Processor of Unknown Genome Sets for Laboratory Identification (of Proteins) (PUGSLI-P):

A simple approach to identifying candidate proteins from mass spectroscopy data using FindPept and BioPerl scripts

Scott Bartels <sbartels@uthsc.edu>

2/13/2010

0 Introduction

0.1 Purpose of this project

pugslip-FindPept is a series of scripts to aid in peptide identification.

It takes a possible amino acid sequence and a mass spec file and submits these to an online service (FindPept, http://www.expasy.ch/tools/findpept.html).

It takes a brute force approach to identifying sequence coverage for the particular mass spec data.

These scripts were used to help identify proteins isolated from ETEC H10407. They may prove useful more broadly in aiding the analysis of other genomes for which complete annotation is not yet available.

0.2 Purpose of this documentation

This documentation intends to demonstrate

- How to install this software and its prerequisites.
- How to use this software to perform a first-order analysis of mass spec data
- General code flow
- Potential optimizations and avenues for further development.

The convention in this document is to format command lines as follows:

```
# perl -V
```

Please note that each line of multi-line commands must generally end with a "\".

Pseudocode is used to explain code flow.

0.3 Basic overview of mass spec identification

The protein is separated based on molecular weight and isoelectric point. The protein is then subjected to mass spectroscopy (MALDI-TOF). The data obtained from the mass spec contains the peaks corresponding (roughly) to the molecular weight of digested fragments of this protein.

All of this information (MW, pI, and mass spec data) can be used to identify the protein.

Unfortunately, currently available search strategies do not readily permit identification of proteins that are not already annotated in online databases (e.g. Genbank, Swiss-Prot). However, in the case of H10407, an unannotated sequence is available. Open reading frames have been predicted given this sequence. Each of these ORFs potential encodes the candidate protein. These candidate proteins are often poorly characterized, often getting "predicated " or "hypothetical" matches as a result of a protein BLAST.

To overcome the minor difficulty of lacking a fully annotated sequence, these scripts use a brute force approach: the entire search space of ORFs and their protein products is considered. Given there may be several thousand ORFs, this can be unwieldy in terms of the amount of time to get back meaningful results. A simple optimization is to take the MW and pI of the protein from experimental observations, and use these data to filter only those ORFs that potentially encode a protein that is in the correct range of MW and/or pI.

FindPept takes the AA sequence of the candidate protein and digests it *in silico* and determines if any of the digested fragments correspond to those in the mass spec file. This "bottom-up" approach is characteristic of peptide mass fingerprinting techniques.

The scripts provided by this project automate queries to FindPept to search for candidate proteins. First, the genomic data, which has been broken down into ORFs, is converted into amino acid sequences. Second, these protein sequences are submitted along with the mass spec data to FindPept, which returns a simple HTML file containing an analysis of the data. This analysis consists of a series of tables showing which amino acid sequences "match" each mass spec peak. These will be generically referred to in this document as "matches." Not all matches support the identification of the protein. Some result from the cleavage of contaminants (including the digesting protease). Fortunately, FindPept sorts out these possibilities.

The most important of the data tables shows the masses matching the (digested) protein of interest. Calculation of how well these fragments "cover" the protein offers a simple metric: sequence coverage or "identity." Such identity may be evaluated in multiple contexts:

- Experimental observations
- The degree of digestion (which is suggested by the number of missed cleavages)
- The number of non-matching masses/contaminant masses

In this context, a "hit" would refer to a sequence which seems to correspond to the mass spec data based on the calculated sequence identity. FindPept does not perform these calculations, which are helpful to compare one protein to another. In the most basic sense, these scripts automate the simple calculation of identity to allow comparisons of multiple candidate proteins.

The technique used in this project differs from the MOWSE algorithm used by MASCOT (a well-known commercial program also employing a MOlecular WEight SEarch technique) in that it does not attempt to perform a statistical analysis of the search space. As a result, a mass peak that occurs rarely (and which might be a strong indicator as to a real "hit") is treated the same as a peak that occurs frequently (i.e. is present in a lot of proteins, therefore not an ideal fingerprint in and of itself.)

