P_q calculator walkthrough

Rhys Barnett
Imperial College London

version 1.1
July 1, 2020

Contents

1 Overview

This Python code applies the Bayesian model comparison (BMC) technique laid out in Mortlock et al. (2012) to survey photometry, to identify good high-redshift quasar candidates. This is a more efficient way to identify high-redshift quasars than colour cuts, or SED model fitting. For fuller discussion of the BMC and comparisons against other methods see, e.g, Mortlock et al. (2012) and Barnett et al. (2019).

The code is run via command line and is designed to be flexible in terms of the accepted photometry options. It can handle input photometry from a number of different telescopes and filters, presented as logarithmic or arcsinh magnitudes, fluxes, or lower magnitude limits (see Sect. 3.2). The latter is a less informative option; measured magnitudes or linear fluxes and a photometric uncertainty are preferable whenever possible.

The data should be given in a .csv formatted file in the `/Pq_server/data/` directory. See Sect. 3.1 for guidance as to how to format the .csv file headings correctly. For each source, the code will return respective high-redshift quasar (HZQ), MLT dwarf, and early-type galaxy (ETG) population weights (W_q, W_s, W_g), as well as the combined quasar probability

$$P_q = \frac{W_q}{W_q + W_s + W_g},$$

which is the key output for a single source. The code can also produce a couple of diagnostic plots for individual sources which may be useful for further investigation of interesting candidates (Sect. 6).

For the remainder of this guide the user is assumed to be in the `/Pq_server/` directory as a start point – this will be indicated using `./` from now on.

2 Quick start

This section briefly outlines the main steps needed to make full use of the code. Further details on each step are given in the relevant sections of this document.

1. Put a correctly-formatted .csv file in the `./data/` directory (or a sub-directory thereof; Sect. 3.1).

2. Check the parameters in the `./options/populations.options` file match your requirements (Sect. 4). *OPTIONAL: If your data file contains asinh magnitudes in a system other than SDSS, you need to also define the zero-flux magnitudes in the file* `./options/asinh_m0.values`

3. Install the supplied Python environment (or use your own, with relevant libraries installed; see Sect. 5.1).

4. With your Python environment activated, run the bash script `./scripts/probability.sh` with suitable input/output filepath arguments (Sect. 5.2).

5. The code will display a progress bar as it works through the candidates in your input file, and output the results where specified once complete. The code does NOT subsequently rank candidates or take any cuts on P_q - this is for the user to carry out separately.

6. If desired, diagnostic plots of the best fitting population models, and the calculated population weights for any candidate can be produced using the `./scripts/plots.sh` script, with the input set as the results file from the probability script (Sect. 6.2). As part of this code the user is asked to specify the candidates of interest by index - it does not run automatically on all sources.

3 Data requirements

This section outlines the data file format required by the code. In terms of what you supply, the minimum requirement is a *J* band measurement from either *Euclid* or UKIDSS/VISTA (COSMOS UltraVISTA *J* band data is also accepted). These filters are sometimes referred to as the 'main' filter in subsequent sections of this guide. However, do not expect insightful results from a single band! Ideally you will have a number of filters for each source – among others, a deep z-band measurement is very useful for discriminating quasars from MLTs and galaxies.

3.1 The input data file

This section outlines how the input photometry file should be formatted to get the P_q calculation to work correctly. The details in this section should be followed precisely, as if the input file columns are not set up correctly an error is likely to occur. The code accepts photometry in a `.csv` format.

Please observe the following:

- The first line of the file should list the column headings. A delimiter is not required.
- The first two columns should be `ra_x` and `dec_x`. Use `x = d` if decimal coordinates are provided or `x = s` if the *hh:mm:ss.ss dd:mm:ss.ss* format is being used. The code expects the coordinate format to be the same for all sources in a given run.
- Either *Euclid* or MKO *J*-band photometry *must* be present for each source for the code to execute. The population density models (Bayesian priors) make use of the *J* band, and model colours are determined relative to *J*. MKO photometry from the UKIDSS Large Area Survey (LAS), the VIKING survey, or UltraVISTA (which is used in COSMOS2015) is supported.
- For each filter for which photometry is provided, there should be a further two columns: a *measurement* column and an *uncertainty* column.
- The *measurement* column name format is `filter_system_phottype`. Valid values for the `filter`, `system` and `phottype` parameters are given below.

