

Note: This guide is usage only. The algorithm is described formally in the LaTeX document.

## Feature detection (MSExtract)

**To build binary:** In current revision of pwiz, (r1222), you have to explicitly build the binary for feature detection. From pwiz/pwiz/analysis/peakdetect, run:

```
% bjam bin
```

Binary will be installed in pwiz/pwiz/analysis/peakdetect/bin.

**Input:** mzXML/mzML file (if using other format, convert with msconvert first)

**Output:** .features file, XML representation of features detected.

## Command line usage string:

```
godzilla@temp-cpas:~/pwiz/pwiz/analysis/peakdetect/bin$ ./msextract
Usage: msextract [options] [file]
```

Options:

-c [ --config ] arg	: specify file of config options, in format optionName=optionValue
-d [ --defaults ]	: print configuration defaults
-o [ --outputPath ] arg (=.)	: specify output path
-f [ --featureDetectorImplementation ] arg (=Simple)	: specify implementation of FeatureDetector to use. Options: Simple, PeakelFarmer
--writeFeatureFile arg (=1)	: write xml representation of detected features (.features file)
--writeTSV arg (=1)	: write tab-separated file
--writeLog arg (=0)	: write log file (for debugging)
--filter arg	: add a spectrum list filter
	filter options:
	index
	[indexBegin,indexEnd]
	...
	mzWindow [mzLow,mzHigh]
	peakPicking prefer
	vendor peak picking:
	<true false>
	[msLevelsBegin,msLevelsEnd] ...
	precursorRecalculation
	(based on ms1 data)
	scanNumber
	[scanNumberBegin,scanNumberEnd] ...
	scanEvent
	[scanEventBegin,scanEventEnd] ...
	scanTime
	[scanTimeLow,scanTimeHigh]
	stripIT (strip ion trap ms1 scans)

#### FeatureDetectorPeakel Options:

<code>--noiseCalculator_2Pass.zValueCutoff arg (=1)</code>	: specify cutoff for NoiseCalculator_2Pass
<code>--peakFinder_SNR.windowRadius arg (=1)</code>	: specify window radius for PeakFinder_SNR
<code>--peakFinder_SNR.zValueThreshold arg (=3)</code>	: specify z threshold for PeakFinder_SNR
<code>--peakFitter_Parabola.windowRadius arg (=1)</code>	: specify window radius for PeakFitter_Parabola
<code>--peakelGrower_Proximity.mzTolerance arg</code>	: specify mz tolerance for PeakelGrower_Proximity
<code>--peakelGrower_Proximity.rtTolerance arg (=10)</code>	: specify rt tolerance for PeakelGrower_Proximity
<code>--peakelPicker_Basic.minCharge arg (=2)</code>	: specify min charge for PeakelPicker_Basic
<code>--peakelPicker_Basic.maxCharge arg (=5)</code>	: specify max charge for PeakelPicker_Basic
<code>--peakelPicker_Basic.minMonoisotopicPeakelSize arg (=3)</code>	: specify min monoisotopic peakel size for PeakelPicker_Basic
<code>--peakelPicker_Basic.mzTolerance arg</code>	: specify mz tolerance for PeakelPicker_Basic
<code>--peakelPicker_Basic.rtTolerance arg (=5)</code>	: specify rt tolerance for PeakelPicker_Basic
<code>--peakelPicker_Basic.minPeakelCount arg (=3)</code>	: specify min peakel count for PeakelPicker_Basic

#### Examples:

```
# print default configuration parameters to config.txt
msextract -d > config.txt

# run using parameters in config.txt, output in outputdir
msextract -c config.txt -o outputdir file1.mzML file2.mzML

# filters: select scan numbers
msextract file1.mzML --filter "scanNumber [500,1000]"
```

#### Notes:

FeatureDetectorSimple aggregates rectangular feature by iterating through scans in retention time and accumulating isotope envelopes (“PeakFamily” objects) with the same number of isotopes, charge state, mz into Feature objects. A Feature must aggregate at least 2 PeakFamily objects (2 scans) to be reported. Currently, (r1222) a Feature is allowed to skip one scan between consecutive PeakFamily observations. Ask Darren re: FeatureDetectorPeakel

For reference, the Feature struct: (can be found in `pwiz/pwiz/data/misc/PeakData.hpp`)

```
struct PWIZ_API_DECL Feature
{
    Feature();
    Feature(const MSIHandler::Record& record);

    std::string id; // assigned by feature detection, for easier lookup
    double mz;
    double retentionTime;
    int charge;
    double totalIntensity;
```

```

double rtVariance; // calculated from child Peakels?
std::vector<PeakelPtr> peakels;

void calculateMetadata(); // Fills in mz, rt, totalIntensity, rtVariance,
    which require all of the Peakels for calculation

// retention time range calculation based on first two Peakels
double retentionTimeMin() const;
double retentionTimeMax() const;

void write(pwiz::minimxml::XMLWriter& xmlWriter) const;
void read(std::istream& is);

bool operator==(const Feature& that) const;
bool operator!=(const Feature& that) const;

private:
    Feature(Feature&);
    Feature operator=(Feature&);

};

```

## AMT Database generation and query (MSEharmony)

To build binaries: (r1222) Binaries are not built by default as the source code is not included in the Jamfile for pwiz/pwiz/analysis. Invoke bjam from pwiz/pwiz/analysis/eharmony (no bin argument necessary). Binaries will be installed in the same directory.

