Greylag User Guide

Overview

The basic workflow of greylag is as follows. First, a shuffled sequence database is prepared with `greylag-shuffle-database`. This generates a sequence database with one decoy sequence for each real sequence, which is required by `greylag-validate`.

Then, the spectra are searched using `greylag-rally`, which takes a configuration file and a set of spectrum files as input, and generates a glw file as output.

The `greylag-rally` program connects to one or more `greylag-chase` programs running on the same or different hosts, and these worker programs do the actual search. The `greylag-chase` programs can be started using any convienient means.

If the search has been broken into multiple pieces, `greylag-merge` can be used to merge the resulting glw files into a single glw file that represents the best matches found in all of the spectrum files.

Finally `greylag-sqt` is used to generate the corresponding sqt output files (one for each input spectrum file) from a glw file.

After the search, the program `greylag-validate` can be used to determine a set of "valid" identifications that satisfy a specified false discovery rate (FDR), if the spectra have been searched against a sequence database that includes decoys.

The auxilliary program `greylag-flatten-fasta` can be used to translate FASTA databases to and from a one-line-per-locus format that makes it easier to manipulate FASTA files using standard Unix utilities (e.g., `grep`).

Simple Example

Here is a transcript for a simple, non-specific search, performed locally on an eight-processor machine:

```
$ ls
greylag.conf  greylag.hosts  test.ms2  yeast.fasta
$ cat greylag.conf
[greylag]
databases = yeast_decoy.fasta
mass_regimes = AVG/MONO
pervasive_mods = +C2H3ON!@C
potential_mods = O@M oxidation '@'
potential_mod_limit = 4
parent_mz_tolerance = 1.25
$ greylag-shuffle-database yeast.fasta > yeast_decoy.fasta
six-mers: 2248190 real 2485103 decoy 315491 both
$ cat greylag.hosts
localhost:10078
```

```
localhost:10079
localhost:10080
localhost:10081
localhost:10082
localhost:10083
localhost:10084
localhost:10085
$ greylag-chase --port-count=8 &
[3] 21736
$ greylag-chase --port-count=8 &
[4] 21737
$ greylag-chase --port-count=8 &
[5] 21744
$ greylag-chase --port-count=8 &
[6] 21745
$ greylag-chase --port-count=8 &
[7] 21746
$ greylag-chase --port-count=8 &
[8] 21747
$ greylag-chase --port-count=8 &
[9] 21748
$ greylag-chase --port-count=8 &
[10] 21755
$ greylag-rally --hostfile=greylag.hosts greylag.conf test.ms2
$ greylag-sqt chase_unknown.glw
$ greylag-validate --kill *.sqt
52 real ids, 0 decoys (FDR = 0.0000) [p1]
```

This is just an example. Different command options and configuration file options will be appropriate, depending on the instrument and situation. Adding the `-v` flag to `greylag-rally`, for example, will show progress during the search.

Use the `--help` flag to see documentation for each command.

Host File

The host file read by `greylag-rally` consists of a series of lines like

```
host1234:10078
```

which indicates that a `greylag-chase` program is (potentially) running on host `host1234` and listening for connections on port `10078`. While `greylag-rally` is running, it will attempt to connect (and reconnect, if later disconnected) to each of these `greylag-chase` instances, once per minute.

> **Caution!**
>
> Currently `greylag-chase` will accept all connections with no authorization required. As such it probably should not be run on an insecure network. (This problem will be addressed in a future version.)

Configuration File

The configuration file specifies the search parameters and is read by `greylag-rally`. It uses a simple, readable format similar to the format of Windows .ini files. The file must begin with the section marker `[greylag]`. See the included example configuration file.

Except for `databases` and probably `pervasive_mods`, all parameters have reasonable defaults and often need not be specified. Note that if a parameter is specified more than once in the configuration file, the final occurrence takes precedence.

Here is a detailed description of each parameter.

databases

This is a whitespace-separated list of filenames. Each filename may be an absolute or relative path. *The named file must be visible at this path to both the* `greylag-rally` *and* `greylag-chase` *processes.*

Each file should contain a set of sequences in FASTA format. Locus names must be unique across all database files, to avoid confusion. Sequence characters are uppercased for uniformity. Non-amino-acid sequence characters are interpreted as "breaks"--candidate peptides do not cross these breaks. Specifically, only the residues

```
A C D E F G H I K L M N P Q R S T V W Y
```

are recognized.

