MS2Network User Manual

Author: Ikemefuna Lawrence Ezechukwu

Supervisor: Dr Simon Rogers

August, 2018

Contents

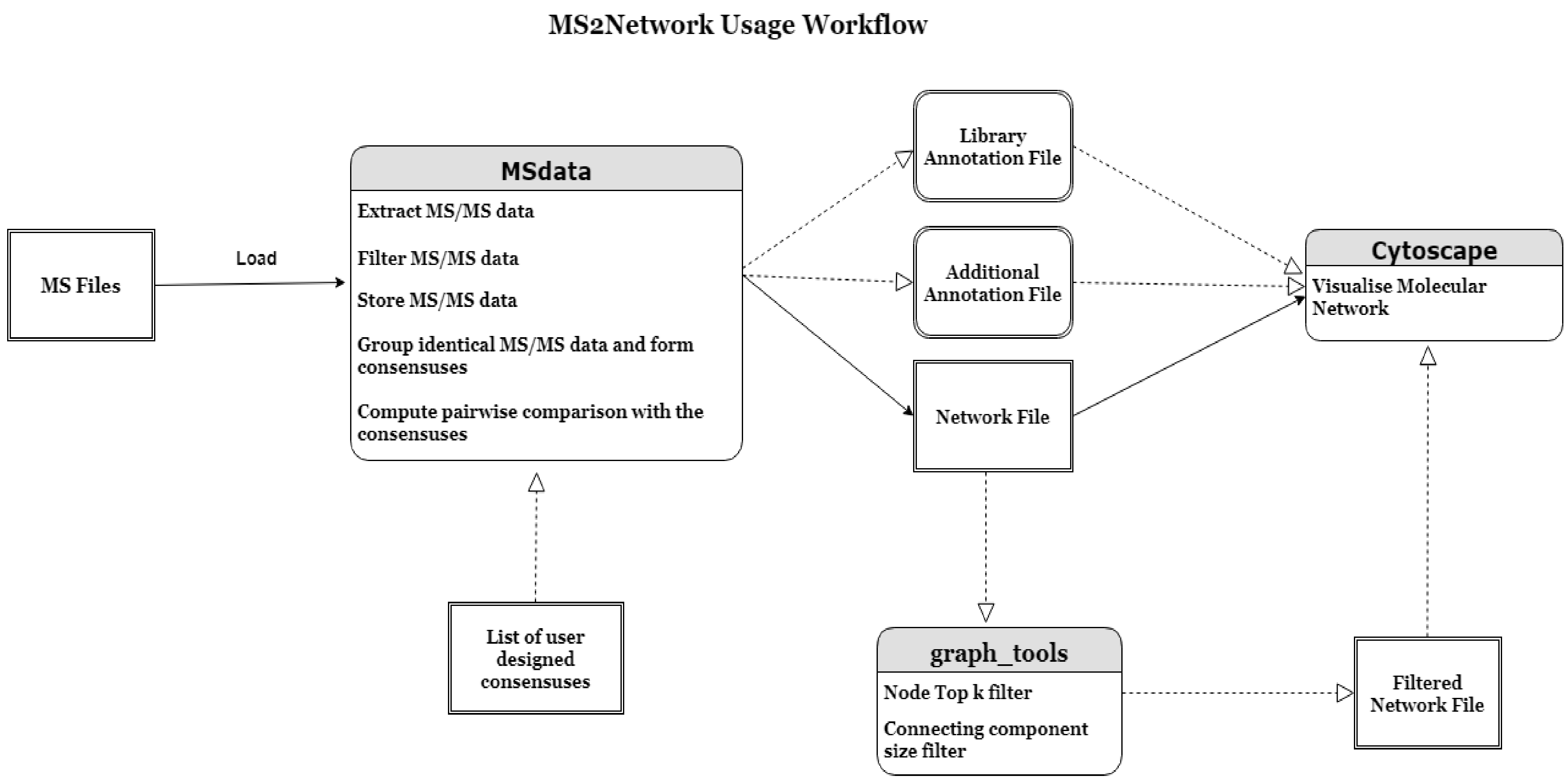

MS2Network Usage Workflow
MS Files
Load
MSdata
Extract MS/MS data
Filter MS/MS data
Store MS/MS data
Group identical MS/MS data and form consensuses
Compute pairwise comparison with the consensuses
List of user designed consensuses
Library Annotation File
Additional Annotation File
Network File
graph_tools
Node Top k filter
Connecting component size filter
Filtered Network File
Cytoscape
Visualise Molecular Network

Figure 1: MS2Network usage workflow. Dashed lines represents optional paths in the workflow. Solid lines represent the basic path of the workflow. The user may start with mass-spectrometry files or a Python list of consensuses (Python objects representing clusters of identical spectra). The list of consensuses are used to generate the network file. The network file can be inputted directly into Cytoscape, or filtered prior to that.

