



X!TandemPipeline is one of the PAPPSO facility software projects

X!TANDEMPIPELINE USER MANUAL

FREE AND OPEN SOURCE PROTEIN IDENTIFICATION SOFTWARE

X!TANDEMPIPELINE 0.4.43

X!TANDEMPIPELINE USER MANUAL: FREE AND OPEN SOURCE PROTEIN IDENTIFICATION SOFTWARE

by Benoît Valot, Olivier Langella, Thomas Renne, Filippo Rusconi, and Michel Zivy

November 23, 2021 , 0.4.43

Copyright 2021 Filippo Rusconi and Olivier Langella

Thierry Balliau and Marlène Davanture are warmly thanked for their outstanding technical help while writing this user manual.

X!TandemPipeline


[HTTP://PAPPSO.INRAE.FR/EN/BIOINFO/](http://pappso.inrae.fr/en/bioinfo/) 

This book is part of the *X!TandemPipeline* project.

The *X!TandemPipeline* project is the successor of the Java language-based homonymous project. This project is a full rewrite of the former project in the C++ language, with many new features added.

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see [HTTP://WWW.GNU.ORG](http://www.gnu.org) ([HTTP://WWW.GNU.ORG/LICENSES/](http://www.gnu.org/licenses/)) .

Revision History

Revision 0.7.24	28 May 2021	Filippo Rusconi
-----------------	-------------	-----------------

- Start actually documenting the software with the Preface.

Revision 0.7.24	21 April 2021	Filippo Rusconi
-----------------	---------------	-----------------

- Very first setting up of the user manual using the mineXpert2 project as a template.

DEDICATION

To all the admirable people acting in the “*Free Software Movement*” for a better and more ethical computing world

To all the readers who helped with this manual.

CONTENTS

PREFACE iv

I GENERALITIES I

- 1.1 GENERAL CONCEPTS AND TERMINOLOGIES I
 - BOTTOM-UP PROTEOMICS OR TOP-DOWN PROTEOMICS? I • TYPICAL CYCLE OF A MASS SPECTROMETER DATA ACQUISITION 2 • OUTLINE OF AN X!TANDEMPipeline WORKING SESSION 2
- 1.2 CITING THE X!TANDEMPipeline SOFTWARE. 3
- 1.3 INSTALLATION OF THE SOFTWARE 3
 - INSTALLATION ON MS WINDOWS AND macOS SYSTEMS 3 • INSTALLATION ON DEBIAN- AND UBUNTU-BASED SYSTEMS 3 • INSTALLATION WITH AN APPIMAGE SOFTWARE BUNDLE 4
- 1.4 BUILDING THE SOFTWARE FROM SOURCE 5
 - THE DEPENDENCIES REQUIRED TO BUILD X!TANDEMPipeline 5 • GETTING THE SOURCE TARBALL 6 • BUILDING OF THE SOFTWARE 6

2 FUNDAMENTALS IN BOTTOM-UP PROTEOMICS 8

- 2.1 THE PROTEIN BIOPOLYMER: STRUCTURE AND CHEMISTRY 8
 - PROTEIN BIOSYNTHESIS 8 • PROTEIN DISRUPTING CHEMISTRIES 10
- 2.2 GENERAL OVERVIEW OF BOTTOM-UP PROTEOMICS 15
 - THE FIRST STEP: DIGESTION OF THE SAMPLE'S PROTEINS 17 • CHROMATOGRAPHIC SEPARATION OF THE PEPTIDIC MIXTURE 18 • MASS SPECTROMETRIC ANALYSIS OF THE PEPTIDES 19 • THE PROTEIN DATABASES AND THEIR USE 20 • MATCHING FRAGMENTATION SPECTRA WITH THEORETICAL SPECTRA 21 • PHOSPHO-PROTEOMICS 30

3 THE MAIN PROGRAM WINDOW 32

- 3.1 STARTING A NEW X!TANDEMPipeline WORKING SESSION 33
- 3.2 RUNNING X!TANDEM IDENTIFICATIONS 33

3.3	SETTING THE X!TANDEM RUN PRESETS	35
	LOADING EXISTING PRESETS CONFIGURATIONS FROM FILE	36 • CREATING
	NEW PRESETS CONFIGURATIONS	37 • ACTUAL X!TANDEM PRESETS
	CONFIGURATION	37 • RUNNING A PROPERLY CONFIGURED X!TANDEM PROCESS
		37
3.4	LOADING IDENTIFICATION RESULTS	38
	CONFIGURATION OF THE PARAMETERS	40 • SAVING X!TANDEMPipeline
	PROJECTS	43
3.5	LOADING X!TANDEMPipeline PROJECTS	43
4	EXPLORING IDENTIFICATION DATA	44
4.1	THE PROTEIN IDENTIFICATIONS LIST WINDOW	44
	THE PROTEIN IDENTIFICATIONS LIST TABLE VIEW	44 • OPERATIONS IN THE PROTEIN
	IDENTIFICATION LIST WINDOW	46 • DELVING INSIDE THE PROTEIN IDENTIFICATION
	DATA	50
4.2	THE PEPTIDE IDENTIFICATIONS LIST WINDOW	51
	THE PEPTIDE IDENTIFICATIONS LIST TABLE VIEW	51 • OPERATIONS IN THE PEPTIDE
	IDENTIFICATION LIST WINDOW	53 • DELVING INSIDE THE PEPTIDE IDENTIFICATION
	DATA	53
4.3	HANDLING PHOSPHO-PROTEOMICS DATA	57
5	STUFF AWAITING INCLUSION	59
A	GNU GENERAL PUBLIC LICENSE VERSION 3	60

PREFACE

I SOFTWARE FEATURE OFFERINGS AND INTENDED AUDIENCE

This manual is about the *X!TandemPipeline* protein identification software project.

X!TandemPipeline has the following features:

- Load mass spectrometry data files in the mzXML or mzML format, thanks to the excellent *libpwiz* library of ProteoWizard¹ fame.
- Configure the way the peptide spectrum matches (PSM) are to be performed;
- Configure the database files to be used (target organism databases and contaminant databases);
- Use the MS/MS data in the file to feed the *X!Tandem* program that produces peptide identification results by matching the measured ion masses with peptide fragments calculated *in silico* on the basis of the databases contents;
- Display the data in powerful ways in a unified graphical user interface to allow the user to inspect the peptide identifications and also control the way these identifications are used to infer the protein identifications.

2 FEEDBACK FROM THE USERS

We are always grateful to any constructive feedback from the users.

The PAPPSO software team might be contacted *via* the following contact page:

[HTTP://PAPPSO.INRAE.FR/EN/TRAVAILLER_AVEC_NOUS/CONTACT/](http://pappso.inrae.fr/en/travailler_avec_nous/contact/)  (search for team members having the “Bioinformatics” specialty mentioned, like Olivier Langella or Filippo Rusconi).

3 PROGRAM AND DOCUMENTATION AVAILABILITY AND LICENSE

The programs and all the documentation that are shipped along with the *X!TandemPipeline* software suite are available at [HTTP://PAPPSO.INRAE.FR/EN/BIOINFO/XTANDEMPIPELINE/](http://pappso.inrae.fr/en/bioinfo/xtandempipeline/) . Most of the time, a new version is published as source, and as binary install packages for *MS-Windows* (64-bit systems only).

¹ [HTTP://PROTEOWIZARD.SOURCEFORGE.NET/](http://proteowizard.sourceforge.net/) .

For *GNU/Linux*, binary packages are created locally (see [HTTP://PAPPSO.INRAE.FR/EN/BIOINFO/XTANDEM-PIPELINE/DOWNLOAD/](http://pappso.inrae.fr/en/bioinfo/xtandempipeline/download/)²) but are also built in the *Debian*² autobuilders and are uploaded to the distribution servers. These packages are available using the system's software management infrastructure (like using the *Debian*'s **apt** command, for example, or the graphical application).

The software and all the documentation are all provided under the Free Software license *GNU General Public License, Version 3, or later, at your option*. For an in-depth study of the *Free Software* philosophy, the reader is kindly urged to visit [HTTP://WWW.GNU.ORG/PHILOSOPHY](http://www.gnu.org/philosophy)².

² [HTTP://WWW.DEBIAN.ORG/](http://www.debian.org/)

I GENERALITIES

In this chapter, I wish to introduce some general concepts around the *X!TandemPipeline* program, the reference to be used to cite the software in publications, the building and installation procedures.

I.1 GENERAL CONCEPTS AND TERMINOLOGIES

This section describes the general concepts at the basis of the analysis of proteomics data that one needs to grok in order to properly assimilate the workings of the *X!TandemPipeline* software.

I.1.1 BOTTOM-UP PROTEOMICS OR TOP-DOWN PROTEOMICS?

Proteomics is a mass spectrometry-based field of endeavour that is aimed at characterizing the “protein complement” of a given genome. The protein complement of a genome is the set of proteins that are expressed at a given instant in the life of a cell, a tissue or an organ, for example. Characterizing that protein complement actually means identifying the proteins expressed by a given living cell or tissue or organ. Optionally, if feasible, the characterization of post-translational modifications might be desirable.

There are two main variants of proteomics: “bottom-up” proteomics and “top-down” proteomics:

- The first variant—bottom-up proteomics—identifies proteins on the basis of the identification of all the peptides obtained by first digesting all the proteins of the sample using an enzyme of known specificity. In this variant, the sample that is injected in the mass spectrometer is the resulting peptide mixture (first resolved by high performance liquid chromatography). The identification of the proteins contained in the initial sample is performed in a number of steps that are actually the focus of *X!TandemPipeline*. Indeed the *X!TandemPipeline* software is a bottom-up-oriented software program.
- The second variant—top-down proteomics—identifies proteins on the basis of intact proteins directly injected in the mass spectrometer. Of course, it might be necessary to fragment the proteins in the mass spectrometer and to use the fragments to actually identify the protein. However, the fact that the protein is first detected and analyzed as one entity (and not as set of peptides), allows for some very useful discoveries, like the identity and number of post-translational modifications, for example.



NOTE

At the moment, *X!TandemPipeline* does not handle top-down proteomics data: it is a bottom-up proteomics software project.

1.1.2 TYPICAL CYCLE OF A MASS SPECTROMETER DATA ACQUISITION

Once the initial sample, containing all the proteins to identify, has been digested using a protease of known cleavage specificity (trypsin, typically), the peptidic mixture (that might be highly complex) needs to be resolved as much as possible using chromatography. In the vast majority of the proteomics experimental settings, the chromatography setup is connected to the mass spectrometer so that when the gradient is developed, all the peptides are immediately injected “on line” to the mass spectrum ion source.

The mass spectrometer runs an analysis cycle that can be summarized like the following:

- Acquire a full scan mass spectrum of the whole set of ions at a given chromatography retention time. This kind of mass spectrum is called a MS spectrum;
- Enter a loop during which ions having the most intense signal are subjected in turn to collision-induced dissociation (CID), that is, are fragmented by accelerating them against gas molecules in a fragmentation cell. The mass spectra that are collected at each one of these fragmentation acquisitions are called MS/MS spectra because they are obtained after two mass analysis events: the first event is the measurement of the intact peptide ion's m/z value (full scan mass spectrum) and the second event is the measurement of all the obtained fragments' m/z values (MS/MS scan).

Each instrument records all the MS and MS/MS spectra in a raw data format file that is specific of the vendor. Free Software developers cannot know the internal structure of the files. To use the mass spectrometric data, they need to rely on a specific software that performs the conversion from the raw data format to an open data format (mzML). That program is called *msconvert*, from the *ProteoWizard* project.



NOTE

Mass spectrometrists used to call ions that were analyzed in full scan mass spectra “parent ions”. They also used to call fragment ions arising upon fragmentation of a parent ion “daughter ions”. This terminology has been deprecated and has been replaced with “precursor ion” and “product ion”, respectively. In our document, we thus use the new terminology.

1.1.3 OUTLINE OF AN *X!TandemPipeline* WORKING SESSION

X!TandemPipeline loads mzXML- and mzML-formatted files and needs for its operations to have access to all the MS and MS/MS spectra. Once data files have been loaded, *X!TandemPipeline* allows the user to perform the following tasks, that will be detailed in later chapters:

- Configure the *X!Tandem* database searching software (that is, the software, external to *X!TandemPipeline* that actually performs the peptide-mass spectrum matches);
- Run the *X!Tandem* software and load its results;
- Display the results to the user in a way that they can be scrutinized and checked. The peptide identification results serve as the basis for another processing step that is integrally performed by *X!TandemPipeline*: the “protein inference”. That step aims at using the peptide identifications to actually craft a list of proteins identities. The user is provided with various means to control that step in various ways.

1.2 CITING THE *X!TandemPipeline* SOFTWARE.

Please, cite the software using the following citation: Olivier Langella, Benoît Valot, Thierry Balliau, Mélisande Blein-Nicolas, Ludovic Bonhomme, and Michel Zivy (2016) X!TandemPipeline: A Tool to Manage Sequence Redundancy for Protein Inference and Phosphosite Identification. *J. Proteome Res.* 2017, 16, 2, 494–503. <https://doi.org/10.1021/acs.jproteome.6b00632>.

1.3 INSTALLATION OF THE SOFTWARE


The installation material is available at [HTTP://PAPPSO.INRAE.FR/EN/BIOINFO/XTANDEMPIPELINE/DOWNLOAD/](http://pappso.inrae.fr/en/bioinfo/xtandempipeline/download/).

1.3.1 INSTALLATION ON MS WINDOWS AND MACOS SYSTEMS

The installation of the software is extremely easy on the MS-Windows and macOS platforms. In both cases, the installation programs are standard and require no explanation.

1.3.2 INSTALLATION ON DEBIAN- AND UBUNTU-BASED SYSTEMS

The installation on Debian- and Ubuntu-based GNU/Linux platforms is also extremely easy (even more than in the above situations). `xtandempipeline` is indeed packaged and released in the official distribution repositories of these distributions and the only command to run to install it is:

```
$ 1 sudo apt install <package_name> 
```

In the command above, the typical *package_name* is in the form `xtandempipeline` for the program package and `xtandempipeline-doc` for the user manual package.

¹ The prompt character might be `%` in some shells, like *zsh*.

