Identifying brown dwarfs in 2MASS and SDSS using the VO

Hendrik Heinl, Markus Demleitner, Dave Morris

November 11, 2018

Abstract

Today, IVOA interoperability standards like Simple Cone Search (SCS), Simple Image Access (SIAP) and TAP/ADQL are used by many astronomers. Most recent surveys like Gaia are making use of these standards for publishing their data, future surveys like LSST or SKA plan to do so. Though accessing data of a single service makes a good bunch of VO use cases so far, familiarity with VO standards opens the possibility to combine and crossmatch the data of several surveys and service within a few steps: knowing how to access one service means knowing how to access all of them. The following tutorial introduces into the usage of some of these standards and how one can remotely crossmatch between different services and survey. As example we chose the use case of identifying brown dwarves in 2MASS and SDSS.
Brown dwarfs are faint objects with a mass below 0.8 solar masses. In their stellar core, conditions don't suffice for hydrogene fusion and therefore the surface 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 filter bands. 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 check for accuracy and eventually use SSAP to access spectra for one object.

Software: TOPCAT Version 4.3, Aladin Version 8.0, SPLAT, TAP, ADQL,

1 Starting out

We will use the VO tools TOPCAT, Aladin and Splat-VO. These client software can be found here:

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 and TAP

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 instead of downloading whole catalogues, we receive only the data we are interested in.
TAP (Table Access Protocol) services can be accessed by using ADQL. Therefore one needs a client software capable of understanding TAP. Here, we use TOPCAT as a TAP client, but you are not restricted to it. You can always use a your favourite TAP client, the ADQL queries will not be different.

Be aware that this is not a comprehensive introduction to ADQL. Though the steps will be understandable without any ADQL skills, we strongly recommend you take 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, we want to search the VO registry. Therefore we open TOPCAT and go to VO →Table Access Protocol. In the TAP-Query window Keywords we enter `2MASS`. In the line below, we check Description (you may need to adjust the window width to see all options). Then we click on Submit Query. In the section below now we can see a list of TAP services related to 2MASS. When searching for services, you may need to spent some time to find the one most appropriate for your needs.

In our case we already have a good guess that the **GAVO Data Center TAP service** will be the right choice. So in this list, we mark the line by clicking once. Then we either click on Use Service at the bottom or on the according tab at the top line.

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. Notice that you get a first glance of the meta data of this table in the

VO Registry

The Registry is the first standard we use. The Registry is the central point for searches for VO-compliant services. VO services identify themselves to the Registry and provide metadata about the data they serve and the service protocols they support. Thus, the Registry is the entry point for data discovery in the VO.

field on the right. Here you find a good overview when you mine for data. We will have a close look at meta data a few steps ahead so we leave this aside for the moment.

Exercise: You are interested in different surveys ? Play around with the registry. Try key words related to your research and see if the VO provides you with table data. In case you lack ideas, just try the following:
How many TAP Services providing Gaia Data are out there?
Are there any Services providing HST data?
What about neutrinos?

▷ **2** *Submitting a first ADQL Query: cone selection* – Within the VO the term of a **cone search** covers basically an area at the sky defined by a position and a radius in degrees around it. You may have performed cone searches already with TOPCAT, so here is a little different approach to give you an idea how ADQL works. Luckily TOPCAT comes with examples for the most common ADQL queries – or more precisely with those 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
  1=CONTAINS(POINT('ICRS', raj2000, dej2000),
            CIRCLE('ICRS', 189.2, 62.21, 0.05))
```

Now at the first glance that may look confusing, so 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 the first 1000 data records will be returned, which match the whole query. With * we select all columns of matching data records. If you just wanted certain columns from a table to be returned, you could specify that 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. A little confusing may be the `1=CONTAINS`. This is due to the boolean result returned by the function `CONTAINS`. A result of `1` means **`true`** whereas `0` means **`false`**. `POINT` expresses a point depending on the coordinate frame (soon to be depricated), the right ascension and the declination in degrees. Our query takes the coordinates from the table using the columns `raj2000` and `dej2000`. Analogously `CIRCLE` expresses a cone in space, except that we need to set an angle in degrees as last number.

Clicking OK will start the query and within a few seconds we retrieve the first 1000 data records. Return to the TOPCAT mainwindow and play around with this data, if you like to. As soon as you feel ready, proceed.

Exercise: Click on Graphics→Sky Plots. Topcat will open the Sky Plot window and you see the data records plotted on the sphere. You can drag the sphere and change the perspective, use your mouse wheel to zoom in and out. But the real question here is: how did TOPCAT know which columns to use for the sky plot? You will find the answer in the next step.

▷ **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 colour filter magnitudes. 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 H,J and K magnitudes, which we see are named hmag, jmag and kmag. Knowing the column names, we now can modify the ADQL query accordingly.

UCDs – VO semantics

UCDs (Unified Content Descriptors) are the semantics in the VO. They describe the content of columnms in a machine readable way. Thus, the machines "know" something about the data and you can make use of this in your programs. A good example are votable objects in astropy.

▷ **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. We also assume brown dwarfs are red sources. So we want to search for sources with an index of

$$jmag - kmag > 0.8.$$

Running the query over the whole 2MASS catalog would take a long time so 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 1=CONTAINS(
    POINT('ICRS', raj2000, dej2000),
    CIRCLE ('ICRS', 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_s filter magnitudes from 2MASS we found first candidates for brown dwarfs. For further data reduction we now apply criteria derived from Zhang et al. (2010). Therefore we need i,r and z filters which we take from SDSS. In the Topcat TAP window we now enter sdssdr7, again check Description and submit the query. Again our choice due to performance issues is the GAVO datacenter TAP service. You notice, that we have the same choices to query tables as when we searched for 2MASS. This is because we use the same TAP service as before and therefore we have to specify the tables we want to query with `FROM`.

We could now do a similar search as for SDSS and then compare the two tables locally. But we rather use a more elegant method and let the GAVO TAP service do some work for us remotely. Since both tables are on the very same service, we can merge the two queries into a single one and obtain the result at once. For this we use the `JOIN... ON` command in ADQL to merge the data. This time we use the `CIRCLE` function to find matches for the data we received from 2MASS and therefore keep the cone very small at 1". We also add criteria from Zhang et al into the query to perform a colour cut on the SDSS colours. Finally we add `TOP 20000` after `SELECT` to limit the data. This is the full query:

```
SELECT TOP 20000
tm.mainid AS twomass_id, sdss.objid AS sdss_objid, tm.*, sdss.*
FROM twomass.data AS tm
```

```
JOIN sdssdr7.sources as sdss
  ON 1=CONTAINS(
    POINT('ICRS', sdss.ra, sdss.dec),
    CIRCLE('ICRS', tm.raj2000, tm.dej2000, 2./3600.))
    AND  1=CONTAINS(
      POINT('ICRS', tm.raj2000, tm.dej2000),
      CIRCLE ('ICRS', 127.0000, 1.2000, 2.0))
    AND tm.jmag > 15.3
    AND tm.jmag-tm.kmag > 0.8
    AND sdss.i - sdss.z BETWEEN 1.5 AND 2
    AND sdss.r - sdss.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. At first glance we already recognise that we do have some identical data records in raj2000 and dej2000 columns or in the column of the 2MASS identyfier. This is due to the TAP search which returned all matches around a source from the 2MASS data, and not the single best match. As we scroll down the records, we see that for all 2MASS positions at least 2 records in SDSS matched our query. How come that as we performed a cone search around 1" around our given positions? It may be helpful to take a look at the observed images and thankfully the VO provides us with the perfect tool for that: Aladin.

Exercise: Consider we may not be interested in colours from SDSS, but astrometry from Gaia instead. There is a gaia table on the very same server. Try to adjust above query to receive proper motions for the crossmatch between 2MASS and Gaia.

Exercise: As you can see, we can use algebraic expressions in the `WHERE` clause. Can you write a crossmatch with the Gaia catalogue that selects objects with absolute magnitude between 6 and 8?

▷ **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, especially we are interested to find out, why we have several matches in SDSS for each of the candidates identified in 2MASS. For this we use the SAMP protocol to send data from TOPCAT to Aladin. We start Aladin and select the 2MASS catalog. Then we go back to the TOPCAT main window and check the table with our candidates from SDSS. We then click in Views →Activation Actions →Send Sky Coordinates. Thus we activate TOPCAT to broadcast data to other SAMP Applications. In our case, Aladin will "listen" and wait for data from the SAMP hub, that is running in the background. Still in Topcat we open the table data and click on any table row. In Aladin we can see the position change. If we compare the images from our doubled 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 datarecords from our crossmatch with SDSS and in the SDSS catalog the different observations to a single object are not merged. So we have to do this on our own.

SAMP and VOTable

SAMP (Simple Application Messaging Protocol) is the protocol that enables VO software tools to interoperate and communicate. It is the interface for astronomical software to work together. In particular the standard needs standardised file formats like VOTable that can be exchanged between different VO tools.
VOTable is the standard within the VO to exchange table data. It is implemented as an XML standard and therfore allows transformations through XSLT engines. VOTables are meta data rich and in combination with UCDs provide a high level of interoperability.

We solve this in TOPCAT. In the main window we mark the table and click on Joins →Internal match. In the new window at Table we chose our current table and below we check Eliminate All But First of Each Group. Of course for real science this would be inappropriate, but for our purpose it works well. Now we reduced our brown dwarfs candidates to 11 and it's about time to check the accuracy of this kind of these methods.

▷ **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 let's 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 dwarves candidates and the SIMBAD database. In the topcat TAP window click on examples→Upload→Upload Join. You will find the query

```
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.raj2000, tc.dej2000, 3./3600.))
```

You may notice that our `JOIN` command now is different to the one we used above. `JOIN` is used when you want to receive only those datarecords where the conditions for both table are matched, here the positions. But in this step we actually are interested to keep those of our local records that do not match records in SIMBAD: actually those could be good candidates for further research, because apparently they are not identified as brown dwarfs. `RIGH OUTER JOIN` roughly translates to "Join table1 with table2 where the following conditions are met. Where the conditions are not met, keep data records from table2." Analogously there is the command `LEFT OUTER JOIN` implemented in ADQL. 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 5 of our objects are listed in SIMBAD as brown dwarfs. One is listed as a candiate. The other 5 objects may be candidates worth further research.

Exercise: With this knowledge you should be able to write a ADQL query with a TAP upload that will perform a cross match between the table we obtained in step "Crossmatching with SDSS" and the gaia catalouge on the GAVO server.

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

We now want to use Splat-VO and the Simple Spectra Access Protocol (SSAP) to search for spectra. We open Splat and open the SSAP window: File→SSAP_. In the left part of the SSAP window we see the server selection. Here one can pre select services according to the spectra you are interested. In our case we will search for all band width and 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 if you find missing metadata! Thus the VO may improve by your contribution.

SSAP

SSAP (Simple Spectra Access Protocol) is used to find spectra within the VO. SSAP clients usually first query the VO Registry to find services of interest based on user demand e.g. related to wave band. In a second step the selected services can be queried for datasets matching the requirements. The result of this will be a VOTable containing rich metadata that enables to decide to actually download the actual spectra.

3 References

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

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