This parameter must be specified--there is no default.

mass_regimes

This specifies the mass regimes to be searched. A *mass regime* is an assignment of masses to atoms, which in turn determines the mass of each residue. The two most familiar mass regimes are monoisotopic masses, denoted `MONO`, and average masses, denoted `AVG`. The former uses atomic masses that correspond to the most commonly found element isotopes. The latter uses atomic masses that are weighted averages, weighted according to the relative prevalence of the isotopes for each element (according to US NIST figures).

In addition to these two, mass regimes may also look like `MONO(N15@90%)`, which denotes a monoisotopic assignment, except that nitrogen is assigned the mass of N15 instead of the standard N14. The `90%` conceptually indicates that 90% of the nitrogen in the sample is N15, but this prevalence information is not currently used in the MONO case.

Similarly, `AVG(N15@95.5%)` would indicate that average masses are to be used, except that nitrogen is assigned the mass

```
avgmass(N) + 0.955 * (mass(N15) - mass(N14))
```

Mass regimes are specified as pairs, where the first member of the pair is used for parent (precursor) calculations, and the second member is used for fragment calculations. For example, `AVG/MONO` will use average masses to calculate the parent mass and monoisotopic masses to do the (virtual) fragment mass calculations. For convenience, if only one mass regime is specified, it is used as both the parent and fragment regimes.

The mass regimes to be searched are specified as a semicolon-separated list. For example,

```
AVG/MONO;AVG/MONO(N15@90%)
```

would specify two mass regime pairs to search with.

The default for this parameter is `MONO`.

pervasive_mods

Pervasive modifications (also known as fixed or static modifications) are changes in the mass of a residue that are assumed to always be present, for the purpose of the search. For example

```
+C2H3ON!@C carboxyamidomethylation
```

is the preferred form of the typical C+57 fixed modification specifier. Instead of a symbolic formula (`+C2H3ON`), a specific numeric value such as +57 may be specified. In this case, the same value is used for all mass regimes.

The `!` indicates that this formula is to be evaluated using the first mass regime and that that value should be used everywhere. This is useful when one has a sample part of which is isotope-enriched, but the modification in question is caused by a later chemical modification that is *not* isotope-enriched.

The label (*e.g.*, `carboxyamidomethylation`) is optional, but must be unique if specified. It is used to mark modifications in the output, which makes them easier for downstream software to track.

Each specifier may specify the modification for only a single residue (`C` in the above example), or `[` or `]` (the peptide N- and C-terminii, respectively). Each residue (or terminus) may have at most one specifier.

The value of this parameter is a comma-separated list of modification specifiers. The default is empty, but usually some value will be needed.

potential_mods

Potential modifications (also known as dynamic modifications) are changes in the mass of a residue that may or may not be present, for the purpose of the search. For example

```
PO3H@STY phosphorylation '*'
```

is a potential modification specifier for the phosphorylation of serine, threonine, or tyrosine. As with pervasive specifiers, the modification delta can be specified numerically, and the modification label can be omitted.

The modification marker (`'*'` in this case) is optional; if present, it is used to mark occuring modifications in peptides in the output.

The value of this parameter is a boolean expression composed of conjuctions and disjunctions of potential modification specifiers. So, for example,

```
potential_mods = (PO3H@STY phosphorylation '*';
                  C2H2O@KST acetylation '^';
                  CH2@AKST methylation '#'),
                 O@M oxidation '@'
```

means that each peptide is searched for

- phosphorylation and oxidation, or
- acetylation and oxidation, or
- methylation and oxidation.

This might find for example, a peptide with two phosphorylations and one oxidation, or a peptide with just one acetylation. It will not find a peptide with a phosphorylation and a methylation.

Syntactically, the `';'` is disjunction (read "or") and the `','` is conjuction (read "and"). Parentheses may be used, as above, for grouping, and can be nested without limit.

The purpose of these expressions is to limit search to likely combinations of potential modifications, as search for arbitrary combinations may be prohibitively costly (in terms of running time).

Internally the expressions are reduced to disjunctive normal form and redundancy is removed. Roughly, this means that the search times for two equivalent potential modification expressions will be the same, even though they are expressed differently.

The default is empty--no potential modifications.

potential_mod_limit

This parameter is non-negative integer that specifies the maximum number of potential modifications (*i.e.*, modified residues) sought for a peptide. (Terminal modifications, including PCA modifications, are not counted in this limit.)

