Data Structure

February 10, 2015

1 Introduction

This is an ongoing documentation to describe the data structures used in this project. Implementations of the object models and data structures described in this report can be found inside `metabolomics_tools/discretisation/models.py`.

Throughout the whole report, N refers to the number of peaks in an input file and K refers to the total number of clusters. Peaks (i.e. features) are indexed by $n = 1, ..., N$ and clusters are indexed by $k = 1, ..., K$.

Remember that by design, generally $N == K$ in some of our models.

2 Data structures

A feature is a tuple of (m/z, RT, intensity) values, stored in a `Feature` object. Additionally, we also have a `DatabaseEntry` object, corresponding to a record from the molecule database. User-defined adduct transformations are stored in the `Transformation` object.

The `PeakData` object is a container of all those stuff above, plus more. Inside `PeakData`, we have the lists of features (`PeakData.features`), database entries (`PeakData.database`) and adduct transformations (`PeakData.transformations`). For computational convenience, the `PeakData` object also contains Numpy matrices of the m/z, RT and intensity values of peak features. Each attribute is stored in an N x 1 matrix. They are `PeakData.mass`, `PeakData.rt` and `PeakData.intensity`.

For the purpose of modelling, features are then discretised into bins – based on some user-defined mass and RT tolerances. The mapping between features to all the valid bins they can go into are stored in `PeakData.possible`, an N x K

Algorithm 1 FileLoader – example usage

```
from models import FileLoader

basedir = '.'

input_file = basedir + '/input/synthetic/synthdata_0.txt'
database_file = basedir + '/database/std1_20_mols.csv'
transformation_file = basedir + '/mulsub_synth.txt'
mass_tol, rt_tol = 2.0, 5.0

loader = FileLoader()
peak_data = loader.load_model_input(input_file, database_file,
    transformation_file, mass_tol, rt_tol)

# do whatever you want to peak_data after this
alpha, sigma, nsamps = 0.01, 20, 20
mbc = MassBinClusterer(peak_data, mass_tol, alpha, sigma, nsamps)
```

matrix where entries in the matrix are indices of the possible adduct transformations connecting a peak to any bin. This is subjected to the constraints on mass tolerance, RT tolerance and the additional constraint that the M+H adduct has to be the one with the largest intensity (i.e. there can be no other adduct transformation from a peak to any bin where the peak intensity is larger than the one for M+H). Additionally, there is another N x K matrix, `PeakData.transformed`, that stores the actual transformed mass values. The class `FileLoader` can be used to load some input file and prepare `PeakData` object and everything inside. See listing 1 for example usage.

The output from any of the clustering model we've developed is an N x K matrix of peak-to-cluster assignments. In the case of Gibbs sampling, this will usually contains the accumulated of counts of peak n in cluster k throughout the collected posterior samples.