Once the package has been installed the program shows up in the *Science* menu. It can also be launched from the shell using the following command:

```
$ xtpcpp RETURN
```



TIP

If the Debian system onto which the program is to be installed is older than *testing*, that is, older than *Buster* (*Debian 10*), then using the AppImage program bundle might be a solution. See below for the method to run *mineXpert2* as an AppImage bundle.

1.3.3 INSTALLATION WITH AN APPIMAGE SOFTWARE BUNDLE

The *AppImage* software bundle format is a format that allows one to easily run a software program on any GNU/Linux-based distribution. From the [HTTP://APPIMAGE.ORG/](http://AppImage.org/) 

The key idea of the AppImage format is one app = one file. Every AppImage contains an app and all the files the app needs to run. In other words, each AppImage has no dependencies other than what is included in the targeted base operating system(s).

—Simon Peter

There are AppImage software bundles for the various *mineXpert2* versions that are available for download. As of writing, the software bundle has been tested on *Centos* version 8.3.2011 and on *Fedora* version 22. These are pretty old distribution versions and thus *mineXpert2* should also run on more recent versions of these computing platforms. The AppImage bundle of *mineXpert2* was created on a rather current Debian version: the *testing Debian 11-to-be* distribution.

In order to run the *mineXpert2* software AppImage bundle, download the latest version (like `mineXpert2-0.7.4-x86_64.AppImage`). Once the file has been downloaded to the desired directory, change to that directory and change the permissions to make it executable:

```
$ chmod a+x mineXpert2-0.7.4-x86_64.AppImage RETURN
```

Finally, execute the file that has become a normal program:

```
$ ./mineXpert2-0.7.4-x86_64.AppImage RETURN
```



TIP

If the program complains about a locale not being found, please, modify the command line to read:

```
$ LC_ALL="C" ./mineXpert2-0.7.4-x86_64.AppImage RETURN
```

I.4 BUILDING THE SOFTWARE FROM SOURCE

The *mineXpert2* software build is under the control of the *CMake* build system. There are a number of dependencies to install prior to trying to build the software, as described below.

I.4.1 THE DEPENDENCIES REQUIRED TO BUILD *X!TandemPipeline*

The dependencies to be installed are listed here with package names matching the packages that are in Debian/Ubuntu. In other RPM-based software, most often the package names are similar, albeit with some slight differences.

DEPENDENCIES

The build system

`cmake`

Conversion of svg files to png files

`graphicsmagick-imagemagick-compat`

For the parallel computations

`libgomp1`

For the isotopic cluster calculations

`libisospec++-dev`

For all the raw mass calculations like the data model, the mass spectral combinations...

`libpappsomsp-dev, libpappsomsp-widget-dev`

For all the plotting

`libqcustomplot-dev`

For the C++ objects (GUI and non-GUI)

`qtbase5-dev, libqt5svg5-dev, qttools5-dev-tools, qtchooser`

For the man page

`docbook-to-man`

For the documentation (optional, with `-DMAKE_USER_MANUAL=1` as a flag to the call of *cmake*, see below.)

`daps, libjeuclid-core-java, libjeuclid-fop-java, docbook-mathml, libjs-jquery, libjs-highlight.js, libjs-mathjax, fonts-mathjax, fonts-mathjax-extras, texlive-fonts-extra, fonts-ebgaramond-extra`

I.4.2 GETTING THE SOURCE TARBALL

In the example below, the version of the software to be installed is 7.3.0. Replace that version with any latest version of interest, which can be looked for at <https://gitlab.com/msxpertsuite/minexpert2/-/releases>.

I.4.2.1 USING GIT

The rather convoluted command below only downloads the branch of interest. The whole git repos is very large...

```
$ git clone https://gitlab.com/msxpertsuite/minexpert2.git --branch mas-  
ter/7.3.0-1 --single-branch minexpert2-7.3.0
```

I.4.2.2 USING WGET TO DOWNLOAD THE TARBALL

```
wget https://gitlab.com/msxpertsuite/minexpert2/-/archive/7.3.0/minex-  
pert2-7.3.0.tar.gz
```

Untar the tarball, which creates the minexpert2-7.3.0 directory:

```
tar xvzf minexpert2-7.3.0.tar.gz
```

I.4.3 BUILDING OF THE SOFTWARE

Change directory:

```
$ cd minexpert2-7.3.0
```

Create a build directory:

```
$ mkdir build
```

Change directory:

```
$ cd build
```

Configure the build:

```
$ cmake ../ -DCMAKE_BUILD_TYPE=Release
```

Build the software:

```
$ make
```

2 FUNDAMENTALS IN BOTTOM-UP PROTEOMICS

This chapter is an optional chapter which the reader might be referred to upon reading other part of this manual.

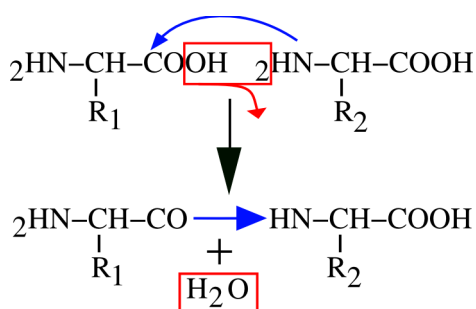
2.1 THE PROTEIN BIOPOLYMER: STRUCTURE AND CHEMISTRY

This section introduces the basics in protein polymer chemistry. The way this topic is going to be covered is admittedly biased towards mass spectrometry and proteins. Moreover, the aim of this chapter is to provide the reader with the specialized words that will later be used to describe and explain the (inner) workings of the *X!Tandem-Pipeline* program. This manual is not a “crash course” in biochemistry.

2.1.1 PROTEIN BIOSYNTHESIS

Proteins are made of amino acids. There are twenty major amino acids in nature, and each protein is made of a number of these amino acids. The combinations are infinite, providing enormous diversity to the protein realm. A protein is a polar polymer: it has a left end and a right end, and polymerization actually occurs from left to right (from N-terminus to C-terminus, see below). **FIGURE 2.1, “PEPTIDIC BOND FORMATION BY CONDENSATION”** shows that the chemical reaction at the basis of protein synthesis is a *condensation*. A protein is the result of the condensation of amino acids with each other in an orderly polar fashion. A protein has a left end, called *N-terminus; amino-terminal end* and a right end, called *C-terminus; carboxy-terminal end*. The left end is an amino group ($_2\text{HN}-$) corresponding to the non-reacted α -amino group of the very first amino acid of the protein sequence. Upon condensation of a new entering amino acid onto the first N-terminal one, the amino group of the entering amino acid reacts (nucleophilic attack) with the α -carboxyl group of the N-terminal amino acid. A water molecule is released, and the formation of an amide bond between the two amino acids yields a dipeptide. The right end of the dipeptide is a carboxyl group ($-\text{COOH}$) corresponding to the un-reacted α -carboxyl group of the last amino acid to have been “polymerized in”.

The bond formed by condensation of two amino acids is an amide bond, also called—in protein chemistry—a *peptidic bond*. The elongation of the protein is a simple repetition of the condensation reaction shown in **FIGURE 2.1, “PEPTIDIC BOND FORMATION BY CONDENSATION”**, granted that the elongation *always* proceeds in the described direction (a new monomer arrives to the right end of the elongating polymer, and elongation is done from left to right).



The left end monomer R_1 is condensed to the right end monomer R_2 to yield a peptidic bond. A water molecule is lost during the process.

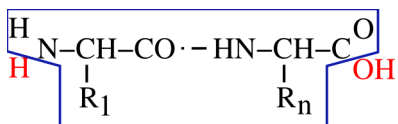
FIGURE 2.1: PEPTIDIC BOND FORMATION BY CONDENSATION



NOTE

Now we should point at a protein chemistry-specific terminology issue: we have seen that a protein is a polymer made of a number of monomers, called *amino acids*. In protein chemistry, there is a subtlety: once an amino acid has been polymerized into a protein, it is no more called an amino acid, but is called a *residue* instead. We may say that a residue is an amino acid less a water molecule.

From what we have seen until now, we may define a protein this way: — “*A protein is a chain of residues linked together in an orderly polar fashion, with the residues being numbered starting from 1 and ending at n, from the first residue on the left end to the last one on the right end*”. This definition is still partly inexact, however. Indeed, from what is shown in **FIGURE 2.2, “END CAPPING CHEMISTRY OF THE PROTEIN POLYMER”**, there is still a problem with the extremities of the residual chain: what about the amino group on the left end of a protein (the amino group sits right onto the first amino acid of the protein), and what about the carboxyl group of the right end of a protein (the carboxyl group sits right onto the last amino acid of the protein)? Because these groups lie at the extremities of the residual chain, they remained unreacted during the polymerization process. But because we are simulating a residual chain using residues and not amino-acids, we still need to put the protein polymer molecule in its “finished state”: by *capping* the left end with a proton *cap* (so as to complete the amino group) and the right end with a hydroxyl cap (so as to complete the carboxyl group). The capping of the residual chain extremities ensures that the polymer is in its finished state, and that it cannot be elongated anymore. The proton is the *left cap* of the protein polymer and the hydroxyl is the *right cap* of the protein polymer.



A protein is made of a chain of residues and of two caps. The left cap is the N-terminal proton and the right cap is the C-terminal hydroxyl. Altogether, the residual chain (enclosed here in the blue polygon) and both the H and OH red-colored caps do form a complete protein polymer in its finished state.

FIGURE 2.2: END CAPPING CHEMISTRY OF THE PROTEIN POLYMER

Now comes the question of unambiguously defining the structure of a protein. It is commonly accepted that the simple ordered sequence of each residue code in the protein, from left to right, constitutes an unambiguous description of the protein's primary structure (that is, its sequence). Of course, proteins have three-dimensional structures, but this is of no interest to a program like *massXpert*, which is aimed at calculating masses of polymers. To enunciate unambiguously the sequence of a protein, one would use a symbology like this:

- Using the 3-letter code of the amino acids:
Ala Gly Trp Tyr Glu Gly Lys
- Using the 1-letter code of the amino acids:
A G W Y E G K
Alanine is thus the residue 1 and Lysine is the last residue ($n = 7$)

2.1.2 PROTEIN DISRUPTING CHEMISTRIES

The “polymer chain disrupting chemistry” was mentioned earlier as a complex subject that was of *enormous* importance to the mass spectrometrists. This is why that subject will be treated in a pretty thorough manner. First of all it should be noted that a chemical modification of a polymer does not necessarily involve the perturbation of the chain structure of the polymer. Here, however, we are concerned specifically with a number of chemical modifications that yield a polymer chain perturbation; *cleavages* and *fragmentations*:

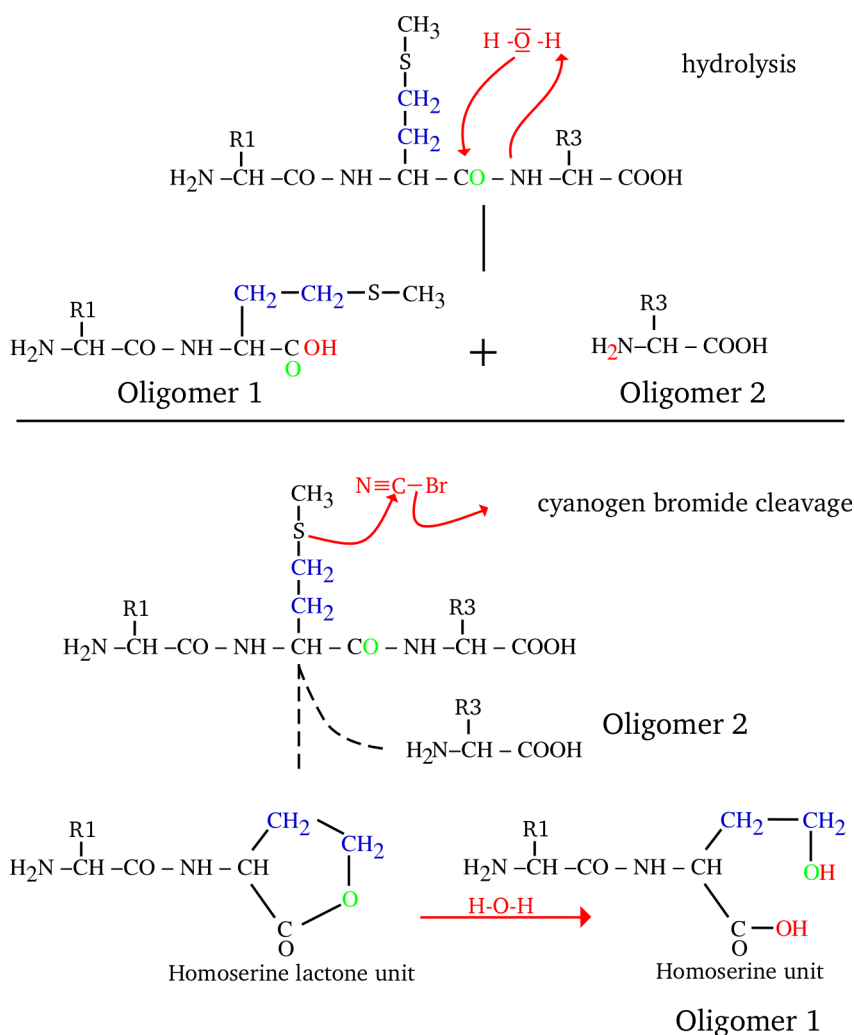
Cleavages. These are chemical processes by which a cleaving agent will act directly on the protein residual chain making it fall into at least two separated pieces (the peptides).

Fragmentations. These are chemical processes by which the polymer structure is disrupted into separated pieces (the *product ions*, or *fragments*) mainly because of energy-dependent electron doublet rearrangements leading to bond breakage.

Upon cleavage of a protein, the cleaving molecule reacts with it, and by doing so directly or indirectly “*dissolves*” an inter-residue bond. A protein cleavage always occurs in such a way as to generate a set of *true* finished polymerization state “proteins” (smaller in size than the parent polymer, evidently, which is why they are called *oligopeptides*, or *peptides*). Indeed, let us take the example shown in [FIGURE 2.3, “PROTEIN CLEAVAGE BY WATER AND CYANOGEN BROMIDE”](#), where a tripeptide (a very little protein, containing a methionyl residue at position 2) is submitted either to a water-mediated cleavage (hydrolysis, upper panel) or to a cyanogen bromide-mediated cleavage (lower panel). The two cases presented in this figure are similar in some respects and different in others:

- In the first case the molecule that is responsible for the cleavage is water, while in the second case it is cyanogen bromide;
- In both cases the bond that is cleaved is the inter-monomer bond (in protein chemistry this is a peptidic bond);
- In both cases the Oligomer 2 has the same structure;
- The structures of the Oligomer 1 species differ, when produced using water or cyanogen bromide as the cleaving molecule.

