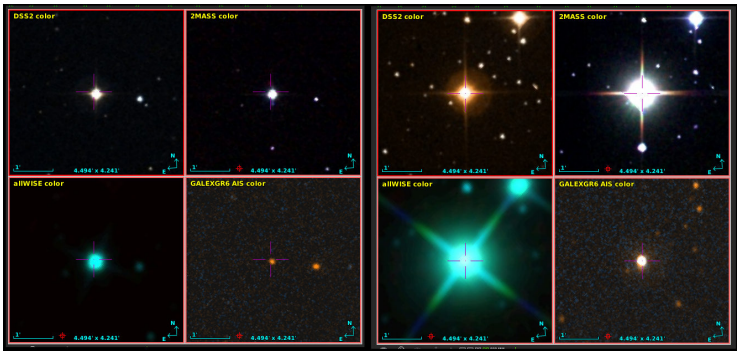
Outlier Analysis in Low-Resolution Spectra: DFBS and Beyond



Demleitner, M.; Mickaelian, A.; Knyazyan, A.;
Baghdasaryan, D.; Mikayelyan, G.

September 7, 2020

Abstract

Low-resolution spectra like those resulting from objective prism observations or the RP/BP instrument on board of the Gaia astrometry satellite enable a wealth of interesting science. This use case investigates one use leading up to combining many VO resources: The identification of misclassified objects from reference databases.

Software: TOPCAT, astropy, Aladin, SPLAT

1 Introduction

Objective prism low-dispersion spectra are useful for search and selection purposes, as they permit a preliminary understanding on the the nature of a large number of objects and are sufficient to distinguish objects with definite features. Historically, such spectra resulted from observations using objective prisms on Schmidt telescopes, which let astronomers observe large parts of the sky and obtain spectra of millions of objects. A modern source of such low-dispersion spectra will be ESA's Gaia satellite, which will produce more than a billion of them. It is expected that they will become publicly available in the early 2020s.

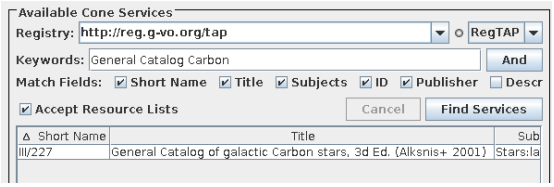


Figure 1: Discovering a candidate catalogue in TOPCAT.

This use case will illustrate the use of such spectra to look for "outliers" in catalogues of special objects – for simplicity, we use carbon stars here. For more background information, in particular on our main data collection, the Digitised First Byurakan Survey (DFBS), see Appendix A.

2 Obtaining a Target List

There are many ways to obtain lists of objects of interest; in particular, you might choose from any object catalog you can find in TOPCAT's VO → Cone Search window. As long as you're working with DFBS, make sure sufficiently many objects of what you're looking at are available off of the galactic plane (other surveys of course have other criteria). Also, the objects should not have redshifts significantly impacting spectra at a resolution of several nanometers in the optical, because we assume here that spectra are different because of intrinsic (rather than kinematic) properties.

Since DFBS has epochs in the 60ies and 70ies, if you have objects that may have high proper motions, you should also fetch and apply the proper motions[1].

For this tutorial, a useful source is the General Catalogue of Carbon Stars – there are numerous carbon stars outside of the Galactic plane, and they're spectrally fairly distinctive thanks to the Swan bands originating from glowing C_2.

To obtain that catalogue, in TOPCAT go to VO → Cone Search. Enter "General catalog carbon" into Keywords. Use the "fetch all" trick of using 0, 0 for RA and Dec and 180 for the search radius and double click the found service (the line with "III/227" in Fig. 1). See Appendix B for how to obtain your source lists from Simbad.

3 Adding Spectra

We now want to add the low-resolution spectra. Since the spectra are so small, they can conveniently be kept right in the database. This means that with

[1] A tutorial on how to do this with VO tools is available at http://www.g-vo.org/tutorials/add-pms.pdf.

a single TAP query, we can pull lots of spectra, which makes bulk analysis somewhat easier.

