



sherlock Documentation

Release v2.2.0

Dave Young

2021

TABLE OF CONTENTS

1	Features	3
1.1	Installation	3
1.1.1	Development	3
1.2	Initialisation	4
1.2.1	Modifying the Settings	4
1.2.2	Basic Python Setup	4
1.3	Todo List	4
1.4	Release Notes	9
2	API Reference	11
2.1	Modules	11
2.1.1	commonutils (<i>module</i>)	11
2.1.2	imports (<i>module</i>)	12
2.1.3	utKit (<i>module</i>)	13
2.2	Classes	13
2.2.1	catalogue_conesearch (<i>class</i>)	14
2.2.2	update_wiki_pages (<i>class</i>)	17
2.2.3	database (<i>class</i>)	18
2.2.4	database_cleaner (<i>class</i>)	19
2.2.5	ifs (<i>class</i>)	20
2.2.6	ned (<i>class</i>)	21
2.2.7	ned_d (<i>class</i>)	23
2.2.8	veron (<i>class</i>)	24
2.2.9	transient_catalogue_crossmatch (<i>class</i>)	26
2.2.10	transient_classifier (<i>class</i>)	30
2.3	Functions	35
2.3.1	get_crossmatch_catalogues_column_map (<i>function</i>)	35
2.3.2	getpackagepath (<i>function</i>)	37
2.4	A-Z Index	37
3	Release Notes	39
	Python Module Index	41
	Index	43

mine a library of historical and on-going astronomical survey data in an attempt to identify the sources of transient/variable events, and predict their classifications based on crossmatched data.

Documentation for sherlock is hosted by [Read the Docs](#) ([development version](#) and [master version](#)). The code lives on [github](#). Please report any issues you find [here](#).

FEATURES

-

1.1 Installation

The easiest way to install `sherlock` is to use `pip` (here we show the install inside of a conda environment):

```
conda create -n sherlock python=3.7 pip
conda activate sherlock
pip install sherlock
```

Or you can clone the [github repo](#) and install from a local version of the code:

```
git clone git@github.com:thespacedoctor/sherlock.git
cd sherlock
python setup.py install
```

To upgrade to the latest version of `sherlock` use the command:

```
pip install sherlock --upgrade
```

To check installation was successful run `sherlock -v`. This should return the version number of the install.

1.1.1 Development

If you want to tinker with the code, then install in development mode. This means you can modify the code from your cloned repo:

```
git clone git@github.com:thespacedoctor/sherlock.git
cd sherlock
python setup.py develop
```

[Pull requests](#) are welcomed!

1.2 Initialisation

Before using sherlock you need to use the `init` command to generate a user settings file. Running the following creates a `yaml` settings file in your home folder under `~/.config/sherlock/sherlock.yaml`:

```
sherlock init
```

The file is initially populated with sherlock's default settings which can be adjusted to your preference.

If at any point the user settings file becomes corrupted or you just want to start afresh, simply trash the `sherlock.yaml` file and rerun `sherlock init`.

1.2.1 Modifying the Settings

Once created, open the settings file in any text editor and make any modifications needed.

1.2.2 Basic Python Setup

If you plan to use `sherlock` in your own scripts you will first need to parse your settings file and set up logging etc. One quick way to do this is to use the `fundamentals` package to give you a logger, a settings dictionary and a database connection (if connection details given in settings file):

```
## SOME BASIC SETUP FOR LOGGING, SETTINGS ETC
from fundamentals import tools
from os.path import expanduser
home = expanduser("~")
settingsFile = home + "/.config/sherlock/sherlock.yaml"
su = tools(
    arguments={"settingsFile": settingsFile},
    docString=__doc__,
)
arguments, settings, log, dbConn = su.setup()
```

1.3 Todo List

Todo:

- update the package tutorial if needed

(The *original entry* is located in `/home/docs/checkouts/readthedocs.org/user_builds/qub-sherlock/envs/master/lib/python3.7/site-packages/qub_sherlock-2.2.0-py3.7.egg/sherlock/catalogue_conesearch.py:docstring of sherlock.catalogue_conesearch.catalogue_conesearch`, line 25.)

Todo:

- write a checklist for adding a new catalogue to the sherlock database and reference it from here (use the image below of the `tcs_helper_catalogue_views_info` table)

(The *original entry* is located in `/home/docs/checkouts/readthedocs.org/user_builds/qub-sherlock/envs/master/lib/python3.7/site-packages/qub_sherlock-2.2.0-py3.7.egg/sherlock/commonutils/get_crossmatch_catalogues_col`

of `sherlock.commonutils.get_crossmatch_catalogues_column_map.get_crossmatch_catalogues_column_map`, line 7.)

Todo:

- update the package tutorial if needed
-

(The *original entry* is located in `/home/docs/checkouts/readthedocs.org/user_builds/qub-sherlock/envs/master/lib/python3.7/site-packages/qub_sherlock-2.2.0-py3.7.egg/sherlock/transient_classifier.py:docstring of sherlock.transient_classifier.transient_classifier`, line 26.)

Todo:

- add usage info
 - create a sublime snippet for usage
 - update package tutorial if needed
-

(The *original entry* is located in `/home/docs/checkouts/readthedocs.org/user_builds/qub-sherlock/envs/master/lib/python3.7/site-packages/qub_sherlock-2.2.0-py3.7.egg/sherlock/transient_classifier.py:docstring of sherlock.transient_classifier.transient_classifier._consolidate_coordinateList`, line 16.)

Todo:

- add usage info
 - create a sublime snippet for usage
 - write a command-line tool for this method
 - update package tutorial with command-line tool info if needed
-

(The *original entry* is located in `/home/docs/checkouts/readthedocs.org/user_builds/qub-sherlock/envs/master/lib/python3.7/site-packages/qub_sherlock-2.2.0-py3.7.egg/sherlock/transient_classifier.py:docstring of sherlock.transient_classifier.transient_classifier.classification_annotations`, line 16.)

Todo:

- add usage info
 - create a sublime snippet for usage
 - write a command-line tool for this method
 - update package tutorial with command-line tool info if needed
-

(The *original entry* is located in `/home/docs/checkouts/readthedocs.org/user_builds/qub-sherlock/envs/master/lib/python3.7/site-packages/qub_sherlock-2.2.0-py3.7.egg/sherlock/transient_classifier.py:docstring of sherlock.transient_classifier.transient_classifier.update_classification_annotations_and_summaries`, line 19.)

Todo:

- add usage info
 - create a sublime snippet for usage
-

- write a command-line tool for this method
 - update package tutorial with command-line tool info if needed
-

(The *original entry* is located in `/home/docs/checkouts/readthedocs.org/user_builds/qub-sherlock/envs/master/lib/python3.7/site-packages/qub_sherlock-2.2.0-py3.7.egg/sherlock/transient_classifier.py:docstring of sherlock.transient_classifier.transient_classifier.update_peak_magnitudes`, line 16.)

Todo:

- add usage info
 - create a sublime snippet for usage
 - write a command-line tool for this method
 - update package tutorial with command-line tool info if needed
-

(The *original entry* is located in `/home/docs/checkouts/readthedocs.org/user_builds/qub-sherlock/envs/master/lib/python3.7/site-packages/qub_sherlock-2.2.0-py3.7.egg/sherlock/transient_classifier.py:docstring of sherlock.transient_classifier.transient_classifier._create_tables_if_not_exist`, line 16.)

Todo:

- add usage info
 - create a sublime snippet for usage
 - write a command-line tool for this method
 - update package tutorial with command-line tool info if needed
-

(The *original entry* is located in `/home/docs/checkouts/readthedocs.org/user_builds/qub-sherlock/envs/master/lib/python3.7/site-packages/qub_sherlock-2.2.0-py3.7.egg/sherlock/transient_classifier.py:docstring of sherlock.transient_classifier.transient_classifier.generate_match_annotation`, line 25.)

Todo:

- update the package tutorial if needed
-

(The *original entry* is located in `/home/docs/checkouts/readthedocs.org/user_builds/qub-sherlock/envs/master/lib/python3.7/site-packages/qub_sherlock-2.2.0-py3.7.egg/sherlock/catalogue_conesearch.py:docstring of sherlock.catalogue_conesearch.catalogue_conesearch`, line 25.)

Todo:

- write a checklist for adding a new catalogue to the sherlock database and reference it from here (use the image below of the `tcs_helper_catalogue_views_info` table)
-

(The *original entry* is located in `/home/docs/checkouts/readthedocs.org/user_builds/qub-sherlock/envs/master/lib/python3.7/site-packages/qub_sherlock-2.2.0-py3.7.egg/sherlock/commonutils/get_crossmatch_catalogues_column_map.get_crossmatch_catalogues_column_map`, line 7.)

Todo:

- add an entry in the tutorial to clean database tables

```
"""python from sherlock.database_cleaner import database_cleaner db = database_cleaner(
    log=log, settings=settings
) db.clean() """
```

(The [original entry](#) is located in `/home/docs/checkouts/readthedocs.org/user_builds/qub-sherlock/envs/master/lib/python3.7/site-packages/qub_sherlock-2.2.0-py3.7.egg/sherlock/database_cleaner.py:docstring of sherlock.database_cleaner.database_cleaner`, line 15.)

