zoom, and right click on a dense region to recenter.
4. Navigate so only the objects in the overdense region (about 200 of them?) are visible inside the wireframe — these are the Hyades.
5. Use **Subsets|Subset From Visible**, **Name**: "`hyades`", **Add Subset**.

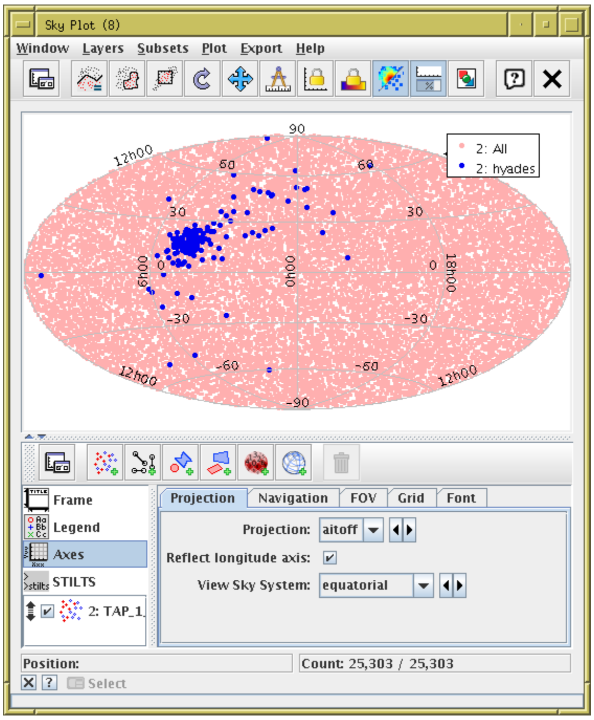
2.6 View positions

1. Go back and plot this subset on the sky (**Graphics|Sky Plot**)

2. Use the **Subsets** tab to make sure both **hyades** and background objects (**All**) are plotted in different colours. You might want to set **Shading Mode**: “`flat`” and fiddle with marker size and shape for clarity.

3. To get an all-sky view, use the **Axes** control, **Projection** tab, and change the **Projection** selector from “**sin**” to “**aitoff**”

4. If you like, you can plot it in 3-d space as well: **Graphics|Sphere Plot**, **Lon**: “`ra`”, **Lat**: “`dec`”, **Radius**: “`1000./parallax`”



Projection: aitoff
Reflect longitude axis: ☑
View Sky System: equatorial

Position:
Count: 25,303 / 25,303

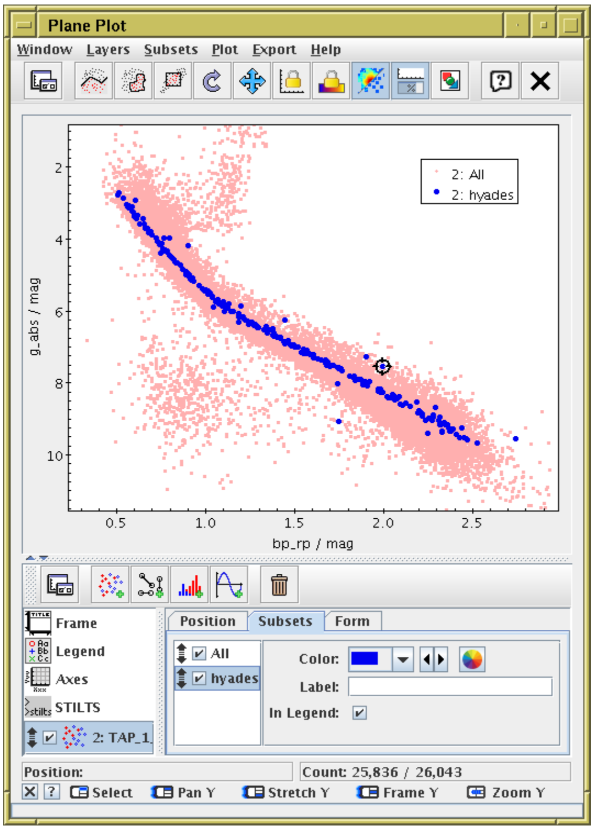
2.7 Colour-magnitude diagram

1. Plot a colour-magnitude diagram: **Graphics|Plane Plot**, **X**: “`bp_rp`”, **Y**: “`g_abs`”.

2. Use the **Axes** control, **Coords** tab, **Y Flip** checkbox to flip it the right way round.

3. Use the **Subsets** tab to make sure both **hyades** and background objects (**All**) are plotted in different colours. Hyades sit on a nice tight main sequence!

4. There are a few outliers. Click on them, see the position show up in the sky plot too. In some cases, you can see by sky position that they are non-members. Open the **Views|Data Window** and see that the relevant row is highlighted when you click on a plotted point.



Color:
Label:
In Legend: ☑

Position:
Count: 25,836 / 26,043

Bonus

- Can you find any other clusters in velocity space?
- Try refining the selection by localising in position space too.
- What is the mean distance to the Hyades?
- Try using Aladin and SAMP along with TOPCAT to investigate the outliers.