▷ **1** *Locate a TAP server holding DFBS spectra.* – Go to the VO → Table Access Protocol (TAP) dialog in TOPCAT and type `dfbs` in the Keywords: input. Hit Find Services, and in the TAP service tree, select a service offering the `dfbsspec.spectra` table[2] and click Use Service.

▷ **2** *Locate the table we want to query.* – In the table browser (that's the upper left pane in the TAP window), look for dfbs again. You'll see a `dfbsspec.spectra` table – that's the one we want. Select it and have a look at the Columns tab. You'll see that apart from quite a bit of information on the object observed there is a `flux` column. You could also retrieve the `spectral` column, which contains the wavelengths belonging to each point in spectrum, but since (on this particular dataset) they're all the same, we don't actually need them here.

▷ **3** *Create the crossmatch query.* – Writing a crossmatch query may seem a bit daunting at first, but TOPCAT helps you: With the table from section 2 selected in the main window and our `dfbsspec.spectra` in the TAP window, hit Examples → Upload → Upload join. You'll get a sample query that you will have to adorn a bit. First, remove the `TOP 1000` – we want as many matches as there are.

Second, it's usually a good idea to think first which columns you actually want in order to avoid wasting storage and processing – this becomes more important as you start dealing with millions of rows, but it's a good idea to develop a routine. TOPCAT's columns browser helps doing this. For our problem, we'd suggest to retrieve the columns `accref` (trust us on this one for now), `specid` (so we know what object we talk about later), `ra`, `dec` (we may want to inspect the piece of sky this comes from), `flux` (obviously), `magb` and `magr` (photometry is always good), `emulsion` (which might be useful for debugging), `cutout_link` (so we can look at what the spectra look like on the objective prism plates), and `recno` and `Names` from our source list so we can go back there if we think we need to. Just replace the `*` in the query with the names of these columns. This would result in a query somewhat like:

```
SELECT
   accref, specid, ra, dec, flux,
   magb, magr, cutout_link, recno, "Names"
   FROM dfbsspec.spectra AS db
   JOIN TAP_UPLOAD.t1 AS tc
   ON 1=CONTAINS(POINT(db.ra, db.dec),
                 CIRCLE(tc."_RAJ2000", tc."_DEJ2000", 5./3600.))
```

(adjust the "t1" to match the number of the carbon star list in TOPCAT's Table List; this will automatically be right if you used Examples to generate

[2] One of these should be the ArVO Byu TAP service at http://arvo-registry.sci.am/tap.

the query). In case you're wondering: We need the double quotes around `Names` because that's a reserved word in SQL; if you're choosing column names yourself, avoid these, but since we're bound to the column names the GCCS authors chose, we have to use this little workaround.

Run the query. Depending on what data you're trying to upload and the server configuration, this may fail with an error like "Error writing request body to server". If that happens to you, just switch Mode in TOPCAT's TAP dialog to "Asynchronous" – what's happened is that you've hit the limit for upload sizes on the service in question, and some services are stingy for synchronous (immediate) queries but generous for async (which can be queued).

▷ **4** *Limit the magnitude range* – Since photographic emulsions had a rather severely nonlinear response, you want to constrain the magnitude of what you're looking at at a time, or the spectra will look fairly differently just because of that nonlinearity. You can make a histogram plot of `magr` to help you decide what magnitude bracket to use, but our adivce is: use a magnitude range from 11.5 to 13.

To do that, in TOPCAT say Views → Row Subsets, and in that window, say Subset → New subset. In Subset Name, type "sample", and in Expression say `magr>11.5 && magr<13`. If you have plotted the histogram, you'll see this subset marked.

To make this subset the one that TOPCAT works with, in TOPCAT's main window, for Row Subset select the *sample* entry.

Rather than subsetting, you'd normally use server-side selection, i.e., just add a `WHERE magr BETWEEN 11.5 AND 13` to the query. In this particular case, this will make the query a lot slower for somewhat obscure reasons. See Appendix C to see how to write such a query nevertheless. There, you will also find other hints on speeding up large upload joins (which are optional here).