The difference between hydrolysis and cyanogen bromide cleavage is in the generation of the Oligomer 1 species: the cyanogen bromide cleavage has a side effect of generating a homoseryl residue at the C-terminus of Oligomer 1, while hydrolysis generates a genuine methionyl residue. This is because water reverses in a very symmetrical manner what polymerization did (hydrolysis is the converse of condensation), while cyanogen bromide did some chemical modification onto the generated Oligomer 1 species.



A tripeptide is cleaved at position 1 either by hydrolysis (top) or by cyanogen bromide (bottom). Cyanogen bromide cleaves specifically on the right of a methionine monomer. Upon cleavage, the methionyl monomer gets converted into homoserine by the cyanogen bromide reagent

FIGURE 2.3: PROTEIN CLEAVAGE BY WATER AND CYANOGEN BROMIDE

Nonetheless, the reader might have noted that—interestingly—all the four oligomers do effectively have their left cap (the proton, making the N-terminal amino group) and their right cap (the hydroxyl, making the C-terminal carboxyl group). This means that in both water- and cyanogen bromide-mediated cleavages, all the generated oligomers are indeed true polymers in the sense that: 1) they are a chain of residues (modified or not) and 2) they are correctly capped (*i.e.* they are polymers in their finished polymerization state). This is important because it is the basis on which we shall make the difference between a cleavage process and a fragmentation process. Thus, our definition of a peptide might be: *a peptide is a protein (of at least one residue) in its finished polymerization state that was generated upon cleavage of a longer protein*. Of course, when we use the term “protein”, above, we mean “protein polymer”, irrespective of its size.

When the protein cleavage reaction precisely reverses the reaction that was performed for the same protein's biosynthesis, there is no special difficulty. But when the cleavage reaction modifies the substrate, then this should be carefully taken into account when using *X!TandemPipeline*. This is true for any chemical modification that happens onto a protein.

Well, all this sounds reasonable. But what about the “normal” case, when the cleavage is done using water? Nothing special: the mass of the oligomer is calculated by summing the mass of each monomer in the oligomer (since the monomers are not modified, this is easily done) and the masses corresponding to the left and right caps (these are defined in the polymer chemistry definition; in our present case it would be a proton on the left end, and a hydroxyl on the right end). In this way, the oligomer complies with its definition, which states that it is a faithful polymer made of monomers and that it is in its finished state.

Yes, but then how should one calculate the mass of the modified oligomer, like our Oligomer 1 in the case of the cyanogen bromide-mediated cleavage? Simple enough: in a first step it does exactly the same way as for the unmodified oligomer. Next, each oligomer is checked for presence or absence of a methionine residue on its right end. If a methionine is found, the mass corresponding to the “-C₁H₂S₁+O₁” chemical reaction is applied. And that's it.

2.1.2.2 PROTEIN FRAGMENTATION

In a fragmentation process, the bond that is broken does not necessarily yield smaller-sized “proteins” because fragmentation does not necessarily break the inter-residue bond the same way that the hydrolysis does. Indeed, fragmentations are oft-times high energy chemical processes that can affect peptidic bonds at different locations, not necessarily between the CO-NH bond of the peptidic bond. This is one of the reasons why fragmentations do differ from cleavages.

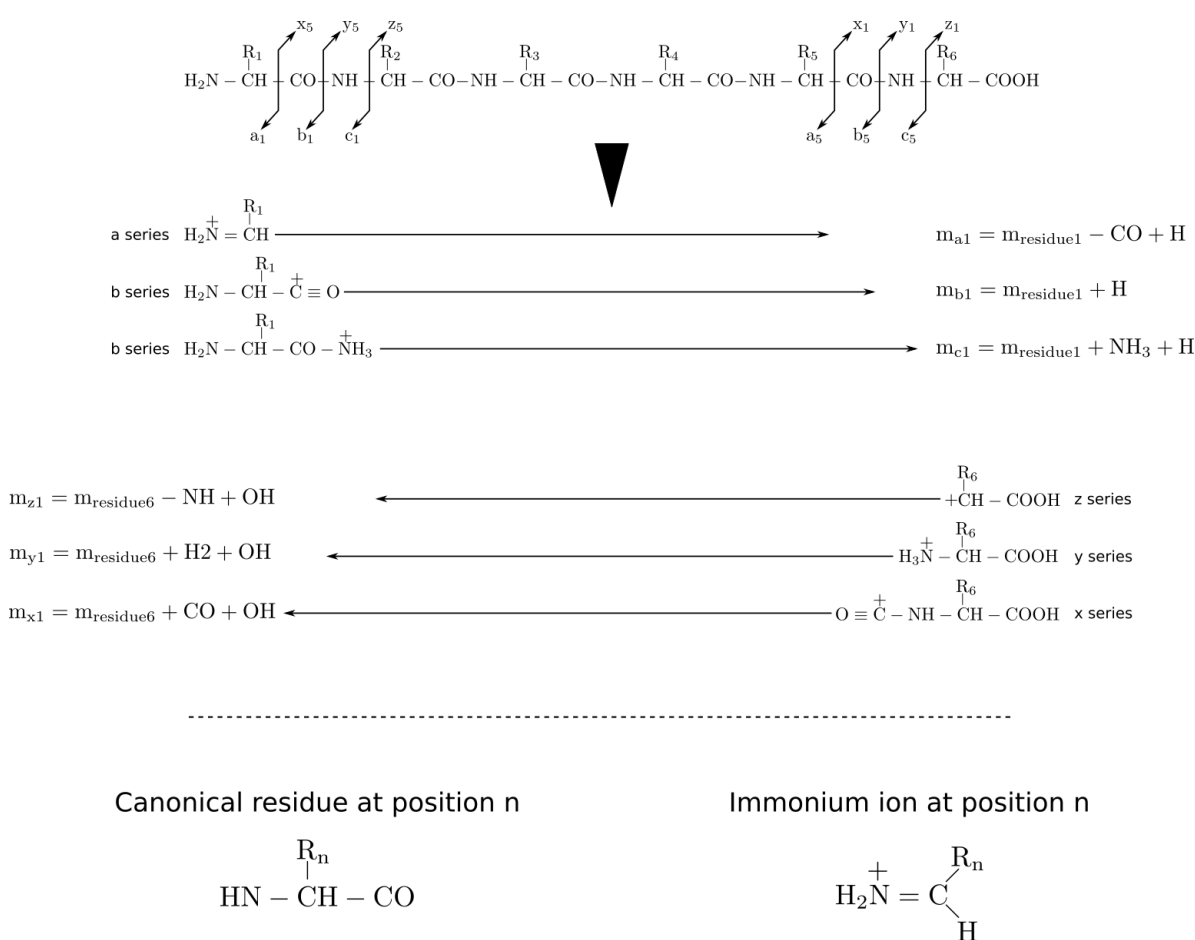
Another peculiarity of fragmentations, compared with cleavages, is the fact that there is no cleaving molecule starting the process, like water or cyanogen bromide, for example. Indeed, in the gas phase, the peptidic ions are “isolated”: that is, very far one from each other. A fragmentation process is often initiated by an intra molecular electron doublet rearrangement that propagates more or less in the polymer structure to eventually break it. Fragmentations are mainly a gas phase process, not some reaction that happens in solution as a result of putting in contact the polymer and some reagent. It is precisely because no cleaving molecule is involved in the fragmentation process that the obtained fragments are not necessarily capped like a normal polymer should be; and this is another really important difference between cleavage and fragmentation. The following examples should illustrate these concepts.



TIP

For the sake of completeness of this section, it must be noted that it is possible to have other “*chemical/physical* entities” intervene during the gas phase fragmentation process by enacting a chemical reaction, be these entities ions, electrons or photons. In bottom-up proteomics, the intervening molecules are gas molecules (nitrogen, most often, or helium) that act as physical entities imposing collisions to the peptidic ions with the effect that the ions acquire internal energy, eventually leading to dissociation (CID, for “collisionally-activated dissociation”).

There is a pretty important number of different kinds of fragments that can be generated upon fragmentation of peptides. We are going to detail the most common ones.



An hexapeptide is fragmented in the seven most widely encountered manners, such as to generate product ions of the a, b, c, x, y, z series and also immonium ions. The figure illustrates the position of the bond dissociation for each kind of fragment (exemplified using the case of the smallest fragment possible) and the mass calculation method is described for each fragment kind; consider that each fragment bears only *one positive* charge.

FIGURE 2.4: PROTEIN FRAGMENTATION PATTERNS MOST WIDELY ENCOUNTERED

As can be seen from [FIGURE 2.4, “PROTEIN FRAGMENTATION PATTERNS MOST WIDELY ENCOUNTERED”](#), the fragmentations do generate fragments of three categories: the ones that include the left end of the precursor polymer (a, b, c), the ones that include the right end of the precursor polymer (x, y, z), and finally the special case in which the fragment is an *internal fragment*, like the immonium ions. When looking at the fragmentations described in the figure, it becomes immediately clear why a fragmentation cannot be mistaken for a cleavage: the ionization of the fragment is not necessarily due to the captation of a proton by the fragment. Furthermore, we can also see that a fragmentation is not a cleavage because the fragment that is generated is *absolutely* not necessarily what we call a polymer, in the sense that the fragment might not be capped the same way as the precursor protein/peptide is (that is, the fragment is not in its finished polymerization state).

By looking at [FIGURE 2.4, “PROTEIN FRAGMENTATION PATTERNS MOST WIDELY ENCOUNTERED”](#), the reader should have noticed that the fragment naming scheme takes into consideration the fact that the fragment bears the N-terminal or C-terminal end of the precursor peptide (or none, also). Indeed, the numbering of fragments holding the N-terminal end of the precursor polymer sequence begins at the left end, and for fragments that hold the C-terminal end, at the right end. Thus the third fragment of series *a* (*a*₃) would involve monomers [1→] and the third fragment of series *y* (*y*₃) would involve monomers [6→] (see arrows in the figure).

2.2 GENERAL OVERVIEW OF BOTTOM-UP PROTEOMICS

Bottom-up proteomics is a field of endeavour where the ultimate goal is to identify the greatest number of proteins in a given sample. This goal might also, depending on the project at hand, be doubled with another goal: characterize at the finest level possible the nature and the position of post-translational/chemical modifications beared by the proteins.

To achieve the best results, proteomics has developed over the years a number of methods and techniques that, taken together, have allowed scientists to obtain impressive results of protein identification on pretty complex samples. These are listed below:

- *Mass spectrometers*: The development of mass spectrometers of ever-greater resolution power has allowed to attain at ever-lower false discovery rates over the years. In particular, the development of the Orbitrap analyzers, along with the huge improvements of the time-of-flight (TOF) mass analyzer technology, have strongly increased the identification results reliability by allowing the downstream data processing step to be more stringent in the protein identification task (see below);
- *Chromatography*: The development of highly resolute chromatography resins along with the elaboration of hardware (columns, chromatography setups) that yields sensitivity improvements have had their share in the way proteomics has evolved over the years;
- *Bioinformatics*: The development and refinement of software that can cope with extremely large data sets (think metaproteomics) is one major field that enabled significant advances in proteomics. Also, refinement of algorithms related to the simulation of isotopic clusters and comparison with experimental data have had their part. Likewise so for algorithms that detect the charge of ions based on the analysis of the isotopic cluster peaks. Being able to single out without error the monoisotopic peak of an isotopic cluster (whatever the ion charge or m/z ratio) is a big part of the successfully tackled challenges at the root of successful proteomics data processing.

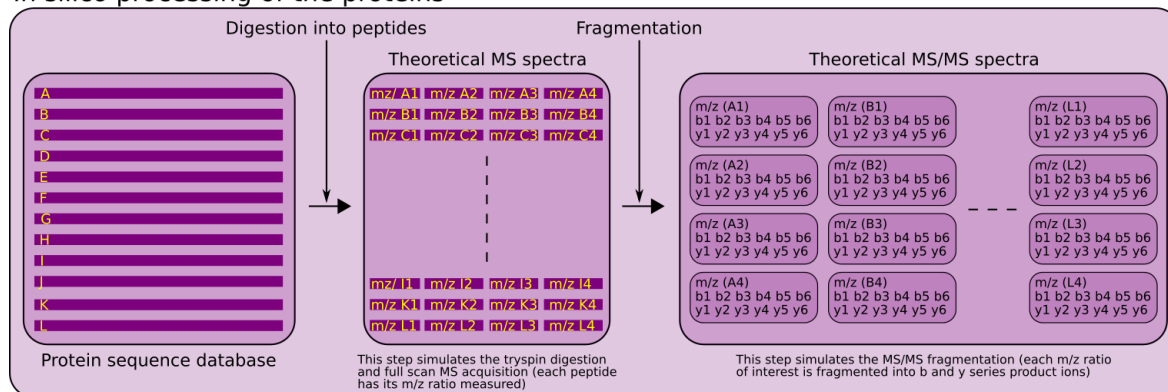
In this section, we will review the bioinformatics-based mass spectrometric data processing, as it is the core subject of this user manual. In particular, we will provide an outline of how the major software packages on the market perform protein identification on the basis of mass spectrometric analyses of biological samples.

This section will outline in not-so-rough terms how bottom-up proteomics works, from the protein sample to the protein identification list. The workflow comprises two sequential processes:

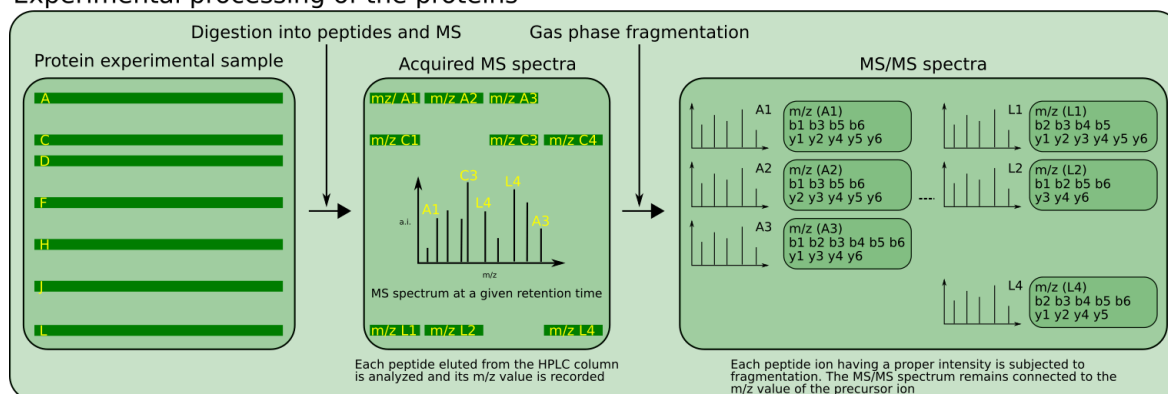
- *From the protein to the sequences of the peptides*: this initial part of the workflow is somehow doubled by having two parallel processes replicating it:
 - *In silico* process;
 - Experimental process.