Todo:

- update the package tutorial if needed
-

(The [original entry](#) is located in `/home/docs/checkouts/readthedocs.org/user_builds/qub-sherlock/envs/master/lib/python3.7/site-packages/qub_sherlock-2.2.0-py3.7.egg/sherlock/transient_classifier.py:docstring of sherlock.transient_classifier.transient_classifier`, line 26.)

Todo:

- add usage info
 - create a sublime snippet for usage
 - write a command-line tool for this method
 - update package tutorial with command-line tool info if needed
-

(The [original entry](#) is located in `/home/docs/checkouts/readthedocs.org/user_builds/qub-sherlock/envs/master/lib/python3.7/site-packages/qub_sherlock-2.2.0-py3.7.egg/sherlock/transient_classifier.py:docstring of sherlock.transient_classifier.transient_classifier.classification_annotations`, line 16.)

Todo:

- add usage info
 - create a sublime snippet for usage
 - write a command-line tool for this method
 - update package tutorial with command-line tool info if needed
-

(The [original entry](#) is located in `/home/docs/checkouts/readthedocs.org/user_builds/qub-sherlock/envs/master/lib/python3.7/site-packages/qub_sherlock-2.2.0-py3.7.egg/sherlock/transient_classifier.py:docstring of sherlock.transient_classifier.transient_classifier.generate_match_annotation`, line 25.)

Todo:

- add usage info
 - create a sublime snippet for usage
 - write a command-line tool for this method
-

- update package tutorial with command-line tool info if needed
-

(The [original entry](#) is located in `/home/docs/checkouts/readthedocs.org/user_builds/qub-sherlock/envs/master/lib/python3.7/site-packages/qub_sherlock-2.2.0-py3.7.egg/sherlock/transient_classifier.py:docstring of sherlock.transient_classifier.transient_classifier.update_classification_annotations_and_summaries`, line 19.)

Todo:

- add usage info
 - create a sublime snippet for usage
 - write a command-line tool for this method
 - update package tutorial with command-line tool info if needed
-

(The [original entry](#) is located in `/home/docs/checkouts/readthedocs.org/user_builds/qub-sherlock/envs/master/lib/python3.7/site-packages/qub_sherlock-2.2.0-py3.7.egg/sherlock/transient_classifier.py:docstring of sherlock.transient_classifier.transient_classifier.update_peak_magnitudes`, line 16.)

Todo:

- document `cl_utils` module
 - tidy usage text
-

(The [original entry](#) is located in `/home/docs/checkouts/readthedocs.org/user_builds/qub-sherlock/envs/master/lib/python3.7/site-packages/qub_sherlock-2.2.0-py3.7.egg/sherlock/cl_utils.py:docstring of sherlock.cl_utils`, line 7.)

Todo:

- add an entry in the tutorial to clean database tables

```
"""python from sherlock.database_cleaner import database_cleaner db = database_cleaner(
    log=log, settings=settings
) db.clean() """
```

(The [original entry](#) is located in `/home/docs/checkouts/readthedocs.org/user_builds/qub-sherlock/envs/master/lib/python3.7/site-packages/qub_sherlock-2.2.0-py3.7.egg/sherlock/database_cleaner.py:docstring of sherlock.database_cleaner.database_cleaner`, line 15.)

Todo:

- add usage info
 - create a sublime snippet for usage
 - write a command-line tool for this method
 - update package tutorial with command-line tool info if needed
-

(The *original entry* is located in `/home/docs/checkouts/readthedocs.org/user_builds/qub-sherlock/envs/master/lib/python3.7/site-packages/qub_sherlock-2.2.0-py3.7.egg/sherlock/database_cleaner.py:docstring of sherlock.database_cleaner.database_cleaner._create_tcs_help_tables`, line 25.)

Todo:

- nice!
-

(The *original entry* is located in `/home/docs/checkouts/readthedocs.org/user_builds/qub-sherlock/checkouts/master/docs/source/_template_.md`, line 1.)

1.4 Release Notes

v2.2.0 - August 2, 2021

- **FEATURE:** Default search algorithm now version-controlled and ship alongside code. Users can override the default algorithm if they need to.
- **FEATURE:** Hidden `database-batch-size` and `cpu-pool-size` settings added - power users can access and change in settings files.
- **ENHANCEMENT:** Speed improvements (5-10 times processing speed increase).
- **REFACTOR/FIX:** synonym match break now removed by default so a location now gets matched against all catalogues regardless of whether or not a synonym match is initially found (e.g. right on top of a catalogued star). Resolves edge cases where true SNe are mis-classified as VS as they are located just too close to a stellar source, or galaxy source mis-identified as stellar in one catalogue but correctly as galaxy in another.
- **FIX:** some URLs to NED objects, found in the human readable Sherlock annotations, that were not resolving now do.

v2.1.8 - April 16, 2021#bugFixFriday

- **REFACTOR:** catalogue quality weights now used *within* the merged source sets to sort the data. This fixed *some* mis-matched annotation issues.
- **REFACTOR:** updating db connection settings to use QUB jumpbox.
- **REFACTOR/FIX** Reduced associated source merging radius from 3 arcsec to 1 arcsec (1/9th of the original matching area). This seems to correct many of the cases in which 2 or more distinct catalogued sources were getting merged/blended into one association. This resulted in in-correct classification prediction and wrongly associated distances.
- **FIXED:** annotation mismatches. If transient is classified as a SN then the underlying sources is identified as a galaxy in the annotation.

v2.1.7 - September 29, 2020

- **FIXED:** lite version introduced a couple of small bugs. Fixed.

v2.1.6 - September 7, 2020

- **ENHANCEMENT:** transient human-readable annotations added to the `classifications` dictionary return of the `transient_classifier.classify()` method.
- **ENHANCEMENT:** a `lite` parameter has been added to the Sherlock `transient_classifier` object. When set to `True` only top-ranked crossmatches are returned with a limited column set.

- **ENHANCEMENT:** snapshot annotation is now added into the classification dictionary return. The classification dictionary is now `{'MyTransientId01': ['Type', 'Annotation'], 'MyTransientId02': ['Type', 'Annotation'],}`
- **REFACTOR:** small change to query that writes NED source magnitudes that vastly improves speed
- **FIXED:** small issue where a missing error in photoz was causing annotations to not complete

v2.1.5 - June 22, 2020

- **refactor:** tunnel changes due to new jump box installation at QUB
- **fixed:** some NED galaxies not reported if redshift does not exist but a semi-major axis is given

v2.1.4 - June 18, 2020

- **fixed:** an empty crossmatch set could cause a transient database update when not required

v2.1.3 - June 5, 2020

- **refactor:** stop sherlock checking for transient database triggers if running in non-update mode
- **fixed:** another sdss photoz issue where photoz ranking about specz

v2.1.2 - May 24, 2020

- **fixed:** merged result parameters are now merged correctly

v2.1.1 - May 21, 2020

- **fixed:** fixes to cl utils

v2.1.0 - May 20, 2020

- Now compatible with Python 3.*

API REFERENCE

2.1 Modules

<code>sherlock.commonutils</code>	<i>common tools used throughout package</i>
<code>sherlock.imports</code>	
<code>sherlock.utKit</code>	<i>Unit testing tools</i>

2.1.1 commonutils (module)

common tools used throughout package

Classes

<code>update_wiki_pages(log[, settings])</code>	<i>Update sherlock's github wiki pages with some useful info regarding the crossmatch database catalogue tables</i>
---	---

Functions

<code>get_crossmatch_catalogues_column_map(dbConn, log)</code>	<i>Summary the sherlock-catalogues helper tables to generate a map of the important columns of each catalogue</i>
<code>getpackagepath()</code>	<i>Get the root path for this python package - used in unit testing code</i>

Sub-modules

<code>get_crossmatch_catalogues_column_map(dbConn, log)</code>	<i>Summary the sherlock-catalogues helper tables to generate a map of the important columns of each catalogue</i>
<code>getpackagepath()</code>	<i>Get the root path for this python package - used in unit testing code</i>

continues on next page

Table 4 – continued from previous page

<code>update_wiki_pages(log[, settings])</code>	<i>Update sherlock’s github wiki pages with some useful info regarding the crossmatch database catalogue tables</i>
---	---

2.1.2 imports (module)

Classes

<code>ifs(log[, settings, pathToDataFile, ...])</code>	<i>Importer for the Multi Unit Spectroscopic Explorer (MUSE) IFS galaxy catalogue stream</i>
<code>ned(log[, settings, pathToDataFile, ...])</code>	<i>Using a list of coordinates, query the online NED database and import sources found within a given search radius of each of the loctions into the sherlock-catalogues database</i>
<code>ned_d(log[, settings, pathToDataFile, ...])</code>	<i>Import the *`NED-D <https://ned.ipac.caltech.edu/Library/Distances/>`_ *galaxy catalogue in to the sherlock-catalogues database</i>
<code>veron(log[, settings, pathToDataFile, ...])</code>	<i>importer object for the VERON AGN catalogue</i>