▷ **5** *Save the data for further processing.* – Once the query is done, in TOPCAT's main window select File → Save Table(s)/Session. In the dialog, set Location to `specs.vot`. That name is hard-coded in the little script we use in the next section, so make sure you get it right.

4 Scoring

We now want to see which of the spectra deviate a lot from what we consider "typical" for the object class we've selected. To do that, we need to compute the mean of what we've pulled and then compute the distances (in some sense of that word) of the individual spectra from that mean. The spectra that are far away are the ones that probably are misclassified.

Since TOPCAT's array facilities are not quite sufficient to do that at the time of writing, we drop into astropy to do that.

▷ **1** *Obtain and understand the source code.* – The source for our spectrum difference computing tool is attached to this PDF. You can also pull it from the document source[3]. Wherever you get the file from, put it next to the `specs.vot` file you created in the last section.

Here's the relevant piece of code:

```
min_len = min(len(x) for x in spec_table["flux"])
normalized = [normalize(flux)[:min_len] for flux in spec_table["flux"]]
template = normalize(sum(normalized))

diff_column = table.Column(
    data=[np.mean((n-template)**2) for n in normalized],
    name="dist_from_tpl",
    description="Distance of this spectrum from the mean"
        " spectrum of this table",
    meta={"ucd": "stat.fit.goodness"})

score_table = table.Table([spec_table["spec_id"], diff_column])
```

So, we crop all spectra to a common length (the minimal one, as we can't fantasize values that aren't there, but we can crop away values that are), then normalize them to integral one, compute a template as the mean of the result, and then make a new column (with description and UCD – if you take that and nothing else home from this tutorial, that's already a win) with a difference measure between each spectrum and the template.

▷ **2** *Add the scores to our table in TOPCAT.* – We amend (rather than replace) the previous table. In this case, that's because there is extra information in the table not preserved by astropy; in general it's much more flexible to aggregate tables through the sort of joins we're doing here.

So, load the files scores.vot you've just been creating into TOPCAT.

Select Joins → Pair Match. As Algorithm use "exact value", then select the spectrum table (its name will probably contain "dfbsspec.spectra") and the scores table (which will be called "scores.vot" by TOPCAT), and in each select specid as column. After you've run the match, you'll have a new table called something like "match(foo,bar)" (or the like). Continue working with this table.

▷ **3** *Find out the outliers.* – In a table display of your match table, right click on the `dist_from_tpl` column and choose Sort down. The worst-fitting spectra end up on top. Also, plot a histogram of the `dist_from_tpl` column; you'll see there's one object that's apart (cf. Fig. 2).

[3] http://svn.ari.uni-heidelberg.de/svn/edu/trunk/arvo_dfbs/res/specdistance.py

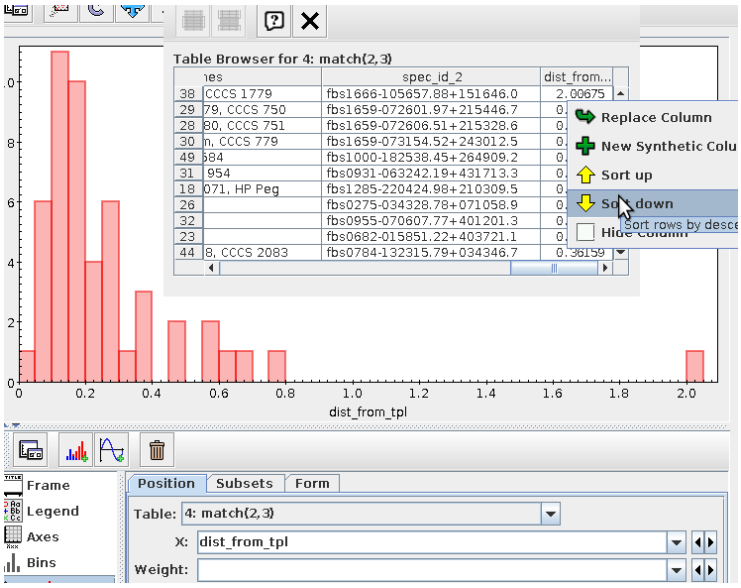


Figure 2: A histogram clearly shows that one of our spectra is quite different from the others. Sorting by score is one way to identify which one is bad.