- For the corresponding *uncertainty* column, the naming convention is `filter_system_err`, with the `filter` and `system` parameters matching the *measurement* column name. Use `err` at the end regardless of the type of photometry – this is dealt with on the basis of the `phottype` parameter in the *measurement* column.

- Where a full photometric measurement is not available, limits (upper flux, or lower magnitude/luptitude) can be provided by putting the limit in the *measurement* column and `nsig` in the *uncertainty* column where `n` is the significance of the limit. You do not need to provide separate columns or labels for photometry and limits as long as everything is on the same system (e.g., fluxes and flux limits only in one column). The effect of using limits instead of a full photometric measurement and uncertainty is discussed in Mortlock et al. (2012), see also Sect. 3.2.

- Simply leave empty *measurement* and *uncertainty* values if photometry is missing in a band for a given source.

- Use separate sets of columns if, e.g., fluxes are available for some sources and AB mags for another, leaving the missing columns empty as appropriate.

- Filters can be repeated and both used in the calculation if two independent measurements exist – provide separate sets of columns. Again entries can be left blank for sources that do not have two measurements in that filter.

- From the final two points above, it follows that column names may occur more than once in your input file (especially the *uncertainty* columns which always have the `_err` label). This is fine and is handled internally by the code. However, to aid human readability, additional characters at the end of column names are ignored[1].

[1] I have not fully stress-tested this so do not make column names too complicated! But, e.g., `J_EUC_abmag_2` should be interpreted properly.

The following `filter` and `system` parameter values are accepted for use in the BMC code:

`filter`	`system`	Comments
i, z	SDSS	
i, z, y	LSST	
i, z, y	PS	Pan-STARRS; colours are almost identical to LSST but can still be quoted separately.
z, Y	DEC	Dark Energy Camera
z, y	COSMOS	Subaru Suprime z' with full transmission (aka z^{++}); HSC *y*. COSMOS uses NIR data from UltraVISTA – MKO filters below are suitable.
Z, Y, J, H, KS	VIK	VISTA VIKING survey
Y, J, H, KS	UV	UltraVISTA survey (used in COSMOS2015)
Y, J, H, K	LAS	UKIDSS LAS
O, Y, J, H	EUC	*Euclid; O* refers to the VIS wide filter *R* + *I* + *Z*
W1, W2	WISE	

The following `phottype` parameter values are accepted:

`phottype`	Photometry format
flux	linear fluxes
abmag	AB logarithmic magnitudes
vegamag	Vega logarithmic magnitudes
ablup	AB arcsinh magnitudes (luptitudes)
vegalup	Vega arcsinh magnitudes (luptitudes)

As noted above, in all cases an upper flux/lower magnitude limit can be handled by supplying `nsig` in the corresponding `err` column.

So, e.g., to provide AB *J* band magnitudes from *Euclid* the *measurement* and *uncertainty* columns should be labelled `J_EUC_abmag` and `J_EUC_err` respectively.

Column names are case sensitive.

Example input file

```
ra_d,dec_d,z_LSST_abmag,z_LSST_err,z_LSST_abmag,z_LSST_err,Y_EUC_abmag,Y_EUC_err,J_EUC_abmag,J_EUC_err,H_EUC_abmag,H_EUC_err
8.86499554,-24.56567992,25.152,0.274,24.9,5sig,22.719,0.067,21.325,0.019,21.17,0.016
350.03626123,-51.90094678,25.201,0.287,,,20.551,0.009,20.369,0.008,,
33.37827355,-12.62661049,25.455,3sig,,,22.134,0.039,21.851,0.03,21.68,0.026
```