Despite the simplified analytical model utilized by these scripts, the results obtained may be useful in generating hypotheses as to which coding sequence may correspond to the observed spot on the gel.

1 Prerequisites

1.1 Operating system

This project was developed and tested using the GNU/Linux operating system. The distribution used was Fedora 12 (http://fedoraproject.org), which influenced packaging of the scripts and modules (via RPM). Other systems not utilizing RPM packages will require manual installation of the scripts.

1.2 Software

The scripts are written in Perl (version 5.10.0). BioPerl (http://www.bioperl.org,, version 1.6.1) is the primary external library used for these scripts.

Shell access is needed to run the command line scripts. The shell used in these examples was GNU bash (version 4.0.35). Other similar shells should perform similarly. Shell scripts, which may be used to automate processing, typically depend on /bin/sh.

A web browser supporting JavaScript is strongly recommended to view output. Text-mode browsers are acceptable but offer fewer data visualization and search options.

A web server is optional, but allows remote viewing of output.

N.B.: If installed from rpm, /usr/local/bin should be in the execution path.

1.2.1 Required external libraries

These libraries can be installed using RPM, CPAN, or other packaging software available on your system. No modifications were made to the libraries.

Library	Description	RPM
Getopt::Long	To allow processing of command line switches	Included with perl
Bio::Seq	From BioPerl, represents NT/AA sequence	perl-bioperl
Bio::SeqIO	From BioPerl, allows reading from fasta/EMBL files	perl-bioperl
Bio::Tools::SeqStats	From BioPerl, holds statistics for a sequence	perl-bioperl
Bio::Tools::pICalculator	From BioPerl, calculates pI	perl-bioperl
Bio::SearchIO	From BioPerl, driver for parsing sequence database searches	perl-bioperl
Bio::Tools::Run::RemoteBlast	From BioPerl, runs blast search on NCBI via http	perl-bioperl-run
Set::Scalar	Used to record matched AA in a peptide	perl-Set-Scalar
File::stat	Used to determine file sizes, and whether files exist	Included with perl

HTML::TableExtract	Used to scrape data from FindPept results	perl-HTML-TableExtract
Pod::Usage	For documentation in libraries	Included with perl
HTTP::Request::Common	Builds POST request to submit to FindPept	perl-libwww-perl
LWP::UserAgent	Allows script to act as a web browser so data can be submitted to website	perl-libwww-perl

1.2.2 Project-specific libraries

These libraries are provided as both RPMs and tarfiles.

Library	Description	RPM
pugslip::FindPept	Submits an AA sequence and mass spec data to FindPept	perl-pugslip-FindPept
pugslip::FindPeptTools	Utility functions for the custom scripts	perl-pugslip-FindPept

1.2.3 Scripts

Multiple scripts are packaged in the pugslip-scripts RPMs or in the tarfile. All are prefixed with "process-" and are numbered in the general order in which they should be executed.

1.2.4 Web-specific libraries

These libraries are recommended to improve functionality of viewing the output.

Library	Description	Source
jQuery	Used to provide filtering/sorting capabilities to data	http://jquery.com/
DataTables	Table plugin for jQuery	http://www.datatables.net/

1.3 Hardware

1.3.1 Networking

Network access is required for multiple reasons:

1. To download sequences
2. To BLAST sequences
3. To submit to FindPept

1.3.2 Storage

The brute force approach utilized here requires a great deal of temporary space (1MB per query, with possibly several hundred candidate queries per mass spec file). Once all processing is completed, however, these text data files are extremely compressible. (More on this below...)

1.4 Required input

The following information should be available prior to proceeding:

1. The **sequence(s)** for the organism.

   For this project, H10407 and its plasmid sequences were obtained from http://www.sanger.ac.uk/Projects/E_coli_H10407/.

   Any major nucleotide sequence format (EMBL, GenBank, fasta, etc.) should theoretically be able to be recognized (and possibly converted), though only EMBL files were tested in this project.

2. **Experimentally derived data**

   To process a particular spot on a gel:

   a) *Required:* Mass spec data. Here, the data was in pkl format.

   b) *Required:* An estimated molecular weight.

   c) *Optional:* The observed isoelectric point.

   (b) and (c) help speed up processing by restricting the search space.