Introduction

The MS2Network package contains five modules that aid in the development of a molecular networking pipeline. This manual aims to briefly, yet comprehensively, explain how to utilise these modules. It was written up very quickly, and focuses of the key functions required to develop a molecular networking pipeline.

Basic Usage

The MS2Network package is a Python extension that is dependent on two other Python extensions, pymzML and Networkx. These dependencies have to be manually installed.

The development of a molecular networking pipeline directly involves the use of two MS2Network modules (ms_profile and graph_tools), and the Networkx package. The following examples are executed within a Python development environment (indicated by ">>>"), but can equally be incorporated in standalone scripts.

```
>>> import networkx as nx
>>> from ms2network import ms_profile as mp
>>> from ms2network import graph_tools as gt
```

The MSdata class, supplied by the ms_profile module, is responsible for mediating the core processes involved in the pipeline development. It is able to load mass spectrometry(MS) datasets, filter and store tandem MS (MS/MS) spectral information, and perform a pairwise similarity comparison on the stored tandem MS spectra. Multiple instances of the class can be created if multiple pipelines are desired to be developed at a time.

```
>>> dataset1 = mp.MSdata()
>>> dataset2 = mp.MSdata()
```

The MS input files can be in the mgf or the mzML format. The format of the file, and the experiment group of the data in the file have to be specified before loading it into an MSdata object. Only one file can be loaded in at a time. The parse function of the MSdata class is used to extract, filter, and store the MS/MS data from the input files. The tolerance for the precursor m/z filter can be specified.

```
>>> dataset1.add_file("E.coli_extract.mzml", "mzml", "E.coli")
>>> dataset1.add_file("Arabidopsis_extract.mgf", "mgf",
"Arabidopsis")
>>> dataset1.parse(tolerance=15)
```

The MSdata object returns an iterator, hence the user can loop over all spectra using classic Python syntax. Each iteration returns a Spectrum object (**ms2network.spec.Spectrum**) which provides information (both extracted and derived) on a spectrum. Additionally, the user can find out the amount of MS/MS data that has been stored, and can retrieve a specific spectrum via its ID and experiment group. The latter is particularly useful if the user is interested in certain spectra after viewing the molecular network.

```
>>> for spectrum in dataset1:
>>>   if spectrum.experiment_group == "E.coli":
>>>         print spectrum.ID
```

```
>>> Ecoli_64 = dataset1.get_spectrum(ID="64", group="E.coli")
>>> len(dataset1)
# 3500
```

To group spectra into clusters of identical spectra, the MSdata "**build_consensus**" function is called. Spectra in a cluster will have identical precursor m/z values (within a specified tolerance) and will all have similar spectral patterns, with a similarity score equal to/above the specified minimum. One spectrum is chosen from a cluster to represent the cluster. This spectrum is referred to as a consensus (a cluster representatives). Each cluster is modelled by a Consensus object (ms2network.consensus.Consensus). The consensus object is very similar to a Spectrum object. It has the major attributes of the consensus (cluster representative), but it stores all the members of a cluster in an attribute called "**spectral_data**". The result of the "**build_consensus**" operation is a list of Consensus objects. This list can be accessed via the MSdata attribute "**consensus_data**".

Note: A Consensus object could contain singletons, or a group of identical spectra. The reason is because a bottom-up hierarchical clustering algorithm is used to generate consensuses. Using this approach, a spectrum may not find any identical spectrum to merge with, and thus remains by itself in a Consensus object. The Consensus objects with more than one members are stored in the MSdata.clustered attribute, while the ones with single members are stored in the MSdata.unclustered attribute.

