This document contains the installation instructions for the Proteomics Pipeline developed at University of Liverpool by Jones Group

Subject - Proteomics Pipeline

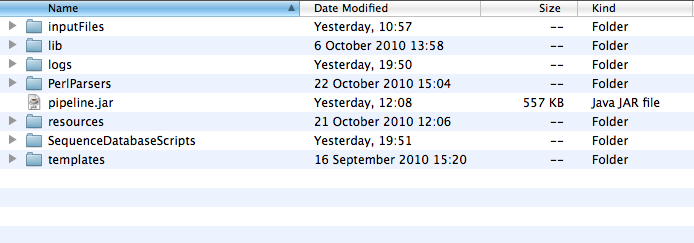
Date – Oct 28, 2010

Author – Ritesh Krishna

Email – [Ritesh.Krishna@liverpool.ac.uk](mailto:Ritesh.Krishna@liverpool.ac.uk)

**Directory structure of the pipeline**

The directory structure of a base distribution should have following directories and files. A screenshot is provided below-



* **inputFiles** – Contains the template for running the search engines. The files inside this folder should be modified for configuring the search criteria.
* **lib** – Contains some dependent libraries for running of the pipeline
* **logs** – Contains log files for logging the events while running the pipeline. The logging for the pipeline has been done using the log4j mechanism, which allows to change the logging behaviour according to the specifications mentioned in the properties files. The log4j properties file is present in the resources folder.
* **PerlParsers**  - Contains perl source code for converting files to mzIdentML format
* **resources** – Contains the information regarding the location of external executables, log property file etc.
* **SequenceDatabaseSripts** – Contains perl scripts for creating a decoy database.
* **Templates** – Contains the templates with placeholders to create Omssa/X!Tandem search engine commands on the fly.
* **Pipeline.jar** – Jar file which will be called to execute the whole pipeline.

**Prerequisites**

The following software/libraries are required for running the pipeline.

* 1. Omssa – The binaries can be downloaded from <http://pubchem.ncbi.nlm.nih.gov/omssa/download.htm>

The pipeline has been tested for

* + - omssa-2.1.9 version on Mac os, and
    - omssa-2.1.7 version on Linux.

The compiled version of omssa-2.1.9 version for Linux is not

available for 32-bit operating systems at the time of writing this

document. It should be noted that Omssa has some kind of bug in

interpreting the description lines in Fasta files, so it’s safer to stick

to 2.1.7 version on Linux. The installation simply requires

unpacking of the tar.gz file in a folder in a preferred location, as for

example, in /opt or /home/username.

* 1. X!Tandem – The binaries can be downloaded from

<ftp://ftp.thegpm.org/projects/tandem/>

The pipeline has been tested for

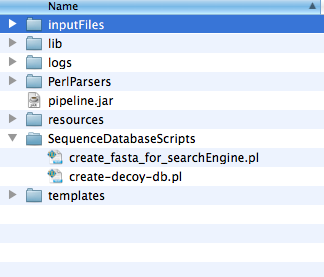
* tandem-10-01-01-4 version on Mac os, and
* tandem-08-02-01-3 version on Linux

The installation simply requires unpacking of the compressed file in a preferred location.

* 1. NCBI Blast libraries – Omssa requires the tool ***formatdb*** (in older distributions)or ***makeblastdb*** (in newer distributions, formatdb is no more available, and has been replaced by makeblastdb). We used makeblastdb on Mac os, and formatdb on Linux.
  2. Perl Modules – Most of the systems may have all the required modules already installed, but on a precautionary measure, I will just mention some of the module which may might be missing. They are –
     + Text::CSV
     + Data::Types
     + XML::Simple
  3. Java Runtime Version 1.6

**Important steps in setting up the pipeline**

* **Creation of the Fasta database** – In order to run the search engines, we need to setup a Fasta database. The database should contain the annotated protein sequences, and the fictional decoy sequences. The merging of decoy sequences with the actual sequences helps us in accessing the validity of the search results. There are various ways to create decoy sequences; we have used random sequence generators for this pipeline. The file to create decoy database is in ***SequenceDatabaseScripts*** folder.



The script create\_fasta\_for\_searchEngine.pl needs to be called at the command line in the following way

*$perl create\_fasta\_for\_searchEngine.pl ~/Ritesh\_Work/Toxo/ToxoDB/TgondiiME49\_ToxoDB-6\_2.fasta 3 ~/Ritesh\_Work/Toxo/ToxoDB/TgondiiME49\_ToxoDB-6\_2\_withdecoy.fasta*

where,

* 1. *~/Ritesh\_Work/Toxo/ToxoDB/TgondiiME49\_ToxoDB-6\_2.fasta* = The annotated Fasta file.
  2. *3* = The decoy ratio, i.e, how many random sequences to produce for each actual sequence.
  3. *~/Ritesh\_Work/Toxo/ToxoDB/TgondiiME49\_ToxoDB-6\_2\_withdecoy.fasta* = The output file produced
* **Blast formatting** - We need to blast-format the produced Fasta file using *formatdb* or *makeblastdb,* depending on the platform we are using.