3 Match Gaia and HST observations

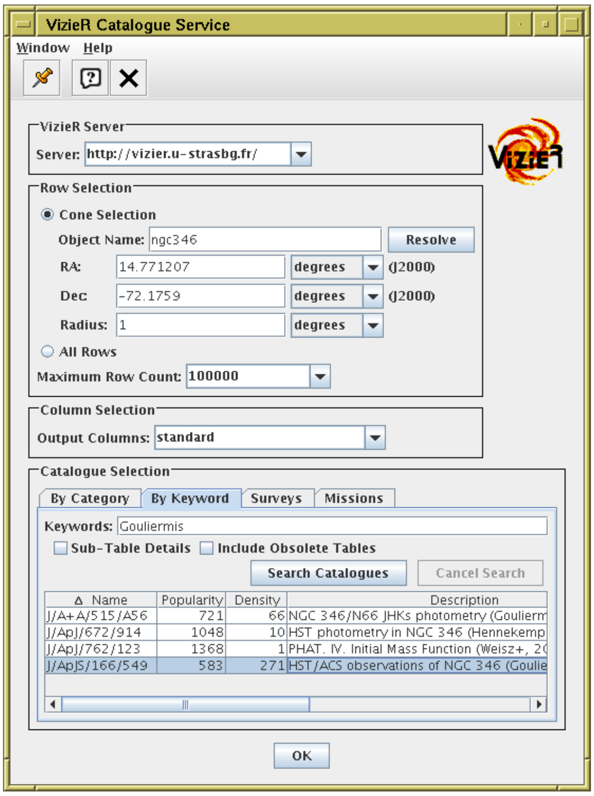
In this example we have a local catalogue from a publication by Gouliermis et al. 2006 [4], available in VizieR as J/ApJS/166/549. This contains about 100 000 sources observed by the ACS instrument on the Hubble Space Telescope at epoch ≈ J2004.6 of stars in NGC346, a cluster in the Small Magellanic Cloud. We match these positions with positions in the main Gaia catalogue at J2016.0.

3.1 Acquire HST observations

There are various ways to do this, but here we will use TOPCAT's VizieR dialogue window, which talks directly to the VizieR catalogue service.

1. Open **VO|VizieR Catalogue Service** window
2. **Object Name**: "`ngc346`", and **Resolve** to fill in **RA** and **Dec**
3. **Radius**: "`1`" (degrees)
4. **Maximum Row Count**: "`100000`" (or some large number)
5. Catalogue selection panel: **By Keyword** tab
6. Fill in **Keywords**: "`Gouliermis`"
7. Select "`J/ApJS/166/549`"
8. Hit the **OK** button at the bottom. A new table with 99 079 rows should be loaded

Alternatively, you could download the table from the VizieR web page.



Server: http://vizier.u-strasbg.fr/

● Cone Selection

Object Name: ngc346

RA: 14.771207 degrees (J2000)

Dec: -72.1759 degrees (J2000)

Radius: 1 degrees

○ All Rows

Maximum Row Count: 100000

Output Columns: standard

Keywords: Gouliermis

□ Sub-Table Details □ Include Obsolete Tables

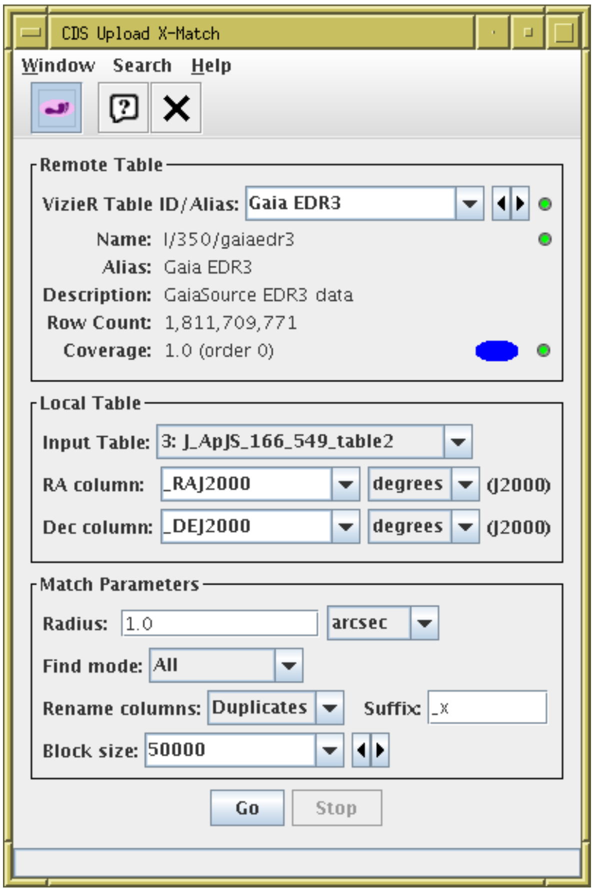
Name	Popularity	Density	Description
J/A+A/515/A56	721	66	NGC 346/N66 JHKs photometry (Goulierm
J/ApJ/672/914	1048	10	HST photometry in NGC 346 (Hennekemp
J/ApJ/762/123	1368	1	PHAT. IV. Initial Mass Function (Weisz+, 2(
J/ApJS/166/549	583	271	HST/ACS observations of NGC 346 (Goulie

3.2 Crossmatch with Gaia

Now we want to find associations of the HST objects with sources from Gaia EDR3. Use the CDS X-Match service from TOPCAT. This uploads a local table to the CDS X-Match service, where the match is made against the Gaia EDR3 catalogue (or any other catalogue in VizieR). The resulting matched catalogue is then received as a new table in TOPCAT.