Sub-modules

<code>ifs(log[, settings, pathToDataFile, ...])</code>	<i>Importer for the Multi Unit Spectroscopic Explorer (MUSE) IFS galaxy catalogue stream</i>
<code>ned(log[, settings, pathToDataFile, ...])</code>	<i>Using a list of coordinates, query the online NED database and import sources found within a given search radius of each of the loctions into the sherlock-catalogues database</i>
<code>ned_d(log[, settings, pathToDataFile, ...])</code>	<i>Import the *`NED-D <https://ned.ipac.caltech.edu/Library/Distances/>`_ *galaxy catalogue in to the sherlock-catalogues database</i>
<code>veron(log[, settings, pathToDataFile, ...])</code>	<i>importer object for the VERON AGN catalogue</i>

2.1.3 utKit (module)

Unit testing tools

Classes

<code>utKit(moduleDirectory[, dbConn])</code>	<i>Override dryx utKit</i>
---	----------------------------

Sub-modules

<code>utKit(moduleDirectory[, dbConn])</code>	<i>Override dryx utKit</i>
---	----------------------------

class utKit (*moduleDirectory, dbConn=False*)

Bases: `fundamentals.utKit.utKit`

Override dryx utKit

get_project_root ()

Get the root of the ``python`` package - useful for getting files in the root directory of a project

Return

- `rootPath` – the root path of a project

refresh_database ()

Refresh the unit test database

setupModule ()

The setupModule method

Return

- `log` – a logger
- `dbConn` – a database connection to a test database (details from yaml settings file)
- `pathToInputDir` – path to modules own test input directory
- `pathToOutputDir` – path to modules own test output directory

tearDownModule ()

The tearDownModule method

2.2 Classes

<code>sherlock.catalogue_conesearch</code>	<i>The worker class for the conesearch module</i>
<code>sherlock.commonutils.update_wiki_pages</code>	<i>Update sherlock's github wiki pages with some useful info regarding the crossmatch database catalogue tables</i>
<code>sherlock.database</code>	<i>the database object for sherlock, setting up ssh tunnels and various database connections</i>
<code>sherlock.database_cleaner</code>	<i>Clean and maintain the database helper tables used by sherlock</i>

continues on next page

Table 9 – continued from previous page

<code>sherlock.imports.ifs</code>	<i>Importer for the Multi Unit Spectroscopic Explorer (MUSE) IFS galaxy catalogue stream</i>
<code>sherlock.imports.ned</code>	<i>Using a list of coordinates, query the online NED database and import sources found within a given search radius of each of the loctions into the sherlock-catalogues database</i>
<code>sherlock.imports.ned_d</code>	<i>Import the *`NED-D <https://ned.ipac.caltech.edu/Library/Distances/>`_ *galaxy catalogue in to the sherlock-catalogues database</i>
<code>sherlock.imports.veron</code>	<i>importer object for the VERON AGN catalogue</i>
<code>sherlock.transient_catalogue_crossmatch</code>	<i>crossmatch a list of transients against a suite of catalogues according to given search algorithm</i>
<code>sherlock.transient_classifier</code>	<i>The Sherlock Transient Classifier</i>

2.2.1 catalogue_conesearch (class)

class catalogue_conesearch (*log, ra, dec, tableName, radiusArcsec, colMaps, dbConn=False, nearestOnly=False, physicalSearch=False, upperMagnitudeLimit=False, lowerMagnitudeLimit=False, magnitudeLimitFilter=False*)

Bases: object

The worker class for the conesearch module

Key Arguments

- `dbConn` – mysql database connection to the catalogues database
- `log` – logger
- `ra` – ra of transient location (sexagesimal or decimal degrees, J2000, single location or list of locations)
- `dec` – dec of transient location (sexagesimal or decimal degrees, J2000, single location or list of locations)
- `tableName` – the name of the database table to perform the conesearch on
- `radius` – radius of the conesearch to perform (arcsec)
- `colMaps` – maps of the important column names for each table/view in the crossmatch-catalogues database
- `nearestOnly` – return only the nearest object. Default *False*
- `physicalSearch` – is this a physical search, so only return matches with distance information. Default *False*
- `upperMagnitudeLimit` – the upper magnitude limit if a magnitude cut is required with the conesearch. Default *False*
 - `lowerMagnitudeLimit` – the lower magnitude limit if a magnitude cut is required with the conesearch. Default *False*
 - `magnitudeLimitFilter` – the filter to use for the magnitude limit if required. Default *False*, (“_u”_g”_r”_i”_z”_y”_U”_B”_V”_R”_I”_Z”_J”_H”_K”_G”)

Usage

To setup your logger, settings and database connections, please use the `fundamentals` package (see [tutorial here](#)).

Todo:

- update the package tutorial if needed

The following examples assume you've connected to the various databases and generated the catalogue column maps in the following manner:

```
# SETUP ALL DATABASE CONNECTIONS
from sherlock import database
db = database(
    log=log,
    settings=settings
)
dbConns, dbVersions = db.connect()
transientsDbConn = dbConns["transients"]
cataloguesDbConn = dbConns["catalogues"]

# GET THE COLUMN MAPS FROM THE CATALOGUE DATABASE
from sherlock.commonutils import get_crossmatch_catalogues_column_map
colMaps = get_crossmatch_catalogues_column_map(
    log=log,
    dbConn=cataloguesDbConn
)
```

To perform a single location conesearch on a catalogue in the database, for example the milliquas AGN catalogue:

```
from sherlock import catalogue_conesearch
cs = catalogue_conesearch(
    log=log,
    ra="23:01:07.99",
    dec="-01:58:04.5",
    radiusArcsec=60.,
    colMaps=colMaps,
    tableName="tcs_view_agn_milliquas_v4_5",
    dbConn=cataloguesDbConn,
    nearestOnly=False,
    physicalSearch=False
)
# catalogueMatches ARE ORDERED BY ANGULAR SEPARATION
indices, catalogueMatches = cs.search()

print(catalogueMatches)
```

The output of this search is:

```
[{'R': 20.1, 'cmSepArcsec': 0.28015184686564643, 'ra': 345.2832267, 'catalogue_
↪object_subtype': u'QR', 'z': 0.777, 'dec': -1.9679629, 'catalogue_object_id': u
↪'PKS 2258-022'}]
```

Note `catalogue_conesearch` can accept coordinates in sexagesimal or decimal degrees (J200). It can also accept lists of coordinates:

```
from sherlock import catalogue_conesearch
cs = catalogue_conesearch(
    log=log,
```

(continues on next page)

(continued from previous page)

```

ra=["23:01:07.99", 45.36722, 13.875250],
dec=["-01:58:04.5", 30.45671, -25.26721],
radiusArcsec=60.,

    colMaps=colMaps,
    tableName="tcs_view_agm_milliquas_v4_5",
    dbConn=cataloguesDbConn,
    nearestOnly=False,
    physicalSearch=False
)

```

When passing a list of transient coordinates the returned ``indices`` value becomes important as this list identifies which transient is matched with which crossmatch results (and possibly multiple crossmatch results)

```

```python
indices, catalogueMatches = cs.search()
for i, c in zip(indices, catalogueMatches):
 print(i, c)
```

```

The output of this search is:

```

```text
0 {'R': 20.1, 'cmSepArcsec': 0.28015184686564643, 'ra': 345.2832267, 'catalogue_
 ↳object_subtype': u'QR', 'z': 0.777, 'dec': -1.9679629, 'catalogue_object_id': u
 ↳'PKS 2258-022'}
2 {'R': 19.2, 'cmSepArcsec': 0.81509715903447644, 'ra': 13.875, 'catalogue_object_
 ↳subtype': u'Q', 'z': 2.7, 'dec': -25.2672223, 'catalogue_object_id': u'Q 0053-
 ↳2532'}
```

```

- update key arguments values **and** definitions **with** defaults
- update **return** values **and** definitions
- update usage examples **and** text
- update docstring text
- check sublime snippet exists
- clip **any** useful text to docs mindmap
- regenerate the docs **and** check redendering of this docstring

Methods

search()

trigger the conesearch

search()

trigger the conesearch

Return

- matchIndies – the indicies of the input transient sources (syncs with uniqueMatchDicts)
- uniqueMatchDicts – the crossmatch results

Usage

See class docstring for usage examples

```
- update key arguments values and definitions with defaults
- update return values and definitions
- update usage examples and text
- update docstring text
- check sublime snippet exists
- clip any useful text to docs mindmap
- regenerate the docs and check redendering of this docstring
```

2.2.2 update_wiki_pages (class)

class update_wiki_pages (log, settings=False)

Bases: object

Update sherlock's github wiki pages with some useful info regarding the crossmatch database catalogue tables