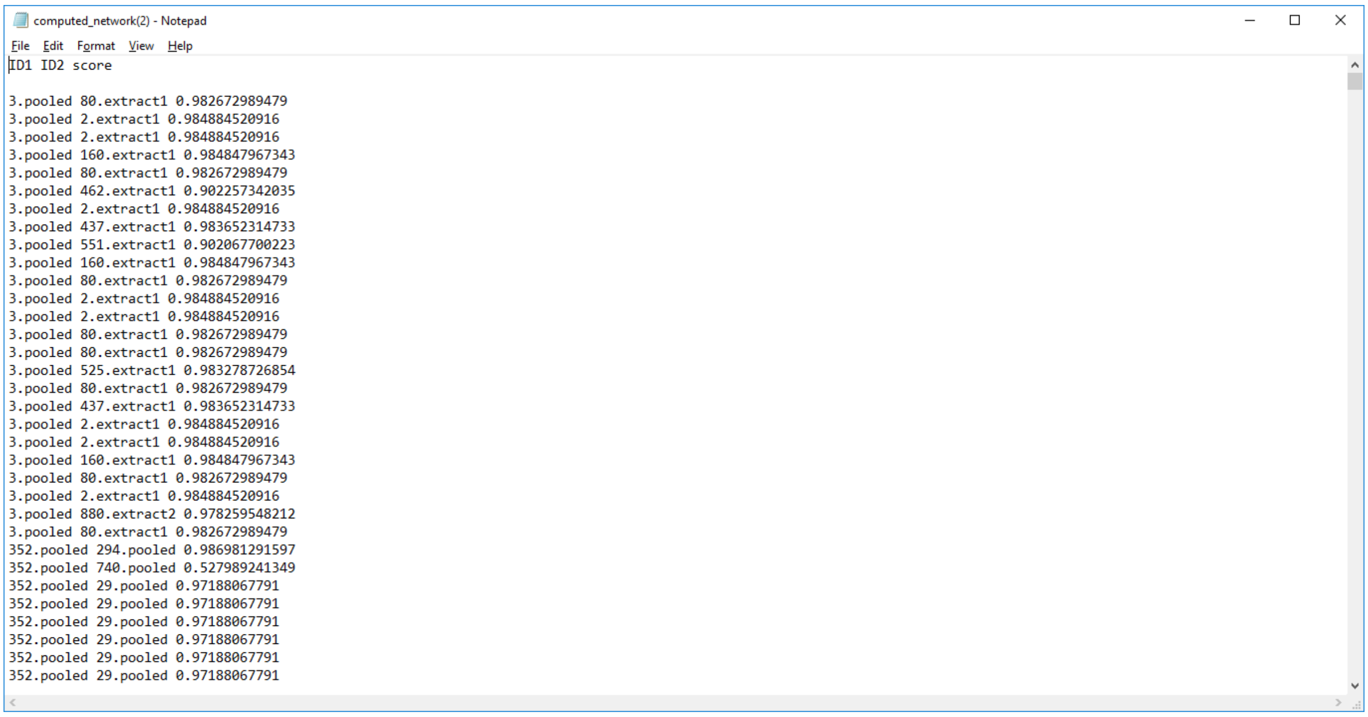
```
>>> dataset1.build_consensus(pm_range=0.5, score_threshold=0.8)
>>> print len(dataset1. consensus_data)
>>> print len(dataset1.clustered)
>>> print len(dataset1.unclustered)

# 1850
# 1126
# 724
```

Alternatively, the user may load in a Python list of their own Consensus objects. It will be compatible with the downstream pairwise comparison operation as long as it possesses three particular attributes: “**normalised_peaks**”, “**n_peaks**”, and “**parent_mz**”.

```
>>> spectra = list()
>>> for spectrum in dataset1:
>>>   spectra.append(spectrum)

…… User designed clustering algorithm

>>> my_consensuses  # User’s list of consensuses

>>> dataset1.load_consensus(my_consensuses)
```

The molecular network is a way of interpreting the pairwise relatedness of MS/MS spectra in the input data. The pairwise similarity scoring is computed from the list of consensuses. The result is a file consisting of a list of weighted edges (Figure 2). A pair of MS/MS spectra (their IDs specifically) represents an edge, and their similarity score represents the weight. The degree of relatedness in a molecular network is controlled by the minimum value of similarity allowed, which can be specified by the user. This file can either be further processed via the graph_tools module or directly inputted into Cytoscape to visualise the molecular network.

```
>>> dataset1.compute_molecular_network(“computed_network(2).txt”,
edge_filter=0.5)
```