5 Investigating the Outliers

Let's now see what that outlier is about. We will make ample use of activation actions in this; essentially, these are things TOPCAT executes when you click on a row or a point in a plot. Activation actions have become a lot more fun since TOPCAT 4.6. Make sure you have that (or a newer) version before proceeding.

To configure activation actions, go to Views → Activation Actions.

▷ **1** *Plot the spectra* – The obvious first step is to plot the spectra themselves. That's the "Plot Table" action. Add a check mark on it. To see what happens, hit the flash button next to Invoke now on row 1 (or whatever row you have selected). You should see something like Fig. 3. When you now click on some other table row, you should see a "normal" spectrum close to the template we've made. Look at a few of them and you'll see that the spikes in our outlier spectrum are really a bit odd.

Future versions of TOPCAT might be able to plot directly from arrays, which would let us plot our flux column rather than retrieve the spectrum from the server – that would be a bit more elegant in this case.

▷ **2** *View the source images* – So – is this an extraction problem or is this, perhaps, really an object with an extremely strong emission line? To get to the ground of this, let's have a look at the image this was generated from. To do that,

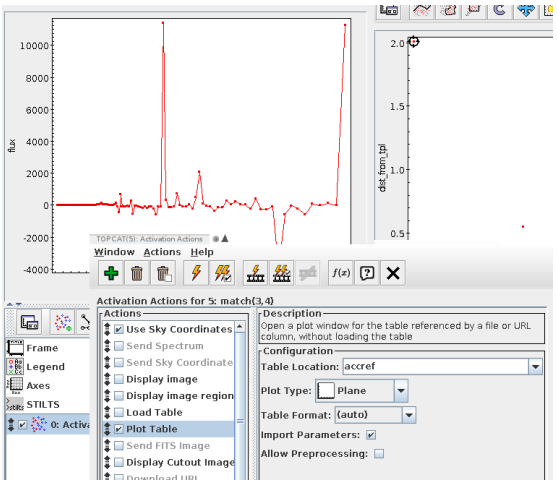


Figure 3: Plotting a spectrum through a TOPCAT activation action. By default, you'll not see lines but dots in the plot; to fix that, go to the Form tab in the plot window and add a Line form.

configure a "Display Image" activation action by clicking on the respective entry in Actions. In "Image Location", select the `cutout_link` column. Don't forget to actually enable the activation action by hitting its check button.

If you run the activation actions now (either by hitting the button or selecting an entry), you'll see the cutout of the spectrum. Which looks inoccuous, which is why in this particular case we suspect a technical problem on the end of spectrum extraction.

▷ **3** *Look up things in a sky atlas.* – Let's pretend we still suspect physics. So, let's see how the object looks like in reality. A great tool to do this is the interactive sky atlas Aladin. Start it. As a sane default, we'll want to see how things look like in the optical, so click DSS right above the image plane to get the venerable Palomar Sky Survey plates. We'll want to look at individual stars, so Zoom in to a field of view of a few arcminutes.

▷ **4** *Connect Aladin and TOPCAT.* – We now want to investigate the objects in the table in TOPCAT in broadband photometry – or, actually, plain images. To make TOPCAT tell Aladin where we're looking at, check the "Send Sky Coordinates" activation action (which is properly pre-configured because TOPCAT has enough metadata to know where the position is encoded in each row). Run the activation action, and you'll see a harmless-looking star. Could this be a Carbon star? Well, split the Aladin display into four view ("multiview" just below the image) and load, in turn, 2MASS (as near infrared), WISE (as mid infrared) and, say, Galex. Hit match (next to the multiview buttons) and you should see something like the left part of the Fig. 4.