When using makeblastdb –

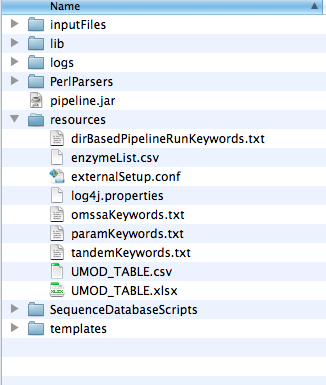
*$/usr/local/ncbi/blast/bin/makeblastdb -in TgondiiME49\_ToxoDB-6\_2\_withdecoy.fasta -dbtype prot*

When using formatdb –

*$/usr/local/ncbi/blast/bin/formatdb -i TgondiiME49\_ToxoDB-6\_2\_withdecoy.fasta -o*

This formatting should produce additional files with suffixes .pin,.psq etc. in the current directory.

* **Specifying the path of executables** - Once the database has been created and blast formatted, we need to specify the location of Ommsa and X!Tandem executable in the pipeline. The locations of the executables can be specified in the file names **externalSetup.conf** in the resources directory (see the screenshot below)

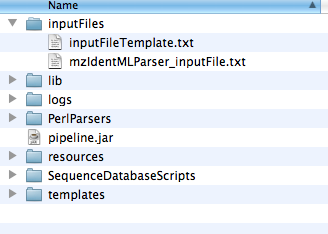


The following lines in the **externalSetup.conf** need to be changed to reflect the actual location of the executables –

omssa\_executable = /opt/omssa-2.1.9.macos/omssacl

tandem\_executable = /opt/tandem-osx-intel-10-01-01-4/bin/tandem

* **Inputting the search criteria** – The directives for the search engines can be specified in the files **inputFileTemplate.txt** and **mzIdentMLParser\_inputFile.txt** in the inputFiles folder shown below –



The contents of **mzIdentMLParser\_inputFile.txt** is here –

*user\_name = Ritesh*

*decoy\_regex = Rnd*

*decoy\_ratio = 3*

*rank\_threshold = 3*

* 1. decoy\_regex = Rnd, is the tag used for creating decoy sequences while creating the decoy database.
  2. Decoy\_ratio = 3, indicates the ratio we used for creating number of decoy sequences for each actual sequence
  3. rank\_theshold = 3, indicates that we are interested in only top 3 hits for peptide spectrum matches reported by the search engines.

The content of **inputFileTemplate.txt** is presented here; the contents that can be changed to specify search criteria are highlighted.

*input\_file= -fm {{ input\_file\_path }}*

*fasta\_file= -d ~/ToxoDB/TgondiiME49\_ToxoDB-6\_2\_withdecoy.fasta*

*enzyme\_name = -e 0*

*product\_tolerance= -to 0.8*

*precursor\_tolerance= -te 1.5*

*fixed\_mod\_id= -mf 3*

*variable\_mod\_id= -mv 1*

*missed\_cleavages = -v 1*

*output\_file= -oc {{ output\_dir\_path }}/testrun\_osx.csv*

*species=Toxoplasma*

*default\_input\_file=/opt/tandem-osx-intel-10-01-01-4/bin/default\_input.xml*

*taxonomy\_file={{ output\_dir\_path }}/taxonomy.xml*

Description of each changeable line is –

* + - *fasta\_file* -- Name of the blast formatted fasta database
    - *enzyme\_name* – 0 represented Trypsin. The enzymes are specified by their numberic representation as specified by Omssa search engine. The numeric representation of other enzymes can be see by running - *$omssacl –el*
    - *product\_tolerance –* User specified number
    - *precursor\_tolerance -* User specified number
    - *fixed\_mod\_id –* Fixed modifications are represented by numeric representation of mods by Omssa. The list of the modifications and their respective numeric identifiers can be seen by running - *$omssacl –ml*
    - *variable\_mod\_id* – Variable modifications, same as above.
    - *missed\_cleavages* – Number of maximum missed cleavages allowed
    - *species –* Any name is allowed
    - *default\_input\_file –* The location of default\_input.xml for X!Tandem search engine. The file is usually present in the bin directory of the X!Tandem installation.

**Running the pipeline**

The pipeline can be run by changing to the root directory where pipeline.jar is present. We need to call the pipeline.jar with the following arguments –

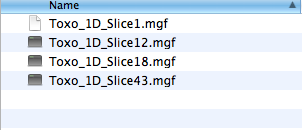
*$ java -jar pipeline.jar inputFiles/inputFileTemplate.txt ~/Ritesh\_Work/Toxo/Toxo\_Test\_MSDataset ~/Ritesh\_Work/TestSpace/pipeline\_test/ = inputFiles/mzIdentMLParser\_inputFile.txt =*