2 Installation

2.1 Install libraries and scripts

2.1.1 Fedora

1. Change to root user (or use sudo).
2. Install pugslip-FindPept & pugslip-scripts rpms using yum or rpm.

   /usr/share/doc/pugslip-scripts will contain files to create the base web directory

   a) A few test files (which are demonstrated below)

   b) Readme/license files

Local installation into a user directory is also possible, but will require

- Updating the execution path (or explicitly specifying it)
- Updating the Perl @INC path (for the pugslip::* modules).

2.1.2 Other systems

1. Install prerequisites.

2. Unzip pugslip-FindPept.tar.gz. Install files in the lib/ subdirectory into the Perl @INC path.
3. Unzip pugslip-scripts.tar.gz into path

2.2 Setup web directory

1. Create a directory to store your files in.

   Copy base_web_files.tar.gz and (optionally) test-data.tar.gz into this directory

2. Uncompress base_web_files.tar.gz in desired directory. If RPM is used, this file is in /usr/share/doc/pugslip-scripts/.

   ```
   # tar -xvf base_web_files.tar.gz
   ```

3. Run the shell script to set up the appropriate subdirectories and scripts.

   ```
   # sh scripts/create_pugslip_directory.sh
   ```

   The directory will have the following structure:

blast/	blast file storage
css/	simple css files to format web output
dataTables/	symbolic link to scripts/dataTables-* directory
fasta/	fasta file storage
jquery.js	symbolic link to scripts/jquery-*.js file
pkl/	pkl file storage
scripts/	jQuery script storage and helper shell script
sequence/	sequence files

4. [Optional] Uncompress test data (several plasmid sequences and a pkl file). If RPM is used, this file is in /usr/share/doc/pugslip-scripts/. **Please note that this is a somewhat artificial setup, since the demonstration below will use a molecular weight range that does not necessarily correspond to this pkl file (to reduce the search space for speed concerns). Note this highlights the need to correlate experimental data with any observed degree of identity.**

   ```
   # tar -xvf test-data.tar.gz
   ```

5. The directory is set up.

   Mass spec data should be placed in the pkl/ directory. A recommended naming scheme is to save these files with a simple name (say, 34.pkl) rather than a longer one (2009-07-06-34.pkl).

   Sequence files should be saved in sequence/.

3 Running the scripts

3.1 Extract features to fasta format: process-1-extract_feature_seq_to_fasta.pl

```
# process-1-extract_feature_seq_to_fasta.pl \
     -i sequence/p52_gene_pred.embl -i sequence/p58_gene_pred.embl \
     -i sequence/p666_gene_pred.embl -i sequence/p948_gene_pred.embl
```

Compare the output of this script to the BioPerl script (http://code.open-bio.org/svnweb/index.cgi/bioperl/view/bioperl-live/trunk/scripts/seq/extract_feature_seq.PLS).

The difference here is that the predicted gene sequences have a different tag (systematic_id) which requires a modification to this original code.

The goal of this script is to create a file with all the desired features (coding sequences) converted to fasta format.

This file is saved by default in fasta/out.fasta

3.2 Cache useful information about coding sequences: process-2-calculate-mw-pi.pl

```
# process-2-calculate-mw-pi.pl
```

This script does the following:

for each coding sequence in fasta/out.fasta{

get translation

calculate the MW

predict the pI

write the sequence ID, MW, and PI to cds.info in CSV format

}

This file will be used to define the search space for FindPept submissions using a particular pkl file.

3.3 Create a list of candidate proteins: process-3-create-candidate-list.pl

```
# process-3-create-candidate-list.pl --pkl 34.pkl --mw 85000
```

This script accomplishes several things:

1. Creates a directory based on the pkl file name (in this case, 34/).
2. Goes through cds.info and matches desired properties to those specified on command line
3. Creates an error file listing results of these comparisons.

4. Creates a candidate protein list (34/candidate.list) with proteins matching the specified requirements. The first line of this file specifies the mwhi, mwlo, pihi, and pilo for the search.

In this example, there is one match in 34/candidate.list.

