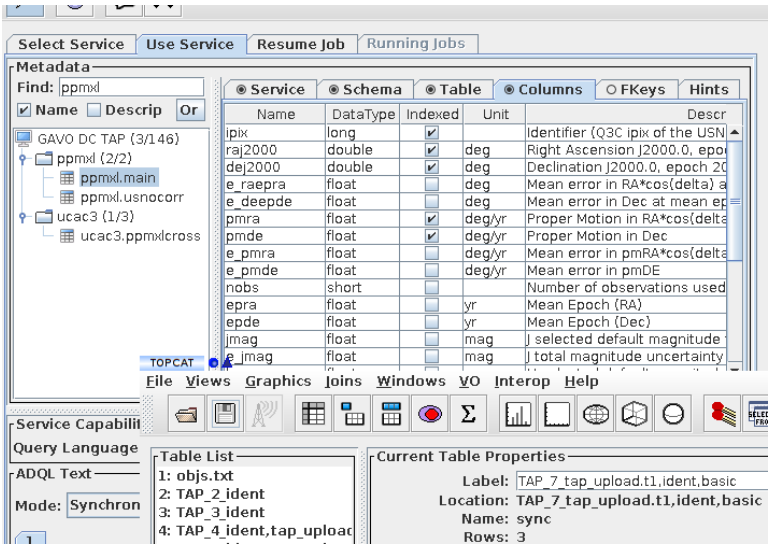
Adding catalog data to object lists using the VO



Markus Demleitner

March 27, 2019

Abstract

This brief tutorial shows you how to quickly add proper motions and photometry from Gaia to (almost) any object list using the Virtual Observatory. The VO protocol most suited to this kind of this is TAP ("table access protocol") and lets you transfer data and queries to database servers. In the example, we will be using TOPCAT as a client. There is no lock-in to it: There are libraries and other tools allowing an integration of TAP operations into arbitrary workflows that's what standards are about. Tutorial supplements apply the techniques to Simbad, show how to use TAP from Python, and introduce UCDs.

Software: TOPCAT, pyVO

1 Describe your data to TOPCAT

Whatever you can load into TOPCAT can work here. To play through the worst case, let's assume you just have a load (millions are possible) of RA/Dec values in one of the text files we astronomers (regrettably) so adore. If you don't have a file of your own handy, we have prepared a sample file for you at

`http://docs.g-vo.org/pmadd-testdata.txt`.

▷ **1** *Load your data* – Click File/Load Table With our test data, you'll manually have set format to ASCII in TOPCAT's load dialog. With the pesky ASCII files, TOPCAT has no way to know it's dealing with positions. Because we later need it to understand that, we have to help it.

▷ **2** *Improving column metadata* – To tell TOPCAT there's RA and Dec here, bring up the columns info window by selecting Views/Column Info from the menu. The easiest way to tell TOPCAT we have a position here is just editing the column names to be `ra` and `dec` (at the end of this document you'll find a better way). Double-click into table cells to edit them. Close the columns info window.

2 Select the service and table to match against

To locate services and data in the VO, use the Registry; there are many ways to do that, and it's possible (and in some cases a very good idea) to use physics to locate data ("Give me an all-sky catalog containing proper motions and infrared photometry"). There is a separate tutorial for that. Here, we already know the name of the data collection, which simplifies data discovery.

▷ **3** *Find a TAP service* – In TOPCAT, select VO/TAP Query, and in Keywords type "Gaia". This will result in a list of services offering at least some Gaia data. Most of the ones near the top should be good for what follows; select it, then hit Use Service.

▷ **4** *Find your table* – Some TAP services have hundreds of even thousands of tables. To find the metadata of yours, in the Find field in the Metadata pane, enter some characters you expect to see in the table name or description. In this case, "dr2" or "source" would work fine. Note the extensive metadata

right of the table tree. It should help you select a pertinent table. In this case, you probably want `gaiadr2.gaia_source` (or `gaia.dr2light` if you chose the GAVO DC TAP service). Select this table in the table tree so TOPCAT knows this is what you are interested in.