These two processes are described in **FIGURE 2.5, “THEORETICAL AND EXPERIMENTAL PARALLEL DATA-PRODUCING PROCESSES”**.

In silico processing of the proteins



Experimental processing of the proteins



The digestion of the proteins, the analysis of the m/z of the peptides and the sequencing of the peptides are processes that exist both *in silico* and experimentally. This figure shows how the processes somehow mirror each other in the virtual and real contexts.

FIGURE 2.5: THEORETICAL AND EXPERIMENTAL PARALLEL DATA-PRODUCING PROCESSES

- **Database searching using experimental data:** this last part of the workflow is entirely based on bioinformatics software and involves the search for peptide *vs* mass spectrum matches and then a process called *protein inference* (see SECTION 2.2.5, “MATCHING FRAGMENTATION SPECTRA WITH THEORETICAL SPECTRA”).

2.2.1 THE FIRST STEP: DIGESTION OF THE SAMPLE'S PROTEINS

The very first step in the bottom-up proteomics workflow is to digest all the proteins in the initial biological sample with a site-specific endoprotease: typically trypsin.

The sample is subjected to proteolysis with all its proteins unresolved. This produces a highly complex mixture of peptides, each having a constant characteristic: each peptide has one predictable end (unless it is either the protein's N-terminal or the C-terminal peptide, as detailed below), either N-terminal or C-terminal:

- *Predictable N-terminus*: when the protease cuts at the N-terminal end of the target residue. For example, EndAspN cleaves left of Asp residues, thus producing peptides that always have Asp as their N-terminal residue. The only exception is when the peptide is the protein's N-terminal peptide and the first residue is not Asp);
- *Predictable C-terminus*: when the protease cuts at the C-terminal end of the target residue. For example, the most used enzyme, trypsin, cuts right of the basic residues Lys and Arg. The generated peptides thus necessarily end with one of these two residues. The only exception is when the peptide is the protein's C-terminal peptide and the last residue is not Lys nor Arg.



TIP

One interesting feature of trypsinolysis is that it generates peptides that—for their major part—will most probably be protonated twice: on their N-terminal end (the primary NH_2 amine group¹ and on the basic residual chain of the basic residue found at their C-terminal position (the ϵ -amine group for Lys and the guanidium group for Arg). Upon fragmentation of the peptide's precursor ion, both the left hand side fragment and the right hand side fragment will bear a proton and will thus be detected, thus potentially providing a better coverage of the peptide's sequence during the MS/MS experiment.

2.2.2 CHROMATOGRAPHIC SEPARATION OF THE PEPTIDIC MIXTURE

One major analytic step in bottom-up proteomics is the separation of the peptides obtained by endoproteolysis of all the proteins in the sample. Indeed, analyzing all the peptides in one single injection without any prior chromatographic separation would yield catastrophic results, similar to having injected nothing in the mass spectrometer.

The typical method for resolving peptides is by separating them on a chromatographic column functionalized with a hydrophobic group (for peptides, that would be a C_{18} reversed phase column).

The chromatographic gradient that will elute the peptides progressively according to their increasing hydrophobicity will be developed over the 5–95 % of acetonitrile (a non-protic organic solvent).



TIP

Using acetonitrile as the non-protic organic solvent has the huge benefit of not injecting protons inside the mass spectrometer as the chromatographic gradient develops.

¹ If not either converted to an amide group by acetylation or formylation or cyclised.

The eluate of the chromatographic column is directly injected into the mass spectrometer's source. The role of the mass spectrometer's source device is to ensure that the analytes are desolvated and ionized upon their entering in the core part of the mass spectrometer. Most often, that source is an electrospray source that is fed a liquid (typically, the eluate from the column). The source is designed to evaporate the solvent (analyte desolvation) and—having an electric potential applied to it—to help ionize the analytes (often the peptides are already ionized in solution, prior to desolvation). The electrically charged analytes in the gas phase are thus ions, the m/z (mass-to-charge) ratio of which can be measured by the mass spectrometer analyzer.



WARNING

There are two main sources used in the mass-spectrometry-for-biology specialty: the matrix-assisted laser desorption ionization (MALDI) source and the electrospray ionization (ESI) source. One important difference between the two is that the MALDI process mostly produces mono-charged ions ($[M+H]^+$), while the ESI process mostly produces multi-charged ions ($[M+nH]^{n+}$). This has huge implications in the mass data analysis.

The source that is mainly used in bottom-up proteomics is the ESI source.

2.2.3 MASS SPECTROMETRIC ANALYSIS OF THE PEPTIDES

Upon elution off the chromatographic column, the peptides are desolvated, ionized and drawn into the mass spectrometer using an electrical field. Once they have entered the mass spectrometer they are analyzed in the mass analyzer of the instrument.



NOTE

There are a variety of mass analyzers commonly used in bottom-up proteomics. In fact, one single instrument might have as many as 4 or 5 mass analyzers. However, not all the analyzers in the instrument are responsible for the m/z measurement.

Sometimes, during the whole cycle of the analysis, two different mass analyzers are used at different steps of the cycle: one analyzer selects the ion for fragmentation and another analyzer measures the m/z value of the fragments.

In bottom-up proteomics, two different kinds of mass spectrometric data are required—ideally, for each peptide eluted from the column—in order to effectively identify the proteins in the initial sample:

- The mass-to-charge ratio value (m/z) of the peptide ion;
- The m/z values of the fragments (the product ions) of the peptidic precursor ion that has undergone an MS/MS gas phase fragmentation².

These two kinds of data are necessary because the protein identification process is based on searches in protein databases using the precursor ions' m/z value and the m/z values of that ion's fragments when it is fragmented. The way the protein databases are used as the substrate of these searches is described in the next section.

2.2.4 THE PROTEIN DATABASES AND THEIR USE

The previous section ended on the idea that the protein identification process, that is based on the analysis of all the peptides of a peptidic mixture resulting from the endoproteolysis of a sample containing many proteins, requires searches into protein databases.

A bottom-up proteomics experiment typically needs at least one protein database: a database listing all the known proteins of the organism from which the initial sample of proteins was prepared. That organism might be a bacterium, a Eucaryote, like a fungus, a protist, a plant, a mammalian... Optional databases might be used, like protein databases listing all known protein contaminants, for example.

The protein databases are files in the following FASTA format:

```
>GRMZM2G009506_P01 NP_001149383 serine/threonine-protein kinase receptor
MEEQHMA GPPYRYRLQHRRLMDIAPASASDDSGHHG SNGMAIMVSILVVIVCTLFYCV
YCWRWRKRNAVRRAQIERLRPMSSDLPLMDLSSIHEATNSFSKENKLGE GFGPVYRGV
MGGGA EIAVKRLSARSRQGAAEFRNEVELIAKLQHRNLVRLLGCCVERDEKMLVYEYLPN
RSLDSFLFDSRKSGQLDWKTRQSI VLG IARGMLYLHEDSCLKVIHRDLKASNVLLDNRMN
PKISDFGMAKIFEEEGNEPNTGPVVGTYGYMAPEYAMEGVFSVKSDVFSFGVLVLEILSG
QRNGSMY LQEHQHTLIQDAWKLNEDRAAEFMDAALAGSYPRDEAWRCFHVGLLCVQESP
DLRPTMSSVVLMLISDQTAQQMPAPAQPPLFASSRLGRKASASDLSLAMKTETTKTQSVN
EVSISMMEPRFWADPGTSNGAATSHPATGACKKRGQGGDRNVKDGLAARTPTHQPVARW
HHDRRIVD
```

This format is really simple, because it only contains three information pieces, grouped in as many stanzas as there are proteins in the database:

² Most often, that fragmentation step is performed using collisionally-activated dissociation (CID). In this process, the peptidic precursor ion is first isolated in the gas phase on the basis of its m/z value and then is accelerated against a gas “fog” inside of the collision cell of the instrument. The ion hits gas molecules multiple times, acquires a lot of energy and finally breaks.

- The *unique* protein's accession id in the database (`GRMZM2G009506_P01`) that comes right after the '>' prompt that signals a new protein stanza;
- The protein description (`NP_001149383 serine/threonine-protein kinase receptor`) that provides some functional data bits for the protein at hand;
- The protein sequence (the rest of the stanza above).

The first (id) and second (description) information bits are used in various places in the *X!TandemPipeline* program.

The protein databases are used by the protein identification software as the very first step in a bottom-up proteomics data analysis process: the proteins in the database are digested *in silico* in order to produce a list of peptides that retain a connection to the protein from which they were generated. For each one of all these peptides, the following data bits are computed (FIGURE 2.5, “THEORETICAL AND EXPERIMENTAL PARALLEL DATA-PRODUCING PROCESSES”, top panel):

- *sequence*: The peptide's sequence;
- *m/z value*: The peptide's m/z value, often computed for the mono-protonated ($[M+H]^+$) ion;
- *MS/MS spectrum*: The peptide's fragmentation spectrum is nothing but an array of m/z values corresponding to the set of calculated fragments (of the b and y ion series). The m/z values of the product ions are crucial for the database search algorithm;

The next step is the establishment of a relation between the experimental MS/MS data acquired by the instrument and the theoretical MS/MS spectra computed from the protein sequences in the database. This next step is described in detail in the next sections.

2.2.5 MATCHING FRAGMENTATION SPECTRA WITH THEORETICAL SPECTRA

This section is about how the protein database searching software sets a relation between the experimental mass data and the theoretical mass data originating in the protein database. The elementary relation is between a given *experimental* MS/MS mass spectrum of a peptide's ion at a given m/z value and its *theoretical* counterpart from the database: when these two MS/MS spectra match at a sufficiently convincing level, then a “*peptide vs mass spectrum match*” was achieved (abbreviated name: PSM). The computing of a PSM is described in detail in FIGURE 2.6, “THE STEPS LEADING TO A SCORED PEPTIDE VS MASS SPECTRUM MATCH (PSM)”.

We have seen in SECTION 2.2.4, “THE PROTEIN DATABASES AND THEIR USE”, that two somehow similar processes are at the basis of the preparation of the data for the subsequent database searches. These processes were described in FIGURE 2.5, “THEORETICAL AND EXPERIMENTAL PARALLEL DATA-PRODUCING PROCESSES”.

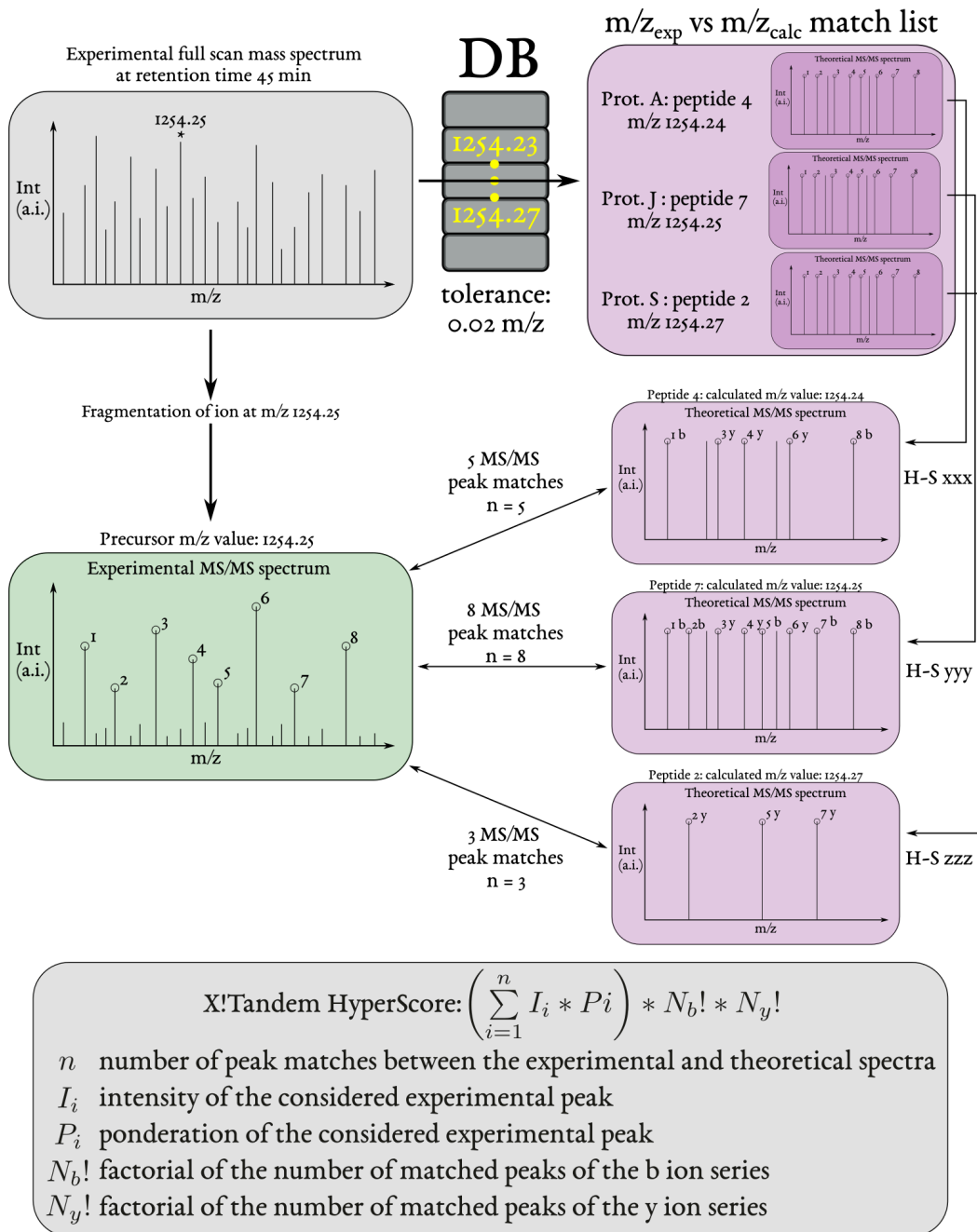
On the one hand (top panel, violet), the protein database is processed to digest *in silico* every protein it contains into a list of peptides. For each peptide arising from the digestion of a protein, the following data elements are recorded:

- The peptide's m/z value is computed. The association between that m/z value, the peptide and its originating protein is maintained;
- The peptide is fragmented into a list of peptidic fragments (product ions' m/z values, that is, the MS/MS spectrum; typically b and y ions series). The connection with the earlier data elements above is also maintained.