Key Arguments

- log – logger
- settings – the settings dictionary

Usage

To trigger an update of sherlock's wiki pages to give an overview of the crossmatch table database tables run the following:

```
from sherlock.commonutils import update_wiki_pages
wiki = update_wiki_pages(
    log=log,
    settings=settings
)
wiki.update()
```

```
- create a new script for updating sherlock wiki with the snippet above, remove
↳wiki command from cl-utils and add stand alone scripts to the sherlock repo
↳(cleans up the usage and docs for sherlock)
- harvest text from wiki pages and then delete them: https://github.com/thespacedoctor/sherlock/wiki
```

Methods

| | |
|----------|-------------------|
| update() | Update wiki pages |
|----------|-------------------|

update ()
 Update wiki pages
 See class docstring for usage

2.2.3 database (class)

class database (*log, settings=False*)

Bases: object

the database object for sherlock, setting up ssh tunnels and various database connections

The returned dictionary of database connections contain the following databases:

- `transients` – the database hosting the transient source data
- `catalogues` – connection to the database hosting the contextual catalogues the transients are to be crossmatched against

Key Arguments

- `log` – logger
- `settings` – the settings dictionary

Return

- `dbConns` – a dictionary of the database connections required by sherlock

Usage

To setup the sherlock database connections, run the following:

```
# SETUP ALL DATABASE CONNECTIONS
from sherlock import database
db = database(
    log=log,
    settings=settings
)
dbConns, dbVersions = db.connect()
transientsDbConn = dbConns["transients"]
cataloguesDbConn = dbConns["catalogues"]
```

- update key arguments values **and** definitions **with** defaults
- update **return** values **and** definitions
- update usage examples **and** text
- update docstring text
- check sublime snippet exists
- clip **any** useful text to docs mindmap
- regenerate the docs **and** check redendering of this docstring

Methods

| | |
|------------------------|---|
| <code>connect()</code> | connect to the various databases, the credentials and settings of which are found in the sherlock settings file |
|------------------------|---|

`connect ()`

connect to the various databases, the credentials and settings of which are found in the sherlock settings file

Return

- `transientsDbConn` – the database hosting the transient source data

- cataloguesDbConn – connection to the database hosting the contextual catalogues the transients are to be crossmatched against

See the class docstring for usage

```
- update key arguments values and definitions with defaults
- update return values and definitions
- update usage examples and text
- update docstring text
- check sublime snippet exists
- clip any useful text to docs mindmap
- regenerate the docs and check redendering of this docstring
```

2.2.4 database_cleaner (class)

class database_cleaner (*log, settings=False*)

Bases: object

Clean and maintain the database helper tables used by sherlock

The helper tables list row counts for tables and views and provide the column maps that help sherlock know which catalogue columns relate to which parameters (e.g. RA, DEC etc)

Key Arguments

- dbConn – mysql database connection
- log – logger
- settings – the settings dictionary

Usage

Todo:

- add an entry in the tutorial to clean database tables

```
```python from sherlock.database_cleaner import database_cleaner db = database_cleaner(
 log=log, settings=settings
) db.clean() ```
```

---

```
- update key arguments values and definitions with defaults
- update return values and definitions
- update usage examples and text
- update docstring text
- check sublime snippet exists
- clip any useful text to docs mindmap
- regenerate the docs and check redendering of this docstring
```

## Methods

clean()	<i>clean up and run some maintance tasks on the cross-match catalogue helper tables</i>
---------	-----------------------------------------------------------------------------------------

**clean()**  
*clean up and run some maintance tasks on the crossmatch catalogue helper tables*

```
- update key arguments values and definitions with defaults
- update return values and definitions
- update usage examples and text
- update docstring text
- check sublime snippet exists
- clip any useful text to docs mindmap
- regenerate the docs and check redrendering of this docstring
```

### 2.2.5 ifs (class)

**class ifs**(log, settings=False, pathToDataFile=False, version=False, catalogueName="", coordinateList=[], radiusArcsec=False)  
 Bases: sherlock.imports.\_base\_importer.\_base\_importer

*Importer for the Multi Unit Spectroscopic Explorer (MUSE) IFS galaxy catalogue stream*

#### Key Arguments

- log – logger
- settings – the settings dictionary

#### Usage

To import the IFS catalogue stream into the sherlock-catalogues database, run the following:

```
from sherlock.imports import IFS
```

```
stream = IFS(
 log=log,
 settings=settings
)
stream.ingest()
```

```
- abstract this module out into its own stand alone script
- check sublime snippet exists
```

## Methods

add_data_to_database_table(dictList[, ...])	<i>Import data in the list of dictionaries in the requested database table</i>
ingest()	<i>Import the IFS catalogue into the sherlock-catalogues database</i>

**add\_data\_to\_database\_table** (dictList, createStatement=False)



*Import data in the list of dictionaries in the requested database table*

Also adds HTMIDs and updates the sherlock-catalogue database helper table with the time-stamp of when the imported catalogue was last updated

### Key Arguments

- `dictList` - a list of dictionaries containing all the rows in the catalogue to be imported
- `createStatement` - the table's mysql create statement (used to generate table if it does not yet exist in database). Default *False*

### Usage

```
self.add_data_to_database_table(
 dictList=dictList,
 createStatement=createStatement
)
```

```
- Write a checklist for creating a new sherlock database importer
```

### `ingest()`

*Import the IFS catalogue into the sherlock-catalogues database*

The method first generates a list of python dictionaries from the IFS datafile, imports this list of dictionaries into a database table and then generates the HTMIDs for that table.

### Usage

See class docstring for usage

## 2.2.6 `ned` (class)

```
class ned(log, settings=False, pathToDataFile=False, version=False, catalogueName="", coordinateList=[], radiusArcsec=False)
```

Bases: `sherlock.imports._base_importer._base_importer`

*Using a list of coordinates, query the online NED database and import sources found within a given search radius of each of the locations into the sherlock-catalogues database*

The code:

```
1. Uses the list of transient coordinates and queries NED (conesearch) for the
 ↪ results within the given search radius
2. Creates the `tcs_cat_ned_stream` table if it doesn't exist
3. Adds the resulting matched NED IDs/Names to the `tcs_cat_ned_stream` table
4. Updates the NED query history table
5. Queries NED via NED IDs (object search) for the remaining source metadata to be
 ↪ added to the `tcs_cat_ned_stream` table
```

Note it's up to the user to filter the input coordinate list by checking whether or not the same area of the sky has been imported into the `tcs_cat_ned_stream` table recently (by checking the `tcs_helper_ned_query_history` table)

### Key Arguments

- `dbConn` - mysql database connection
- `log` - logger
- `settings` - the settings dictionary

- `coordinateList` – list of coordinates (a list of strings with RA and DEC space separated)
- `radiusArcsec` - - the radius in arcsec with which to perform the initial NED conesearch. Default `*False*`

### Usage

To import the ned catalogue stream, run the following:

```
from sherlock.imports import ned
```

```
stream = ned(
 log=log,
 settings=settings,
 coordinateList=["23.12323 -12.34343", "345.43234 45.26789"],
 radiusArcsec=180
)
stream.ingest()
```

- test this code is still working after changes
- add option to filter coordinate list via the ``tcs_helper_ned_query_history`` table
- check sublime snippet exists
- clip any useful text to docs mindmap

### Methods

---

<code>add_data_to_database_table(dictList[, ...])</code>	<i>Import data in the list of dictionaries in the requested database table</i>
<code>ingest()</code>	<i>Perform conesearches of the online NED database and import the results into a the sherlock-database</i>

---

**`add_data_to_database_table`** (*dictList*, *createStatement=False*)  
*Import data in the list of dictionaries in the requested database table*

Also adds HTMIDs and updates the sherlock-catalogue database helper table with the time-stamp of when the imported catalogue was last updated

#### Key Arguments

- `dictList` - a list of dictionaries containing all the rows in the catalogue to be imported
- `createStatement` - the table's mysql create statement (used to generate table if it does not yet exist in database). Default *False*

#### Usage

```
self.add_data_to_database_table(
 dictList=dictList,
 createStatement=createStatement
)
```

- Write a checklist **for** creating a new sherlock database importer

**`ingest`** ()  
*Perform conesearches of the online NED database and import the results into a the sherlock-database*

The code:

```
1. uses the list of transient coordinates and queries NED for the results,
↳ within the given search radius
2. Creates the `tcs_cat_ned_stream` table if it doesn't exist
3. Adds the resulting NED IDs/Names to the `tcs_cat_ned_stream` table
4. Updates the NED query history table
5. Queris NED via NED IDs for the remaining source metadata to be added to,
↳ the `tcs_cat_ned_stream` table
```

### Usage

Having setup the NED object with a coordinate list and cone-search radius, run the `ingest()` method

```
stream.ingest()
```

```
- check sublime snippet exists
- clip any useful text to docs mindmap
- regenerate the docs and check redrendering of this docstring
```

## 2.2.7 ned\_d (class)