3 Write the query

Since positional crossmatches to uploaded data are so frequent, TOPCAT can generate you a sample query to start from:

▷ **5** *Have TOPCAT help you with your query* – Hit the Examples button (right below the large text field near the bottom of the TAP dialog) and select upload/Upload join. A query adapted to joining the current TOPCAT table with the TAP table selected appears in the query field. It should look pretty much like this:

```
SELECT
   TOP 1000
   *
   FROM gaia.dr2light AS db
   JOIN TAP_UPLOAD.t1 AS tc
   ON 1=CONTAINS(POINT('ICRS', db.ra, db.dec),
                 CIRCLE('ICRS', tc.ra, tc.dec, 5./3600.))
```

Don't worry if you don't understand all of it now – that can wait a bit. Suffice it to say that it's fairly standard SQL[1]

▷ **6** *Customise the match limit* – Edit it a bit to better fit your needs: Instead of `TOP 1000`, say `TOP 5000` or something appropriate to the size of your input data (but remember that an input line can match multiple objects, so be generous here). You will have to raise Max Rows above the query input if you want to retrieve more rows than shown there.

The `*` after the `TOP` clause means: Get all columns. That's more than we care to see here. We want our columns – that's `tc.*`, the `tc` being the alias given in the line starting with `JOIN` –, and the proper motions and

[1] "Structured Query Language", the de-facto standard for communicating with database systems since the seventies. In this incarnation, it's actually ADQL, where AD stands for "Astronomical Data", but that makes no real difference.

photometry from the database. Obtain the names from the Columns pane in the TAP Window's table browser.

▷ **7** *Customise the fields to retrieve* – So, replace the * with:

```
tc.*, pmra, pmdec,
phot_g_mean_mag, phot_bp_mean_mag, phot_rp_mean_mag
```

(you could have written `db.pmra` and so on here, but as long as the names are unique, you can skip the table name). And in a real science settings, you should probably get the errors and `astrometric_params_solved`, too...

▷ **8** *Customise the match radius* – Finally, you may want to change the match radius. That's the `5./3600.`, and it's in degrees. Whether 5 arcsec is too much or too little for your application is a matter of your scientific judgement (or a principled analysis).

▷ **9** *Run the query* – Then hit Run Query and enjoy, plot, or devour your results. If you have large object sets, you may want to change Mode to Asynchronous before running the query; that lets you run queries that take, potentially, hours (but if you think you need to do this, think again and perhaps ask the operators; there are often better and faster ways to the same result).

4 Supplement: Simbad

One nice thing about standards is that if you can operate one site, you can (basically) operate all of them. So, since you know how to crossmatch with PPMXL on GAVO's TAP server, let's see how things work with SIMBAD.

Locating Simbad's TAP service works exactly as above: In the TAP window, go back to the Select Service tab. In the Keywords field, enter "simbad" and hit Find Services (that's the button in the upper third of the window, *not* the Use Service button near the bottom). You should receive at least one result, and you want to doubleclick SIMBAD TAP.

After Use Service, you can now browse Simbad's table metadata. Most of the standard stuff is in `basic`. Use the search field to find the table, select the table.

Just for demonstration, try again what you've done above, i.e., Examples, Upload Join again, fix the search radius as you see fit and send off the query. You should fairly quickly see a couple of rows identifying the "well-known" among the objects in the list as (possibly just being near to)

anything from an absorption line system or a galaxy to a semi-regular variable star.

A variation of the example above ("get proper motions when all you have is positions") is getting positions when all you have is a list of object names (as long as Simbad can resolve them). We just need to be a bit more creative with our joins, and TOPCAT doesn't have an example (yet).