It is thus easy to determine the filiation between any given MS/MS theoretical mass spectrum, the precursor ion's m/z value, the peptidic sequence and, finally, the protein whence that peptide came.

On the other hand (bottom panel, green), the mass spectrometric data acquisition yields a huge set of the following pairs of data elements that are recorded over time:

- The m/z value of the peptidic precursor ion undergoing fragmentation (keeping a connection with the retention time at which it is recorded);
- The list of peptidic fragments (product ions' m/z values, that is, the MS/MS spectrum). The connection with the precursor ions' m/z value and with the retention time is maintained.



H-S yyy >> H-S xxx >> H-S zzz

The process starts with a full scan mass spectrum from which the mass spectrometer selects one precursor ion at a definite m/z value. That ion is fragmented and thus generates a MS/MS spectrum. During the data exploration, the software extracts from the database all the peptides having the same m/z value as that of the fragmented ion (top right, violet background). Next, the experimental MS/MS spectrum is compared in turn to each one of the MS/MS spectra of the extracted peptide list. A HyperScore is computed at each comparison. Because *X!TandemPipeline* uses *X!Tandem* as its preferred protein database search engine, the HyperScore calculation, as performed by *X!Tandem*, is described.

FIGURE 2.6: THE STEPS LEADING TO A SCORED PEPTIDE VS MASS SPECTRUM MATCH (PSM)

Once the acquisition of the experimental data is complete, the analysis of these data involves going through all the fragmentation data of the acquisition and performing these steps for *each* MS/MS spectrum (as evidenced in [FIGURE 2.6, “THE STEPS LEADING TO A SCORED PEPTIDE VS MASS SPECTRUM MATCH \(PSM\)”](#)):

- Get the precursor ion's m/z value;
- Compute the match m/z range. For example, if the software is configured with a m/z tolerance for the m/z matches set to 0.02 and the precursor ion's m/z value is 1254.25, then the match m/z range would be [1254.23–1254.27];
- Construct a list of all the peptides in the database that have their m/z value contained in the match m/z range;
- For *each* peptide in the list returned from the database, compare its theoretical MS/MS spectrum with the experimental one. Compute a HyperScore for comparison.

2.2.5.1 COMPUTATION OF THE PSM HYPERSCORE

Of course, it is extremely rare that an experimental MS/MS spectrum matches fragment-by-fragment an identical theoretical spectrum. Most often, some theoretical product ions (MS/MS spectrum peaks) are missing from the experimental fragmentation spectrum. Also, there will almost certainly be dozens (if not hundreds) of peptides having a m/z value in the searched m/z range. Most certainly, the vast majority of these peptides are not of the right sequence (that is, do not have their MS/MS theoretical mass spectrum matching the experimental one). To make without any human scrutiny of the matches, it is necessary to compute a score that somehow assesses the extent to which both the experimental and theoretical MS/MS spectra match. That score, in *X!Tandem*, is called *HyperScore* and is described at the bottom of the figure.

The HyperScore computation process is relatively straightforward. First off, it is necessary to stress the fact that a HyperScore is computed each time an *experimental* MS/MS spectrum is compared to a theoretical (*calculated*) MS/MS spectrum (see *m/z_{exp} vs m/z_{calc} match list* in [FIGURE 2.6, “THE STEPS LEADING TO A SCORED PEPTIDE VS MASS SPECTRUM MATCH \(PSM\)”](#)).

In the example, three peptides from the database have their m/z value matching the searched m/z range (the m/z value of the precursor ion with accounting for the tolerance). So, the program checks the similarity between the experimental MS/MS spectrum and each one of the three theoretical ones. Each similarity test is associated to a HyperScore value.

The HyperScore is computed by summing—for each tested fragment peak *in the theoretical MS/MS spectrum*—the product of two variables described below. Once that sum is computed, it is compounded by two factorial numbers also described below:

- I_i : the intensity of the matching mass peak in the experimental MS/MS spectrum (if found);
- P_i : the ponderation factor of the matching mass peak in the experimental MS/MS spectrum. That variable can take a number of values, depending on the presence or not of this fragment peak in the experimental MS/MS spectrum (if not found, then P_i is naught and the peak is disregarded entirely). There are other values greater than naught, accounting for the physico-chemical properties of the peptidic bond that was cleaved to obtain that fragment (presence of proline will lower the P value, for example).
Intuitively, the HyperScore will end up larger if there are a lot of fragment peaks in the theoretical MS/MS spectrum that are matched with experimental ones (each P_i value compounded by the I_i value is being summed into the HyperScore final value).
- $N_b!$: the sum computed above is then compounded by the factorial of the number of ions of the b ion series that are found in the experimental MS/MS spectrum;
- $N_y!$: the product computed at the previous step is then compounded by factorial of the number of ions of the y ion series that are found in the experimental MS/MS spectrum.

This last compounding operation terminates the computation of the HyperScore value.

It is apparent now that the HyperScore value will tend to be greater if there are numerous fragment peaks in the theoretical MS/MS spectrum that are matched by fragment peaks in the experimental MS/MS spectrum. Also, the score value is incremented if the intensity of the matching peaks is greater and if the number of matching peaks of the two b and y ions series is greater.

This, however, cannot be all of it, because the HyperScore does not really answers the question: “*what are—if any—, of all the PSMs found for a given experimental MS/MS spectrum, the one (or ones) that we can faithfully tell as true match(es)?*”. To answer that question, some more computational steps need to be carried over, that should lead to a numerical value that is truly indicative of the confidence we may have that a given PSM is a *real* match. In *X!Tandem*, that numerical value is called *expectation value* (abbreviation: *E-value*). We describe the whole process of its computation below.

2.2.5.2 COMPUTATION OF THE PEPTIDE EXPECTATION VALUE (E-VALUE)

First of all, it needs stating that we describe the *peptide E-value*, not the protein E-value. A peptide E-value is obtained for a single experimental MS/MS spectrum. It is computed by looking into the HyperScore values obtained for all the MS/MS spectra comparisons described at the previous section. The HyperScore values (for example, the three values denoted *H-S xxx*, *H-S yyy* and *H-S zzz* in [FIGURE 2.6, “THE STEPS LEADING TO A SCORED PEPTIDE VS MASS SPECTRUM MATCH \(PSM\)”](#)) are used to perform the E-value computation. In the following text, we’ll assume that there are many more PSMs than these three, for a given experimental MS/MS spectrum (which is actually the reality, with hundreds of peptides in the database that match a given searched m/z range). As illustrated in [FIGURE 2.7, “COMPUTATION OF A PEPTIDIC EXPECTATION VALUE \(E-VALUE\)”](#), a histogram is crafted

plotting the count of MS/MS spectral pair comparisons (let us call them “wannabe PSMs”) against a number of HyperScore bins. This histogram is a good representation of the distribution of the HyperScore values among the various peptides in the m/z value-matching list (see previous section). In this example, the very best HyperScore value is 82 and the number of PSMs having that score is obviously very low! Instead, the distribution clearly shows that there are a vast majority of wannabe PSMs that have very low HyperScore values and that will not ultimately be considered as real PSMs.

In order to be able to use the distribution pattern further, the second half of the distribution’s main peak is replotted by computing the natural logarithm of the count of MS/MS spectral pair comparisons, still against the HyperScore value bins. The new plot is easily fitted into a line, of which the equation is computed.

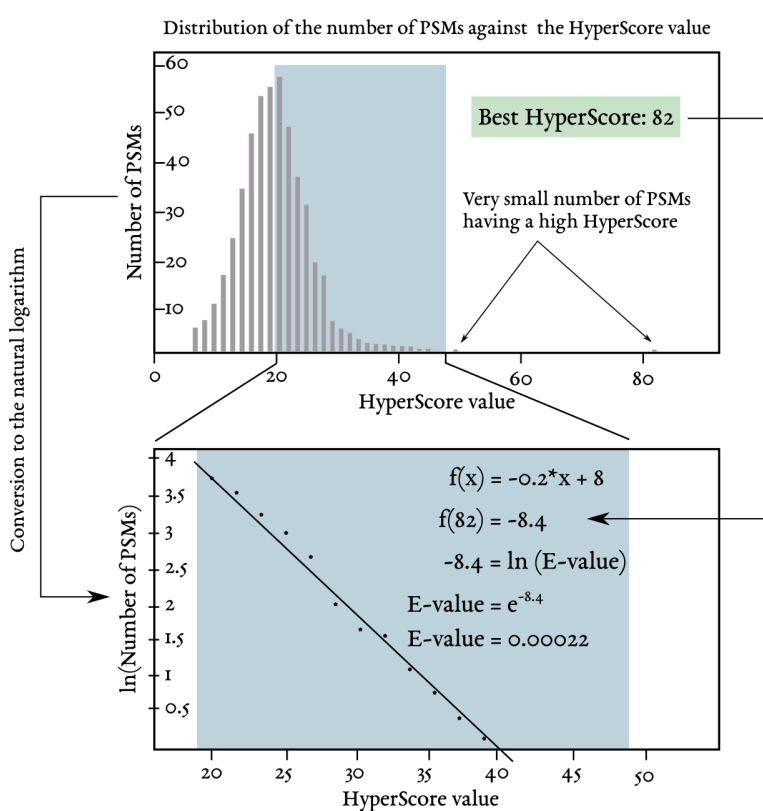
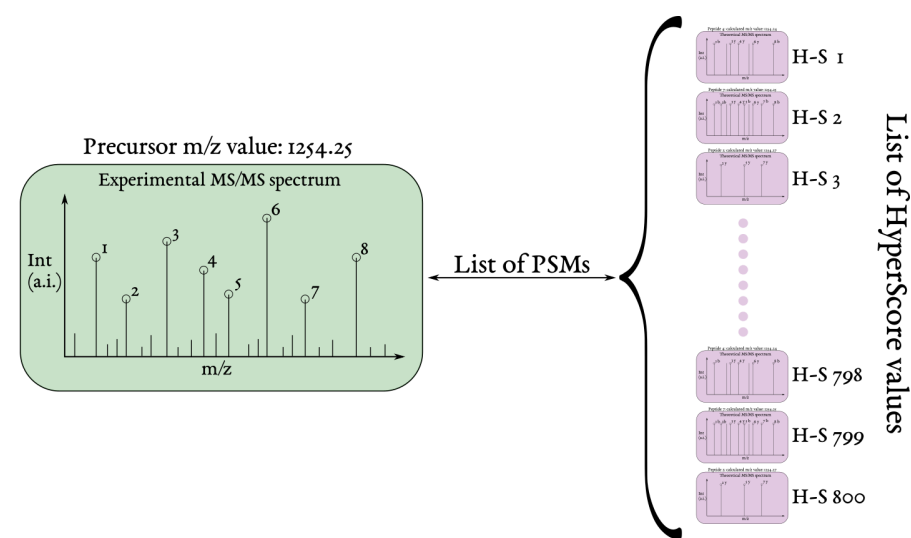
The best HyperScore value (82, in the example) is then used in the line equation to compute the corresponding ordinate (the natural logarithm of the PSMs count having that HyperScore). That value (-8.4, in the example) corresponds to the natural logarithm of the expectation value (E-value). By using the exponential function, the E-value is thus computed to be 0.00022, which a pretty low number. Since the E-value somehow gives an idea that a given PSM was obtained by chance, the very small obtained result shows that the match almost certainly was a faithful one.



NOTE

The expectation value is defined as the probability that the peptide sequence would match an experimental tandem mass spectrum by chance, if the trial is repeated many times. For example, if the E-value is found to be 1, then that means that the match can occur by chance or not with an equal probability. Instead, if the E-value is found to be 0.01, then that means that there is one event over 100 trials that the match has occurred by chance.

The smaller the E-value, the more confidence one has that the match is correct and that the PSM is a faithful one.



For each experimental MS/MS spectrum, gather all the peptides in the database that have a m/z value matching the precursor ion's m/z value. For each peptide sequence, compute the HyperScore. With all the HyperScore values, go on with the calculation of the expectation value for the peptide set. The peptidic E-value should be the smallest possible, as it is an indication of the possibility that the match between the experimental MS/MS spectrum and the theoretical mass spectrum occurred by chance.

FIGURE 2.7: COMPUTATION OF A PEPTIDIC EXPECTATION VALUE (E-VALUE)



TIP

The user configures the software to only consider PSMs if their peptidic E-value is below a given threshold. Typically, that threshold is given a value of 0.05 (FIGURE 3.5, “CONFIGURATION OF THE LOADING OF THE IDENTIFICATION RESULTS”).

When a reliable match between an experimental MS/MS spectrum and a theoretical MS/MS spectrum is found (that is, a true PSM), the software reports the following set of data elements:

- *m/z*: the *m/z* value of the precursor peptidic ion that underwent fragmentation;
- *sequence*: the sequence of the peptide that was matched in the present PSM;
- *protein name*: the protein accession number that produced the matched peptide upon enzymatic digestion of the sample;
- *E-value*: the peptide expectation value, as described above.

