msp2db Documentation

Release 0.0.1

Thomas N. Lawson

May 04, 2021

Contents:

Python package to create an SQLite database from a collection of MSP mass spectromertry spectra files. Currently works with MSP files formated as MassBank records or as MoNA records.

The resulting SQLite database can be used for spectral matching with msPurity Bioconductor R package, see vigenette.

CHAPTER 1

Quick start

1.1 Installation

```
$ pip install msp2db
```

1.2 Command line

```
$ msp2db --msp_pth [msp file or directory of msp files] --source [name of source of␣
↪msp e.g. massbank] -out_pth [out dir]
$ msp2db --help

usage: PROG [-h] -m MSP_PTH -s SOURCE [-o OUT_PTH] [-t TYPE] [-d] [-l MSLEVEL]
        [-c CHUNK] [-x SCHEMA]

Convert msp to SQLite or MySQL database

optional arguments:
    -h, --help            show this help message and exit
    -m MSP_PTH, --msp_pth MSP_PTH
                          path to the MSP file (or directory of msp files)
    -s SOURCE, --source SOURCE
                          Name of data source (e.g. MassBank, LipidBlast)
    -o OUT_PTH, --out_pth OUT_PTH
                          file path for SQLite database
    -t TYPE, --db_type TYPE
                          database type [mysql, sqlite]
    -d, --delete_tables   delete tables
    -l MSLEVEL, --mslevel MSLEVEL
                          ms level of fragmentation if not detailed in msp file
    -c CHUNK, --chunk CHUNK
                          Chunks of spectra to parse data (useful to control
```

(continues on next page)

(continued from previous page)

```
                            memory usage)
    -x SCHEMA, --schema SCHEMA
                            Type of schema used (by default is "mona" msp style
                            but can use "massbank" style)

---------------
```

1.3 API

```
db_pth = 'spectral_library_07112018v1.db'
create_db(file_pth=db_pth, db_type='sqlite', db_name='spectra')
libdata = LibraryData(msp_pth='MoNA-export-FAHFA.msp',
                  db_pth=db_pth,
                  db_type='sqlite',
                  d_form=None,
                  schema='mona',
                  source='fahfa',
                  mslevel=None,
                  chunk=200)
```

CHAPTER 2

Spectral matching with msPurity

msPurity is an R package that assess precursor ion contribution within a gas phase fragmentation isolation window for mass spectrometry.

The package also has functionality for spectral matching against an SQLite database. A default SQLite library database is provided within Bioconductor but any MSP files can be used to generate a library SQLite database using this python package.

CHAPTER 3

Spectral library creation with Django

The msp2db package can be used with the django-mbrowse package to populate a Django SQL database with any MSP files.

Currently under development.

CHAPTER 4

API

class msp2db.parse.**LibraryData**(*msp_pth*, *db_pth=None*, *mslevel=None*, *polarity=None*, *source=u'unknown'*, *db_type=u'sqlite'*, *password=None*, *user=None*, *mysql_db_name=None*, *chunk=200*, *schema=u'mona'*, *user_meta_regex=None*, *user_compound_regex=None*, *compound_lookup=True*, *celery_obj=False*)

MSP file parser to SQL databases

After creating a SQL database for the library spectra using create_db, MSP files can be parsed into the database using the LibraryData class.

Example

```
>>> from msp2db.db import create_db
>>> from msp2db.parse import LibraryData
>>> db_pth = 'spectral_library.db'
>>> create_db(file_pth=db_pth, db_type='sqlite', db_name='spectra')
>>> libdata = LibraryData(msp_pth='MoNA-export-FAHFA.msp',
>>>                       db_pth=db_pth,
>>>                       db_type='sqlite',
>>>                       schema='mona',
>>>                       source='fahfa',
>>>                       chunk=200)
```

Parameters

- **msp_pth** (*str*) – path to msp file or directory [required]
- **db_pth** (*str*) – path to sqlite database (only required when using SQLite database) [default None]
- **source** (*str*) – Source of the msp files (e.g. massbank) [default 'unknown']

- **mslevel** (*int*) – If the msp file does not contain the mslevel this can be defined here [default None]
- **polarity** (*str*) – If the msp file does not contain the polarity this can be defined here [default None]
- **db_type** (*str*) – The type of database to submit to (either 'sqlite', 'mysql' or 'django_mysql') [default sqlite]
- **user** (*str*) – Username for database (only required for non Django mysql databases) [default None]
- **password** (*str*) – Password for database (only required for non Django mysql databases) [default None]
- **mysql_db_name** (*str*) – Name of the mysql database (only required for non Django mysql databases) [default None]
- **chunk** (*int*) – Chunks of spectra to parse data (useful to control memory usage) [default 200]
- **schema** (*str*) – MSP files can vary based on how they were made, two standard schemas are available either 'mona' based on the MassBank of North America (MoNA) MSP files. And 'massbank' which is based on the more controlled MassBank MSP files https://github.com/MassBank/MassBank-data [default 'mona']
- **user_meta_regex** (*dict*) – For other MSP files not derived from either MoNA or MassBank a custom dictionary of regexes can be used [default None]
- **user_compound_regex** (*dict*) – For other MSP files not derived from either MoNA or MassBank a custom dictionary of regexes can be used [default None]
- **compound_lookup** (*boolean*) – Include compound lookup
- **celery_obj** (*boolean*) – If using Django a Celery task object can be used to keep track on ongoing tasks [default False]