```
class ned_d(log, settings=False, pathToDataFile=False, version=False, catalogueName="", coordi-
nateList=[], radiusArcsec=False)
```

Bases: `sherlock.imports._base_importer._base_importer`

Import the \* *NED-D* <<https://ned.ipac.caltech.edu/Library/Distances/>>`\_\*galaxy catalogue in to the *sherlock-catalogues database*

### Key Arguments

- `log` – logger
- `settings` – the settings dictionary
- `pathToDataFile` – path to the `ned_d` data file
- `version` – version of the `ned_d` catalogue
- `catalogueName` – the name of the catalogue

### Usage

To import the `ned_d` catalogue catalogue, run the following:

```
from sherlock.imports import ned_d
catalogue = ned_d(
 log=log,
 settings=settings,
 pathToDataFile="/path/to/ned_d.txt",
 version="1.0",
 catalogueName="ned_d"
)
catalogue.ingest()
```

```
- abstract this module out into its own stand alone script
```

## Methods

<code>add_data_to_database_table(dictList[, ...])</code>	<i>Import data in the list of dictionaries in the requested database table</i>
<code>ingest()</code>	Import the ned_d catalogue into the catalogues database

**add\_data\_to\_database\_table** (*dictList*, *createStatement=False*)

*Import data in the list of dictionaries in the requested database table*

Also adds HTMIDs and updates the sherlock-catalogue database helper table with the time-stamp of when the imported catalogue was last updated

### Key Arguments

- `dictList` - a list of dictionaries containing all the rows in the catalogue to be imported
- `createStatement` - the table's mysql create statement (used to generate table if it does not yet exist in database). Default *False*

### Usage

```
self.add_data_to_database_table(
 dictList=dictList,
 createStatement=createStatement
)
```

```
- Write a checklist for creating a new sherlock database importer
```

**ingest** ()

Import the ned\_d catalogue into the catalogues database

The method first generates a list of python dictionaries from the ned\_d datafile, imports this list of dictionaries into a database table and then generates the HTMIDs for that table.

### Usage

See class docstring for usage

```
- update docstring text
- check sublime snippet exists
- clip any useful text to docs mindmap
- regenerate the docs and check redendering of this docstring
```

## 2.2.8 veron (class)

**class veron** (*log*, *settings=False*, *pathToDataFile=False*, *version=False*, *catalogueName=""*, *coordinateList=[]*, *radiusArcsec=False*)

Bases: `sherlock.imports._base_importer._base_importer`

*importer object for the VERON AGN catalogue*

### Key Arguments

- `dbConn` – mysql database connection
- `log` – logger
- `settings` – the settings dictionary

- pathToDataFile – path to the veron data file
- version – version of the veron catalogue

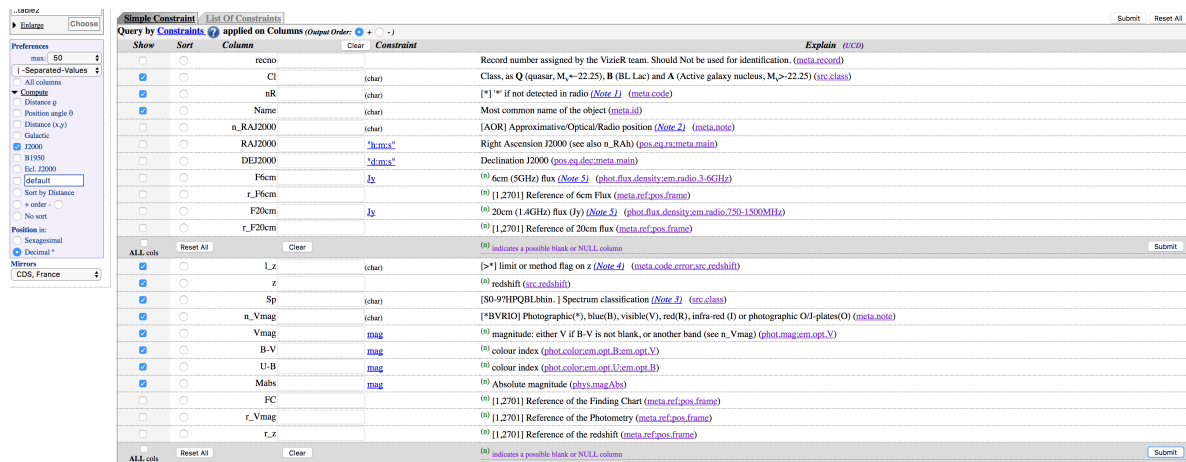
## Usage

To import the veron catalogue catalogue, run the following:

```
from sherlock.imports import veron
```

```
catalogue = veron(
 log=log,
 settings=settings,
 pathToDataFile="/path/to/veron.txt",
 version="1.0",
 catalogueName="veron"
)
catalogue.ingest()
```

Whenever downloading a version of the Veron catalogue from Vizier use the following column selection:



The screenshot shows the Vizier column selection interface. The 'Simple Constraint' tab is active, displaying a list of columns and their constraints. The columns are sorted by constraint, and the 'ALL cols' button is visible at the bottom left. The columns listed include:

Show	Sort	Column	Clear	Constraint	Explain (EXPL)
<input type="checkbox"/>	<input type="checkbox"/>	recno	<input type="checkbox"/>		Record number assigned by the Vizier team. Should Not be used for identification. (meta:recno)
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Cl	<input type="checkbox"/>	(char)	Class, as Q (quasar, M, < 22.25), B (BL Lac) and A (Active galaxy nucleus, M, > 22.25) (src:class)
<input checked="" type="checkbox"/>	<input type="checkbox"/>	nR	<input type="checkbox"/>	(char)	[*] "" if not detected in radio (Note 1) (meta:code)
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Name	<input type="checkbox"/>	(char)	Most common name of the object (meta:sd)
<input type="checkbox"/>	<input type="checkbox"/>	n_RAJ2000	<input type="checkbox"/>	(char)	[AOR] Approximative/Optical/Radio position (Note 2) (meta:note)
<input type="checkbox"/>	<input type="checkbox"/>	RAJ2000	<input type="checkbox"/>	Jy	Right Ascension J2000 (see also n_RA) (pos.eq.ra:meta:main)
<input type="checkbox"/>	<input type="checkbox"/>	DEJ2000	<input type="checkbox"/>	'd,m,s'	Declination J2000 (pos.eq.dec:meta:main)
<input type="checkbox"/>	<input type="checkbox"/>	F6cm	<input type="checkbox"/>	Jy	6cm (5GHz) flux (Note 3) (phot.flux.density:em.radio.3-6GHz)
<input type="checkbox"/>	<input type="checkbox"/>	r_F6cm	<input type="checkbox"/>		(1,2701) Reference of 6cm Flux (meta.ref:pos:frame)
<input type="checkbox"/>	<input type="checkbox"/>	F20cm	<input type="checkbox"/>	Jy	20cm (1.4GHz) flux (Jy) (Note 3) (phot.flux.density:em.radio.750-1500MHz)
<input type="checkbox"/>	<input type="checkbox"/>	r_F20cm	<input type="checkbox"/>		(1,2701) Reference of 20cm flux (meta.ref:pos:frame)
<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>		(2) indicates a possible blank or NULL column
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Lz	<input type="checkbox"/>	(char)	[>=] limit or method flag on z (Note 4) (meta:code.error.src:redshift)
<input type="checkbox"/>	<input type="checkbox"/>	z	<input type="checkbox"/>		redshift (src:redshift)
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Sp	<input type="checkbox"/>	(char)	[S0-9]HPQBLbhhin ] Spectrum classification (Note 5) (src:class)
<input checked="" type="checkbox"/>	<input type="checkbox"/>	n_Vmag	<input type="checkbox"/>	(mag)	[*BVRI] Photographic(*), blue(B), visible(V), red(R), infra-red (I) or photographic OJ-plates(O) (meta:note)
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Vmag	<input type="checkbox"/>	mag	(2) magnitude: either V if B-V is not blank, or another band (see n_Vmag) (phot.mag:em.opt.V)
<input checked="" type="checkbox"/>	<input type="checkbox"/>	B-V	<input type="checkbox"/>	mag	(2) colour index (phot.colour:em.opt.B:em.opt.V)
<input checked="" type="checkbox"/>	<input type="checkbox"/>	U-B	<input type="checkbox"/>	mag	(2) colour index (phot.colour:em.opt.U:em.opt.B)
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Mabs	<input type="checkbox"/>	mag	(2) Absolute magnitude (nsys.mag:Abs)
<input type="checkbox"/>	<input type="checkbox"/>	FC	<input type="checkbox"/>		(1,2701) Reference of the Finding Chart (meta.ref:pos:frame)
<input type="checkbox"/>	<input type="checkbox"/>	r_Vmag	<input type="checkbox"/>		(1,2701) Reference of the Photometry (meta.ref:pos:frame)
<input type="checkbox"/>	<input type="checkbox"/>	r_z	<input type="checkbox"/>		(1,2701) Reference of the redshift (meta.ref:pos:frame)
<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>		(2) indicates a possible blank or NULL column

- abstract this module out into its own stand alone script
- add ppmxl to stand-alone `import scripts`
- update key arguments values and definitions with defaults
- update `return` values and definitions
- update usage examples and text
- update docstring text
- check sublime snippet exists
- clip any useful text to docs mindmap
- regenerate the docs and check rendering of this docstring

## Methods

<code>add_data_to_database_table(dictList[, ...])</code>	<i>Import data in the list of dictionaries in the requested database table</i>
<code>ingest()</code>	ingest the veron catalogue into the catalogues database

**add\_data\_to\_database\_table** (*dictList, createStatement=False*)

*Import data in the list of dictionaries in the requested database table*

Also adds HTMIDs and updates the sherlock-catalogue database helper table with the time-stamp of when the imported catalogue was last updated

### Key Arguments

- `dictList` - a list of dictionaries containing all the rows in the catalogue to be imported
- `createStatement` - the table's mysql create statement (used to generate table if it does not yet exist in database). Default *False*

### Usage