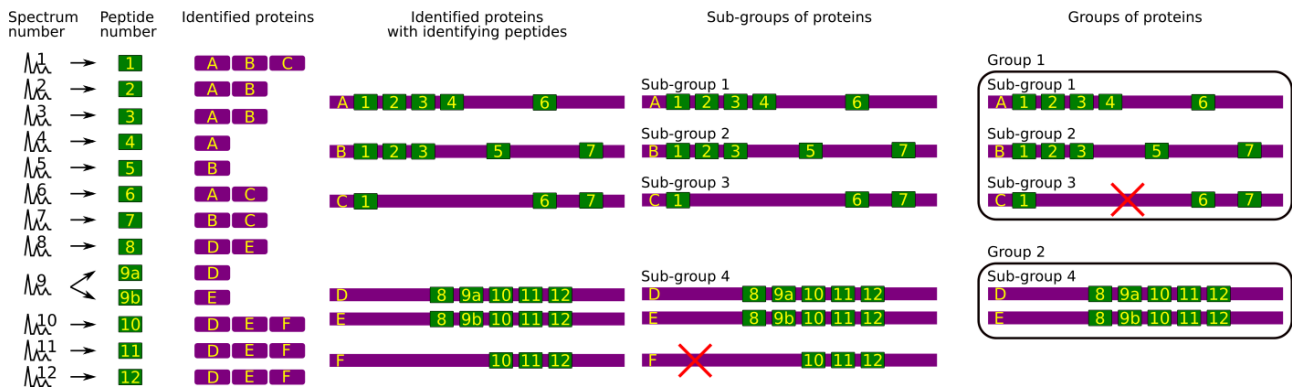
2.2.5.3 COMPUTATION OF THE PROTEIN EXPECTATION VALUE (E-VALUE)

The last step in the computation of values that help the software and the user determine if identifications are faithful (for peptides and for proteins) is the computation of the protein expected value. This value is very easily computed: it is the product of the E-values of all the peptides that participated in the identification of the protein. By necessity, then, the protein E-value will be less than the threshold peptide E-value (since that last value is below 1). By default, the protein E-value is set to 0.01 (FIGURE 3.5, “CONFIGURATION OF THE LOADING OF THE IDENTIFICATION RESULTS”).

2.2.5.4 PROTEIN INFERENCE: FROM PSMs TO PROTEIN IDENTITIES

One remaining critical question is: “— *How is the list of protein identifications returned by the database searching software verified and modified?*” Indeed, there are a number of situation where the proteomics data user may want to tweak the identification results. But also, the protein identification list returned by the database software may not be as perfect as one would expect. Bioinformaticians working in proteomics have come up with a number of algorithms to better the reliability of the identification results returned by database searching software.

In *X!TandemPipeline* we use an algorithm that is impinged on the concept of *parsimony*. That algorithm is detailed in an article describing *X!TandemPipeline* that was published in *The Journal of Proteome Research* in 2017 by Olivier Langella and Colleagues. The general concepts are presented here for the sake of completeness of this user manual.



The process of establishing a consolidated protein identity list from the results reported by the database searching software is illustrated (see text).

FIGURE 2.8: PROTEIN INFERENCE: CONSTRUCTING A CONSOLIDATED PROTEIN IDENTIFICATIONS LIST

The protein inference process, depicted in **FIGURE 2.8, “PROTEIN INFERENCE: CONSTRUCTING A CONSOLIDATED PROTEIN IDENTIFICATIONS LIST”**, is a multi-step one. The starting point is the huge list of PSMs that are reported by the database searching software. These PSMs are displayed in the figure as the two columns on the left hand side: one *experimental* MS/MS spectrum (*Spectrum number*) has provided a convincing PSM and thus allowed the identification of a peptide (MS/MS $i \rightarrow$ Pep i , *Peptide number*). Of course, a given peptide (Pep i) might have allowed the identification of multiple proteins (for example, homologous proteins that share the same peptidic sequence). Thus, Pep i is found in proteins A, B and C (column *Identified proteins*). The structure of the identified proteins can thus be partially reconstructed, and that is shown in column *Identified proteins with identifying peptides*. All the other PSMs are listed below that first one.

The general concept of the algorithm is that, by going through all the PSM data it is possible to check if some form of degraded redundancy allows pruning off some proteins from the list. This pruning off of some proteins is meant to increase the confidence that the identifications are reliable. That might be at the cost of having a smaller number of identified proteins, but with an improved false discovery rate (that is, a reduced FDR). As described below, the pruning off of proteins from the protein identifications list occurs at two different steps in the inference process.

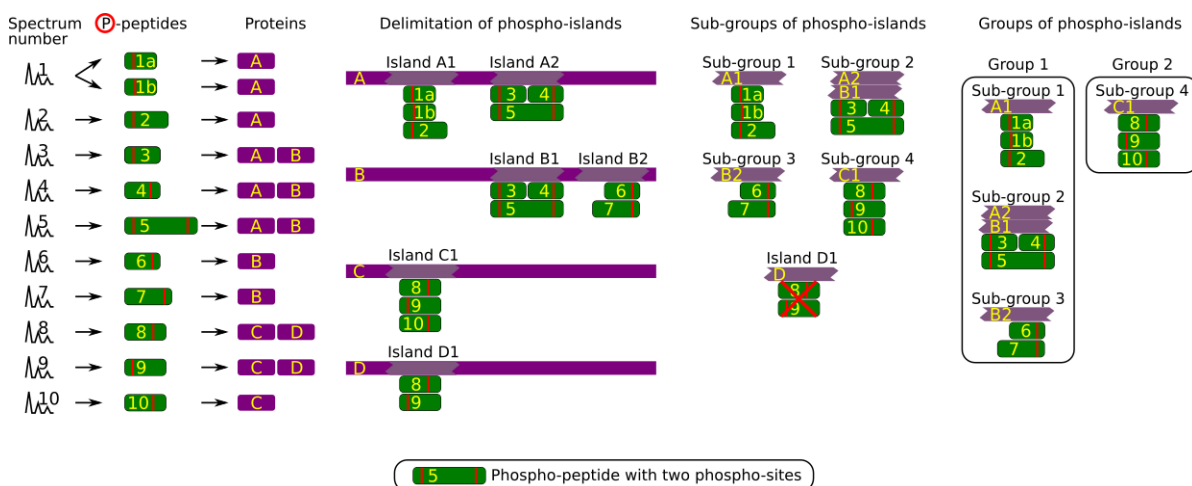
The first step is the creation of sub-groups of the identified proteins. In this step, all the proteins that could be identified thanks to the exactly same set of peptides are gathered into a sub-group. In the example, the sub-group that contains more than one protein happens to be sub-group 4. Note how protein F in this sub-group is identified by a set of three peptides. This is two peptides less than the number of peptides that identified the other two proteins (D and E) in the sub-group. The principle of parcimony allows thus to remove Protein F as that protein is not justified *per se*, that is, it is unnecessary to explain the presence of the three peptides.

The second step is the creation of groups that gather all the sub-groups that share at least one peptide. Thus, group 1 contains sub-groups 1, 2 and 3, while group 2 contains the sub-group 4. According to exactly the same philosophy as for the previous step, the sub-groups that contain proteins identified only by peptides also shared by proteins present in other sub-groups are pruned off.

The whole process described here is dubbed “*protein grouping*” in the *X!TandemPipeline* language. The output of this protein grouping process is displayed in the protein identification window, to be described below.

2.2.6 PHOSPHO-PROTEOMICS

An analogous algorithm as the one used for protein inference is at play when *X!TandemPipeline* is handling phospho-proteomics data. That algorithm is described below and in **FIGURE 2.9, “PHOSPHO-SITE INFERENCE: CONSTRUCTING A CONSOLIDATED PHOSPHO-SITE LIST”**.



The process of establishing a consolidated phospho-site list from the results reported by the database searching software is illustrated (see text).

FIGURE 2.9: PHOSPHO-SITE INFERENCE: CONSTRUCTING A CONSOLIDATED PHOSPHO-SITE LIST

The phospho-island inference process, depicted in **FIGURE 2.9, “PHOSPHO-SITE INFERENCE: CONSTRUCTING A CONSOLIDATED PHOSPHO-SITE LIST”**, is a multi-step one, most similarly to what was described in **SECTION 2.2.5.4, “PROTEIN INFERENCE: FROM PSMs TO PROTEIN IDENTITIES”**. The starting point is the list of peptides that were identified and determined to bear one or more phospho-sites (thus called phospho-peptides; see the red vertical bar in the figure). Two difficulties here are, on the one hand, the fact that phospho-sites may be shared by more than one peptide and, on the other hand, the fact that more than one phospho-site might be determined on the *same* peptide. These are the reasons that the concept of *phospho-island* was elaborated: it is a protein region that bears at least one phospho-site, in turn beared by one or several overlapping phospho-peptides. It is important to note that the position and number of phospho-sites are not necessarily the same in all of the overlapping phospho-peptides.

In this inference process, the analogy with the previously described one is the following:

- Peptides are replaced by phospho-peptides;
- Proteins are replaced by phospho-islands.

In the first step, the phospho-islands are delimited on the phosphorylated proteins. In the second step, sub-groups of phospho-islands are created using all the phospho-islands identified in different proteins and that share exactly the same set of phospho-peptides. At this step, any remaining phospho-island defined by a subset of phospho-peptides only partially defining a sub-group is disregarded. In the example, phospho-island D₁ is defined by two phospho-peptides, 8 and 9, that also are part of a sub-group defined by these two peptides but also by phospho-peptide 10. Phospho-island D₁ is thus disregarded.

In the third step, all the sub-groups that contain phospho-islands beared by the same protein are gathered in a group.

3 THE MAIN PROGRAM WINDOW

Proteomics data explorations, with *X!TandemPipeline*, entail, for a large part, the following steps:

- Configuration of the *X!Tandem* external software that runs the database searches (producing peptide vs mass spectrum matches—PSMs—, leading to the peptide identifications and ultimately to protein identifications);
- Configuration of the protein database files (both the organism-specific protein databases and optional contaminant-containing databases);
- Loading of the mass spectrometry data acquisition files (the mzML format is recommended);
- Running *X!Tandem* from inside of *X!TandemPipeline*;
- Loading of the identification results produced during the previous step;



NOTE

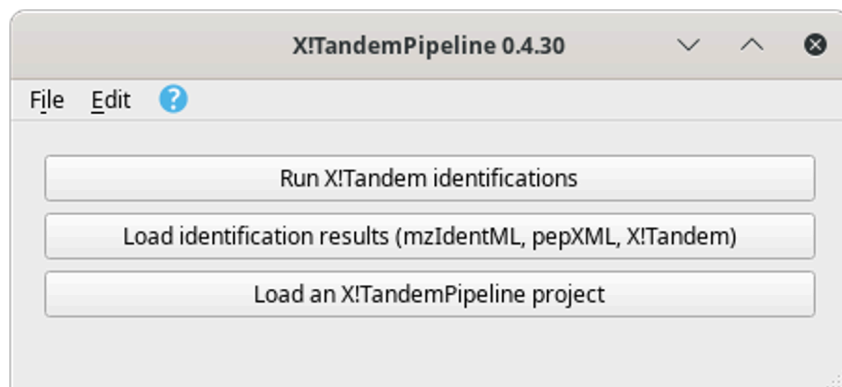
X!TandemPipeline can also handle peptide vs spectrum matches data (peptide identification data) from other software with the following formats:

- mzIdentML;
 - pepXML;
 - Mascot DAT files
-
- Relentless scrutiny of the peptide identification results. Optional modification of the results;
 - Protein inference, that is, protein identification on the basis of the peptide identifications. *X!TandemPipeline* implements a protein grouping algorithm, as described in [FIGURE 2.8, “PROTEIN INFERENCE: CONSTRUCTING A CONSOLIDATED PROTEIN IDENTIFICATIONS LIST”](#), that leads to consolidated protein identifications. The program has an interface geared towards the tweaking of the protein grouping process so as to let the user in full control of the stringency with which the protein identifications list is ultimately generated.

In this chapter, *X!TandemPipeline*'s main window's user interface is described in detail, in particular in the way it is a starting point for the main tasks briefly mentioned above.

3.1 STARTING A NEW *X!TandemPipeline* WORKING SESSION

To start a session, run *X!TandemPipeline* and the main program window shows up as described in [FIGURE 3.1](#), “MAIN PROGRAM WINDOW”.



The main program window contains three buttons described in detail in the text.

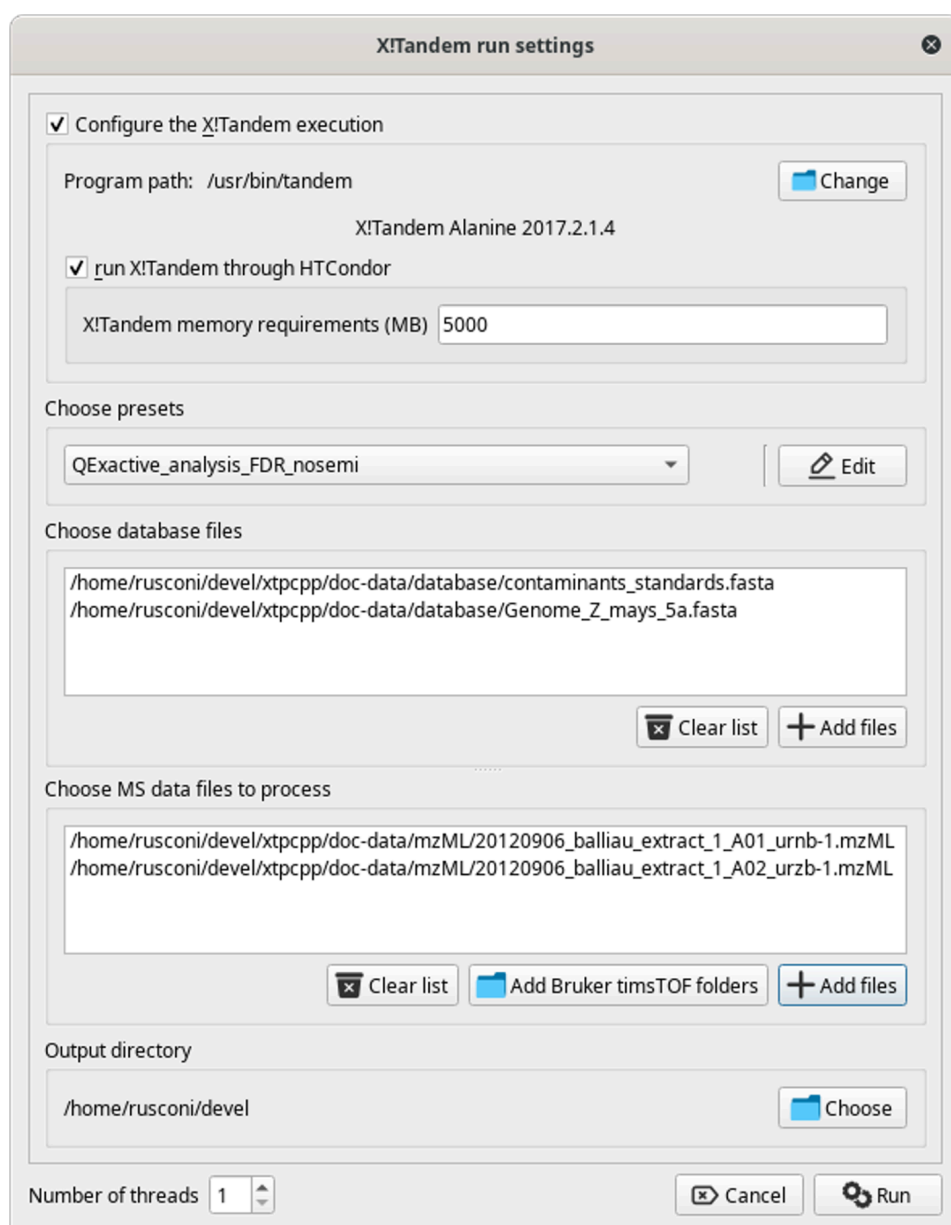
FIGURE 3.1: MAIN PROGRAM WINDOW

The main program window contains three buttons that start the following main tasks:

- *Run X!Tandem identifications*. See [SECTION 3.2](#), “RUNNING X!TANDEM IDENTIFICATIONS”.
- *Load identification results (mzIdentML, pepXML, Mascot, X!Tandem)*. See [SECTION 3.4](#), “LOADING IDENTIFICATION RESULTS”.
- *Load an X!TandemPipeline project*. See [SECTION 3.5](#), “LOADING X!TANDEMPipeline PROJECTS”.