In general, run time is an exponential function of this search depth, so large values are likely to be prohibitive.

The default is `2`.

enable_pca_mods

This parameter determines whether PCA modifications are searched for, and must be `yes` (the default) or `no`.

This feature is borrowed from the X!Tandem implementation. Roughly, a PCA is a special kind of N-terminal modification.

Enabling this option will typically increase search time by around 10% and increase the number of valid matches found by 5% or so.

charge_limit

This parameter is a positive integer specifying the largest spectrum precursor charge. This should be set to the largest precursor charge expected for the input spectra, and is used to scale the `parent_mz_tolerance` parameter.

The default value is `3`.

min_peptide_length

This parameter is a positive integer specifying the minimum peptide length to search. Peptides shorter than this will not be searched.

The default value is `5`.

cleavage_motif

This parameter is a string and follows the X!Tandem syntax for specifying cleavage sites. In particular `[X]|[X]` indicates non-specific digestion, and `[KR]|{P}` indicates tryptic digestion (cleavage following lysine or arginine, except when followed by proline).

The default value is `[X]|[X]`.

maximum_missed_cleavage_sites

This parameter is a non-negative integer specifying the maximum number of missed cleavage sites in a candidate peptide. (If non-specific cleavage is being used, the value of this parameter is not used.)

The default value is `1000000000`.

min_parent_spectrum_mass

This parameter is a non-negative floating point value specifying the minimum parent mass for spectra, measured in Daltons. Spectra having a smaller parent mass are skipped.

The default value is `0`.

max_parent_spectrum_mass

This parameter is a non-negative floating point value specifying the maximum parent mass for spectra, measured in Daltons. Spectra having a larger parent mass are skipped.

The default value is `10000`.

TIC_cutoff_proportion

This parameter is a floating point value between zero and one, inclusive. It's used to filter spectra before search. The spectrum peaks are sorted by intensity, and then all but the N most intense peaks are discarded, where N is the smallest value for which the summed intensity of the remaining peaks is at least the specified proportion of the total ion current (TIC).

The default value is `0.98`.

parent_mz_tolerance

This parameter is a non-negative floating point value. It's used to select candidate spectra that might match a peptide--candidates within the specified tolerance are considered possible matches and scored.

The unit of this parameter is m/z--the actual tolerance in Daltons will depend on the charge of the spectrum in question. If the value of this parameter is `1.25`, the mass tolerance will be `1.25 Da` if the charge is +1. If the charge is +3, the mass tolerance is `3.75 Da`, meaning that the difference between the masses of the actual and predicted spectrum precursors must be less than `3.75 Da`.

Search run time will be roughly proportional to this value (at least across the range of interest), so it's convenient to choose the smallest value that will work. Sufficiently small values will start excluding true matches, so a choice that's too small will be counterproductive. (Excessively large values may also reduce the number of valid matches found.)

The default value is `1.25`.

fragment_mass_tolerance

This parameter is a non-negative floating point value, measured in Daltons. It determines how close together a real and a predicted fragment peak have to be to be counted as overlapping (or matching).

The default value is `0.5`.

intensity_class_count

This parameter and the next control how spectrum intensities are classified. During classification, spectrum peaks are ordered according to decreasing intensity. The most intense peaks are placed in the first class, next most intense into the second class, and so on. This classification scheme is drawn from MyriMatch's scoring algorithm--see the paper for more details.

This parameter is a positive integer, and determines how many classes the classifier uses.

The default value is `3`.

intensity_class_ratio

This parameter is a positive floating point value. It determines the relative size of the succeeding intensity classification classes. (See `intensity_class_count` above.)

So, for example, if the parameter value is `2.0`, the second class will be twice as large as the first, the third twice as large as the second, and so on.

The default value is `2.0`. (This parameter should be greater than one in normal usage.)

best_result_count

This parameter is a positive integer and describes the number of candidate matches output for each spectrum.

The default value is `5`.

mass_regime_debug_delta

This parameter is only for debugging use. The value is a float and will be added to the mass of every residue, across all mass regimes. The purpose is to support certain potential modification test scenarios.

The default value is `0`, which gives normal behavior.

Detailed Command Descriptions

Usage: `greylag-rally [options] <configuration-file> <ms2-file>...`

[to be completed]