InsPecT Tutorial

Sam Payne and Natalie Castellana

January 2012

InsPecT is a tool for interpreting peptide tandem mass spectra. Information about the program and its authors can be found at http://peptide.ucsd.edu/. The purpose of this tutorial is to help guide you through the steps necessary to complete a successful run of the program. This tutorial is divided into four parts:

- Preparing your computer
- Setting up the InsPecT run
- The InsPecT run
- Analysis of the InsPecT results

The InsPecT package includes the
source code as well as a Windows executable. The executable was built on a 64-bit machine running Windows 7.

This tutorial covers only installing and setting up a basic InsPecT run. It does not cover most options in our basic workflow. Please consult the InsPecT Advanced Tutorial and the Unrestrictive Search Tutorial. Both of these introduce features that we routinely use.

Part 1. Preparing Your Computer

Download the InsPecT package from http://proteomics.ucsd.edu/Software/Inspect.html and unzip it. This will create an InsPecT directory containing the source code for InsPecT as well as an analysis toolkit. InsPecT and its accompanying scripts are command line tools. We recommend using the web interface available at http://proteomics.ucsd.edu/LiveSearch/ which has an intuitive graphical interface.

The Python programming language

To help you analyze the InsPecT results, the authors included several Python scripts with the InsPecT download. Using these scripts requires Python version 2.5 or greater on your machine. The newest Python release can be found at http://www.python.org/download/releases/. The post-processing scripts included with InsPecT also require the following python libraries:

- Python Image Library (PIL)
- Numerical Python Library (NumPy)

Please install these libraries for your system, and include them in the python path.

Part2. Setting Up The InsPecT Run

InsPecT expects your data to be in the right format prior to the run: MS/MS spectra in a common format, a trie database, and a parameters file.

MS/MS Spectra

The spectra files for InsPecT must be in a common, non-proprietary format. The preferred format is mzXML, but the mgf, mzData, ms2, and dta formats are acceptable. See the conversion tools provided by the Seattle Proteome Center (http://sashimi.sourceforge.net/) for mzXML conversion tools.

Database Setup

InsPecT requires pre-processing of the protein database before running. The processed database can be created from a FASTA file using the script PrepDB.py. From the InsPecT directory run

```
> python PrepDB.py FASTA myDB.fasta
```

where “myDB.fasta” is the name of your database. This creates two files which will be used by InsPecT: myDB.trie and myDB.index. Once these files are created, they can be reused for later InsPecT runs.

Parameters File

A simple input file feeds arguments to InsPecT. Each line in the parameters file is of the same form: a parameter name followed by a comma followed by the value for the parameter. The permitted parameters are

- **spectra,[FILENAME]** - Specifies a spectrum file to search. You can specify the name of a directory to search every file in that directory (non-recursively).
  Preferred file formats: .mzXML and .mgf
  Other accepted file formats: .mzData, .ms2 .dta. Note that multiple spectra in a single .dta file are **not** supported.
- **db,[FILENAME]** - Specifies the name of a database (.trie file) to search. The .trie file contains one or more protein sequences delimited by asterisks, with no whitespace or other data. You should specify at least one database. You may specify several databases; if so, each database will be searched in turn.
- **SequenceFile,[FILENAME]** - Specifies the name of a FASTA-format protein database to search. If you plan to search a large database, it is more efficient to preprocess it using PrepDB.py and use the "db" command instead. You can specify at most one SequenceFile.
- **protease,[NAME]** - Specifies the name of a protease. "Trypsin", "None", and "Chymotrypsin" are the available values. If tryptic digest is specified, then matches with non-tryptic termini are penalized.
- **mod,[MASS],[RESIDUES],[TYPE],[NAME]** - Specifies an amino acid modification. The delta mass (in daltons) and affected amino acids are required. The first four characters of the name should be unique. Valid values for "type" are "fix", "cterminal", "nterminal", and "opt" (the default). For a guide to various known modification types, consult the following databases:
  Examples:
  mod,+57,C,fix - Most searches should include this line. It reflects the addition of CAM (carbamidomethylation, done by adding iodoacetamide) which prevents cysteines from forming disulfide bonds.
  mod,80,STY,opt,phosphorylation
  mod,16,M,opt - Oxidation of methionine, seen in many samples
  mod,43,*,nterminal - N-terminal carbamylation, common if sample is treated with urea
  Important note: When searching for phosphorylation sites, use a modification with the name "phosphorylation". This lets Inspect know that it should use its model of phosphopeptide fragmentation when generating tags and scoring matches. (Phosphorylation of serine dramatically affects fragmentation, so modeling it as simply an 80Da offset is typically **not** sufficient to detect sites with high sensitivity)
- **Mods,[COUNT]** - Number of PTMs permitted in a single peptide. Set this to 1 (or higher) if you specify PTMs to search for.