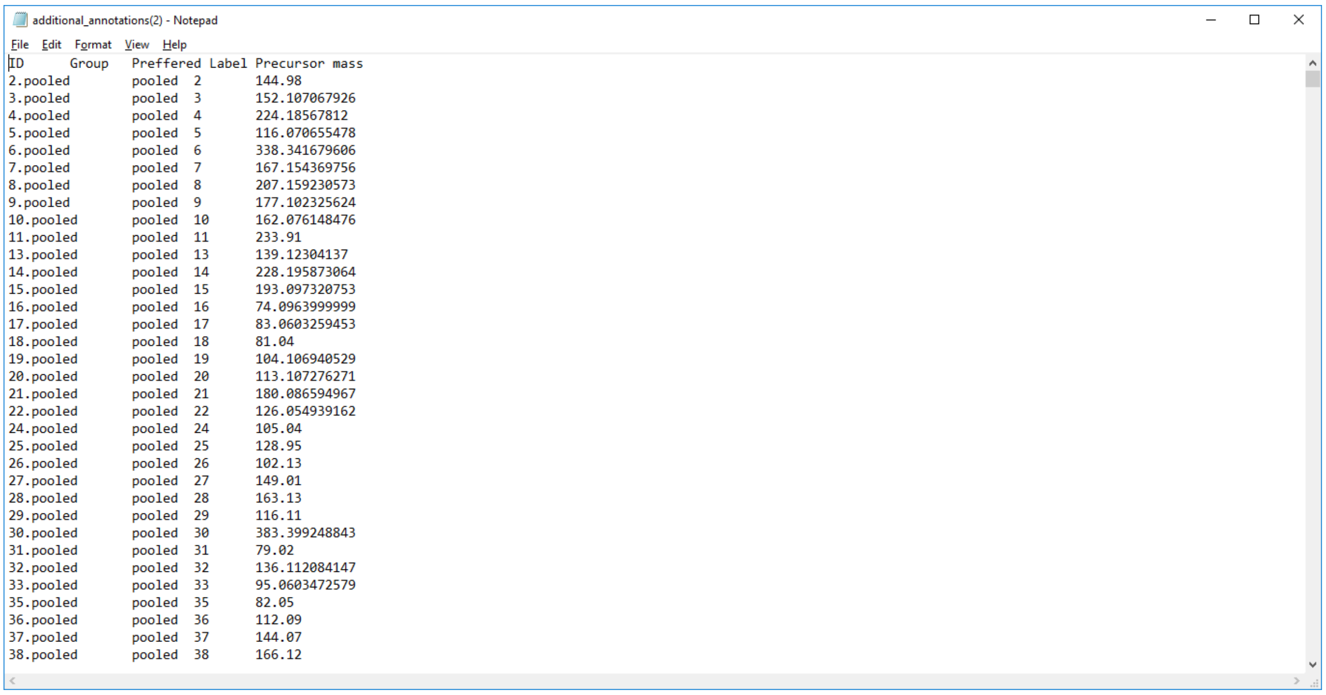
```
ID1 ID2 score

3.pooled 80.extract1 0.982672989479
3.pooled 2.extract1 0.984884520916
3.pooled 2.extract1 0.984884520916
3.pooled 160.extract1 0.984847967343
3.pooled 80.extract1 0.982672989479
3.pooled 462.extract1 0.902257342035
3.pooled 2.extract1 0.984884520916
3.pooled 437.extract1 0.983652314733
3.pooled 551.extract1 0.902067700223
3.pooled 160.extract1 0.984847967343
3.pooled 80.extract1 0.982672989479
3.pooled 2.extract1 0.984884520916
3.pooled 2.extract1 0.984884520916
3.pooled 80.extract1 0.982672989479
3.pooled 80.extract1 0.982672989479
3.pooled 525.extract1 0.983278726854
3.pooled 80.extract1 0.982672989479
3.pooled 437.extract1 0.983652314733
3.pooled 2.extract1 0.984884520916
3.pooled 2.extract1 0.984884520916
3.pooled 160.extract1 0.984847967343
3.pooled 80.extract1 0.982672989479
3.pooled 2.extract1 0.984884520916
3.pooled 880.extract2 0.978259548212
3.pooled 80.extract1 0.982672989479
352.pooled 294.pooled 0.986981291597
352.pooled 740.pooled 0.527989241349
352.pooled 29.pooled 0.97188067791
352.pooled 29.pooled 0.97188067791
352.pooled 29.pooled 0.97188067791
352.pooled 29.pooled 0.97188067791
352.pooled 29.pooled 0.97188067791
352.pooled 29.pooled 0.97188067791
```

Figure 2: Example of a network file (list of weighted edges)

Additional files (Figure 3) can be generated from the MSdata object and inputted as node tables into Cytoscape to facilitate group and attribute mapping. The file generated from the MSdata "**spectrum_annotation_file**" function is used for group mapping. The second file is generated from an MS/MS spectral library search. Spectral library files first have to be loaded and parsed, before it can be searched for a match to the input MS/MS data. A match in the MS/MS library is one with an identical precursor m/z value and a similarity score above a defined threshold.