This will install two binaries, eharmony and mscupid.

### ./eharmony

**Input:** pep.xml and .features files for every experiment that should be aggregated into the database.

**Output:** database.xml, an xml representation of the database as a set of SpectrumQuery objects. (These are the same objects that we read pep.xml into – they store all the information that is in a single MS2.)

database.tsv, a tab-delimited file with the mass, retention time, and a few other attributes of the database objects.

### Command line usage string:

```

godzilla@temp-cpas:~/pwiz/pwiz/analysis/eharmony$ ./eharmony
Usage: eharmony [options] [filenames]

```

#### Options:

```

-i [ --inputPath ] arg          : specify location of input files
-o [ --outputPath ] arg         : specify output path
-f [ --filename ] arg           : specify file listing input runIDs (e.g
    .., 20090109-B-Run)
-w [ --warpFunctionCalculator ] arg : specify method of calculating the rt-c
    alibrating warp function.
    Options:
    default, linear, piecewiseLinear
-d [ --distanceAttribute ] arg   : specify distance attribute.
    Options:
    hammingDistance, numberOfMS2IDs, randomDi
    stance, rtDistributionDistance, weightedH
    ammingDistance

```

Notes: the -i option is not perfected. I think, for example, if you end the path with a /, it will add its own / also and fail to find your file. This is something I never handled, but it throws immediately when it can't open your file and writes out the name of the complete path, so you'll be able to see

what went wrong.

“default” retention time calibration is no calibration – retention time values go unchanged.

The distance attributes are summarized with the algorithm description.

## **./mscupid**

**Input:** pep.xml and .features file for the query experiment, existing AMT database in xml format.

**Output:** A variety of files for quality analysis, including an estimate of sensitivity and specificity using known MS2 ids. A pep.xml file storing all of the matches with scores, for downstream analysis (i.e., ProteinProphet). Output is handled through the Exporter struct (pwiz/pwiz/analysis/eharmony/Exporter.hpp). It's easy enough to add a new output file by adding a new write function to the struct and calling it within the query(args) function in pwiz/pwiz/analysis/eharmony/AMTDatabase.cpp.

### **Command line usage string:**

```
godzilla@temp-cpas:~/pwiz/pwiz/analysis/eharmony$ ./mscupid
Usage: mscupid [options] filename.pep.xml filename.features database.xml
```

Options:

```
-w [ --warpFunctionCalculator ] arg          : specify method of
                                              calculating the rt-calibrating
                                              warp function.
                                              Options:
                                              linear, piecewiseLinear
                                              Default:
                                              no calibration
-t [ --threshold ] arg (=0.9419999999999995) : specify threshold for match
                                              acceptance.
```

**Notes:** Database “Islandization”. The postprocessing step of making the database into islands is run as part of mscupid. The Island itself is described in the algorithm documentation. This speeds up the query quite a bit but has proven problematic in that the conversion from mz to mass shows big differences in mass for both true positives and false negatives – peptides whose mass we observed, according to the FeatureDetector's mz and charge, hundreds of daltons from where we ought to have. I think that this concept is a good one but it needs to be refined to work correctly. In particular, scores are extremely low and hard to differentiate. The method is also overly specific and tends to miss large amounts of true positives. (Related to the mass difference problem.)

You can run mscupid with or without this step. Without, if you are trying to get ROC info, it is slow since it continues to search until finding the “next closest” match for a query feature, meaning lots of search time. If you have a threshold and you only want to look up to that threshold, it is much faster as it is optimized to search within a small region.

### **Scripts for analyzing the result of the query:**

Run pwiz/pwiz/analysis/eharmony/scripts/eharmonyAnalyser.sh from the scripts directory itself. Takes two arguments – the first is the original pep.xml from your DB query and the second is the path to the directory containing the output files from your DB query (e.g., ../amtdb\_query/amtdb\_query95 for a default query with threshold 95.)

This will make four files:

**sequences.txt** : a list of all unique sequences found by MS1.5

**diff\_sequences.txt** : a list of the subset of the above that were not originally found by MS2

**roc.txt** : a two-column FPR\TPR text file for roc plotting

**roc.png** : a gnuplot image generated from roc.txt