- **Unrestrictive,[FLAG]** - If FLAG is 1, use the MS-Alignment algorithm to perform an **unrestrictive** search (allowing arbitrary modification masses). Running an unrestrictive search with one mod per peptide is slower than the normal (tag-based) search; running time is approximately 1 second per spectrum per megabyte of database. Running an unrestrictive search with two mods is significantly slower. We recommend performing unrestrictive searches against a small database, containing proteins output by an earlier search.
- **MaxPTMSize,[SIZE]** - For blind search, specifies the maximum modification size (in Da) to consider. Defaults to 250. Larger values require more time to search.
- **PMTolerance,[MASS]** - Specifies the parent mass tolerance, in Daltons. A candidate's flanking mass can differ from the tag's flanking mass by no more than ths amount. Default value is 2.5. Note that secondary ions are often selected for fragmentation, so parent mass errors near 1.0Da or -1.0Da are not uncommon in typical datasets, even on FT machines.
- **IonTolerance,[MASS]** - Error tolerance for how far ion fragments (b and y peaks) can be shifted from their expected masses. Default is 0.5. Higher values produce a more sensitive but much slower search.
- **MultiCharge,[FLAG]** - If set to true, attempt to guess the precursor charge and mass, and consider multiple charge states if feasible.
- **Instrument,[TYPE]** - Options are ESI-ION-TRAP (default), QTOF, and FT-Hybrid. If set to ESI-ION-TRAP, Inspect attempts to correct the parent mass. If set to QTOF, Inspect uses a fragmentation model trained on QTOF data. (QTOF data typically features a stronger y ladder and weaker b ladder than other spectra).
- **RequiredMod,[NAME]** - The specified modification MUST be found somewhere on the peptide.
- **TagCount,[COUNT]** - Number of tags to generate
- **TagLength,[LENGTH]** - Length of peptide sequence tags. Defaults to 3. Accepted values are 1 through 6.
- **RequireTermini,[COUNT]** - If set to 1 or 2, require 1 or 2 valid proteolytic termini. Deprecated, because the scoring model already incorporates the number of valid (tryptic) termini.

An example params file might look like this:

```
spectra,Fraction01.mzxml
instrument,ESI-ION-TRAP
protease,Trypsin
DB,TestDatabase.trie
# Protecting group on cysteine:
mod,57,C,fix
```

The example above reflects the addition of CAM (carbamidomethylation, done by adding iodoacetamide) to cysteine, inhibiting disulfide bonds. Most searches should include this line.

Part 3. The InsPecT Run

Now that you have gotten this far, the actual InsPecT run is very simple. Type the following at the command prompt

```
> InsPecT.exe –i InputFile.txt –o OutputFile.txt (windows)
> ./inspect –i InputFile.txt –o OutputFile.txt (unix)
```

Depending on the size of the database and the number of spectra, InsPecT may take only a few minutes, or several hours.

Part 4. Analysis of the Results

The InsPecT output file contains an annotation for every MS/MS spectrum, most of which are not statistically significant. Basic filtering and analysis can be done with the toolkit, i.e. python scripts included in the distribution. It is essential that you do postprocessing. At the very least you must run the ComputeFDR.jar script (see below and Advanced tutorial).

ComputeFDR.jar

The purpose of this script is to weed out insignificant results. It requires that a decoy search was also performed with the spectra. The usage is

```
usage: java -jar ComputeFDR.jar VERSION
       -f resuleFileName protCol decoyPrefix or -f targetFileName decoyFileName
       -n scanNumCol (the scanNum comlun number)
       -p pepCol (the peptide column number)
       -s scoreCol 0/1 (0: smaller better, 1: greater better)
       [-o outputFileName (default: stdout)]
       [-delim delimeter] (default: \t)
       [-m colNum keyword (the column 'colNum' must contain 'keyword'. If 'keyword' 
is delimited by ',' (e.g. A,B,C), then at least one must be matched.)]
       [-h 0/1] (0: no header, 1: header (default))
       [-fdr fdrThreshold]
       [-pepfdr pepFDRThreshod]
       [-decoy 0/1 (0: include decoy, 1: don't include decoy (default))
```

Typical usage with InsPect would be

```
> java -jar ComputeFDR.jar -f OutputFile.txt 3 XXX -n 1 -p 2 -s 14 1 
-fdr 0.05 -o FilteredResultsFile.txt
```

This indicates that the protein name is provided in the 3rd column (all columns are 0-based) and decoy proteins start with XXX. If you use MS2DBShuffler or ShuffleDB, then the decoy proteins all start with XXX. The scan number is found in column 1 and the peptide sequence is found in column 2. The score column (the F-Score) is found in column 14 and a larger F-Score is better.

Summary.py

So a big text file is not what you want? Well, we kind of figured that. The summary script creates a webpage with peptides grouped into proteins. It uses the Python Image Library to make pictures of annotated spectra.

The usage for Summary.py is

```
Required options:
 -r [FileName] - The name of the results file to parse.  If a directory is
    specified, then all .txt files within the directory will be combined into
    one report
 -d [FileName] - The name of the database file (.trie format) searched.
          (allows more than one database file; use multiple -d options)

Additional options:
 -b [FileName] - Second-pass database filename.  If specified, the proteins
```

```
    selected will be written out to a database (.fasta, .trie and .index files)
    suitable for unrestrictive search.
 -w [FileName] - Webpage directory.  If specified, a webpage will be written
    to the specified directory, summarizing the proteins identified, and the
    degree of coverage.

 (Note: Either -b or -w, or both, must be provided)

 -e [Count] - Minimum number of peptides that a protein must annotate in order
    to add it to the report or the filtered database.  Defaults to 1.
 -m [Count] - Minimum number of spectra that a protein must annotate in order
    to add it to the report or the filtered database.  By default, this count
    is set to (SpectrumCount / ProteinsInDatabase) * 2.  If the protein
    database has already been filtered, set this parameter to 1.
 -v [Count] - Verbose spectrum output count.  If set, report [Count] spectra
    for each distinct peptide identified.  This option is slower and
    consumes more memory, but can be more informative.
 -i [SpectraPath] - For use if verbose spectrum output (-v) is enabled.
    Images will be generated for each annotation, if the Python Imaging
    Library (PIL) is installed.  This option generates many files on disk,
    so it's recommended that you set the summary file (-w option) in its own
    directory.  "SpectraPath" is the path to the folder with MS2 spectra
```

Typical use might be

```
> python Summary.py -r FilteredResultsFile.txt -d myDB.trie -w
SummaryDir/FilteredResultsSummary.html -v 10 -i mySpectraDir/
```