1. Open the **VO|CDS Upload X-Match** window
2. Fill in the fields:
   VizieR Table ID/Alias: "`GAIA EDR3`"
   Input Table: "`J_ApJS_166_549_table2`"
   (or whatever the HST table is called)
   RA column: "`_RAJ000`", **Dec column**: "`_DEJ2000`"
   (should be filled in automatically)
   Radius: "`1`" (arcsec)
   Find Mode: "`All`" *Important!*
3. Hit **Go**; within a few seconds, it should inform you that a new table has been loaded, with about 24 000 rows.
4. Look at the columns of the new table (all HST followed by all Gaia) in the **Views|Column Info** window

Note: the match is done with Gaia coordinates rolled back (using Gaia proper motions) to J2000.0. These propagated columns are in the matched table as `ra_epoch2000`, `dec_epoch2000`.



VizieR Table ID/Alias: Gaia EDR3

Name: I/350/gaiaedr3

Alias: Gaia EDR3

Description: GaiaSource EDR3 data

Row Count: 1,811,709,771

Coverage: 1.0 (order 0)

Input Table: 3: J_ApJS_166_549_table2

RA column: _RAJ2000 degrees (J2000)

Dec column: _DEJ2000 degrees (J2000)

Radius: 1.0 arcsec

Find mode: All

Rename columns: Duplicates Suffix: _x

Block size: 50000

3.3 Visualise the crossmatch

1. Open a **Graphics|Sky Plot** window
2. Plot the HST observations: **Table**: "`J_ApJS_166_549_table2`", **Lon**: "`_RAJ2000`", **Lat**: "`_DEJ2000`"
3. Overplot the actual matches. Add a new Pair layer: **Layers|Add Pair Control** and fill in:
   Table: "`4xGAIA EDR3`" (or whatever the xmatch result table is called) and both sets of coordinates:
   Lon(1): "`_RAJ2000`", **Lat(1)**: "`_DEJ2000`" (HST)
   Lon(2): "`ra`", **Lat(2)**: "`dec`" (Gaia)
4. Zoom in to look at the associations. There are too many! What is this plot telling you?
5. You can fiddle around with the **Form** tab to make the plot clearer, e.g. add a **Mark2** layer; change marker size, shape or colour.

Visualising the results of a crossmatch is very often a good idea, unless you're pretty sure what you're going to get. Here, you can see it was crucial to understand the results: most of these matches are spurious, because there is a high density of HST sources.

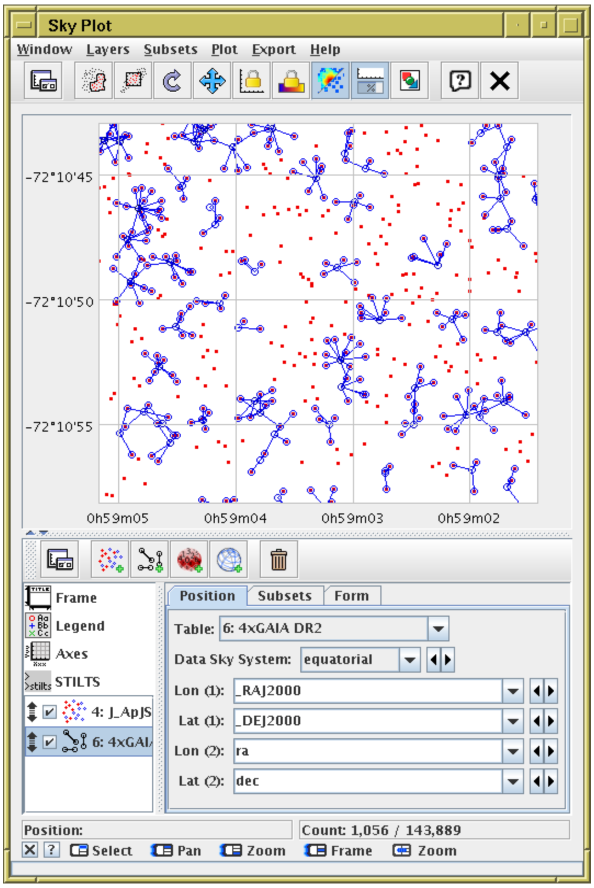


Table: 6: 4xGAIA DR2
Data Sky System: equatorial
Lon (1): _RAJ2000
Lat (1): _DEJ2000
Lon (2): ra
Lat (2): dec
Position:
Count: 1,056 / 143,889

3.4 Investigate and identify matches

1. Add new columns giving RA/Dec discrepancies between HST and Gaia positions:
   Open **Views|Column Info** window,
   then define new columns using **Columns|New Synthetic Column**:
   Name: "`deltaRa`", **Expression**: "`3600*(ra - _RAJ2000)`",
   Units: "`arcsec`"
   Name: "`deltaDec`", **Expression**: "`3600*(dec - _DEJ2000)`",
   Units: "`arcsec`"
2. Use **Graphics|Plane Plot** window,
   plot **X**: "`deltaRa`", **Y**: "`deltaDec`"
3. Identify overdense region, select as in section 1.2, define new subset **true_match**.
4. Go back to the sky associations plot from the previous section, and use the **Subsets** tab to visualise which are the true matches. Why do you think this offset is not zero?
5. Make a colour-colour diagram combining HST and Gaia photometry:
   Use **Graphics|Plane Plot** window, plot **X**: "`Vmag-Imag`", **Y**: "`bp_rp`", display **true_match** subset only.
   What are the two populations?