The arguments are as following –

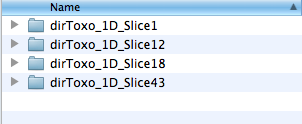
* *inputFiles/inputFileTemplate.txt* = location of the inputFileTemplate.txt which contains the criteria for search engines
* *~/Ritesh\_Work/Toxo/Toxo\_Test\_MSDataset*  = location of the folder which contains .mgf files
* *~/Ritesh\_Work/TestSpace/pipeline\_test/* = location of the folder where the results will be stored
* = is a placeholder
* *inputFiles/mzIdentMLParser\_inputFile.txt* – location of the mzIdentMLParser\_inputFile.txt which contains the tags to identify random sequences, decoy ratio and rank threshold.
* = is a placeholder

**Output from the pipeline**

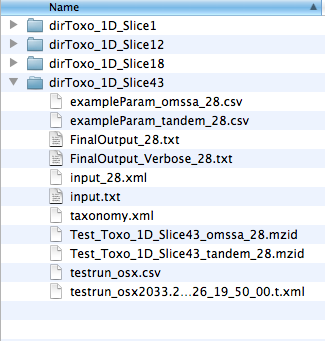
The pipeline produces a number of files for *each* input .mgf file. All the input .mgf files can be put in a single directory, and the directory can be specified as input directory while calling the pipeline ( in the above section we had the directory *~/Ritesh\_Work/Toxo/Toxo\_Test\_MSDataset* with all the input .mgf files.). A screenshot of the input directory with 4 .mgf files is given below –



The pipeline will create 4 directories in the output directory specified ( in the previous section - *~/Ritesh\_Work/TestSpace/pipeline\_test/*  was the output directory mentioned) as shown below. Each directory contains files specific to a given input .mgf file –



Each directory contains the same number of files. The pipeline runs in different stages, and each stage produces a set of files. A typical structure of the output directory (dirToxo\_1D\_Slice43) is shown in the following screenshot –



**Description of the files in the above directory –** Each output directory should have 11 files as shown in the screenshot above. If we are missing some files, it means that the pipeline is failing at some stage and the logs must be checked to find the break point. The pipeline runs in three stages, and the output files are grouped according to the stages. We have appended a random number (in this example *28*) to most of the files produced, this mechanism was introduced as a identify the set of files while belonged together, but is redundant now and can be dropped in the later versions.

**Stage 1 - Call the search engines Omsaa and X!Tandem.**

The files used while calling the search engines are –

input\_28.xml

input.txt

tanonomy.xml

The files produced by Omssa is – testrun\_osx.csv

The file produced by X!Tandem is – testrun\_osx2033.2…….xml

**Stage 2 – Convert the outout files produced by search engines to a standard MzIdentML format**

The Omssa output file *testrun\_osx.csv*  is converted to Test\_Toxo\_1D\_Slice43\_omssa\_28.mzid

The X!Tandem output file testrun\_osx2033.2…….xml is converted to Test\_Toxo\_1D\_Slice43\_tandem\_28.mzid

The intermediate files used were - exampleParam\_omssa\_28.csv and exampleParam\_tandem\_28.csv.

The mzIdentML files can be used for extract other information as well using an independent parser developed in collaboration with European Bioinformatics Institute. The java based parser can be found at - (<http://code.google.com/p/jmzidentml/>)

**Stage 3 – Call the multiple search engine consensus algorithm to find the consensus between the search engines and correct the FDR score**

The files produced at this stage are –

FinalOutput\_28.txt

FinalOutput\_Verbose\_28.txt

These two files are the most important files produced by the pipeline.

**FinalOutput\_Verbose\_28.txt** – Contains two sections with headings, *X!Tandem***,** and, *Omssa***.** The **individual search results** are reported in this file.

All the peptides reported by X!Tandem and Omssa are listed under their respective headings. This is a CSV file where each row reports the information about a peptide and Protein found by the search engine. The columns in the file are –

* Line number
* Spectrum Identifier
* Peptide Sequence
* Estimated FDR score
* Calculated Mass
* Group-ID (this is a placeholder in this version, and can be used for Protein Ambiguity Grouping in the later version)
* Protein Accession
* Start location in the protein sequence
* End location in the protein sequence
* Simple FDR score
* Modifications used (multiple modifications are separated by ##)
* Charge
* Experimental Mass

An example row is shown below –

1 , 43.5500.5521.2.dta , ALLELQMDGEEIYQTFSR , 4.0362105E-10 , 2159.59 , GROUP\_ID , gb|TGME49\_005470 , 157, 174 , 0.0 , ## 15.9949\_M:7 , 2 , 2159.033

**FinalOutput\_28.txt**

The file contains information about the **consensus search result**, i.e., the spectrums which were identified by both the search engines. The results in this file are categorised under three headings –

* *Sequences in the container : [ot]* – The peptide sequences which were identified by both Omssa and X!Tandem
* *Sequences in the container : [o]* - The peptide sequences which were identified by both Omssa only
* *Sequences in the container : [t]* - The peptide sequences which were identified by X!Tandem only

The columns in this CSV file are same as the ones in FinalOutput\_Verbose\_28.txt, except the Estimated FDR column, which reports the new FDR values, computed after making the consensus among the search engine results.