In order to restrict the search space based on observed molecular weight and pI:

- To search for proteins around 10kd, specify "--mw 10000". Default is 85000 (and could be omitted from the above command line if this is desired).
- To search for proteins from 10kd +/- 2kd, specify "--mw 10000 --tolerance 20". Default tolerance is 10%.
- To search for proteins with pI from 10 to 11, specify "--pilo 10 --pihi 11". Default range is 1-14.

These can of course be combined.

3.4 Cache NT and AA fasta files for all cds.info: process-3b-create-fastap.pl

```
# process-3b-create-fastap.pl
```

This script performs the following:

```
for each coding sequence in fasta/out.fasta{
        write that sequence to (id).nt.fasta.txt
        write the translation to (id).fasta.txt
        write errors to fasta/error-3b-fastap
}
```

N.B.: This script can be run after (3.1), which creates fasta/out.fasta. *It only needs to be run once*—it generates all the fasta files (for nucleotides & peptides). This is a fairly rapid procedure.

These files are used for the web output (hence the addition of the "txt" extension so the web browser will not choke on the file format).

3.5 Get blastp information for candidate coding sequences: process-3c-create-blastp.pl

```
# process-3c-create-blastp.pl --pkl 34.pkl
```

This will perform a blastp on all the sequences in 34/candidate.list.

The blast search is saved by default in (id).blastp.txt, and will be used in the web output.

Speed optimizations:

1. Only blast the proteins that are candidates.
2. Any particular coding sequence will only get blasted once. For example, if 34/candidate.list and 35/candidate.list both contain a sequence id "ETEC0001", only one blastp will need to be

performed. If for some reason this file is corrupted, it must be deleted from the blast/ directory to be regenerated.

3.6 Submit AA sequence and mass spec data to FindPept: process-4-submit-candidate-proteins.pl

```
# process-4-submit-candidate-proteins.pl --pkl 34.pkl
```

The basic process:

for each protein (id) in the appropriate candidate list for pkl file (pkl){

1. bundle the AA sequence along with the mass spec data file and post this to the URL for FindPept queries
2. save results in (pkl)/(id)-(pkl).html

}

Errors may result if some older libwww-perl libraries are used. In this case, version 5.833 was used.

Speed optimizations

1. If "--threshold #" is specified, then the query will only be made if the output file is smaller than the specified # (in bytes). If FindPept fails on a query and returns a small file, this will help automatically detect that, unless the files are small to begin with (few mass spec peaks/small proteins). Default is 200000 bytes.
2. If "--sleep #" is specified, then will sleep # seconds between queries. Default is 5. This does a couple of things:
   a) Reduces likelihood of timeouts from FindPept (which may result in small/bogus output files)
   b) Shows good netiquette

3.7 Optimize the FindPept results: process-5-cleanup-expasy-output.pl

```
# process-5-cleanup-expasy-output.pl --pkl 34.pkl
```

This removes some of the verbosity from the FindPept results. (Usually a few hundred lines).

More importantly, it modifies the html file to include a "base href," which allows the local html file to pick up the stylesheets from expasy.ch

The stripped file is saved to (id)-(pkl)-stripped.html

Future optimizations: more aggressive pruning.

3.8 Look for matches between mass spec data and peptide sequences: process-6a-cache-matches.pl

```
# process-6a-cache-matches.pl --pkl 34.pkl
```

This is admittedly the least elegant and most involved of all the scripts, as it has to scrape the tables in the FindPept file.

This file consists of a series of tables with contaminant masses, protease masses, (possibly) specific masses, nonspecific masses, and (possibly) unmatched masses.

Contaminant masses are fragments that match mass spec data but would be predicted to come from a common contaminant (e.g. human keratins).

Protease masses are fragments that match mass spec data but would be predicted to come from autodigestion by a protease. (N.B.: The module has been hard coded to assume that the protease is trypsin. This can be modified within the FindPept module...)

Specific masses refer to those fragments which would be expected from specific cleavage of the protein in question from the protease.

Nonspecific masses are AA sequences that match a mass spec peak, but result from nonspecific cleavage from the protease.

Unmatched masses are those mass spec peaks which do not match a fragment from a contaminant, the protease, or the protein in question.

Scraping is a time-consuming process, so the prime optimization is to do it all once and cache which AA are "covered" (matched) by a particular mass spec peak.