Things to observe from the column headings/data:

- Coordinates have been provided in decimal degrees (note the trailing `_d`'s).
- This dataset includes the LSST z band, and *Euclid YJH* filters.
- All photometry is in AB magnitudes (note the trailing `_abmag`'s).
- The first candidate has two LSST z measurements. These are included in separate pairs of columns with the same heading. For the final two candidates, which only have one z measurement each, the second entries are just left blank.
- The second candidate does not have a H-band measurement – the entries are again simply left blank.
- In the first row, the second LSST z measurement is a $5\,\sigma$ upper limit. In the third row a $3\,\sigma$ upper limit is provided for the LSST filter. These do not necessarily need their own column, as long as the limit is on the same magnitude system as other photometry in that filter.

3.2 Upper flux/lower magnitude limits

It is more informative to provide a photometric measurement and its error, even if the background-subtracted counts are very low, or negative. However, it may not always be possible to supply these values, in which case a limit can be provided instead. For the avoidance of doubt this section spells out what is meant by a limit.

As described above, if you wish to include a limit, the required input is the limit which we now label *lim*, and `nsig`, where `n` is the significance of the limit, i.e., a detection threshold. Explicitly, *lim* represents the magnitude or flux of a source that would be detected with significance S/N = `n`.

By providing a limit, the implication is that the source is fainter than this value (hence we might equivalently describe an upper flux limit, or a lower magnitude limit). Beyond this, however, there is no information about the flux of the source;

only the photometric uncertainty is known. If a lower magnitude limit is quoted, the likelihood becomes the probability that the measured flux is below the detection threshold. This changes the functional form of the likelihood to the error function (assuming the flux errors are distributed as a Gaussian).

3.3 Model colours

The code comes supplied with model colours suitable for all the filters listed in the table above. The `./models/` subdirectories contain the respective AB colours for each population needed for the BMC code in .ab ascii files as functions of redshift for quasars and galaxies, and spectral type for the MLTs.

To make the probability calculation as fast as possible, these colours are read in by the code, interpolated, and stored as look-up tables. This is also the case for the luminosity distance and comoving volume, which are needed as functions of z for the W_q calculation.

To save the user having to wait for the look-up tables to be created every time the code is run, the tables are created once and then stored for future calls of the code. This is facilitated using the *shelve* library.

For each population there should also be a `./models/pop/db/` subdirectory containing the following files:

pop.db.bak
pop.db.dat
pop.db.dir

These are the lookup tables of the colours (and cosmological quantities in the case of HZQs) to be read in by *shelve*.

If these files are missing they will be created automatically by running the probability code; they are then available for subsequent runs. Changing the cosmology (Sect. 4) will also write values to these files, and the code will hence load more slowly the first time a given cosmology is used.

4 Option files

In this section a brief overview of the files in the `./options/` folder is provided.

4.1 populations.options

This section describes the parameters in the `./options/populations.options` file. The user should always check these settings before running the code to ensure sensible probabilities for a given run.

Quasar template distribution

There are three sets of parameters in this section, which control the quasar templates used in the W_q calculation.

1. The first four parameters begin with `l` and refer to the distribution of linewidths in the quasar template. Emission lines in the templates have fixed ratios, with different templates defined by the C IV EW.

2. The next three parameters begin `c` and refer to the continuum slope of the quasar template, defined by the ratio of fluxes at 1315 Å and 2225 Å.

3. The final parameter is `nz` which can either be set to 0 or 1. Setting it to 1 will add a 3 Mpc (proper) near-zone to the template. In practice this makes only a small difference to the template colours.

The linewidths and continuum slopes are provided in the table below, along with a suggested distribution provided by Paul Hewett. For each combination `lx_cx` there is a quasar template, with or without a near-zone, i.e., 24 sets of colours are provided in total. The distribution of the four linewidths must sum to 1, as must the sum of the three continuum slopes. Values of W_q are calculated for each template and these values are weighted according to the specified fractions.