Returns LibraryData object

close()
Close the database connections

get_compound_ids()
Extract the current compound ids in the database. Updates the self.compound_ids list

get_db_dict()
Get a dictionary of the library spectra from the associated database

Example

```
>>> from msp2db.db import create_db
>>> from msp2db.parse import LibraryData
>>> db_pth = 'spectral_library.db'
>>> create_db(file_pth=db_pth, db_type='sqlite', db_name='spectra')
>>> libdata = LibraryData(msp_pth='MoNA-export-FAHFA.msp',
>>>                       db_pth=db_pth,
>>>                       db_type='sqlite',
>>>                       schema='mona',
>>>                       source='fahfa',
```

(continues on next page)

(continued from previous page)

```
>>>                     chunk=200)
>>> libdata.db_dict()
```

If using a large database the resulting dictionary will be very large!

Returns A dictionary with the following keys 'library_spectra', 'library_spectra_meta', 'library_spectra_annotations', 'library_spectra_source' and 'metab_compound'. Where corresponding values for each key are list of list containing all the rows in the database.

insert_data(*remove_data=False*, *db_type=u'sqlite'*)

Insert data stored in the current chunk of parsing into the selected database

Parameters

- **remove_data** (*boolean*) – Remove the data stored within the LibraryData object for the current chunk of processing
- **db_type** (*str*) – The type of database to submit to either 'sqlite', 'mysql' or 'django_mysql' [default sqlite]

msp2db.parse.**add_splash_ids**(*splash_mapping_file_pth*, *conn*, *db_type=u'sqlite'*)

Add splash ids to database (in case stored in a different file to the msp files like for MoNA)

Example

```
>>> from msp2db.db import get_connection
>>> from msp2db.parse import add_splash_ids
>>> conn = get_connection('sqlite', 'library.db')
>>> add_splash_ids('splash_mapping_file.csv', conn, db_type='sqlite')
```

Parameters **splash_mapping_file_pth** (*str*) – Path to the splash mapping file (needs to be csv format and have no headers, should contain two columns. The first the accession number the second the splash. e.g. AU100601, splash10-0a4i-1900000000-d2bc1c887f6f99ed0f74

msp2db.db.**create_db**(*file_pth*)

Create an empty SQLite database for library spectra.

Example

```
>>> from msp2db.db import create_db
>>> db_pth = 'library.db'
>>> create_db(file_pth=db_pth)
```

Parameters **file_pth** (*str*) – File path for SQLite database

msp2db.db.**db_dict**(*c*)

Get a dictionary of the library spectra from a database

Example

```
>>> from msp2db.db import get_connection
>>> conn = get_connection('sqlite', 'library.db')
>>> test_db_d = db_dict(conn.cursor())
```

If using a large database the resulting dictionary will be very large!

Parameters **c** (*cursor*) – SQL database connection cursor

Returns A dictionary with the following keys 'library_spectra', 'library_spectra_meta', 'library_spectra_annotations', 'library_spectra_source' and 'metab_compound'. Where corresponding values for each key are list of list containing all the rows in the database.

msp2db.db.**get_connection**(*db_type*, *db_pth*, *user=None*, *password=None*, *name=None*)
Get a connection to a SQL database. Can be used for SQLite, MySQL or Django MySQL database

Example

```
>>> from msp2db.db import get_connection
>>> conn = get_connection('sqlite', 'library.db')
```

If using "mysql" mysql.connector needs to be installed.

If using "django_mysql" Django needs to be installed.

Parameters **db_type** (*str*) – Type of database can either be "sqlite", "mysql" or "django_mysql"

Returns sql connection object

msp2db.db.**insert_query_m**(*data*, *table*, *conn*, *columns=None*, *db_type=u'mysql'*)
Insert python list of tuples into SQL table

Parameters

- **data** (*list*) – List of tuples
- **table** (*str*) – Name of database table
- **conn** (*connection object*) – database connection object
- **columns** (*str*) – String of column names to use if not assigned then all columns are presumed to be used [Optional]
- **db_type** (*str*) – If "sqlite" or "mysql"

msp2db.re.**get_compound_regex**(*schema=u'mona'*)
Create a dictionary of regex for extracting the compound information for the spectra

msp2db.re.**get_meta_regex**(*schema=u'mona'*)
Create a dictionary of regex for extracting the meta data for the spectra

msp2db.utils.**get_blank_dict**(*d*)
Remove values from dictionary

Parameters **d** (*dict*) – any dictionary

Returns dictionary with blank values

msp2db.utils.**get_precursor_mz**(*exact_mass*, *precursor_type*)
Calculate precursor mz based on exact mass and precursor type

Parameters

- **exact_mass** (*float*) – exact mass of compound of interest

- **precursor_type** (*str*) – Precursor type (currently only works with '[M-H]-', '[M+H]+' and '[M+H-H2O]+'

Returns neutral mass of compound

msp2db.utils.**line_count**(*fn*)

Get line count of file

Parameters **fn** (*str*) – Path to file

Returns Number of lines in file (int)

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

m

Index