Consider a file like

```
"HR 1014"
"M 31"
"Q2237+0305"
```

(the quotes just help TOPCAT to figure out that names with blanks in them still make up a single column; if your object list doesn't have them, add them with `sed -i 's/.*/"&"/' objs.txt`).

Load that (or a similar) file into TOPCAT (manually select the CSV format), then go back to the TAP window pointing at Simbad and run the following query:

```
SELECT col1, ra, dec
FROM TAP_UPLOAD.t1
LEFT OUTER JOIN ident
ON (id=normId(col1))
LEFT OUTER JOIN basic
ON (oidref=oid)
```

(you will probably have to change the "t1" after TAP_UPLOAD to match the index of the table with the identifiers, which in turn is the number TOPCAT shows in the table list in the main window).

Explaining what's going on in this query is a bit beyond the scope of this little tutorial – please refer to our ADQL course if you want to understand the details.

What this returns as of this writing is:

col1	ra	dec
HR 1014	49.49614153531403	-66.92685427016318
M 31	10.68470833333333	41.26875000000000
Q2237+0305		

Note again that there's no problem doing this for a thousand identifiers at a time.

5 Supplement: Use python

Another nice thing about standards is that you get to choose your client (i.e., the software you use to operate the services). To use the VO from within Python programs, there is a (pip-installable, Debian-packaged) package called pyVO.

If you have it, you can run our initial example (minus the discovery part; you *could* do that from within pyVO, too, but frequently it's much better to do that part interactively) using a program like this:

```
import urllib

import pyvo
from astropy.table import Table

# Load our example table
our_data = Table.read(
  "http://docs.g-vo.org/pmadd-testdata.txt",
format="ascii")

# Fix its metadata
# (not required with a sensible input format)
our_data["col1"].name, our_data["col2"].name = "ra", "dec"

# construct a service; I've taken the URL from TOPCAT's
# TAP service browser # ("Selected TAP Service" near the
# foot of the dialog)
service = pyvo.dal.TAPService(
  "http://gea.esac.esa.int/tap-server/tap")

# run the query and retrieve the result; note that the
# "t1" after TAP_UPLOAD must match the key in the uploads
# dictionary.
result = service.run_sync("""
  SELECT
   tc.*, pmra, pmdec,
   phot_g_mean_mag, phot_bp_mean_mag, phot_rp_mean_mag
   FROM gaiadr2.gaia_source AS db
```

```
    JOIN TAP_UPLOAD.t1 AS tc
    ON 1=CONTAINS(POINT('ICRS', db.ra, db.dec),
                  CIRCLE('ICRS', tc.ra, tc.dec, 1./3600.))""",
  uploads = {'t1': our_data}).to_table()

# save the results; use a useful format now.
result.write("our-data-amended.vot", format="votable")
```

It wouldn't be hard to send the result to, say, TOPCAT rather than save it. See http://docs.g-vo.org/pyvo for an informal course on this. The source above is also attached to this PDF.

6 Supplement: Marginalia

The metadata we added above is of course extremely shoddy. For it to be more understandable outside of TOPCAT, you would add UCDs[2] to the column metadata: For that you'd select Display/UCD from the window and enter `pos.eq.ra;meta.main` in the UCD field for the RA column and correspondingly `pos.eq.dec;meta.main` for the Dec column.

The query as written returns one line per *match*, i.e., input objects that have no counterpart in the database will be missing in the output; and indeed, you'll see that for the 4000 input objects (which, really, are typically fairly weak infrared sources if you have to know), only about 1500 output lines result.

If you want non-matching input objects in the output, with NULLs wherever database information would be, write `RIGHT OUTER JOIN` instead of `JOIN` ("right" because your table is the right operand in the join expression). If you used the sample data set, you'll see that 253 objects had multiple matches in PPMXL.

Again, more on TAP and ADQL in GAVO's little course at http://docs.g-vo.org/adql

[2] "Unified Content Descriptors", standard labels for physical entities; you might have seen those on VizeR