At present it is not possible to change the distribution of quasar templates with redshift, although there is some evidence that weaker-lined objects are more common at high redshift (Shen et al. 2019).

The code does not include any 'extreme' templates such as BAL objects, although the colours of these may be quite well approximated by using the `l0` templates. The user may wish to investigate this independently!

label	description	comments	suggested fraction
`l0`	zero lines	continuum emission only	0
`lh`	half line width		0.3
`ls`	standard line width	EW(C IV) = 39.1 Å	0.6
`ld`	double line width		0.1
`cb`	blue continuum	$f_{1315}/f_{2225} = 1.16$	0.05
`cs`	standard continuum	$f_{1315}/f_{2225} = 1.0$	0.7
`cr`	red continuum	$f_{1315}/f_{2225} = 0.84$	0.25

Quasar weight integration limits

Use the `zlo` and `zhi` parameters to set the redshift range over which W_q is calculated. This should be thought of as the redshift range you are interested in. That said, to maximise the quasar weight it is recommended to integrate slightly beyond the redshift range of interest in case the quasar posterior peaks near the boundary. For example, if you are interested in $z = 7-9$ you could set `zlo 6.9` and `zhi 9.1`.

IMPORTANT At present calculations require a J-band detection. This implies upper redshift limits of $z \sim 10$ for MKO and $z \sim 11.5$ for Euclid. Beyond these limits, Lyman α leaves the J band, the predicted $f_J \to 0$ which is not consistent with a J-band detection anyway $P_q \to 0$, and k-corrections become unreliable. If `zhi` is set beyond these limits the code will automatically change the upper integration bound to the relevant maximum, depending on which J band is provided. If higher-redshift quasars are of interest the code will need to be extended to use H band colours/k-corrections/density models.

Quasar colours are currently provided for $z > 5.5$ so in theory it is possible to investigate any redshift above this value. However, the contaminant weights are only ever calculated for dwarf stars in the range M0 – T8 and ETGs between $z = 1-2$, which is relevant for $z \gtrsim 6.5$. Probabilities will be less reliable for $z \lesssim 6$ where the contamination may be different.

Cosmology

These cosmological parameters are needed to calculate luminosity distances and comoving volumes, which are required to calculate W_q. Explicitly:
`Cosmology_h` = h, where $H_0 = 100\,h$ km/s/Mpc.
`Cosmology_Omega_M` = Ω_m

`Cosmology_Omega_k` = Ω_k
`Cosmology_Omega_L` = Ω_Λ

MLT scatter

The quasar and ETG colours are determined from template spectra. In contrast, the MLT colours are median values determined from a range of sources. As such there is a measured scatter in the colours. The `MLT_scatter` parameter accounts for this, and should be stated in magnitudes.

There are two ways in which this parameter should be considered. The first is to add an additional uncertainty in quadrature in the MLT likelihood calculation. The effect of this is to increase the MLT weight relative to the other populations, and consequently there is an adverse effect on quasar selection. To calculate P_q values in the past we have in fact used a default of `MLT_scatter 0.00` which maximises P_q values and produces the most inclusive candidate list possible.

However, when simulating MLT sources to assess potential contamination from a survey (a planned addition to the code), a value of `MLT_scatter` ~ 0.05 should be considered to represent the fact intrinsic MLT colours do not follow a single locus. This will give a more representative sample of simulated measured MLT colours once survey noise is added.

4.2 asinh_m0.values

Important: at present the supplied VIKING asinh mag conversions are unreliable and do not consistently result in correct fluxes. I'm hoping to get this issue sorted in future versions. Try to supply VIKING fluxes instead - which should be possible if you also have access to the luptitudes.)

The file `./options/asinh_m0.values` is only relevant if you are using photometry on the asinh magnitude system (luptitudes).

The code can handle SDSS *iz* and VIKING $ZYJHK_s$ luptitudes automatically. If your data only includes these luptitudes you can also skip this section.