It is helpful to review the output from this script. Refer to the original FindPept output (available in web output) to see where this info comes from.

1. *Caching all masses for 34/p666.0160-34-stripped.html*

   In this step, get a count of all masses in the mass spec file.

2. *Caching contaminant masses for 34/p666.0160-34-stripped.html*

   In this step, the contaminant masses are counted.

3. *Caching trypsin masses for 34/p666.0160-34-stripped.html.*

   In this step, the protease masses are counted.

4. *Getting match header for 34/p666.0160-34-stripped.html*
   2905,910,26,146,93,12964,11705,358

   In this step, a variety of stats are summarized. It is helpful to understand these statistics, as they form the basis for the simple analysis this script performs.

The number of masses in the pkl file:	2905
The number of masses matched by contaminants:	910
The number of masses matched by the protease:	26

The number of masses matched by the protein specifically: This is the number of masses that match specific digestion of the protein in question. Theoretically this may exceed the number of masses in the pkl file (for example, if the protein contains a number of repetitive motifs, which will cause one mass to "cover" multiple sections of the protein).	146
The number of specific masses which matched "perfectly": This should be less than or equal to the previous number. It reflects matches where there are no missed cleavages. This may imply a completely digested protein.	93
The number of nonspecific matches: This is included as a sanity check. • A mass spec peak may match multiple permutations of AA. For example, RGD would match the same peak as RDG, etc. • Note that this number exceeds the number of masses in the mass spec data. Because fingerprinting depends on measurement of small masses, a small tolerance factor has to be specified to FindPept. As a result, a mass may "cover" two or more AA sequences if these would be predicted to have the same mass. Additionally, many of the nonspecific matches are actually 1 AA in length, which would predictably "cover" any occurrence of that AA in the protein.	12964
The number of nonspecific "perfect" matches: Should be <= nonspecific matches	11705
The number of unmatched masses: This may be 0 if all masses are matched. It should never be negative! A high number of unmatched masses may argue against a "hit." For example, say there are two copies of a gene, one of which is a slightly truncated version of the other (but still within the mass tolerance specified in (3.3)). If the mass spec data corresponds to the full-length version, the truncated version will have a fairly good "identity" but a lot of unmatched masses. Alternately, if the mass spec data corresponds to the truncated version, the full-length copy might be expected to have few unmatched masses but also a lower sequence identity.	358

5. *p666.0160: caching match file 34/p666.0160-34-specific.match*
   p666.0160: caching match file 34/p666.0160-34-nonspecific.match

   The positions of the specific and nonspecific matches are saved in (pkl)/(id)-(pkl)-(specific|nonspecific).match.

   The "match header" printed out in output (4) above is used as the first line in the match file.

Speed optimizations:

Only parse a table if it has not been parsed already. This means that if for some reason the data needs to be rerun, you need to remove the match files (for that particular protein/mass spec combination). For example:

```
# rm 34/p666.0160*.match
```

Potential improvements/areas to watch:

1. Speedier scraping techniques.
2. Not all matches are created equal. This algorithm does not apply statistical modeling, nor does it tally how frequently a mass is "used" to cover a protein.
   For example, if the AA sequence "GGGG" only occurs once in the entire genome, and if a mass peak corresponding to the MW of "GGGG" is found, it makes it more likely that that AA sequence corresponds to that mass spec file, regardless of the other noise. Similarly, if the sequence "RRRR" occurs in every protein, its presence in the mass spec data means a lot less. See the discussion in the introduction above.
   One clue to this phenomenon would be to cache how often a particular mass peak "covers" a peptide in the protein. Currently, only the reverse—which particular peptide fragments are matched by any mass peak—is taken into account in caching. A high "peak usage count" might imply that peak is less useful in identifying a protein by itself. Alternatively, it may indicate a highly repetitive motif within the protein.
   This is overkill for this project, as other tools can accomplish a more sophisticated analysis if this level of detail is desired.
3. If proteins interact (dimers/binding), this may result in a number of findings.
   a) It may increase the number of non-matching masses (some of the peptide fragments are from the contaminating/interacting protein, which cannot be explained as a protease fragment or a keratin/other common contaminant). This finding would also be expected any time an impure sample is submitted to mass spec.
   b) More likely, the change in observed MW on the gel will result in that protein's exclusion from screening based on MW (see 3.3).
4. If no specific masses are noted, there is the possibility that the nonspecific mass table will be interpreted incorrectly as representing specific matches. A clue that this is happening is 100% sequence identity (which, while the goal with fingerprinting, does not typically pan out as such). If the FindPept output format changes, it may result in similar "off-by-one" table errors occurring and necessitate modification of the script; this is an unavoidable result of scraping techniques.