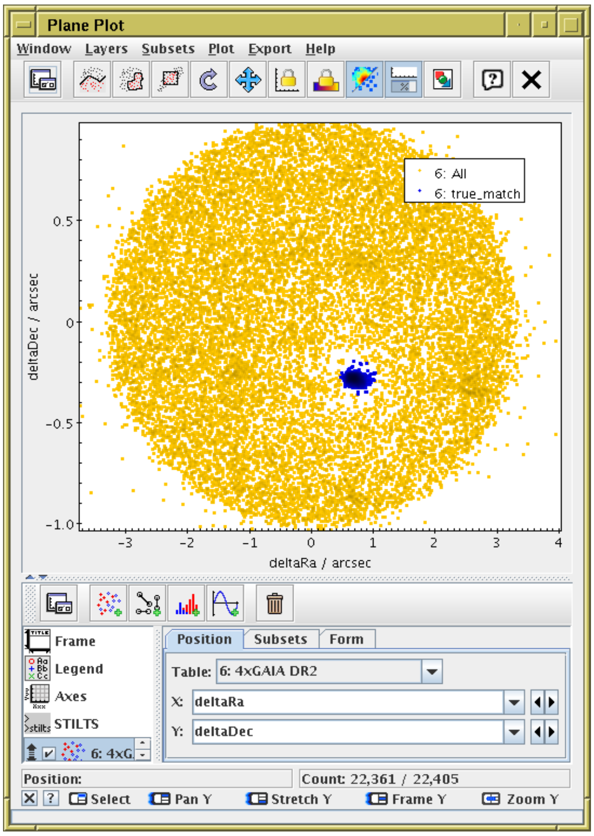


Table: 6: 4xGAIA DR2
X: deltaRa
Y: deltaDec
Position:
Count: 22,361 / 22,405

Note: in the above we have used the Gaia columns `ra` and `dec`, giving the positions observed at J2016.0. It would be a bit better to use `ra_epoch2000` and `dec_epoch2000`, the positions rolled back to J2000.0 using Gaia proper motions, since the HST observations were at about J2014.6, and even better to apply proper motions to get the positions at J2014.6. But the differences (motion over 10–15 years) are fairly small.

3.5 Alternative crossmatch: use local files

The crossmatch in section 3.2 was done by sending a local file to an external service. This is often an efficient way to do it, but there are other options. Here, we will do the same crossmatch by operating on two local files with positions covering the same sky region.

1. First, retrieve Gaia data in the region of interest. Use the **VO|Cone Search** window to a Gaia service as in section 1.1, but this time fill in **Object Name**: "`ngc346`", **Radius**: "`0.05`" (degrees).

2. Plot the Gouliermis (from VizieR) and Gaia (from Cone Search) datasets on the sky: open the **Graphics|Sky Plot** window, fill in RA and Dec as **Lat** and **Lon** for one of the tables, then use the **Layers|Add Position Control** action to overplot the same thing for the other dataset.

3. Open the **Joins|Pair Match** window from the main control window. Default **Match Criteria** (Sky, 1 arcsec) are OK in this case. Fill in the **Table 1** and **Table 2** details for the Gouliermis and Gaia tables. Entry **Match Selection**: "`All matches`".

4. Hit **Go** and wait a few seconds for the match to complete.

5. When complete, a popup window will tell you, and offer to **Plot Result**. If you select this option, you will see a plot like the one from section 3.3, except that the unmatched Gaia sources are also plotted (which can sometimes be useful information).

Other matching options are available in the local match windows, including identifying objects that don't match, matching internally within a table, matching between three or more tables, etc. Note that unlike most things in TOPCAT, crossmatching can take up significant amounts of memory, so matching multi-million-row tables can sometimes grind to a halt or fail.

Bonus

- Use the **Graphics|Histogram Plot** (as in section 1.5) to find the mean values of δ RA, δ Dec for the true matches. What do these values tell you?

- The match done here is between Gaia positions at J2016.0 and HST positions taken at approximately J2004.6. When using the X-Match service in section 3.2, Gaia proper motions were automatically applied to predict the Gaia positions at J2000.0. Use the `epochProp` function in the expression language to do the match with the positions as Gaia proper motions predict for J2004.6. Does it make much difference?