Traditional logarithmic (Pogson) magnitudes are defined using a zero-magnitude flux. On the AB system this is equal to 3631 Jy, while on the Vega system the zero point is different for each filter. These values are included already for all supported bands.

However, luptitudes also require a second parameter which in the literature is often expressed as a *softening parameter, b*, or equivalently as a zero-flux magnitude, $m_0 = -2.5 \log_{10} b$.

m_0 should be specified for any additional filters for which you wish to provide asinh magnitudes in the BMC code. You should specify one filter and m_0 value per line in `./options/asinh_m0.values`, where the filter format is as described in Sect. 3.1 but without a *phottype*, e.g.,

```
i_SDSS 24.36
z_SDSS 22.83
```

5 Running the code

This section outlines the Python environment needed to run the P_q code, and how to run the main probability code from the command line.

The P_q calculation is quite involved and can take up to 0.5 s for a single source on a standard desktop setup. A few seconds are also required at the start of every run, to verify the input data file and load the required model colours.

5.1 Python environment

For the most part this code uses well-known Python libraries such as *numpy* and *scipy* which may already be installed on your system. However, the quickest and safest way to get started is to use the supplied file `./env/pq_code_env.yml` – this file can be used to create a Python environment with all the required packages, versions, and dependencies within which the code can be run.

It is assumed you have conda installed on your system[2]. In the terminal, change to the `/Pq_server/` directory and extract the environment using

```
conda env create -f env/pq_code_env.yml
```

You can use the `-n` option to specify a different name for this environment; but the assumed (for this guide) default is `pq_code_env`. Once installed, The conda environment can be activated, and later shut down, using

```
source activate pq_code_env
```

and

```
source deactivate
```

With the environment active you will be ready to run the code using the supplied bash scripts as described in Sect. 5.2.

[2] https://docs.conda.io/projects/conda/en/latest/user-guide/install/index.html

Otherwise: If you wish to build your own environment from scratch, you will need the following libraries and their dependencies. The versions listed match the conda environment described above, and should definitely work with each other.
python 3.6.2
numpy 1.13.1
scipy 0.19.1
matplotlib 2.0.2
astropy 2.0.1
pandas 0.20.3
os 0.1.4
tqdm 4.43.0
Python 3.6 or greater is required as the code makes use of f-strings[3].

Important: Make sure you are **not** using *scipy 0.19.0.* There is a very nasty memory leak bug in this version which affects the *scipy.integrate.quad* function. This can called over 500 times for a single source so can quickly result in a difficult-to-fathom failure of the whole code.

[3]https://docs.python.org/3/whatsnew/3.6.html. If you are not already using a recent-enough version of Python, f-strings are a highly recommended reason to upgrade!

5.2 Calculating probabilities

Before running, check the settings in the `./options/populations.options` file are satisfactory. There are a number of flexible parameters in this file which are needed for the Wq and Ws calculations.

Make sure the input photometry file is located in the `./data/` directory (or a custom sub-directory thereof). Then, from the command line, load your Python environment and run the bash script `./scripts/probability.sh` as follows:

```
$ bash scripts/probability.sh input_file output_file
```

Both the `input_file` and `output_file` should include the full path to the file starting in the`./data/` directory. In fact the arguments are optional, but note the following:

- If neither argument is given, the you will be asked to specify an input file located in the `./data/` directory. You can then choose to specify an output file which will be created in the `./data/` directory; alternatively, you can choose to just append the results to the input file.
- If one argument is given, the code assumes this is the input file. In this case you *will not* be asked to specify an output file and results will just be appended to the input automatically.
- If two arguments are given, the first is assumed to be the input and the second is assumed to be the output file, which will be created in the `./data/` directory.
- The code always checks the input file exists in the `./data/` directory before proceeding. If the file is not found you will be asked to re-specify the file path. Output folders will be automatically created if they do not already exist.

A couple of examples:

`$ bash scripts/probability.sh euclid/fluxdata.csv` will look in the directory `./data/euclid/` for the file `fluxdata.csv`, and append the results to that file.