3.2 RUNNING *X!Tandem* IDENTIFICATIONS

To run *X!Tandem*-based identifications, click onto the *Run X!Tandem identifications* button. This triggers the opening of the window pictured in [FIGURE 3.2](#), “X!TANDEM-BASED IDENTIFICATION CONFIGURATION”.



The configuration of a *X!Tandem* run is performed in this configuration window (see text for details).

FIGURE 3.2: **X!TANDEM-BASED IDENTIFICATION CONFIGURATION**

The configuration of an *X!Tandem* run entails defining the following:

- *Configure the X!Tandem execution*: This setting allows one to specify the path to the *X!Tandem* software program. The version of the program, if found, is displayed below (in this case, *Alanine 2017.2.1.4*). This feature is useful when the user wants to test multiple versions of the *X!Tandem* software.
- *Run X!Tandem through HTCondor*: This setting is useful when running *X!Tandem* over the network on a server supporting HTCondor¹.

¹ See [HTTPS://RESEARCH.CS.WISC.EDU/HTCONDOR/](https://research.cs.wisc.edu/htcondor/) .

- *Choose presets*: This setting defines the parameters that *X!Tandem* must use. Either load already known presets from the drop-down list widget or edit them (or create a new set) by clicking onto the *Edit* button. Note that to load an existing presets file, it might be necessary to point *X!TandemPipeline* to the directory that contains the presets file. Use the folder icon for this.
- *Choose database files*: Add protein database files in the FASTA format. There must be at least one protein database that contains all the known proteins for the organism of interest (there might be as many such database files as necessary) and optionally protein databases containing known contaminant proteins (there might be as many such database files as necessary). Click onto the *Clear list* button to clear the database files list and start anew if an error occurred (it is not possible to remove files one at a time).
- *Choose MS data files to process*: Add the mass spectrometry data files (mzML or mzXML format) to be processed by the *X!Tandem* software. As many files as necessary might be added in the list.



TIP

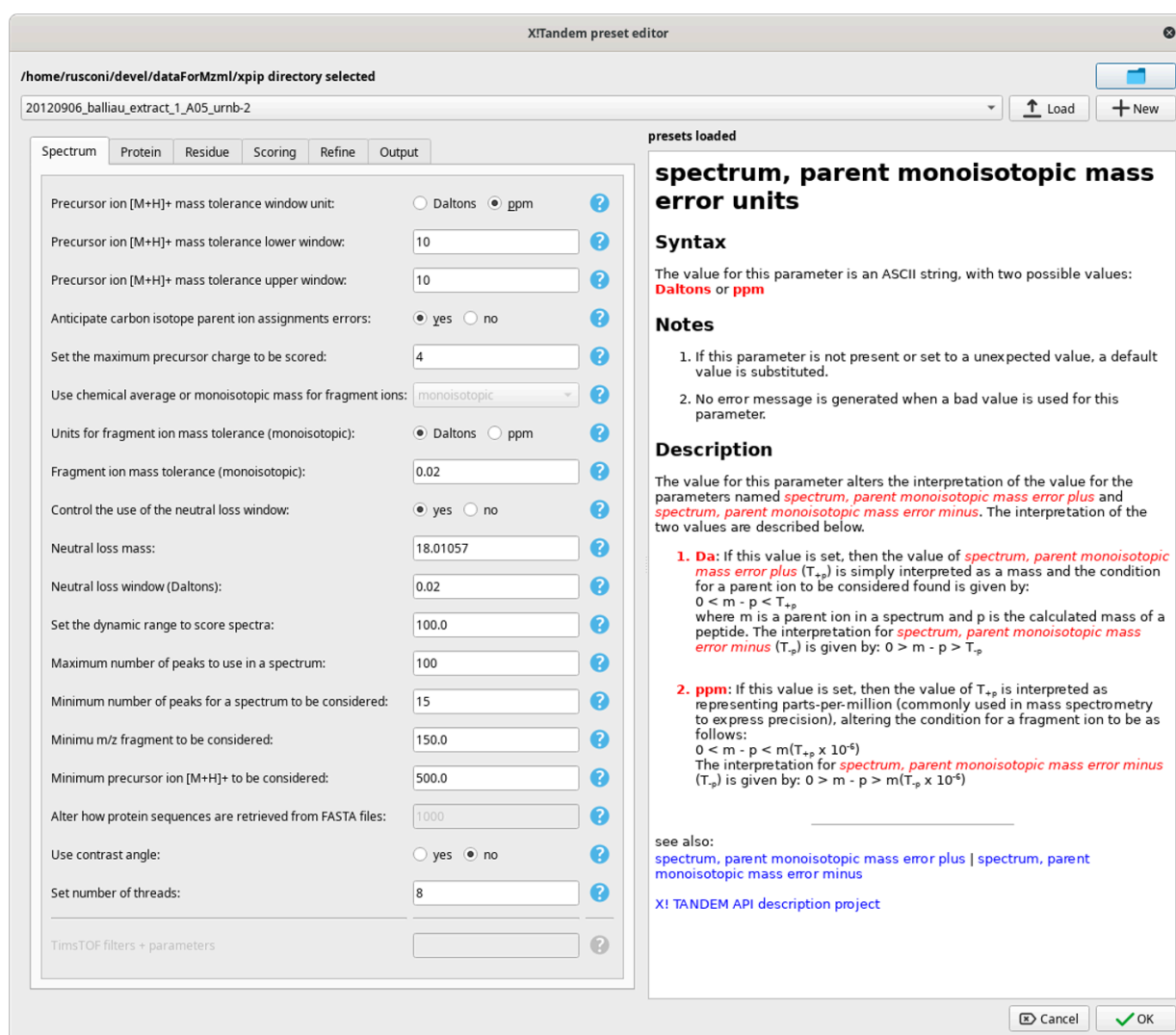
When using Bruker timsTOF data, click onto the *Add Bruker timsTOF folders* button to select folders containing this kind of data. Bruker timsTOF data come as two files that must sit in the same directory.

- *Output directory*: This setting specifies the directory into which new files output by the *X!Tandem* process need to be created. *X!Tandem* produces identification results in files in an XML format that *X!TandemPipeline* reads during a later step.
- *Number of threads*: This setting defines the maximum number of execution threads that *X!Tandem* might be using during its run.

3.3 SETTING THE *X!Tandem* RUN PRESETS

The *Edit* button of the *Choose presets* group box described above triggers the opening of a dialog window where the user might configure in the most detailed way the *X!Tandem* parameters. That dialog window is pictured in

FIGURE 3.3, “*X!TANDEM PRESETS CONFIGURATION WINDOW*”.



The configuration of the *X!Tandem* presets is performed in this configuration window. Each parameter is associated to a manual page that can be displayed by clicking on the interrogation mark button next to it. It is possible to load existing presets from file or to create brand new ones.

FIGURE 3.3: X!TANDEM PRESETS CONFIGURATION WINDOW

3.3.1 LOADING EXISTING PRESETS CONFIGURATIONS FROM FILE

It is possible to load existing *X!Tandem* presets configurations (which is useful in particular if the samples most often come from the same instrument using the same configuration) by first pointing *X!TandemPipeline* to the right directory that contains the presets configuration file of interest (folder icon in **FIGURE 3.3, “X!TANDEM PRESETS CONFIGURATION WINDOW”**). The presets configuration files in the chosen directory are automatically detected and listed in the drop-down list widget. At this point, select from that list the file of interest and click onto the *Load* button.



WARNING

It is compulsory to click onto the *Load* button to confirm loading of the file contents, because these are not updated solely upon choosing the file name from the drop-down list.

3.3.2 CREATING NEW PRESETS CONFIGURATIONS

It is possible to create a new presets configuration in a new file by clicking onto the *New* button. This opens an input dialog window for the user to define a new file name (the edit widget is preset with the currently loaded file's name suffixed with *_copy*).



TIP

One interesting feature of the new presets configuration creation process is that, if presets are already loaded, *X!TandemPipeline* copies the currently displayed settings to the new file. From there, it is possible to create a variant *X!Tandem* presets configuration, which eases the exploration of the right *X!Tandem* parameters for a given sample data set.

3.3.3 ACTUAL *X!Tandem* PRESETS CONFIGURATION

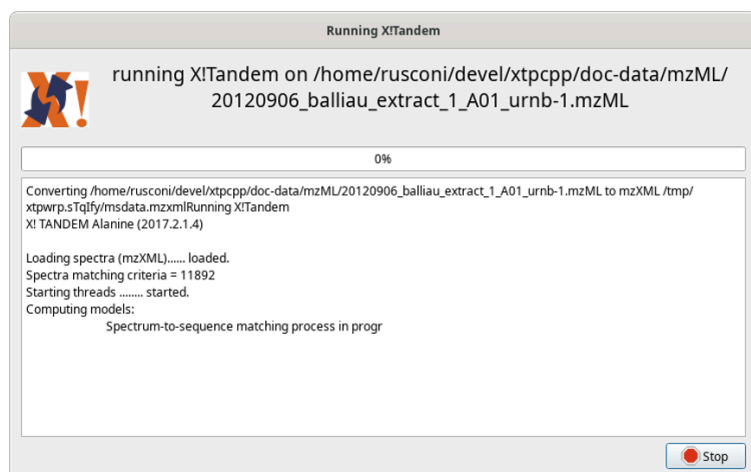
The dialog window pictured in [FIGURE 3.3, “X!TANDEM PRESETS CONFIGURATION WINDOW”](#) contains a number of tabs where various aspects of the *X!Tandem* run settings are handled. Each parameter's documentation can be seen on the pane on the right hand side of the window by clicking onto the question mark button next to it. These manual pages are authoritative because they are taken from the *X!Tandem* software package with no transformation whatsoever.

Once the configuration has been performed, click onto the *OK* button. If the parameters were modified, *X!TandemPipeline* asks if they should be stored in the file.

3.3.4 RUNNING A PROPERLY CONFIGURED *X!Tandem* PROCESS

Once the *X!Tandem* settings configuration dialog window has been closed, it is possible to run *X!Tandem* from inside *X!TandemPipeline* by clicking onto the *Run* button at the bottom of the window pictured in [FIGURE 3.2, “X!TANDEM-BASED IDENTIFICATION CONFIGURATION”](#).

While the computation is carried over, the program shows the feedback dialog window pictured in [FIGURE 3.4, “X!TANDEM PRESETS CONFIGURATION WINDOW”](#).



The text in this feedback dialog window is getting incrementally printed all along the computation.

FIGURE 3.4: *X!TANDEM* PRESETS CONFIGURATION WINDOW

Once the computation is finished, the feedback dialog window closes and the user is returned to the main program window (FIGURE 3.1, “MAIN PROGRAM WINDOW”). From there, it is possible to open the *X!Tandem* results file located in the output directory configured above. There are as many output files (XML-based format, and xml extension) as there were mass spectrometry data files to process. The loading of the results files is described in SECTION 3.4, “LOADING IDENTIFICATION RESULTS”.

3.4 LOADING IDENTIFICATION RESULTS

The loading of identification results comes with a minimal set of configuration required to instruct *X!Tandem-Pipeline* on the way to handle contaminant proteins, for example. This process is pictured in FIGURE 3.5, “CONFIGURATION OF THE LOADING OF THE IDENTIFICATION RESULTS” and is described in the following section.