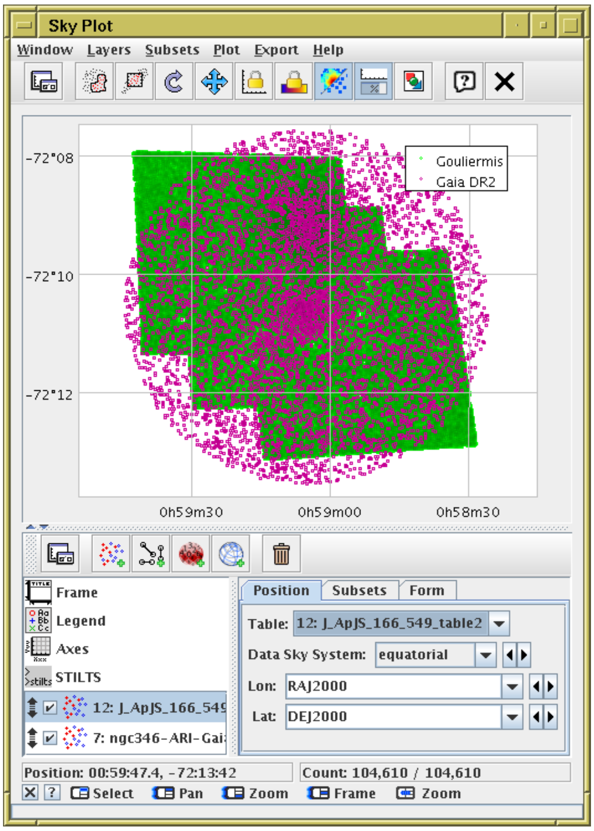


Table: 12: J_ApJS_166_549_table2

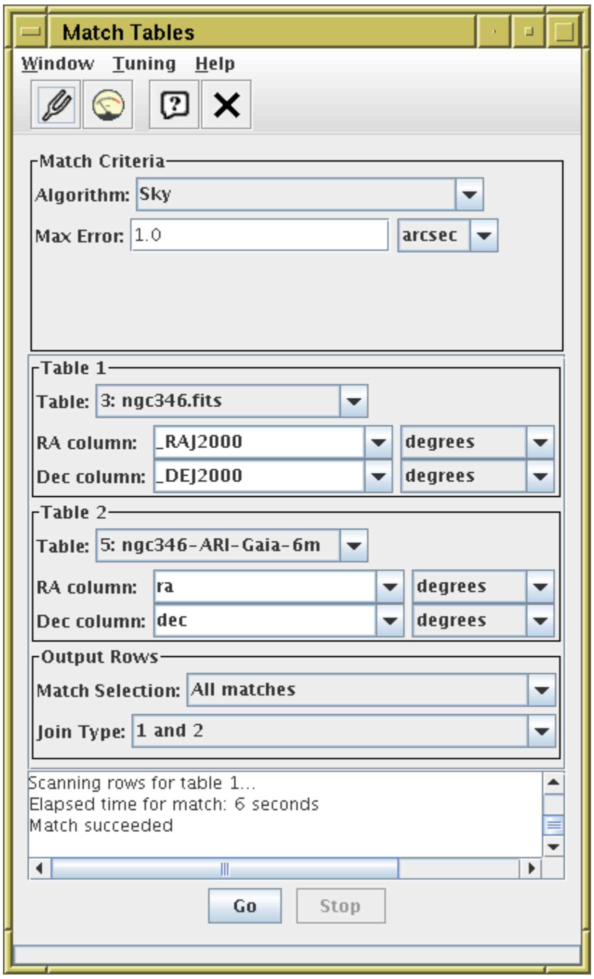
Data Sky System: equatorial

Lon: RAJ2000

Lat: DEJ2000

Position: 00:59:47.4, -72:13:42

Count: 104,610 / 104,610



Match Criteria

Algorithm: Sky

Max Error: 1.0 arcsec

Table 1

Table: 3: ngc346.fits

RA column: _RAJ2000 degrees

Dec column: _DEJ2000 degrees

Table 2

Table: 5: ngc346-ARI-Gaia-6m

RA column: ra degrees

Dec column: dec degrees

Output Rows

Match Selection: All matches

Join Type: 1 and 2

```
Scanning rows for table 1...
Elapsed time for match: 6 seconds
Match succeeded
```

4 Messier 4 in proper motion space using STILTS

In this section we will re-do the determination of the distance to M4 from section 1, but this time from the command line, though we will use some of the results of the earlier GUI activity. Using TOPCAT interactively to work out how to write STILTS scripts for later batch use like this is a common pattern of work.

4.1 Get STILTS running

1. Unlike TOPCAT, you can't bluff your way through STILTS by pointing'n'clicking; you'll need some documentation. Find the manual here at `http://www.starlink.ac.uk/stilts/sun256/` or maybe just google for "stilts" and go to the **Documentation** section.

2. Make sure you have it installed. "`java -jar stilts.jar ...`" should work. If you have topcat, you can run "`java -jar topcat-full.jar -stilts ...`" instead. More convenient (on Un*x), download the `stilts` script into the same directory as `topcat-full.jar` (or `stilts.jar`). Then the directory needs to be on your path, or you can use the full pathname or set up an alias. We will write just "`stilts`" from now on.

3. Run a command:

   ```
   stilts calc expression="1+2"
   ```

   should print out "3"

4. Find the documentation for the `calc` command in the manual (Appendix B). Look at the **Usage** and **Examples** subsections.