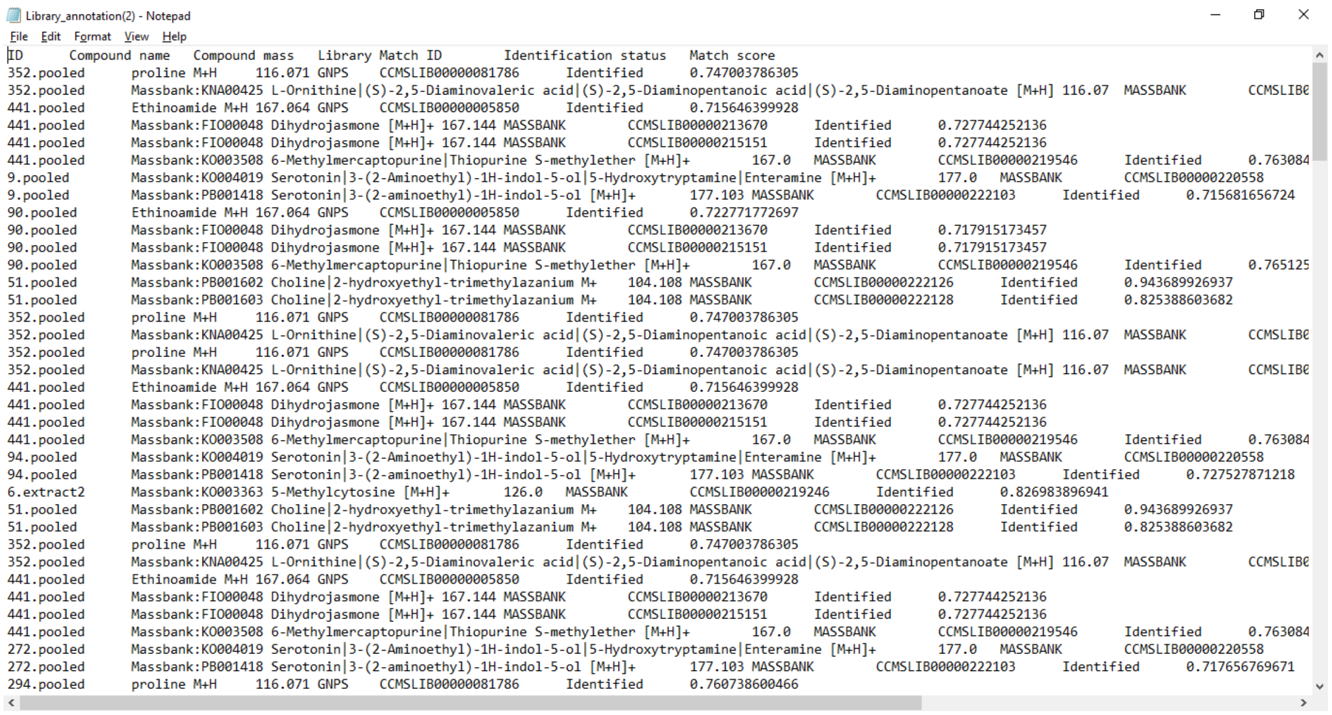
```
>>> dataset1.spectrum_annotation_file(“Group_annotation.txt”)
```

```
>>> dataset1.add_library(“ALL_GNPS.mgf”, “mgf”, “GNPS”)
>>> dataset1.add_library(“MASSBANK.mgf”, “mgf”, “MASSBANK”)
>>> dataset1.parse_library()
>>> dataset1.find_library_match(“Library_file.txt”,
score_threshold=0.7, pm_range=0.5)
```



additional_annotations(2) - Notepad

```
ID	Group	Preffered Label	Precursor mass
2.pooled	pooled	2	144.98
3.pooled	pooled	3	152.107067926
4.pooled	pooled	4	224.18567812
5.pooled	pooled	5	116.070655478
6.pooled	pooled	6	338.341679606
7.pooled	pooled	7	167.154369756
8.pooled	pooled	8	207.159230573
9.pooled	pooled	9	177.102325624
10.pooled	pooled	10	162.076148476
11.pooled	pooled	11	233.91
13.pooled	pooled	13	139.12304137
14.pooled	pooled	14	228.195873064
15.pooled	pooled	15	193.097320753
16.pooled	pooled	16	74.0963999999
17.pooled	pooled	17	83.0603259453
18.pooled	pooled	18	81.04
19.pooled	pooled	19	104.106940529
20.pooled	pooled	20	113.107276271
21.pooled	pooled	21	180.086594967
22.pooled	pooled	22	126.054939162
24.pooled	pooled	24	105.04
25.pooled	pooled	25	128.95
26.pooled	pooled	26	102.13
27.pooled	pooled	27	149.01
28.pooled	pooled	28	163.13
29.pooled	pooled	29	116.11
30.pooled	pooled	30	383.399248843
31.pooled	pooled	31	79.02
32.pooled	pooled	32	136.112084147
33.pooled	pooled	33	95.0603472579
35.pooled	pooled	35	82.05
36.pooled	pooled	36	112.09
37.pooled	pooled	37	144.07
38.pooled	pooled	38	166.12
```



