greylag: software for tandem mass spectrum peptide identification

Greylag is a suite of programs for MS/MS peptide identification by sequence database search. It solves the same basic problem that programs such as SEQUEST, Mascot, X!Tandem, OMMSA, and MyriMatch do.

Greylag is Free Software, distributed under the GNU General Public License.

Caution!

Note that greylag is currently in a "beta" state. You're welcome to try it out, but be aware that there may still be some bugs and sharp edges.

Features

- **No arbitrary limits:** Greylag can search any number of modifications simultaneously, and can simultaneously search multiple potential modifications on the same residue (*e.g.*, K+14, K+16, K+28).

- **User-friendly notation:** The search configuration file has a simple, easy-to-read format. Here is a typical configuration file for LCQ spectra:

  ```
  [greylag]

  databases = /data1/proteomics/db/Hs_2006-03-03_wSHUFFLED.fasta

  mass_regimes = AVG/MONO

  pervasive_mods = +C2H3ON!@C carboxyamidomethylation

  parent_mz_tolerance = 1.25
  ```

  Pervasive or potential modifications may be specified symbolically, as above, or numerically. The symbolic form is both easier to read and typically more accurate, as the mass deltas are calculated using the best available atomic mass measurements from US NIST.

  Similarly, isotope prevalence can be specified symbolically, *e.g.*, `MONO(N15@90%)`.

- **Advanced search:** Greylag can simultaneously search alternate potential modification sets. So, for example, one can search for oxidation of {M} together with either methylation of {K,S} or acetylation of {K,S,T}, without searching for peptides having all three modifications:

  ```
  potential_mods = O@M oxidation '@',
                  (C2H2O@KST acetylation '^';
                   CH2@KS methylation '#')
  ```

Greylag can search multiple mass regimes simultaneously. (A *mass regime* is an assignment of atomic masses, which in turn determines residue masses. Common regimes would be "average", "monoisotopic", and "monoisotopic with 90% N15".) Symbolic modifications masses are calculated according to the regime being searched, which is important if the modification includes an isotope, for example.

Different mass regimes may be specified for parent and fragment mass calculations. For example, `AVG/MONO` means that average masses are used for parent mass calculations and monoisotopic masses are used for fragment mass calculations.

Greylag also automatically searches for PCA (pyrrolidone carboxyl acid, also known as pyroglutamic acid) potential modifications, using the method of X!Tandem.

- **Concise, high-level implementation:** Greylag is implemented using a high-level language for most code, with just performance-critical parts implemented in a lower-level language. The main search programs, `greylag-rally` and `greylag-chase` comprise less than 1500 lines of Python and 700 lines of C++ (as measured by David A. Wheeler's sloccount). This is less than a quarter of the size of other search programs for which source code is available, even though greylag has quite a number of additional features.

  For a typical large nonspecific, non-PTM search of 236,000 spectra searched against a sequence database of 6,000,000 residues, 95% of CPU time is spent in C++ code. For a long-running PTM search, this figure would be greater than 99%. Thus, the performance penalty for this mixed-language approach is negligible.

  Greylag's relatively small size, together with the fact that only a small part is written in a low-level language, makes it easier to understand, debug, experiment with, and (we hope) learn from.

- **Integral parallel capability:** Greylag's parallel implementation works the same on single multiprocessor machines and clusters. There are no external dependencies (*e.g.*, MPI), and the design should be flexible enough to fit into a broad range of cluster scheduling architectures.

  The speedup achieved depends on the size of the search--larger searches will see linear speedup. For smaller searches--tryptic searches or searches of small spectrum sets--the speedup will be smaller due to the greater fraction of overhead.

  The implementation is designed to be fault-tolerant. The "master" program (`greylag-rally`) will make use of whatever running "slave" (`greylag-chase`) instances are available, and will continue even if they appear, disappear, crash, etc., during a processing run.

- **Incremental search:** If a search for one set of modifications is performed, and then later a search for a second set of modification is done, the two can easily be combined, producing the same result as if the entire search had been done all at once. (This feature is not yet implemented.)

- **Emphasis on correctness:** Greylag carefully checks the parameters specified in the search configuration file and on the command line, and also the format of the spectrum files.

  More generally, greylag is designed to never crash or silently give incorrect results with respect to its input. Depending on its input, it may run "forever" or exceed available memory, but if it completes the results will be correct. (This is the goal--bug reports are welcome.)

  Greylag's source code has been carefully checked for errors, and its test suite includes dozens of unit tests.

  Greylag is implemented with numerical stability in mind, and all calculations performed using double-precision floating point. This helps to avoid quantization problems and other errors.

- **Compatibility:** Greylag currently supports search of spectrum files in MS2 format and generates output in SQT format. Translation to and from other formats could easily be added.

- **Reasonable Performance:** For a number of reasons it is difficult to directly compare the performance of different search programs, but greylag's performance appears to be comparable to that of SEQUEST and MyriMatch for equivalent-as-possible searches.

More information about greylag's design and implementation, as well as background information on tandem mass spectrum search, is available in the theory of operation (pdf) document.

Getting greylag

You can download the latest release from the greylag project page (https://sourceforge.net/projects/greylag) at SourceForge--see the Release Notes (pdf). This site also hosts the mailing lists and bug tracker.

Greylag uses git for distributed source code management. You can browse the project repository from the public mirror at http://repo.or.cz/w/greylag.git. You can also pull your own copy with

```
$ git clone git://repo.or.cz/greylag.git
```

Installing greylag

See the Installation Guide (pdf) for information about the compiling and installing the various greylag programs.

Using greylag

See the User Guide (pdf) for information about the configuration and usage of the various greylag programs.

Feedback

Comments, suggestions, and bug reports are welcome! If you try it out, please drop us a line and tell us how it worked (or didn't work) for you.

Credits and Acknowledgements

Greylag would probably not have been possible without access to the source code of MyriMatch, X!Tandem, and OMSSA, which their authors have made available under various Open Source licenses.

Greylag's main spectrum filtering and scoring algorithms are currently based closely on those of MyriMatch. The PCA modification method is based on that of X!Tandem.

The idea for the `greylag-validate --bootstrap` flag is from Roger Moore (City of Hope).

The initial greylag implementation was written by Mike Coleman, supported by the Stowers Institute for Medical Research.