```
self.add_data_to_database_table(
 dictList=dictList,
 createStatement=createStatement
)
```

```
- Write a checklist for creating a new sherlock database importer
```

**ingest** ()

ingest the veron catalogue into the catalogues database

See class docstring for usage.

## 2.2.9 transient\_catalogue\_crossmatch (class)

**class transient\_catalogue\_crossmatch** (*log, settings=False, colMaps=False, transients=[], dbSettings=False, dbConn=False*)

Bases: `object`

*crossmatch a list of transients against a suite of catalogues according to given search algorithm*

### Key Arguments

- `dbConn` - mysql database connection for the catalogues
- `log` - logger
- `settings` - the settings dictionary
- `colMaps` - maps of the important column names for each table/view in the crossmatch-catalogues database
- `transients` - the list of transients

### Usage

To setup your logger, settings and database connections, please use the `fundamentals` package (see [tutorial here](#)).

To initiate a `transient_catalogue_crossmatch` object, use the following:

```

from sherlock import transient_catalogue_crossmatch
xmatcher = transient_catalogue_crossmatch(
 log=log,
 settings=settings,
 colMaps=colMaps,
 transients=transients,
 dbSettings=settings["database settings"]["static catalogues"]
)

```

Then to run the transient through the search algorithm found in the settings file, use the `match` method:

```
classifications = xmatcher.match()
```

```

- update key arguments values and definitions with defaults
- update return values and definitions
- update usage examples and text
- update docstring text
- check sublime snippet exists
- clip any useful text to docs mindmap
- regenerate the docs and check redendering of this docstring

```

## Methods

<code>angular_crossmatch_against_catalogue(objectList)</code>	<i>perform an angular separation crossmatch against a given catalogue in the database and annotate the crossmatch with some value added parameters (distances, physical separations, sub-type of transient etc)</i>
<code>match()</code>	<i>match the transients against the sherlock-catalogues according to the search algorithm and return matches alongside the predicted classification(s)</i>
<code>physical_separation_crossmatch_against_catalogue(objectList)</code>	<i>perform a physical separation crossmatch against a given catalogue in the database</i>

**angular\_crossmatch\_against\_catalogue** (*objectList*, *searchPara*={}, *search\_name*="", *brightnessFilter*=False, *physicalSearch*=False, *classificationType*=False)

*perform an angular separation crossmatch against a given catalogue in the database and annotate the crossmatch with some value added parameters (distances, physical separations, sub-type of transient etc)*

### Key Arguments

- `objectList` – the list of transient locations to match against the crossmatch catalogue
- `searchPara` – the search parameters for this individual search as lifted from the search algorithm in the sherlock settings file
- `search_name` – the name of the search as given in the sherlock settings file
- `brightnessFilter` – is this search to be constrained by magnitude of the catalogue sources? Default *False*. [bright|faint|general]
- `physicalSearch` – is this angular search a sub-part of a physical separation search
- `classificationType` – synonym, association or annotation. Default *False*

## Return

```
- matchedObjects -- any sources matched against the object
```

## Usage

Take a list of transients from somewhere

```
transients = [
 {'ps1_designation': u'PS1-14aef',
 'name': u'4L3Piiq',
 'detection_list_id': 2,
 'local_comments': u'',
 'ra': 0.02548233704918263,
 'followup_id': 2065412L,
 'dec': -4.284933417540423,
 'id': 1000006110041705700L,
 'object_classification': 0L
 },
 {'ps1_designation': u'PS1-13dcr',
 'name': u'3I3Phzx',
 'detection_list_id': 2,
 'local_comments': u'',
 'ra': 4.754236999477372,
 'followup_id': 1140386L,
 'dec': 28.276703631398625,
 'id': 1001901011281636100L,
 'object_classification': 0L
 },
 {'ps1_designation': u'PS1-13dhc',
 'name': u'3I3Pixd',
 'detection_list_id': 2,
 'local_comments': u'',
 'ra': 1.3324973428505413,
 'followup_id': 1202386L,
 'dec': 32.98869220595689,
 'id': 1000519791325919200L,
 'object_classification': 0L
 }
]
```

Then run the `angular_crossmatch_against_catalogue` method to crossmatch against the catalogues and return results:

```
ANGULAR CONESEARCH ON CATALOGUE
search_name = "ned_d spec sn"
searchPara = self.settings["search algorithm"][search_name]
matchedObjects = xmatcher.angular_crossmatch_against_catalogue(
 objectList=transients,
 searchPara=searchPara,
 search_name=search_name
)
```

```
- update key arguments values and definitions with defaults
- update return values and definitions
```

(continues on next page)



(continued from previous page)

```

- update usage examples and text
- update docstring text
- check sublime snippet exists
- clip any useful text to docs mindmap
- regenerate the docs and check redrendering of this docstring

```

**match()**

*match the transients against the sherlock-catalogues according to the search algorithm and return matches alongside the predicted classification(s)*

**Return**

- `classification` – the crossmatch results and classifications assigned to the transients

See the class docstring for usage.

```

- update key arguments values and definitions with defaults
- update return values and definitions
- update usage examples and text
- update docstring text
- check sublime snippet exists
- clip any useful text to docs mindmap
- regenerate the docs and check redrendering of this docstring

```

**physical\_separation\_crossmatch\_against\_catalogue** (*objectList*, *searchPara*,  
*search\_name*, *brightness-*  
*Filter=False*, *classification-*  
*Type=False*)

*perform an physical separation crossmatch against a given catalogue in the database*

This search is basically the same as the angular separation search except extra filtering is done to exclude sources outside the physical search radius (matched sources require distance info to calculate physical separations)

**Key Arguments**

- `objectList` – transients to be crossmatched
- `searchPara` – parameters of the search (from settings file)
- `search_name` – the name of the search
- `brightnessFilter` – is this search to be constrained by magnitude of the catalogue sources? Default *False*. [*bright|faint|general*]
- `classificationType` – synonym, association or annotation. Default *False*

**Return**

- `matchedObjects` – any sources matched against the object

To run a physical separation crossmatch, run in a similar way to the angular separation crossmatch:

**Usage**

```

search_name = "ned spec sn"
searchPara = self.settings["search algorithm"][search_name]
matchedObjects = xmatcher.physical_separation_crossmatch_against_catalogue(
 objectList=transients,
 searchPara=searchPara,

```

(continues on next page)

(continued from previous page)

```

 search_name=search_name
)

```

```

- update key arguments values and definitions with defaults
- update return values and definitions
- update usage examples and text
- update docstring text
- check sublime snippet exists
- clip any useful text to docs mindmap
- regenerate the docs and check redrendering of this docstring

```

### 2.2.10 transient\_classifier (class)

**class transient\_classifier** (*log, settings=False, update=False, ra=False, dec=False, name=False, verbose=0, updateNed=True, daemonMode=False, updatePeakMags=True, oneRun=False, lite=False*)

Bases: object

*The Sherlock Transient Classifier*

#### Key Arguments

- `log` – logger
- `settings` – the settings dictionary
- `update` – update the transient database with crossmatch results (boolean)
- `ra` – right ascension of a single transient source. Default *False*
- `dec` – declination of a single transient source. Default *False*
- `name` – the ID of a single transient source. Default *False*
- `verbose` – amount of details to print about crossmatches to stdout. 0|1|2 Default *0*
- `updateNed` – update the local NED database before running the classifier. Classification will not be as accurate the NED database is not up-to-date. Default *True*.
- `daemonMode` – run sherlock in daemon mode. In daemon mode sherlock remains live and classifies sources as they come into the database. Default *True*
- `updatePeakMags` – update peak magnitudes in human-readable annotation of objects (can take some time - best to run occasionally)
- `lite` – return only a lite version of the results with the topped ranked matches only. Default *False*
- `oneRun` – only process one batch of transients, useful for unit testing. Default *False*

#### Usage

To setup your logger, settings and database connections, please use the `fundamentals` package (see [tutorial here](#)).

To initiate a `transient_classifier` object, use the following:

---

#### Todo:

- update the package tutorial if needed

The sherlock classifier can be run in one of two ways. The first is to pass into the coordinates of an object you wish to classify:

```
from sherlock import transient_classifier
classifier = transient_classifier(
 log=log,
 settings=settings,
 ra="08:57:57.19",
 dec="+43:25:44.1",
 name="PS17gx",
 verbose=0
)
classifications, crossmatches = classifier.classify()
```

The crossmatches returned are a list of dictionaries giving details of the crossmatched sources. The classifications returned are a list of classifications resulting from these crossmatches. The lists are ordered from most to least likely classification and the indices for the crossmatch and the classification lists are synced.

The second way to run the classifier is to not pass in a coordinate set and therefore cause sherlock to run the classifier on the transient database referenced in the sherlock settings file:

```
from sherlock import transient_classifier
classifier = transient_classifier(
 log=log,
 settings=settings,
 update=True
)
classifier.classify()
```

Here the transient list is selected out of the database using the `transient` query value in the settings file:

```
database settings:
 transients:
 user: myusername
 password: mypassword
 db: nice_transients
 host: 127.0.0.1
 transient table: transientBucket
 transient query: "select primaryKeyId as 'id', transientBucketId as 'alt_
↪id', raDeg 'ra', decDeg 'dec', name 'name', sherlockClassification as 'object_
↪classification'
 from transientBucket where object_classification is null
 transient primary id column: primaryKeyId
 transient classification column: sherlockClassification
 tunnel: False
```

By setting `update=True` the classifier will update the `sherlockClassification` column of the transient table with new classification and populate the `sherlock_crossmatches` table with key details of the crossmatched sources from the catalogues database. By setting `update=False` results are printed to stdout but the database is not updated (useful for dry runs and testing new algorithms),

```
- update key arguments values and definitions with defaults
- update return values and definitions
- update usage examples and text
- update docstring text
- check sublime snippet exists
```

(continues on next page)

(continued from previous page)

- ```
- clip any useful text to docs mindmap
- regenerate the docs and check redendering of this docstring
```

Methods

| | |
|--|--|
| <code>classification_annotatons()</code> | <i>add a detaied classification annotation to each clas-sification in the sherlock_classifications table</i> |
| <code>classify()</code> | <i>classify the transients selected from the transient se-lection query in the settings file or passed in via the CL or other code</i> |
| <code>generate_match_annotation(match[,...])</code> | <i>generate a human readale annotation for the transient-catalogue source match</i> |
| <code>update_classification_annotatons_and_update_classification_annotatons()</code> | <i>update classification annotations and summaries</i> |
| <code>update_peak_magnitudes()</code> | <i>update peak magnitudes</i> |

`classification_annotatons ()`

add a detaied classification annotation to each classification in the sherlock_classifications table

Key Arguments

-

=

Return

- None

Usage

Todo:

- add usage info
- create a sublime snippet for usage
- write a command-line tool for this method
- update package tutorial with command-line tool info if needed

```
usage code
```

- ```
- update key arguments values and definitions with defaults
- update return values and definitions
- update usage examples and text
- update docstring text
- check sublime snippet exists
- clip any useful text to docs mindmap
- regenerate the docs and check redendering of this docstring
```

### `classify ()`

*classify the transients selected from the transient selection query in the settings file or passed in via the CL or other code*

#### Return

- `crossmatches` – list of dictionaries of crossmatched associated sources
- `classifications` – the classifications assigned to the transients post-crossmatches (dictionary of rank ordered list of classifications)

See class docstring for usage.

```

- update key arguments values and definitions with defaults
- update return values and definitions
- update usage examples and text
- update docstring text
- check sublime snippet exists
- clip any useful text to docs mindmap
- regenerate the docs and check redendering of this docstring

```

**generate\_match\_annotation** (*match*, *updatePeakMagnitudes=False*)  
*generate a human readable annotation for the transient-catalogue source match*

#### Key Arguments

- `match` – the source crossmatched against the transient
- `updatePeakMagnitudes` – update the peak magnitudes in the annotations to give absolute magnitudes. Default *False*

#### Return

- None

#### Usage

```
usage code
```

---

#### Todo:

- add usage info
  - create a sublime snippet for usage
  - write a command-line tool for this method
  - update package tutorial with command-line tool info if needed
- 

**update\_classification\_annotations\_and\_summaries** (*updatePeakMagnitudes=True*,  
*cl=False*, *crossmatches=False*,  
*classifications=False*)

*update classification annotations and summaries*

#### Key Arguments

- `updatePeakMagnitudes` – update the peak magnitudes in the annotations to give absolute magnitudes. Default *True*
- `cl` – reporting only to the command-line, do not update database. Default *False*
- `crossmatches` – crossmatches will be passed for the single classifications to report annotations from command-line
- `classifications` – classifications will be passed for the single classifications to have annotation appended to the dictionary for stand-alone non-database scripts

#### Return

- None

### Usage

---

#### Todo:

- add usage info
  - create a sublime snippet for usage
  - write a command-line tool for this method
  - update package tutorial with command-line tool info if needed
- 

```
usage code
```

```
- update key arguments values and definitions with defaults
- update return values and definitions
- update usage examples and text
- update docstring text
- check sublime snippet exists
- clip any useful text to docs mindmap
- regenerate the docs and check redrendering of this docstring
```

### `update_peak_magnitudes()`

*update peak magnitudes*

#### Key Arguments

- 

=

#### Return

- None

### Usage

---

#### Todo:

- add usage info
  - create a sublime snippet for usage
  - write a command-line tool for this method
  - update package tutorial with command-line tool info if needed
- 

```
usage code
```

```
- update key arguments values and definitions with defaults
- update return values and definitions
- update usage examples and text
- update docstring text
- check sublime snippet exists
- clip any useful text to docs mindmap
- regenerate the docs and check redrendering of this docstring
```

## 2.3 Functions

---

<code>sherlock.commonutils. get_crossmatch_catalogues_column_map</code>	<i>Query the sherlock-catalogues helper tables to generate a map of the important columns of each catalogue</i>
<code>sherlock.commonutils.getpackagepath</code>	<i>Get the root path for this python package - used in unit testing code</i>

---

### 2.3.1 `get_crossmatch_catalogues_column_map` (function)

`get_crossmatch_catalogues_column_map` (*dbConn*, *log*)

*Query the sherlock-catalogues helper tables to generate a map of the important columns of each catalogue*

Within your sherlock-catalogues database you need to manually map the inhomogeneous column-names from the sherlock-catalogues to an internal homogeneous name-set which includes *ra*, *dec*, *redshift*, *object name*, *magnitude*, *filter* etc. The column-name map is set within the two database helper tables called `tcs_helper_catalogue_views_info` and `tcs_helper_catalogue_views_info`. See the 'Checklist for Adding A New Reference Catalogue to the Sherlock Catalogues Database' for more information.

---

**Todo:**

- write a checklist for adding a new catalogue to the sherlock database and reference it from here (use the image below of the `tcs_helper_catalogue_views_info` table)
-

tcs_helper_catalogue_tables_info	
id	SMALLINT(5)
table_name	VARCHAR(100)
description	VARCHAR(60)
url	VARCHAR(255)
number_of_rows	BIGINT(20)
reference_url	VARCHAR(200)
reference_text	VARCHAR(70)
notes	TEXT
vizier_link	VARCHAR(200)
in_ned	TINYINT(4)
object_types	VARCHAR(100)
version_number	VARCHAR(45)
last_updated	DATETIME
legacy_table	TINYINT(4)
old_table_name	VARCHAR(100)
raColName	VARCHAR(45)
decColName	VARCHAR(45)
catalogue_object_subtypeColName	VARCHAR(45)
catalogue_object_idColName	VARCHAR(100)
zColName	VARCHAR(100)
distanceColName	VARCHAR(45)
object_type_accuracy	TINYINT(2)
semiMajorColName	VARCHAR(100)
semiMajorToArcsec	FLOAT
transientStream	TINYINT(4)
photoZColName	VARCHAR(45)
photoZErrColName	VARCHAR(45)
UColName	VARCHAR(45)
UErrColName	VARCHAR(45)
BColName	VARCHAR(45)
BErrColName	VARCHAR(45)
VColName	VARCHAR(45)
VErrColName	VARCHAR(45)
RColName	VARCHAR(45)
RErrColName	VARCHAR(45)
IColName	VARCHAR(45)
IErrColName	VARCHAR(45)
JColName	VARCHAR(45)
JErrColName	VARCHAR(45)
HColName	VARCHAR(45)
HErrColName	VARCHAR(45)
KColName	VARCHAR(45)
KErrColName	VARCHAR(45)
_uColName	VARCHAR(45)
_uErrColName	VARCHAR(45)
_gColName	VARCHAR(45)
_gErrColName	VARCHAR(45)
_rColName	VARCHAR(45)
_rErrColName	VARCHAR(45)
_iColName	VARCHAR(45)
_iErrColName	VARCHAR(45)
_zColName	VARCHAR(45)
_zErrColName	VARCHAR(45)
_yColName	VARCHAR(45)
_yErrColName	VARCHAR(45)
unkMagColName	VARCHAR(45)
unkMagErrColName	VARCHAR(45)
GColName	VARCHAR(45)
GErrColName	VARCHAR(45)
Indexes	
PRIMARY	



**Key Arguments**

- dbConn – the sherlock-catalogues database connection
- log – logger

**Return**

- colMaps – dictionary of dictionaries with the name of the database-view (e.g. tcs\_view\_agn\_milliquas\_v4\_5) as the key and the column-name dictary map as value ({{view\_name: {columnMap}}}).

**Usage**

To collect the column map dictionary of dictionaries from the catalogues database, use the `get_crossmatch_catalogues_column_map` function:

```
from sherlock.commonutils import get_crossmatch_catalogues_column_map
colMaps = get_crossmatch_catalogues_column_map(
 log=log,
 dbConn=cataloguesDbConn
)
```

**2.3.2 getpackagepath (function)****getpackagepath()**

*Get the root path for this python package - used in unit testing code*

**2.4 A-Z Index****Modules**

<code>sherlock.commonutils</code>	<i>common tools used throughout package</i>
<code>sherlock.imports</code>	
<code>sherlock.utKit</code>	<i>Unit testing tools</i>

**Classes**

<code>sherlock.catalogue_conesearch</code>	<i>The worker class for the conesearch module</i>
<code>sherlock.commonutils.update_wiki_pages</code>	<i>Update sherlock's github wiki pages with some useful info regarding the crossmatch database catalogue tables</i>
<code>sherlock.database</code>	<i>the database object for sherlock, setting up ssh tunnels and various database connections</i>
<code>sherlock.database_cleaner</code>	<i>Clean and maintain the database helper tables used by sherlock</i>
<code>sherlock.imports.ifs</code>	<i>Importer for the Multi Unit Spectroscopic Explorer (MUSE) IFS galaxy catalogue stream</i>

continues on next page

Table 22 – continued from previous page

<code>sherlock.imports.ned</code>	<i>Using a list of coordinates, query the online <a href="#">NED</a> database and import sources found within a given search radius of each of the loctions into the sherlock-catalogues database</i>
<code>sherlock.imports.ned_d</code>	<i>Import the * `NED-D` &lt;<a href="https://ned.ipac.caltech.edu/Library/Distances/">https://ned.ipac.caltech.edu/Library/Distances/</a>&gt; *_galaxy catalogue in to the sherlock-catalogues database</i>
<code>sherlock.imports.veron</code>	<i>importer object for the <a href="#">VERON AGN</a> catalogue</i>
<code>sherlock.transient_catalogue_crossmatch</code>	<i>crossmatch a list of transients against a suite of catalogues according to given search algorithm</i>
<code>sherlock.transient_classifier</code>	<i>The Sherlock Transient Classifier</i>

### Functions

<code>sherlock.commonutils.get_crossmatch_catalogues_column_map</code>	<i>Query the sherlock-catalogues helper tables to generate a map of the important columns of each catalogue</i>
<code>sherlock.commonutils.getpackagepath</code>	<i>Get the root path for this python package - used in unit testing code</i>

## RELEASE NOTES

### v2.2.0 - August 2, 2021

- **FEATURE:** Default search algorithm now version-controlled and ship alongside code. Users can override the default algorithm if they need to.
- **FEATURE:** Hidden `database-batch-size` and `cpu-pool-size` settings added - power users can access and change in settings files.
- **ENHANCEMENT:** Speed improvements (5-10 times processing speed increase).
- **REFACTOR/FIX:** synonym match break now removed by default so a location now gets matched against all catalogues regardless of whether or not a synonym match is initially found (e.g. right on top of a catalogued star). Resolves edge cases where true SNe are mis-classified as VS as they are located just too close to a stellar source, or galaxy source mis-identified as stellar in one catalogue but correctly as galaxy in another.
- **FIX:** some URLs to NED objects, found in the human readable Sherlock annotations, that were not resolving now do.

### v2.1.8 - April 16, 2021#bugFixFriday

- **REFACTOR:** catalogue quality weights now used *within* the merged source sets to sort the data. This fixed *some* mis-matched annotation issues.
- **REFACTOR:** updating db connection settings to use QUB jumpbox.
- **REFACTOR/FIX** Reduced associated source merging radius from 3 arcsec to 1 arcsec (1/9th of the original matching area). This seems to correct many of the cases in which 2 or more distinct catalogued sources were getting merged/blended into one association. This resulted in in-correct classification prediction and wrongly associated distances.
- **FIXED:** annotation mismatches. If transient is classified as a SN then the underlying sources is identified as a galaxy in the annotation.

### v2.1.7 - September 29, 2020

- **FIXED:** lite version introduced a couple of small bugs. Fixed.

### v2.1.6 - September 7, 2020

- **ENHANCEMENT:** transient human-readable annotations added to the `classifications` dictionary return of the `transient_classifier.classify()` method.
- **ENHANCEMENT:** a `lite` parameter has been added to the Sherlock `transient_classifier` object. When set to `True` only top-ranked crossmatches are returned with a limited column set.
- **ENHANCEMENT:** snapshot annotation is now added into the classification dictionary return. The classification dictionary is now `{'MyTransientId01': ['Type', 'Annotation'], 'MyTransientId02': ['Type', 'Annotation'],}`

- **REFACTOR**: small change to query that writes NED source magnitudes that vastly improves speed
- **FIXED**: small issue where a missing error in photoz was causing annotations to not complete

**v2.1.5 - June 22, 2020**

- **refactor**: tunnel changes due to new jump box installation at QUB
- **fixed**: some NED galaxies not reported if redshift does not exist but a semi-major axis is given

**v2.1.4 - June 18, 2020**

- **fixed**: an empty crossmatch set could cause a transient database update when not required

**v2.1.3 - June 5, 2020**

- **refactor**: stop sherlock checking for transient database triggers if running in non-update mode
- **fixed**: another sdss photoz issue where photoz ranking about specz

**v2.1.2 - May 24, 2020**

- **fixed**: merged result parameters are now merged correctly

**v2.1.1 - May 21, 2020**

- **fixed**: fixes to cl utils

**v2.1.0 - May 20, 2020**

- Now compatible with Python 3.\*

## PYTHON MODULE INDEX

### C

`sherlock.commonutils`, 11

### i

`sherlock.imports`, 12

### U

`sherlock.utKit`, 13



## A

add\_data\_to\_database\_table() (*ifs method*), 20  
 add\_data\_to\_database\_table() (*ned method*), 22  
 add\_data\_to\_database\_table() (*ned\_d method*), 24  
 add\_data\_to\_database\_table() (*veron method*), 26  
 angular\_crossmatch\_against\_catalogue() (*transient\_catalogue\_crossmatch method*), 27

## C

catalogue\_conesearch (*class in sherlock*), 14  
 classification\_annotations() (*transient\_classifier method*), 32  
 classify() (*transient\_classifier method*), 32  
 clean() (*database\_cleaner method*), 20  
 connect() (*database method*), 18

## D

database (*class in sherlock*), 18  
 database\_cleaner (*class in sherlock*), 19

## G

generate\_match\_annotation() (*transient\_classifier method*), 33  
 get\_crossmatch\_catalogues\_column\_map() (*in module sherlock.commonutils*), 35  
 get\_project\_root() (*utKit method*), 13  
 getpackagepath() (*in module sherlock.commonutils*), 37

## I

ifs (*class in sherlock.imports*), 20  
 ingest() (*ifs method*), 21  
 ingest() (*ned method*), 22  
 ingest() (*ned\_d method*), 24  
 ingest() (*veron method*), 26

## M

match() (*transient\_catalogue\_crossmatch method*), 29

module

sherlock.commonutils, 11  
 sherlock.imports, 12  
 sherlock.utKit, 13

## N

ned (*class in sherlock.imports*), 21  
 ned\_d (*class in sherlock.imports*), 23

## P

physical\_separation\_crossmatch\_against\_catalogue() (*transient\_catalogue\_crossmatch method*), 29

## R

refresh\_database() (*utKit method*), 13

## S

search() (*catalogue\_conesearch method*), 16  
 setupModule() (*utKit method*), 13  
 sherlock.commonutils  
   module, 11  
 sherlock.imports  
   module, 12  
 sherlock.utKit  
   module, 13

## T

tearDownModule() (*utKit method*), 13  
 transient\_catalogue\_crossmatch (*class in sherlock*), 26  
 transient\_classifier (*class in sherlock*), 30

## U

update() (*update\_wiki\_pages method*), 17  
 update\_classification\_annotations\_and\_summaries() (*transient\_classifier method*), 33  
 update\_peak\_magnitudes() (*transient\_classifier method*), 34  
 update\_wiki\_pages (*class in sherlock.commonutils*), 17  
 utKit (*class in sherlock.utKit*), 13

## V

veron (*class in sherlock.imports*), 24