5. Get command-line help by running "`stilts calc help`" and "`stilts calc help=expression`".

4.2 Acquire data from Cone Search service

1. Investigate the STILTS `cone` command; see the documentation in Appendix B of the manual (`http://www.starlink.ac.uk/stilts/sun256/cone.html`) and run

   ```
   stilts cone help
   ```

   There is a documentation bug! (STILTS v3.4); the `out` parameter is also required to give an output file

2. Find out the cone search parameters we used in 1.1. This includes the service URL (from the **Cone URL** field) and the central RA (longitude) and Dec (latitude) parameters of M4, as well as the search radius.

3. Putting those together, we need to run

   ```
   stilts cone serviceurl="http://dc.zah.uni-heidelberg.de/gaia/q3/cone/scs.xml?"
               lon=245.89675 lat=-26.52575 radius=0.3 out=m4.fits
   ```

   to give the output file `m4.fits` — the output format has been determined by file extension (you can write e.g. `m4.vot` if you prefer).

Note: some values containing whitespace or other strange characters need to be quoted to prevent the shell expanding them. This quoting can get a bit messy. I *believe* the commands as written in this section will work in common Un*x shells, also Windows command prompt and PowerShell.

4.3 Manipulate the downloaded table using `tpipe`

We will perform some processing on the downloaded table using the `tpipe` command. This reads an input table, optionally performs operations on it, and writes it to output, either to a file (in the same or different format) or some other destination. It works like a Unix pipeline.

The operations are defined by adding *filters* such as "`select`" (retain only some rows) or "`addcol`" (add a new column). Filters are specified by adding using zero or more "`cmd=`" parameters on the command line, and documented in Section 6.1: "Processing Filters" of the STILTS manual.

The default output mode is write to a file, but other options such as "`meta`" (display column metadata), "`count`" (count rows) and "`stats`" (calculate mean, st.dev etc) are also available. Select non-default output modes using the "`omode=`" parameter on the command line, documented in Section 6.4: "Output Modes" of the manual.

1. Count the rows in the query result table:

```
stilts tpipe in=m4.fits omode=count
```

2. See what columns the query result table has:

   ```
   stilts tpipe in=m4.fits omode=meta
   ```

   or for a more manageable output (customised metadata; the “meta” filter turns the table into a table with one row per input table column):

   ```
   stilts tpipe in=m4.fits cmd="meta name units class"
   ```

3. For convenience, write a new table with only a few of the columns. The keepcols filter produces a table with only the listed columns (the list is space-separated and needs to be quoted). The clearparams filter removes per-table metadata.

   ```
   stilts tpipe in=m4.fits cmd="keepcols ’ra dec pmra pmdec parallax’" cmd=’clearparams *’
                out=m4mini.fits
   ```

4. Calculate statistics using the stats output mode:

   ```
   stilts tpipe in=m4mini.fits omode=stats
   ```

5. View a few rows of the table. The using the head filter keeps only the first few rows:

   ```
   stilts tpipe in=m4mini.fits cmd="head 10"
   ```

   You can select rows using the select filter, and combine filters by stringing them together on the command line. Try both:

   ```
   stilts tpipe in=m4mini.fits cmd="head 10" cmd="select parallax>0"
   stilts tpipe in=m4mini.fits cmd="select parallax>0" cmd="head 10"
   ```

   Why are they different?

6. Add a column calculating estimated distance:

   ```
   stilts tpipe in=m4mini.fits cmd="addcol dist_est 1000/parallax" cmd="head 10"
   ```

7. Write a new table containing only those rows in the comoving region of proper motion space. Looking at the proper motion plot from section 1.2 you can estimate the expression by eye as being approximately a circle radius 2 centered on (-12.6,-18.9). The select filter includes only rows for which the given expression is true, and the hypot(a,b) function is short for sqrt(x*x,y*y).

   ```
   stilts tpipe in=m4mini.fits cmd="select ’hypot(pmra+12.6,pmdec+18.9)<2’" out=cluster.fits
   ```

   In fact TOPCAT can produce these geometric expressions for you: you can use the **Subsets|Draw Algebraic Subset** (Ellipse) action in the plot window to generate them graphically.

8. Calculate the mean and S.D. of cluster parallax using the previous selection:

   ```
   stilts tpipe in=cluster.fits cmd="keepcols ’parallax’" omode=stats
   ```

   or do it all in one go:

   ```
   stilts tpipe in=m4.fits cmd="select ’hypot(pmra+12.6,pmdec+18.9)<2’"
                           cmd="keepcols ’parallax’"
                           cmd="stats name mean stdev"
   ```

4.4 Plot using STILTS

1. Plot the cluster on the screen:

   ```
   stilts plot2sky in=cluster.fits layer1=mark lon=ra lat=dec
   ```

   Note that this plot is interactive — you can zoom and drag it around as in a TOPCAT plot window.

2. Create a plot graphics file. Just add an output parameter like “out=plot.png” or “out=plot.pdf” to the above plot2sky command to write static output files.

3. Interoperate with TOPCAT: STILTS plotting commands can get quite complicated. If you have a plot in TOPCAT and want to reproduce it in STILTS, look at the **STILTS** control, which shows you the invocation needed to reproduce it.

There is *lots* more that tpipe, and STILTS in general, can do. Explore the manual!