3.9 Create a "map file": process-6b-create-protein-map.pl

```
# process-6b-cache-matches.pl --pkl 34.pkl
```

For each of the candidate matches, create an "executive summary" of how well a particular AA sequence compares to the mass spec file.

Reviewing the "match" files shows which AA are "covered" by a particular peak. For example, the line "318-321,0" means that amino acids 318-321 matched some mass spec peak, and that there were no missed cleavages. Which peak matched is not recorded; see potential improvments #2 from previous step. Since the length of the protein is known, sequence identity can by simply tallying which amino acids are "covered" by matches:

$$\frac{number\ of\ \text{covered}\ amino\ acids}{amino\ acid\ length} = sequence\ identity$$

N.B. The non-specific matches should almost always cover the entire protein... It does not mean much, but it can be a way to pick out problems with the screen scraping.

These map files must be removed if this script is rerun on updated data, as a speed optimization.

```
# rm 34/*-map.html
```

3.10 Create an index file: process-7-create-index.pl

```
# process-7-create-index.pl --pkl 34.pkl
```

Create an executive summary of all the comparisons of AA/mass spec data. One measure which may be of interest is the a simple metric calculated as

$$\frac{number\ of\ amino\ acids\ without\ missed\ cleavages}{number\ of\ amino\ acids\ covered\ by\ any\ cleavages}$$

Mathematically, the number of non-missed cleavages of the protein of interest must be less than or equal to the sum of "non-missed" and "missed" cleavages tallied from the table. It is a simple way to clue one in that the either digestion has been suboptimal or that the degree of identity may be misleading.

3.11 To automate multiple runs: process-0-create-scripts.pl

```
# process-1-extract_feature_seq_to_fasta.pl \
     -i sequence/p52_gene_pred.embl -i sequence/p58_gene_pred.embl \
     -i sequence/p666_gene_pred.embl -i sequence/p948_gene_pred.embl
# process-2-calculate-mw-pi.pl
# process-3b-create-fastap.pl
# echo "34,100,10
     35,80,10" >> AUTOMATION_FILE
# process-0-create-scripts.pl AUTOMATION_FILE > batch_process.sh
# sh batch_process.sh
```

AUTOMATION_FILE should be in CSV format, with columns containing pkl file number, MW, and MW tolerance as demonstrated above.

This Perl script needs to be modified if pIs are to be specified. It is straightforward but beyond the scope of this document.