Library_annotation(2) - Notepad

```
ID	Compound name	Compound mass	Library Match ID	Identification status	Match score
352.pooled	proline M+H	116.071 GNPS	CCMSLIB00000081786	Identified	0.747003786305
352.pooled	Massbank:KNA00425 L-Ornithine|(S)-2,5-Diaminovaleric acid|(S)-2,5-Diaminopentanoic acid|(S)-2,5-Diaminopentanoate [M+H] 116.07	MASSBANK	CCMSLIB0
441.pooled	Ethinoamide M+H 167.064 GNPS	CCMSLIB00000005850	Identified	0.715646399928
441.pooled	Massbank:FI000048 Dihydrojasmone [M+H]+ 167.144 MASSBANK	CCMSLIB00000213670	Identified	0.727744252136
441.pooled	Massbank:FI000048 Dihydrojasmone [M+H]+ 167.144 MASSBANK	CCMSLIB00000215151	Identified	0.727744252136
441.pooled	Massbank:KO003508 6-Methylmercaptopurine|Thiopurine S-methylether [M+H]+	167.0	MASSBANK	CCMSLIB00000219546	Identified	0.763084
9.pooled	Massbank:KO004019 Serotonin|3-(2-Aminoethyl)-1H-indol-5-ol|5-Hydroxytryptamine|Enteramine [M+H]+	177.0	MASSBANK	CCMSLIB00000220558
9.pooled	Massbank:PB001418 Serotonin|3-(2-aminoethyl)-1H-indol-5-ol [M+H]+	177.103 MASSBANK	CCMSLIB00000222103	Identified	0.715681656724
90.pooled	Ethinoamide M+H 167.064 GNPS	CCMSLIB00000005850	Identified	0.722771772697
90.pooled	Massbank:FI000048 Dihydrojasmone [M+H]+ 167.144 MASSBANK	CCMSLIB00000213670	Identified	0.717915173457
90.pooled	Massbank:FI000048 Dihydrojasmone [M+H]+ 167.144 MASSBANK	CCMSLIB00000215151	Identified	0.717915173457
90.pooled	Massbank:KO003508 6-Methylmercaptopurine|Thiopurine S-methylether [M+H]+	167.0	MASSBANK	CCMSLIB00000219546	Identified	0.765125
51.pooled	Massbank:PB001602 Choline|2-hydroxyethyl-trimethylazanium M+	104.108 MASSBANK	CCMSLIB00000222126	Identified	0.943689926937
51.pooled	Massbank:PB001603 Choline|2-hydroxyethyl-trimethylazanium M+	104.108 MASSBANK	CCMSLIB00000222128	Identified	0.825388603682
352.pooled	proline M+H	116.071 GNPS	CCMSLIB00000081786	Identified	0.747003786305
352.pooled	Massbank:KNA00425 L-Ornithine|(S)-2,5-Diaminovaleric acid|(S)-2,5-Diaminopentanoic acid|(S)-2,5-Diaminopentanoate [M+H] 116.07	MASSBANK	CCMSLIB0
352.pooled	proline M+H	116.071 GNPS	CCMSLIB00000081786	Identified	0.747003786305
352.pooled	Massbank:KNA00425 L-Ornithine|(S)-2,5-Diaminovaleric acid|(S)-2,5-Diaminopentanoic acid|(S)-2,5-Diaminopentanoate [M+H] 116.07	MASSBANK	CCMSLIB0
441.pooled	Ethinoamide M+H 167.064 GNPS	CCMSLIB00000005850	Identified	0.715646399928
441.pooled	Massbank:FI000048 Dihydrojasmone [M+H]+ 167.144 MASSBANK	CCMSLIB00000213670	Identified	0.727744252136
441.pooled	Massbank:FI000048 Dihydrojasmone [M+H]+ 167.144 MASSBANK	CCMSLIB00000215151	Identified	0.727744252136
441.pooled	Massbank:KO003508 6-Methylmercaptopurine|Thiopurine S-methylether [M+H]+	167.0	MASSBANK	CCMSLIB00000219546	Identified	0.763084
94.pooled	Massbank:KO004019 Serotonin|3-(2-Aminoethyl)-1H-indol-5-ol|5-Hydroxytryptamine|Enteramine [M+H]+	177.0	MASSBANK	CCMSLIB00000220558
94.pooled	Massbank:PB001418 Serotonin|3-(2-aminoethyl)-1H-indol-5-ol [M+H]+	177.103 MASSBANK	CCMSLIB00000222103	Identified	0.727527871218
6.extract2	Massbank:KO003363 5-Methylcytosine [M+H]+	126.0	MASSBANK	CCMSLIB00000219246	Identified	0.826983896941
51.pooled	Massbank:PB001602 Choline|2-hydroxyethyl-trimethylazanium M+	104.108 MASSBANK	CCMSLIB00000222126	Identified	0.943689926937
51.pooled	Massbank:PB001603 Choline|2-hydroxyethyl-trimethylazanium M+	104.108 MASSBANK	CCMSLIB00000222128	Identified	0.825388603682
352.pooled	proline M+H	116.071 GNPS	CCMSLIB00000081786	Identified	0.747003786305
352.pooled	Massbank:KNA00425 L-Ornithine|(S)-2,5-Diaminovaleric acid|(S)-2,5-Diaminopentanoic acid|(S)-2,5-Diaminopentanoate [M+H] 116.07	MASSBANK	CCMSLIB0
441.pooled	Ethinoamide M+H 167.064 GNPS	CCMSLIB00000005850	Identified	0.715646399928
441.pooled	Massbank:FI000048 Dihydrojasmone [M+H]+ 167.144 MASSBANK	CCMSLIB00000213670	Identified	0.727744252136
441.pooled	Massbank:FI000048 Dihydrojasmone [M+H]+ 167.144 MASSBANK	CCMSLIB00000215151	Identified	0.727744252136
441.pooled	Massbank:KO003508 6-Methylmercaptopurine|Thiopurine S-methylether [M+H]+	167.0	MASSBANK	CCMSLIB00000219546	Identified	0.763084
272.pooled	Massbank:KO004019 Serotonin|3-(2-Aminoethyl)-1H-indol-5-ol|5-Hydroxytryptamine|Enteramine [M+H]+	177.0	MASSBANK	CCMSLIB00000220558
272.pooled	Massbank:PB001418 Serotonin|3-(2-aminoethyl)-1H-indol-5-ol [M+H]+	177.103 MASSBANK	CCMSLIB00000222103	Identified	0.717656769671
294.pooled	proline M+H	116.071 GNPS	CCMSLIB00000081786	Identified	0.760738600466
```

Figure 3: Additional files used for group and attribute mapping in Cytoscape.
The top file is used for group mapping. The bottom file is used for mapping library annotated compounds.