5 Local Herzsprung-Russell Diagram

In this example we will use a TAP query to download all the nearby Gaia sources with good astrometry and photometry, and calculate their absolute magnitudes to construct an HR diagram, performing a couple of cleaning operations to improve the data. This procedure loosely follows Appendix C of the Gaia DR2 astrometry paper Lindegren et al. 2018 [3].

5.1 Acquire data from TAP service

1. Open the TAP window **VO|Table Access Protocol (TAP) Query**

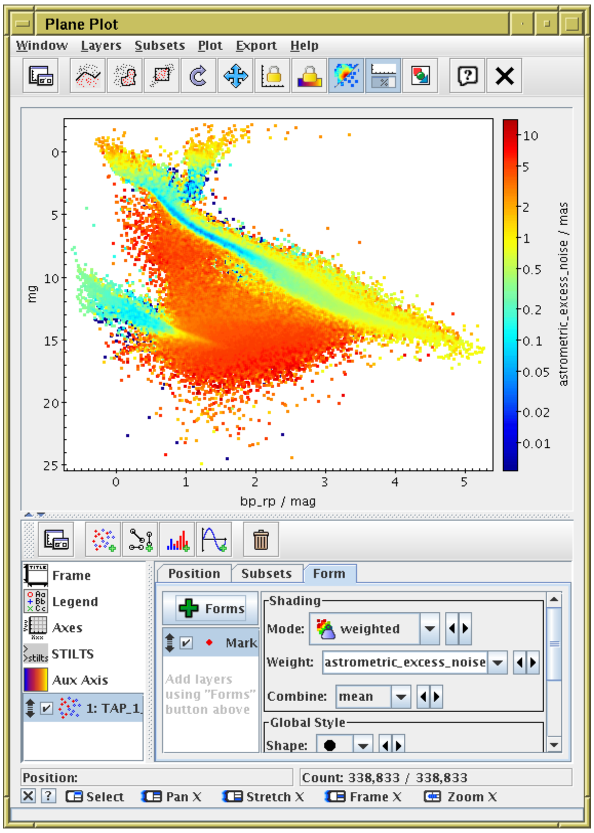
2. Select one of the Gaia services (probably the ESA one) and **Use Service**

3. Choose **Mode**: "`Asynchronous`" (just above the ADQL text entry panel). This query may take a minute or two, so a synchronous query might time out (with the unhelpful result, probably, "*TAP response is not a VOTable*").

4. Execute the following query:

```
SELECT ra, dec, parallax, phot_g_mean_mag, bp_rp,
       astrometric_excess_noise,
       phot_bp_rp_excess_factor
FROM gaiadr2.gaia_source
WHERE parallax > 10
  AND parallax_over_error > 10
  AND phot_bp_mean_flux_over_error > 10
  AND phot_rp_mean_flux_over_error > 10
```

   You should get a table with 338 833 sources; they are nominally within 100 pc, and have ϖ, BP and RP with small errors. In particular, the parallax error is small enough that ϖ^{-1} is a reasonable estimate of distance.



Mode: weighted

Weight: astrometric_excess_noise

Combine: mean

Shape: ●

Position:

Count: 338,833 / 338,833

5.2 Plot HRD

1. Add a new column calculating absolute G magnitude, using parallax: **Columns|New Synthetic Column** in **Column Info** window:
   Name: "`g_abs`",
   Expression: "`phot_g_mean_mag + 5*log10(parallax/100)`",
   Units: "`mag`"

2. Make a **Graphics|Plane Plot**, with **X**: "`bp_rp`" ($BP - RP$ colour), **Y**: "`g_abs`" (absolute G magnitude). Use the **Axes** control, **Coords** tab, **Y Flip** checkbox to flip it the right way round. Structure visible, but lots of interlopers.

3. Play around with different **Shading Modes** in **Form** tab.

4. Colour points using the other columns using modes **Aux**, **Weighted**. What's the difference between the two?

5.3 Exclude astrometrically suspect sources

1. Try different weighting/aux columns to see which one can be used to exclude points in the unwanted region between the main sequence and white dwarf branch.

2. Experiment with the colour map settings (**Aux Axis** control) to find a suitable threshold.

3. Create a subset using an algebraic expression that excludes the spurious points: go to the **Views|Row Subsets** window and use the **Subsets|New Subset** action: **Subset Name**: "`astrom_ok`", **Expression**: "`astrometric_excess_noise < 1`"

4. Go back to the plot, and make sure only the `astrom_ok` subset is plotted (**Subsets** tab). Now there are fewer spurious sources.

The `astrometric_excess_noise` column characterises the goodness of fit of the astrometric solution to the observations. But you didn't need to know that to improve the data like this.