4 Web interface

4.1 Local browsing

1. Open a web browser, and navigate to the directory where the files created in (2.2) are located.

2. Navigate to a directory corresponding to a particular pkl file (say, 34/)
3. Open index.html

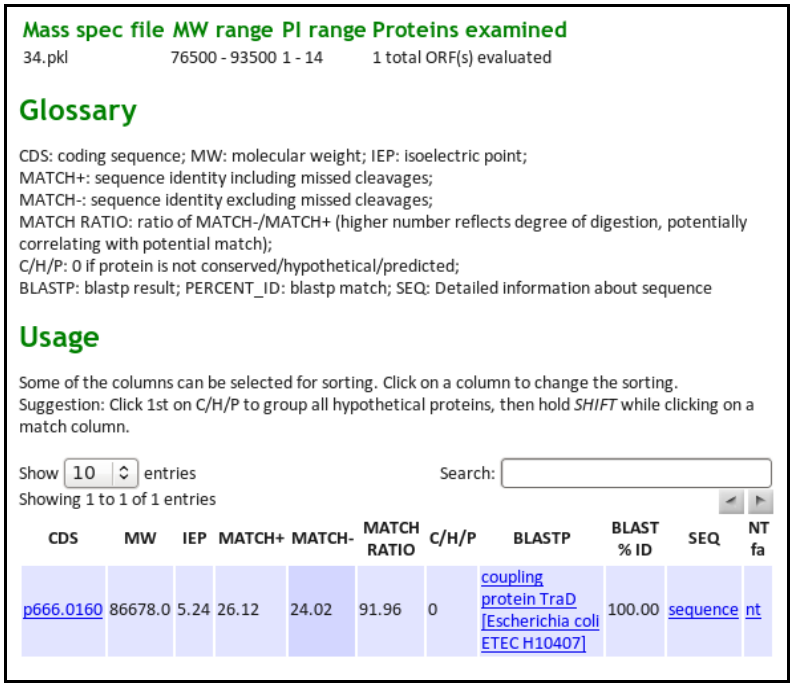
4.2 Web server

1. Migrate the root directory created in (2.2) to your web server
2. Navigate to the appropriate URL

4.3 What is available via the web interface

1. The original FindPept output
2. The results of the blastp search, as well as a link to open a web blastp via NCBI
3. The NT and AA sequences in fasta format for manual blasts.
4. Summaries of the sequence identity (i.e. AA matched by mass peaks from mass spec data) along with a "map" to visualize where these AA lie in the protein. (See figures below).
5. Ability to sort matches, search for a specific match, etc.

Figure 1. Representative output from step (3.10)



Mass spec file MW range PI range Proteins examined
34.pkl 76500 - 93500 1 - 14 1 total ORF(s) evaluated

Glossary

CDS: coding sequence; MW: molecular weight; IEP: isoelectric point;
MATCH+: sequence identity including missed cleavages;
MATCH-: sequence identity excluding missed cleavages;
MATCH RATIO: ratio of MATCH-/MATCH+ (higher number reflects degree of digestion, potentially correlating with potential match);
C/H/P: 0 if protein is not conserved/hypothetical/predicted;
BLASTP: blastp result; PERCENT_ID: blastp match; SEQ: Detailed information about sequence

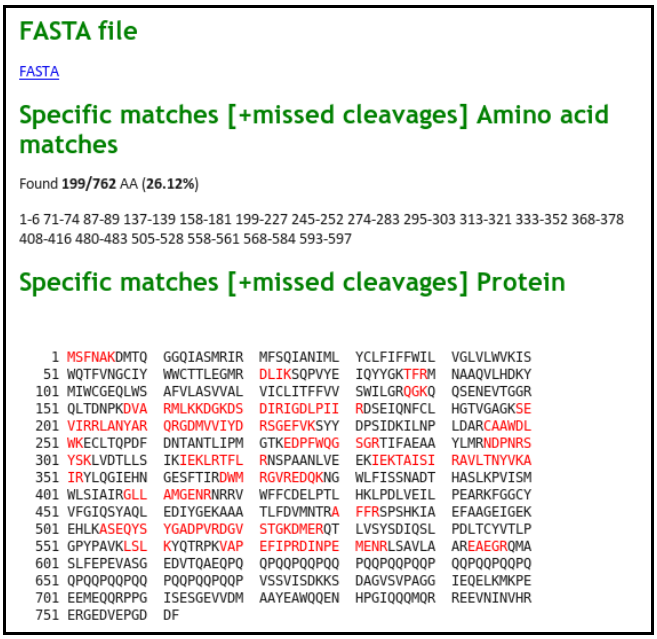
Usage

Some of the columns can be selected for sorting. Click on a column to change the sorting.
Suggestion: Click 1st on C/H/P to group all hypothetical proteins, then hold *SHIFT* while clicking on a match column.