▷ **5** *Compare with a non-outlier.* – What you should have seen in the last step is that our star is bright far into the mid-infrared. This is about what we'd expect for carbon stars, right? Well, click on one of the non-outliers, and you'll see

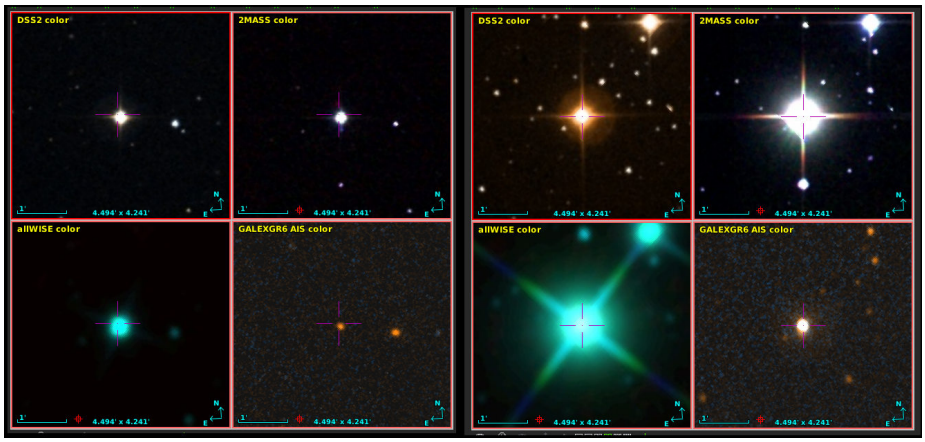


Figure 4: A comparsion of the broadband appearance of our outlier and some tame carbon star.

something like what's shown on the right part of Fig. 4. Again we'd say our outlier is a false alarm.

6 Discovering Spectra

You don't have to be content with a broadband, SED-like impression of what the star is. You can look for actual spectra as well. Discovering those involves finding spectral services in the VO that have data for the area you're looking at and then querying them. Here's how to do that in Aladin.

▷ **1** *Get a star you are interested in into the view.* – Hit the column that has `fbs1659-072606.51+215328.6` for `spec_id` in TOPCAT and in Aladin zoom to make your field of view a few arcminutes (other objects may have spectra as well, but this one is a safe fall-back).

▷ **2** *Restrict discovery to spectral services.* – You can discover all kinds of data in the VO. To only look for spectra, in Aladin's discovery tree, open the Others branch and left-click on SSA (spectrum).

A dialog will pop up, and in it there is a scan button ( ) that lets you filter your selected resources for those that have data for your current view. Aladin will ask you whether you're sure you want to query more than 100 services – just agree (unless you'd be querying 1000s of services, in which case you messed up step 1; don't worry, these kinds of queries are fairly cheap).

When you've done that, you'll see the color of the service entries turn green or orange depending on whether or not they have data. Any green one you can mouse over, check in view and hit Load. We suggest LAMOST DR3 from

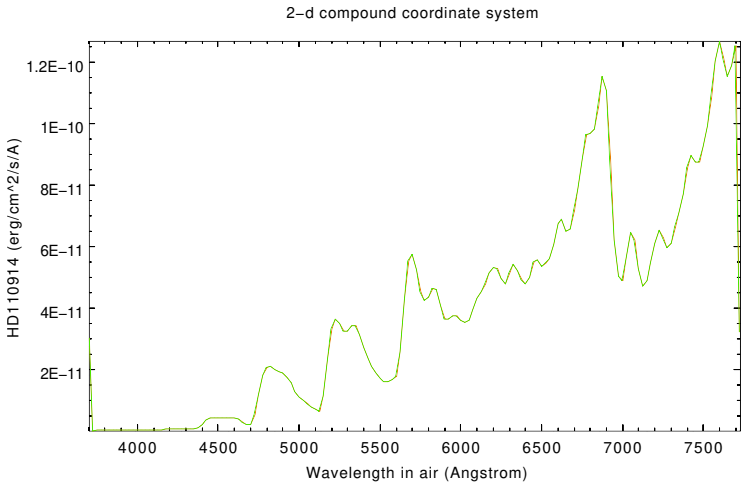


Figure 5: A prototypical spectrum of a carbon star; this plot was made by discovering spectra for Y CVn in Splat's SSA dialog and plotting a result form the HyperLeda FITS Archive (reference URL `http://leda.univ-lyon1.fr`) service.

org.gavo.dc at the time of writing.