Module ms_profile

This module supplies an MSdata class, and a function for generating a dictionary that facilitates the easy retrieval of compound names for library-annotated MS/MS spectra.

ms2Network.ms_profile.MSdata() – returns an instance of the MSdata class

Class MSdata Functions

Note: Not all the functions are mentioned below, only the ones I feel the user should be concerned with.

add_library(library_path, file_format, library_name) – adds the path of an MS/MS library file to MSdata.libraries

Parameters:
- library_path (*String*) – the path to the MS/MS library file
- file_format (*String*) – the format of the MS/MS library file. Can only be either "mgf" or "mzml".
- library_name (*String*) – the name of the library. It can be any name. It is used to keep track of the library in which a match was found.

add_file(path, file_format, group_name) – adds the path of an MS file to MSdata.files

Parameters:
- path (String) – the path to the MS file
- file_format (*String*) – the format of the MS file. Can only be either "mgf" or "mzml".
- group_name (*String*) – the name of the experiment group from which the data was derived. It can be any name. It required for group mapping in Cytoscape.

parse_library(limit=False, NUM_OF_SPECTRA=10, tolerance=17) – extracts the information of each MS/MS spectra in the file, and uses that information to create a Spectrum object for each MS/MS spectra. These Spectrum objects are stored in MSdata.library_spec_data

parse(limit=False, NUM_OF_SPECTRA=10, tolerance=17) – operates the same as parse_library except that Spectrum objects are stored in MSdata.spectral_data

Parameters:
- limit (*Boolean*) – to specify whether or not the user wants to parse a specific number of spectra. The default is FALSE. If it is TRUE,

then the parameter NUM_OF_SPECTRA will affect the operation of the parse Function.

• NUM_OF_SPECTRA (*Integer*) – the number of spectra the user wants to parse from the MS_file(s) that has/have been loaded into MSdata.

• tolerance (*Float*) – the deviation allowed for the precursor m/z value when filtering out the precursor peak.

reset_library() - removes all libraries that have been stored in MSdata.libraries.

next() – will return an instance of the ms2network.spec.Spectrum, stored in MSdata.spectral_data

len() – returns the length of MSdata.spectral_data.

build_consensus(tolerance=0.3, min_match=6, pm_range=0.5, score_threshold=0.9) – populates MSdata.consenus_data with clusters of identical spectra (represented as Consensus objects). Consensus objects with more than one spectra are added to MSdata.clustered, while those with only one are added to MSdata.unclustered.

Parameters:

• tolerance (*Float*) – the deviation allowed for peak positions (their m/z values) when searching for matched peaks. Required for calculating the spectral similarity score. The default is 0.3.

• min_match (*Integer*) – the minimum number of matched peaks that must be found before the similarity score is computed. If the minimum is not met the score will be 0. The default number is 6.

• pm_range (*Float*) – the deviation allowed for the precursor m/z value when grouping spectra of identical precursor m/z values. The default is 0.5.

• score_threshold (*Float*) – the minimum pairwise similarity score in a cluster of identical spectra.

compute_molecular_network(path, tolerance=0.3, min_match=6, pm_range=1.0, edge_filter=0.7) – performs a pairwise similarity scoring of consensuses in MSdata.consensus_data, and returns a file containing a list of weighted edges.

Parameters: Mostly the same as **build_consensus**.

- edge_filter (*Float*) – the minimum weight of an edge required for the edge to be outputted in the resulting network file. The default is 0.7.

- path (*String*) – the path for the output file.

find_library_match(path, tolerance=0.3, min_match=6, pm_range=0.5, score_threshold=0.9) – searches the loaded spectral libraries for MS/MS spectra that match to the ones extracted from experiment MS files. The result is a file that can be used for attribute mapping in Cytoscape.

Parameters: The same as **build_consensus** and **compute_molecular_network**

load_consensus(consensus) – reinitiates MSdata.consensus_data with a list of user designed Consensus objects.

Parameters: • consensus (*list*) – a list of user designed Consensus objects.

spectrum_from_library(**params) – returns Spectrum object representation of spectra stored in MSdata.library_spec_data.

get_consensus(**params) – returns Consensus object representation of a cluster of identical spectra, stored in MSdata.consensus_data.

get_spectrum(**params) – returns Spectrum object representation of spectra stored in MSdata.spectral_data

Parameters: Keyword arguments “ID” (*String*) and “group” (*String*).
ID is the index assigned to each spectrum. Files in the mzml format are usually assigned an index during the parsing process. Files in the mgf format are usually indexed in the file, and this index is extracted.
group is the experiment group that the spectrum belongs to. Spectrum from different files can share the same index, but never the same index and experiment group.