`$ bash scripts/probability.sh fluxdata.csv results/euclid.csv` will look for the file `fluxdata.csv` in `./data/` and output the results to `./data/results/euclid.csv`.

6 Outputs

This section outlines the different outputs available for a quasar candidate.

6.1 Output columns

Whether the user has chosen to add the output to the original photometry file, or a new output filepath, the end result will be a file containing the input coordinates and photometry, and either five or seven additional columns. These are:

ra_d, dec_d: Coordinates will be repeated in decimal degrees only if originally supplied in *hh:mm:ss.ss, dd:mm:ss.ss* format.

sin *b* : This is the absolute value of the sine of the Galactic latitude, calculated from the input coordinates. It is needed for the W_s calculation, as the MLT density decreases as you move vertically from the Galactic plane.

$\mathbf{W_q}, \mathbf{W_g}, \mathbf{W_s}$: These are the values of the individual population weights. By themselves they are not especially informative, especially if the photometry comprises a mixture of upper limits and fluxes/magnitudes, as the likelihood functions differ in these two cases.

$\mathbf{P_q}$: This is the calculated probability based on the combined weights. It is recommended to use the P_q column as a ranking system for candidates. It is not right to think of this column as the absolute probability that that source is a quasar, it just reflects the strength of the quasar weight against two other populations.

The threshold used to select candidates is typically decided empirically and depends on the length of the candidate list you wish to deal with afterwards. This should be assessed against contamination rates (which could be explored through simulations of the MLT/ETG populations for example). However a threshold as low as $P_q = 0.05$ is certainly within the realms of possibility.

6.2 Diagnostic plots

The code has the functionality to produce some diagnostic plots to explore interesting candidates in more detail. This may be useful to, e.g., identify candidates that are in fact very poor fits to all three sets of population models, and their high P_q values arise only by chance. To do this, in your Python environment, call the `./scripts/plots.sh` script as follows:

```
$ bash scripts/plots.sh input_file
```

Where the `input_file` argument is the path to a file found in the `./data/` directory, which as in the case of the probability calculation, may include subdirectories. If you do not specify this argument you will be asked to do so on the command line. It is recommended that this file already has probability information included; however it will work without.

After the code has set up the template colours etc., you will be asked to specify the rows of the candidates you are interested in. Do not worry about Python indexing here: entering `1` will analyse the first candidate in the photometry file. As indicated by the command-line display, you can enter a single number, comma-separated row values, or a range `first-last`.

Expected output

The code will automatically create a subdirectory in the location of your input file, named `/input_file_output/`. Within this folder, there will be a pdf file for each candidate you selected, with two diagnostic plots:

1. Minimum-χ^2 model fits to the input photometry. This figure shows the single-best fitting SED from each population, as fit to the photometry in Jy. It is a useful indicator as to whether a source with a high P_q value is well fit by a quasar model, or if the high probability rose by chance (bad fits to all models). If upper limits are supplied these will be displayed, but are not used in the χ^2 calculation as there is no measured flux in this case.

   This plot quotes a few useful values for the candidate, including the best fitting template, the reduced χ^2 model, and the results of the BMC calculation if these are provided (this plot does not calculate P_q values anew).

2. Bayesian estimate of the best redshift/spectral type. Integrating the curves with respect to redshift (or summing over spectral types for the MLTs)

yields the weights quoted in the output file. The best SED from χ^2 fitting may not completely agree with the peak of these curves, depending on the prior. For instance, the quasar surface density declines with redshift, so the best model from a Bayesian perspective may be at a slightly lower redshift than the model suggested from minimum-χ^2. These curves tend to be sharply peaked and if the quasar curve is pushed against your upper or lower redshift boundary, you may wish to consider widening the redshift range of interest slightly to maximise the quasar weight.

7 F(?)AQs

Can I use this code to produce selection functions?