▷ **3** *Select a spectrum.* – After a little while, the results will be in. If there is a point on the object you're interested in, click on it; if not, try a different result star.

After clicking on a point from your spectral result plane, you should see at least one matching spectrum in the result area below Aladin's image display (you may need to pull it up if there are just three little arrows instead of a result area).

To view the mid-resolution spectra, having an actual spectral analysis program like Splat would be handy. If you don't have it, TOPCAT will do, too, but in that case make sure you're steering clear of image/fits spectra, because TOPCAT can't deal with those.

To load a spectrum, look for button-like objects in the result display (cf. Fig. 6). Don't use previews here (which for the LAMOST DR3 service are the first button-like things). Instead, scroll right until you see the accref column. Click on an accref and tell Aladin to send your spectrum to Splat (or TOPCAT). You can plot it there, which yields a display like the in Fig. 6.

Comparing with Fig. 5, it's fairly clear that our test star is, indeed, a Carbon star. Of course, given that here we've even started with one in our main bunch, that doesn't exactly come as a surprise.

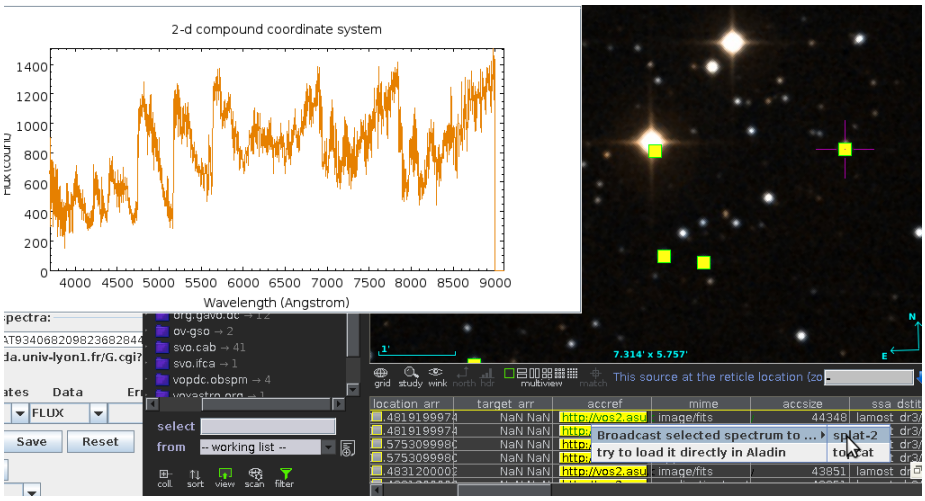


Figure 6: The LAMOST spectrum of our fbs1659-072606.51+215328.6 example, together with the Aladin result display for the LAMOST SSA-query and the action we used to send the spectrum to Splat.

Appendix A
Background Information

A.1 More on Low Dispersion Spectra

Even with the low resolution given by such surveys – we will be using the DFBS (see below) in this example, which has about 50 Å per bin –, one can distinguish many spectral lines (absorption lines of white dwarfs and some subdwarfs, emission lines of QSOs and other AGN, sometimes cataclysmic variables), and follow the spectral energy distribution and colors. The discoverabiltiy of the lines depends on their equivalent width – clearly, when summarizing flux over 50 Å range only very prominent features will be discernable in the spectra.

The large coverage of collections of low-dispersion spectra also helps the optical identification of non-optical sources, for instance from X-ray, IR, and radio observations. While direct images give only the positional difference and brightness, spectra give an understanding on the nature of objects, and having our knowledge on possible counterparts of the given source (AGN, CVs, WDs, late-type stars, etc. for X-ray sources, galaxies, QSOs, late-type stars, etc. for IR, and mostly extragalactic objects for radio). Even when the true counterpart is too faint and was not observed in the spectral survey, the spectra help to exclude the detected objects as non-relevant for the given source.