5.4 Exclude photometrically suspect sources

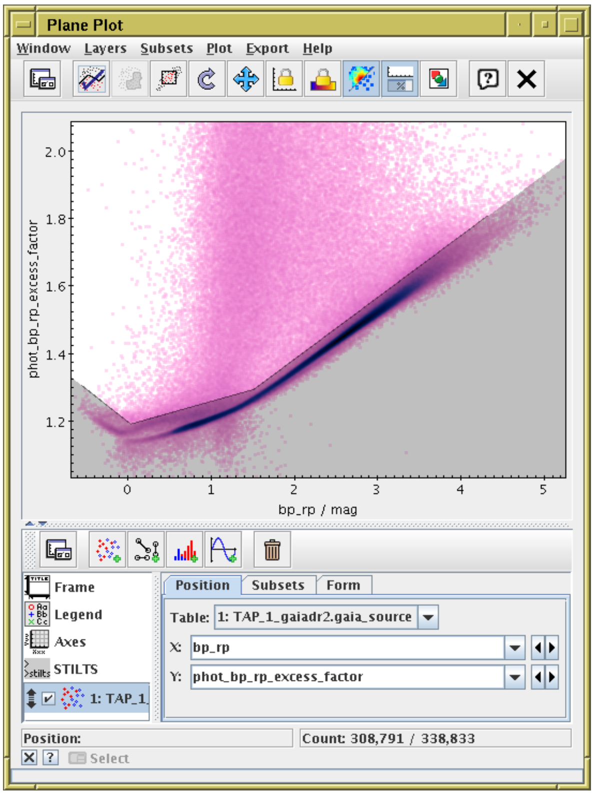


Table: 1: TAP_1_gaiadr2.gaia_source

X: bp_rp

Y: phot_bp_rp_excess_factor

Count: 308,791 / 338,833

1. Open a new **Graphics|Plane Plot**, with **X**: "`bp_rp`" ($BP - RP$ colour), **Y**: "`phot_bp_rp_excess_factor`".

The quantity on the Y axis is some measure of photometric reliability. High values are bad, but how high is colour-dependent. We will define a region in this space to exclude the unusually high values. This time we will interactively draw a polygon rather than a blob.

2. Click the **Subsets|Draw Subset Polygon** action

3. Select **Point inclusion mode**: "`BELOW`" in the popup

4. Click on a few points above the overdense region until the shaded area roughly covers it.

5. When you're done, click on the button again.

6. Fill in the **Subset Name** field in the popup (e.g. `photom_ok`) and hit **OK**

7. Go to the **Views|Row Subsets** window, where you can see the new subset alongside `astrom_ok`.

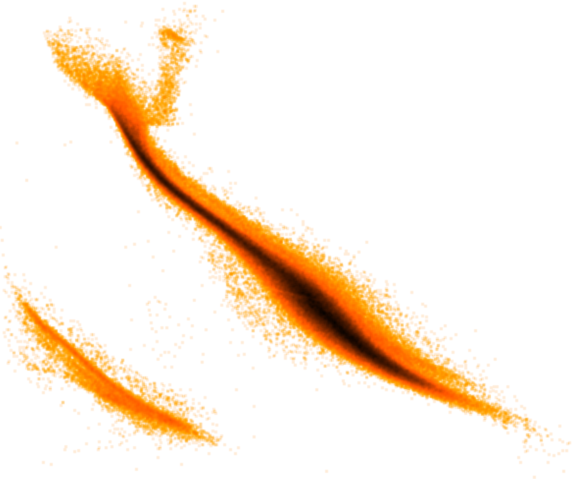
8. Create a new subset combining the two:
 Subsets|New Subset action: **Subset Name**: "`ok`"
 Expression: "`astrom_ok && photom_ok`"

9. Go back to the plot, and make sure only the `ok` subset is plotted (**Subsets** tab). Now there are fewer spurious sources.

Leave this session open, we'll need some of it for the next section.

5.5 Explore the HRD

The photometry and astrometry in Gaia DR2 is so good that plotting a Herzsprung-Russell Diagram by following the steps above gives a lot of astrophysical information. Identify the different populations with the help of the zoom and shading options in TOPCAT: main sequence, giant branches, the double-stranded white dwarfs sequence representing the split between helium and hydrogen burning, and if you look closely a notch in the main sequence near absolute G magnitude of 10 (Jao et al 2018 [5]).



Bonus

Go back and reproduce all the other exercises using STILTS!

References

[1] Gaia Collaboration et al., "Gaia Early Data Release 3: Summary of the contents and survey properties", Astronomy and Astrophysics `https://doi.org/10.1051/0004-6361/202039657` (2021)

[2] X.Luri et al., "Gaia DR2: Using Gaia parallaxes", Astronomy and Astrophysics *616*, A9 (2018), `2018A&A...616A...9L`

[3] L.Lindegren et al., "Gaia DR2: The astrometric solution", Astronomy and Astrophysics *616*, A2 (2018), `2018A&A...616A...2L`

[4] D.A.Gouliermis, A.E.Dolphin, W.Brandner and Th.Henning, "The Star-forming Region NGC 346 in the Small Magellanic Cloud with Hubble Space Telescope ACS Observations. I. Photometry", ApJS, 166 p.549 `2006ApJS..166..549G`

[5] W.-C.Jao, T.J.Henry, D.R.Gies, N.C.Hambly, "A Gap in the Lower Main Sequence Revealed by Gaia Data Release 2", ApJ Letts, 861, L11 (2018), `2018ApJ...861L..11J`

Acknowledgements

This work has made use of data from the European Space Agency (ESA) mission *Gaia* (`https://www.cosmos.esa.int/gaia`), processed by the *Gaia* Data Processing and Analysis Consortium (DPAC, `https://www.cosmos.esa.int/web/gaia/dpac/consortium`). Funding for the DPAC has been provided by national institutions, in particular the institutions participating in the *Gaia* Multilateral Agreement.