Spectrum_annotation_file(path) – returns a file used for facilitating group mapping in Cytoscape.

Parameters: • path (*String*) – the path for the output file.

Comparison_plot(spec1, spec2, path, tolerance=0.3, min_match=6, shift=True) – returns a comparison plot of two spectral plots. Blue spectral lines indicate matching peaks that were used in calculating the spectral similarity.

Parameters: Shares some parameters with **build_consensus.**

- spec1 and spec2 (*ms2network.spec.Spectrum*) – Spectrum objects.
- path (*String*) – the path for the output file. The png format is adviced.
- shift (*Boolean*) – specifies whether or not peaks are allowed to shift by the difference between matching precursor m/z values

---

ms2Network.ms_profile.Store_library_annotation(path**) –** returns a library annotation dictionary. The keys are the unique IDs (in the format ID.group) assigned to each MS/MS spectrum in the dataset. The values are the compound names of the MS/MS spectra.

Parameters: • path (*String*) – the path to the file generated from the **find_library_match** operation.

Module spec

This module supplies a Spectrum class, which is used to model the data present for an MS/MS spectrum..

ms2Network.spec.Spectrum(ID) - returns an instance of the Spectrum class.

Parameters: • ID (*String*) – The index assigned to an MS/MS spectrum.

Class Spectrum Functions

plot_spectrum(path) – returns an image of the spectral plot.

Parameters: • path (*String*) - the path for the output file. The png format is adviced.

Module consensus

This module supplies a Consensus class, and an additional function for merging instances of the Consensus class. These facilitate the grouping of identical spectra into clusters.

ms2Network.consensus.Consensus(spec**)** – returns an instance of the Consensus class. The consensus class is instantiated with a Spectrum object.

Parameters: • spec (*ms2network.spec.Spectrum*) – Spectrum object

Class Consensus Functions

Add(new_spec**)** – adds a Spectrum object to Consensus.spectral_data

Parameters: • new_spec (*ms2network.spec.Spectrum*) – Spectrum object

next() – The python iterator. It returns an instance of the ms2network.spec.Spectrum, stored in Consensus.spectral_data

Update() – the Spectrum with the highest sum of peak intensities will have its major attributes as the major attributes of the Consensus object. These class attributes are Consensus.ID, Consensus.parent_m/z, Consensus.normalised_peaks, Consensus.n_peaks, and Consensus.experiment_group.

plot_spectrum(path**) -** returns an image of the representatives spectral plot.

len() - returns the length of Consensus.spectral_data.

---

ms2Network.consensus.merge_consensus(con1**,** con2**)** – returns a Consensus object whose "**spectral_data**" attribute stores the Spectrum objects from both inputted Consensus objects. In this way, the resulting Consensus object appears as a combination of the two inputted Consensus objecs.

Parameter: • con1 and con2 (*ms2network.consensus.Consensus*) – Consensus objects

Module graph_tools

This module supplies functions that help to further simplify molecular networks.

ms2network.generate_edgeList(path**) –** returns an edge list. An edge list is a list of tuples, with each tuple containing three elements: the start node, the end node, and a dictionary of the edge attributes (in this case the attribute is the similarity score)

Parameters: • path (*String*)– the path of the network file generated from the **compute_molecular_network** operation.

ms2network.filter_k(graph, k**) –** sets the maximum number of neighbouring nodes per node. An edge between two nodes is kept only if it is in the top k scoring edges of both nodes.

Parameters: • graph (*networkx.classes.graph.Graph*) – a networkx Graph object

• k (*Integer*)– the maximum number of neighbouring nodes allowed per node.

ms2network.manage_component_size(graph, size**)** – sets the maximum number of nodes allowed in a connecting component. It removes the lowest scoring edge in a connecting component until the size of the component is equal to/ below the specified size.

Parameters: • graph (*networkx.classes.graph.Graph*) – a networkx Graph object

• size (*Integer*) – the maximum number if nodes allowed in a connecting Component.

Module scoring_functions

Supplied by my supervisor, Dr Simon Rogers.

ms2network.scoring_functions.fast_cosine(spec1, spec2, tolerance, min_match**)** - computes the spectral similarity score between two MS/MS spectra.

ms2network.scoring_functions.fast_cosine_shift(spec1, spec2, tolerance, min_match**)** – computes the spectral similarity score between two MS/MS spectra. This function allows peaks to shift by the difference between their precursor m/z values.

Parameters:

- spec1 and spec2 (Spectrum objects) – any spectrum object containing the following attributes: “**normalised_peaks**”, “**n_peaks**”, and “**parent_mz**”.
- tolerance (*Float*) – the deviation allowed for peak positions (their m/z values) when searching for matched peaks.
- min_match (*Integer*) – the minimum number of matched peaks that must be found before the similarity score is computed. If the minimum is not met the score will be 0.