Show 10 entries Search:
Showing 1 to 1 of 1 entries

CDS	MW	IEP	MATCH+	MATCH-	MATCH RATIO	C/H/P	BLASTP	BLAST % ID	SEQ	NT fa
p666.0160	86678.0	5.24	26.12	24.02	91.96	0	coupling protein TraD [Escherichia coli ETEC H10407]	100.00	sequence	nt

Figure 2. Representative output ("map file") from step (3.9)



FASTA file

FASTA

Specific matches [+missed cleavages] Amino acid matches

Found **199/762** AA (**26.12%**)

1-6 71-74 87-89 137-139 158-181 199-227 245-252 274-283 295-303 313-321 333-352 368-378 408-416 480-483 505-528 558-561 568-584 593-597

Specific matches [+missed cleavages] Protein

```
  1 MSFNAKDMTQ  GGQIASMRIR  MFSQIANIML  YCLFIFFWIL  VGLVLWVKIS
 51 WQTFVNGCIY  WWCTTLEGMR  DLIKSQPVYE  IQYYGKTFRM  NAAQVLHDKY
101 MIWCGEQLWS  AFVLASVVAL  VICLITFFVV  SWILGRQGKQ  QSENEVTGGR
151 QLTDNPKDVA  RMLKKDGKDS  DIRIGDLPII  RDSEIQNFCL  HGTVGAGKSE
201 VIRRLANYAR  QRGDMVVIYD  RSGEFVKSYY  DPSIDKILNP  LDARCAAWDL
251 WKECLTQPDF  DNTANTLIPM  GTKEDPFWQG  SGRTIFAEAA  YLMRNDPNRS
301 YSKLVDTLLS  IKIEKLRTFL  RNSPAANLVE  EKIEKTAISI  RAVLTNYVKA
351 IRYLQGIEHN  GESFTIRDWM  RGVREDQKNG  WLFISSNADT  HASLKPVISM
401 WLSIAIRGLL  AMGENRNRRV  WFFCDELPTL  HKLPDLVEIL  PEARKFGGCY
451 VFGIQSYAQL  EDIYGEKAAA  TLFDVMNTRA  FFRSPSHKIA  EFAAGEIGEK
501 EHLKASEQYS  YGADPVRDGV  STGKDMERQT  LVSYSDIQSL  PDLTCYVTLP
551 GPYPAVKLSL  KYQTRPKVAP  EFIPRDINPE  MENRLSAVLA  AREAEGRQMA
601 SLFEPEVASG  EDVTQAEQPQ  QPQQPQQPQQ  PQQPQQPQQP  QQPQQPQQPQ
651 QPQQPQQPQQ  PQQPQQPQQP  VSSVISDKKS  DAGVSVPAGG  IEQELKMKPE
701 EEMEQQRPPG  ISESGEVVDM  AAYEAWQQEN  HPGIQQQMQR  REEVNINVHR
751 ERGEDVEPGD  DF
```

5 Caveats

5.1 Compression

Consider compression of all this data once processing is completed, as suggested above.

For the H10407 project, 14GB of files were created. Compression reduced this to 1.6GB (which can be more easily transferred on a USB drive/DVD).

Two suggested options for compressing:

1. Using standard archiving/compression tools (tar/zip/gzip/bzip2/xz).

   Pro: Good for transporting files; can be opened with widely available tools on various operating systems.

   Con: Requires special configuration to the web server to decompress on-the-fly

2. Using a compressed filesystem (fusecompress or squashfs).

   Pro: Good compression. Once mounted with proper permissions, the compressed nature of the filesystem is transparent to the web server.

   Con: Typically read-only, which is not a problem for fully processed data. Less portable as it requires ability to read a particular compressed filesystem, which may not be supported in other operating systems. Might not be feasible without user permissions necessary to mount filesystems.

5.2 Speed

This script has some speed optimizations, but because every possible protein in the (narrowed) search space is queried along with each mass spec file, these searches take a long time.

5.3 Interpretation of data

These algorithms will generate lists of potential hits based on peptide identity matches. Therefore, the primary utility of these tools is to identify potential candidate proteins, that can be further statistically and experimentally validated using other techniques.

5.4 Legal

The scripts and modules are distributed under the Perl Artistic License 2.0.

Except as noted, all other scripts and modules referred to or distributed here are licensed using various Open Source licenses (e.g. GPL, GPLv2, BSD, or Perl Artistic License). Derived works should refer to the specific license for further information on requirements.

The documentation for this project is licensed under the GNU Free Documentation License 1.3.

Copies of the PAL and GNU FDL are available with this distribution (in the documentation directory).

Except where otherwise noted, all scripts and documentation were written by Scott Bartels.