Not yet, but I would like to add this soon. If you are in a hurry, producing the selection functions seen in, e.g., the *Euclid* paper Barnett et al. (2019) requires three main steps. Namely:

1. Generate a large grid[4] of quasars in luminosity/redshift space. Use model k-corrections and colours to generate 'true' photometry in each band, and use the noise properties of the relevant surveys to produce noisy observed photometry. (Coordinates should also be generated from the survey footprint). This step is not supported in the code yet.

2. Calculate P_q for all these objects. As long as the quasars are listed in a .csv file this can be done with the code already.

3. Take a cut on P_q and produce a contour plot of the proportion of objects that pass as a function of luminosity/redshift. Producing this plot is not supported in the code yet.

Can I use a custom surface density model for one or more of the populations?

Yes - but a little coding work will be required. In the folder `./py/` you will find the custom modules `etg.py` (galaxies), `mlt.py` (MLTs), and `hzq.py` (quasars). In the relevant module, write your custom density function. You then need to find the `_popintegrand()` method (where `pop` is the name of the module you are working on). This function is just the product of the default surface density function and a likelihood function. Change the surface density function called there to your custom function.

Note the surface density functions are defined per steradian, rather than per square degree – it is important to get the normalisation right. It is assumed that your new surface density will still be a function of the *Euclid* or MKO *J* band - if this is not the case you will need to check some of the additional steps below.

[4]For publications I have used grids of 500 quasars in each cell of $\Delta z = 0.1, \Delta M_{1450} = 0.1$ but this can be relaxed for a 'quick look'.

Can I extend this code to other surveys?

Yes - although this is a fairly significant undertaking. I would first strongly suggest you check if your new survey can be approximated by the filters that are already available. If you still wish to proceed, the steps to work through are as follows:

1. Determine model colours for the HZQ, ETG and MLT templates. Add these to the .ab files in the `./models/` folders. If you wish to use a new 'main' filter, i.e., you no longer wish to base the calculation on the *Euclid* or MKO *J* band, you will also need new quasar k-corrections based on your new main filter.

2. Add the Vega magnitude zero points (if needed) and effective wavelengths for your new filters to the lookup tables towards the end of the `dataload.py` module in `./py/`. If you wish to base the calculation on a new main filter you will also need to change the `_J_filter()` function in this module to accept your new main filter. TIP - labelling your new main filter as either the *Euclid* or MKO *J* band will make the following code changes a bit less painful - but make sure the zero point and wavelength have been changed correctly.

3. In each of the `./py/` modules `etg.py`, `mlt.py`, and `hzq.py` you will need to edit the `_colour_shelf()` function to accept your new colours. If you are not changing the main filter you will need to make sure the functions are providing $X - J$ colours, where X is each of your new filters. If you are changing the main filter then extensive checks and changes will be needed here, to relate all filters to your new main filter. As noted above it may therefore be useful to keep the label of this new main filter as *J*.

4. Delete the files in the `./models/pop/db/` subdirectory for each population to include the new bands in future runs of the code (Sect. 3.3).

5. If you are changing the main filter, determine new surface density models for each population and implement these in each module. By keeping the label as *J* you save yourself the pain of having to modify the other functions related to the weight calculation. You should however check the upper redshift limits are still appropriate in the `_quasar_options()` method of the `hzq.py` module.

8 To-do list

This section just lays out a few additions to the code that will hopefully be implemented soon.

1. Add support for bluer optical bands
2. More comprehensive results from plots.py including best fitting model parameters both from SED fitting and BMC.
3. Ability to produce selection functions - full end-to-end process of simulating quasars (likely with a few assumed survey specifics), calculating P_q, and producing contour plots.
4. Add functionality to turn off individual contaminating populations
5. Simulating MLT and ETG populations, and calculating probabilities to estimate contamination rates.

Please report code issues and additional desirable filters/functionality ideas[5] and I will endeavour to update the code regularly! Suggestions/improvements for this document are also gratefully received.

[5] rhys.barnett09@imperial.ac.uk