When, as in this use case, one wants to use low-dispersion spectra for object classification, there are a few object classes that are particularly well suited because of their unique spectral characteristics. For the DFBS data set, examples include:

- UV-excess (Markarian) galaxies; this was the main goal of the survey and 1515 such objects have been found (these are spectra much bluer that those of ordinary galaxies).
- Quasars (QSOs; blue spectra with (often) emission lines; high redshift QSOs are red spectra somewhat similar to late-type stars), other AGN and peculiar galaxies (blue spectra with (sometimes) emission lines).
- starburst galaxies, due to the length of the blue part.
- white dwarfs (very blue spectra with (often) absorption lines).
- hot subdwarfs (very blue spectra with (sometimes) absorption lines).
- cataclysmic variables (blue spectra with (sometimes) emission lines).
- central stars of planetary nebulae (very blue spectra with (sometimes) absorption lines, similar to WDs and hot sd-s).
- planetary nebulae (emission lines).
- late-type M stars (short spectra with very weak tail to the blue part; absorption bands are also observed).
- carbon stars (C; short spectra with sharp edge, very often resembling dots for faint objects; bright objects show spectra similar to triangles; absorption bands are also observed).

A.2 Some Background on the DFBS

In this use case, we will use the Digitized First Byurakan Survey (DFBS) as the basic spectra collection. It is the digitized version of the famous Markarian survey, also called the First Byurakan Survey (FBS). As of 2019, it is still the largest spectroscopic database in the world, providing some 42,000,000 low-dispersion spectra.

The FBS has been carried out by Markarian, Lipovetski and Stepanian in 1965-1980 on the Byurakan Astrophysical Observatory (BAO) 102/132/213cm (40"/52"/84") Schmidt telescope with 1.5° prism. 2050 Kodak IIAF, IIaF, IIF, and 103aF photographic plates in 1133 fields ($4° \times 4°$ each, the size being $16\,\text{cm} \times 16\,\text{cm}$) have been taken. The FBS covers $17,000$ square degrees of the sky at high galactic latitudes ($|b| > 15°$); due to the telescope location, only declinations north of $\delta = -15°$ are covered.

The limiting magnitude of the survey is $16.5^{\mathrm{m}} \cdots 19.5^{\mathrm{m}}$ in V depending on the plate. For the majority of the plates it is $17.5^{\mathrm{m}} \cdots 18^{\mathrm{m}}$. The dispersion is about 1800 Å/mm near Hγ and 2500 Å/mm near Hβ. The spectra obtained from the plates cover the range $3400 \cdots 6900$ Å, and there is a sensitivity gap near 5300 Å, dividing the spectra into red and blue parts. Each FBS plate contains low-dispersion spectra of some 15,000 to 20,000 objects.

Note that Gaia's RP/BP spectra will of course reflect the technical progress of 50 years over DFBS – they will be much more homogeneous, better calibrated, and there will be about a hundred times more of them. So – if it works just so with DFBS, it'll work marvellously with Gaia!

Appendix B
Going Further: Confirming Doubtful Cases

While we would bet that there *are* misclassifications even in respected sources like the General Catalogue we have been working with so far, it is of course much more interesting to try and confirm doubtful cases. Like, for instance, things marked as doubtful in Simbad.

So, let's repeat the exercise we just did with candidates for carbon stars from Simbad. We will query Simbad through TAP. So, in TOPCAT, go to VO → TAP Query. Enter `Simbad` in Keywords and hit Find Services. Double-click on the label for Simbad TAP.

As you can see when inspecting the columns for Simbad's `basic` table, there is a column otype, which has more or less magic values. To see what object classes Simbad might have for our problem, query the `otypesdef` table, perhaps like this:

```
SELECT * FROM otypedef
WHERE otype_longname like '%Carbon%'
```

If you're new to TAP and wonder how we figured out the `otype_longname`: Select the table in TOPCAT's table browser and check the Columns tab. And LIKE does a wildcard query. A bit funkily, SQL's "any character" wildcard (the shell's question mark) is the underscore, whereas "zero or more characters" (the asterisk in the shell) is the percent sign.

You'll see that the class for carbon star candidates is `C*?`. For each of these we want a position (that's what we use to search in DFBS) and the `main_id` if we ever want to refer back to Simbad with our findings.

```
SELECT main_id, ra, dec
FROM basic
WHERE otype='C*?'
```

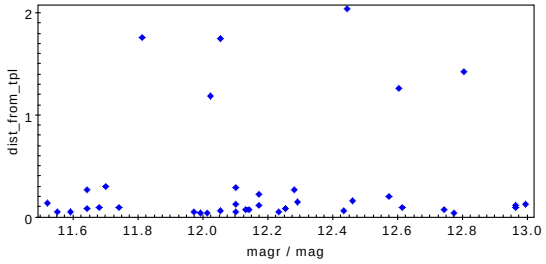


Figure 7: A plot of distance from the template spectrum over the estimated r magnitude of the generating objects. Do you have an interpretation for the... well, two populations here?

With this list, repeat what we've been doing in Sections 3 and 4; you will, of course, have to slightly modify the spectra-fetching query because we want to keep different columns this time (essentially, `main_id`) and we need to qualify the `ra` and `dec` columns because that name now is in both the remote and our local table. This yields something like:

```
SELECT
   accref, spec_id, db.ra, db.dec, flux,
   magb, magr, main_id
FROM dfbsspec.spectra 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.))
```

For the magnitude cut we do to cope with the nonlinear response of photographic emulsions, let's again create a subset; `magr>11.5 && magr<13` should do nicely again. Don't forget to select "sample" (or whatever you used as the subset name) in Row Subset in TOPCAT's main window.

Save that, run the scoring, and join the scores as before. Then do a plot of `magr` versus `dist_to_tmpl`. The result is fairly suggestive (Fig. 7). If you, by the methods developed above or any other means, manage to figure out the difference between the ones with the small deviations and the ones up in the air, that'd be great. Be sure to let us know[4].

Appendix C
Server-Side Object Selection

If you want to have server-side constraints (like the magnitude cut) with the upload joins, you'll notice queries to which you simply add a `WHERE` run forever (actually, you'll have to change to async mode in order to escape the timeout). This is because the query planner (a component of every database system)

[4] msdemlei@ari.uni-heidelberg.de

here believes it's much smarter to first filter out all objects of, say, magnitude between 11.5 and 13, and only then use our object list, which would be much faster. That's something of a bug, but it's non-trivial to fix it.

To work around it, modern TAP servers give you WITH, a SQL construct useful for many things, but in particular for dealing with planner failures. What's within a WITH gets planned and executed separately. So, re-writing the query as:

```
WITH sample AS (
SELECT
   accref, spec_id, ra, dec, flux,
   magb, magr, recno, "Names"
   FROM dfbsspec.spectra AS db
   JOIN TAP_UPLOAD.t1 AS tc
   ON 1=CONTAINS(POINT('ICRS', db.ra, db.dec),
        CIRCLE('ICRS', tc."_RAJ2000", tc."_DEJ2000", 5./3600.)))
SELECT * FROM sample
WHERE magr BETWEEN 11.5 and 13
```

(again, adjust the "t3" as necessary) will make it run quickly (in a matter of seconds).

Note that the query will also be faster if you reduce the amount of data to upload and ingest. For our purposes, we only need the `_RAJ2000`, `_DEJ2000`, `recno`, and `Names` columns from the catalog. You'll save time and (server-side) CPU if you restrict the upload to these columns. To do that, hit Views → Column Info in TOPCAT's main window, and in the window popping up Columns → Hide all Columns. Then in the Visible column check the columns we need. Incidentally, most of the actions can also be operated from buttons, but it's harder to describe these, which is why we're mentioning the menu entries here (as in the rest of this tutorial).

Find me on VO Text Treasures http://dc.g-vo.org/VOTT