Load identification results

Results handling mode

☒ Combine ☐ Individual

Choose result files

/home/rusconi/devel/xtpcpp/doc-data/output.d/20120906_balliau_extract_1_A01_urnb-1.xml
/home/rusconi/devel/xtpcpp/doc-data/output.d/20120906_balliau_extract_1_A02_urzb-1.xml

Number of files: 2

Contaminants

☒ Contaminants file ☐ Contaminant regular expression

contaminants_standards.fasta

Contaminant removal mode

☐ Protein list ☒ Groups

Peptide and protein filters

Peptide threshold on: ☒ Evalue ☐ FDR

Peptide Evalue: 0.050000

Peptide FDR: 1.0%

Number of peptides per protein: 2

Overall samples: ☒

Protein Evalue: 0.01000000

Protein Evalue (log10): -2.00

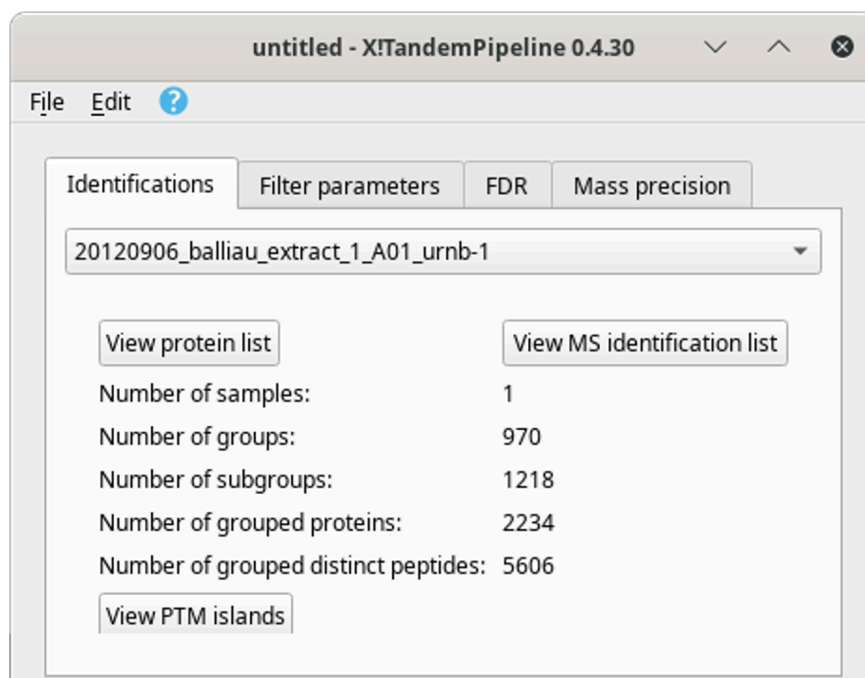
Pep repro: 1

Loading identification results comes with some configuration that is described in the text.

FIGURE 3.5: CONFIGURATION OF THE LOADING OF THE IDENTIFICATION RESULTS

3.4.1 CONFIGURATION OF THE PARAMETERS

- *Results handling mode*: there are two possibilities:
 - *Combine*: in this mode, all the identification results coming from different identification results files are merged into a single set. That single set is the basis for the protein inference step and the identified proteins are listed into a single protein list window.
 - *Individual*: in this mode, the identification results coming for various files are kept separated. Thus, the identification results coming from each file are used for a separate protein inference step. The identified proteins list is thus displayed for *each single file* in turn. The selection of the file for which the protein list needs to be displayed is done via the main program window that changes its appearance:



When loading multiple identification results files in *Individual* mode, the selection of any given identification results file is performed by selecting its name from the drop-down list widget *and* by clicking onto the *View protein list* button. Note that some metadata about the identifications are updated beneath the drop-down list widget.

FIGURE 3.6: SELECTING A PARTICULAR IDENTIFICATION RESULTS FILE'S DATA SET

Right after having selected an identification results file, click onto the *View protein list* to display the protein identifications list. That list has been obtained by performing the protein inference on the file's protein identification results (see [SECTION 2.2.5.4, "PROTEIN INFERENCE: FROM PSMs TO PROTEIN IDENTITIES"](#)). The window that opens up will be described later [FIXME cross ref.](#)



TIP

It is possible to open multiple protein list windows, each showing the identifications from a different file: to that end, when clicking onto the *View protein list* button, keep the `Ctrl` keyboard key pressed.

- *Choose results files*: by clicking onto the *Add files* button, the user is provided a file selection dialog box from which any number of protein identification results files might be selected for loading. Note that it is possible to list all the opened protein identification results files by clicking onto the *View MS identification list* button. The window that opens up will be described later [FIXME cross ref.](#)
- *Contaminants*: there are two possibilities here.
 - *Contaminants files*: when this radio button widget is selected, the list of contaminant proteins will be loaded from the files selected by clicking onto the *Add files* button.
 - *Contaminant regular expression*: when this radio button widget is selected, a text edit widget is shown, replacing the widget listing the contaminants database files. In this text edit widget, the user may enter a regular expression to match the accession number field of the protein databases that were used for the protein identification step. In this situation, the user must use specially crafted protein databases in which the contaminant proteins were tagged on the accession number using a particular pattern. That pattern is then matched against the *Contaminant regular expression* that the user enters in the text edit widget.
- *Contaminant removal mode*: there are two possibilities. The contaminant removal is the process by which when identified proteins match proteins in the contaminants realm (either from the contaminants database files or as determined using the regular expression) they are disregarded for the later protein visualization steps.
 - *Protein list*: in this mode, as soon as a protein identification loaded from a protein identification results file matches a contaminant protein, it is disregarded.
 - *Groups*: in this mode, the protein inference process goes all the way through to the determination of the proteins groups (see [FIGURE 2.8, “PROTEIN INFERENCE: CONSTRUCTING A CONSOLIDATED PROTEIN IDENTIFICATIONS LIST”](#)). When protein groups have sub-groups that contain a contaminant protein, then the whole group is disregarded. This might appear drastic, but our experience is that most often, the sub-groups in a group identify proteins belonging to the same family. Therefore, if one protein is contaminant, all the other proteins are supposed to be also.
- *Peptide and protein filters*: this group box widget holds some parameters that configure the way protein inference is to be performed.

- *Peptide threshold on*: there are two possibilities:
 - *E-value*: all the PSMs having an expectation value higher than that value are disregarded. Enter the value in the spin box widget labelled *Peptide E-value*. A typical value for the *X!Tandem* engine is 0.05. When more stringent results are desirable, setting 0.02 should yield satisfactory results. See [SECTION 2.2.5.2, “COMPUTATION OF THE PEPTIDE EXPECTATION VALUE \(E-VALUE\)”](#) of a detailed explanation of the E-value computation.
 - *FDR* (false rate discovery): the PSMs are disregarded if their FDR value does not match this parameter. Enter the value in the spin box widget labelled *Peptide FDR*. A typical setting is 1%.



TIP

Using *FDR* is most useful when the identification results come from a database searching engine that does not compute an E-value. However, it does only work if the searching step was performed also on a decoy database. In *X!Tandem* the decoy database is crafted by reversing the peptide sequences. In this case, when proteins are identified on the basis of the reversed peptide PSM, then the protein identity is tagged with the “reversed” string, which might be used with the *Contaminant regular expression* setting defined earlier.

- *Number of peptides per protein*: this is the minimal required number of peptides that must be found to identify a protein. These peptides have to be from non-contaminant proteins, of course.
- *Overall samples*: when checked and if multiple identification results files are to be loaded, then the *Number of peptides per protein* requirement might be fulfilled by looking for peptides in all the loaded files. For example, if one results file provides one peptide for a protein identification and another file provide another peptide (different from the first one) to identify the same protein, and

if the *Number of peptides per protein* is 2, then the protein is considered identified. If not checked, that number of peptides requirement must be fulfilled by looking into each results file separately. This last setting is more stringent. A typical value for this setting is 2.



TIP

This setting needs to be checked in at least one case: when a complex peptidic mixture is separated by ion chromatography (typically on an SCX—strong cation exchange—resin) and the different fractions are analyzed by bottom-up proteomics. The peptides coming from a given protein might be located in different fractions, and thus in different protein identification results files!

- *Protein Evalve*: threshold above which a protein identification is disregarded (see [SECTION 2.2.5.3, “COMPUTATION OF THE PROTEIN EXPECTATION VALUE \(E-VALUE\)”](#)).
- *Protein Evalve (log10)*: convenience spin box widget for the user to easily set the protein E-value.
- *Pep repro*: if set to 1, a peptide, to be accounted for, needs to be found in one protein identification results file. If set to a greater number, then that peptide needs to be found in that number of results files. This setting sets more stringent protein identification conditions each time it is incremented.

3.4.2 SAVING *X!TandemPipeline* PROJECTS

Once exploration and optional modification of the identification data have been performed, the user can save the resulting data set into a *X!TandemPipeline* project by selecting the *Save project* menu item of the *File* menu in the main program window (the extension of the file name typically should be xpip). See [SECTION 3.5, “LOADING X!TANDEMPIPELINE PROJECTS”](#) for loading such a project.

3.5 LOADING *X!TandemPipeline* PROJECTS

Loading of *X!TandemPipeline* project files (file of xpip extension) is only possible if the user has previously

- Loaded identification results;
- Saved the data to an *X!TandemPipeline* project file using the *Save project* menu item of the *File* menu in the main program window.

4 EXPLORING IDENTIFICATION DATA

This chapter describes in detail all the steps that the user accomplishes in their data exploration session. The general workflow is to start by looking at a protein identification results window and then by going into the details of the various identifications listed in it. This latter task entails looking into the peptides that provided the protein identification and then looking at the mass spectrum that provided the peptide identification. The mass spectrum, that is, the MS/MS spectrum has features aimed at allowing the user to make an informed opinion on the validity of the peptide *vs* mass spectrum match (PSM) at hand. At each moment, it is possible to invalidate a PSM and the identification results are recomputed automatically by taking into account the modification entered by the user.

4.1 THE PROTEIN IDENTIFICATIONS LIST WINDOW

When identification results files are loaded, *X!TandemPipeline* automatically perform the protein inference process by using the configuration settings set as described in [SECTION 3.4.1, “CONFIGURATION OF THE PARAMETERS”](#).

4.1.1 THE PROTEIN IDENTIFICATIONS LIST TABLE VIEW

When the protein inference process is finished, the *X!TandemPipeline* displays the protein identifications list in a table view, as pictured in [FIGURE 4.1, “”](#).

checked	group	accession	description	log(Evalue)	Evalue	spectra	specific spectra	sequences	specific sequence	coverage	MW	PAI	empPAI
<input checked="" type="checkbox"/>	a1.a1.a1	GRMZM2G083841_P01	P04711 Phosphoenolpyruvate ...	-436.204	0	269	251	58	54	56.49 %	109272	1.84615	69.1704
<input checked="" type="checkbox"/>	c117.a1.a1	GRMZM2G137839_P01	NP_001152746 ascorbate ...	-71.1121	7.72497e-72	32	7	9	2	52.00 %	27353.9	1.33333	20.5443
<input checked="" type="checkbox"/>	c117.a2.a1	GRMZM2G054300_P01	NP_001150192 APx1 - Cytosolic ...	-59.8189	1.5174e-60	27	2	8	1	44.80 %	27290.8	1.16667	13.678
<input checked="" type="checkbox"/>	c117.a2.a2	GRMZM2G054300_P04	NP_001150192 APx1 - Cytosolic ...	-59.8189	1.5174e-60	27	2	8	1	37.46 %	32330.3	1	9
<input checked="" type="checkbox"/>		GRMZM2G054300_P02	NP_001150192 APx1 - Cytosolic ...	-42.1156	7.66366e-43	17		5		35.94 %	20841.5	1.125	12.3352
<input checked="" type="checkbox"/>		GRMZM2G054300_P03	NP_001150192 APx1 - Cytosolic ...	-42.1156	7.66366e-43	17		5		31.80 %	23404.6	0.9	6.94328
<input checked="" type="checkbox"/>	c407.a1.a1	GRMZM2G046841_P01	B65IF9 Histone H2B ...	-24.4482	3.56257e-25	8	3	5	2	37.33 %	16133.9	0.714286	4.17947
<input checked="" type="checkbox"/>	c407.a1.a2	GRMZM2G119071_P01	P30756 Histone H2B.2 ...	-24.4482	3.56257e-25	8	3	5	2	37.33 %	16163.9	0.714286	4.17947
<input checked="" type="checkbox"/>	b36.a1.a1	GRMZM2G044946_P01	NP_001169327 hypothetical ...	-148.005	9.88939e-149	37	37	19	19	49.19 %	51820.9	0.88	6.58578
<input checked="" type="checkbox"/>	b78.a1.a1	GRMZM2G027995_P01	Q41741 Eukaryotic initiation ...	-102.438	3.64897e-103	29	2	12	1	36.47 %	46952.9	0.9	6.94328
<input checked="" type="checkbox"/>	b78.a1.a2	GRMZM2G027995_P02	Q41741 Eukaryotic initiation ...	-102.438	3.64897e-103	29	2	12	1	36.47 %	46952.9	0.9	6.94328
<input checked="" type="checkbox"/>	b78.a2.a1	GRMZM2G116034_P01	NP_001104874 translation ...	-97.3364	4.60922e-98	29	2	12	1	36.47 %	46922.9	0.85	6.07946
<input checked="" type="checkbox"/>		GRMZM2G116034_P01	NP_001104874 translation ...	-91.1493	7.09111e-92	28		11		33.58 %	46663.8	0.8	5.30957
<input checked="" type="checkbox"/>	b60.a1.a1	GRMZM2G0845611_P01	B4F8L7 Glyceraldehyde-3-...	-98.5797	2.63223e-99	52	36	14	11	37.64 %	47150.2	0.954545	8.00628
<input checked="" type="checkbox"/>	b60.a2.a1	GRMZM2G337113_P02	P09315 Glyceraldehyde-3-...	-80.9409	1.14578e-81	41	25	10	7	33.25 %	42830	1.14286	12.895
<input checked="" type="checkbox"/>		GRMZM2G129246_P01	NP_001146005 hypothetical ...	-73.2632	5.45513e-74	24		12		32.02 %	53278.8	0.576923	2.77505
<input checked="" type="checkbox"/>	b82.a1.a1	GRMZM2G129246_P02	NP_001146005 hypothetical ...	-86.6197	2.40026e-87	25	25	13	13	46.07 %	40021.1	0.842105	5.95193
<input checked="" type="checkbox"/>		GRMZM2G129246_P05	NP_001146005 hypothetical ...	-60.7567	1.75125e-61	20		10		40.45 %	32985.5	0.8125	5.49382
<input checked="" type="checkbox"/>		GRMZM2G129246_P03	NP_001146005 hypothetical ...	-34.8504	1.41117e-35	10		7		38.89 %	24162.3	0.538462	2.45511
<input checked="" type="checkbox"/>		GRMZM2G129246_P04	NP_001146005 hypothetical ...	-34.8504	1.41117e-35	10		7		38.89 %	24162.3	0.538462	2.45511
<input checked="" type="checkbox"/>	c117.a3.a1	GRMZM2G140667_P01	NP_001105500 ascorbate ...	-24.1051	7.85039e-25	10	8	5	4	24.13 %	30900.7	0.357143	1.27585
<input checked="" type="checkbox"/>	c117.a3.a2	GRMZM2G140667_P02	NP_001105500 ascorbate ...	-24.1051	7.85039e-25	10	8	5	4	24.47 %	30482.5	0.384615	1.42446
<input checked="" type="checkbox"/>		GRMZM2G140667_P04	NP_001105500 ascorbate ...	-20.5201	3.01938e-21	8		4		31.41 %	20781.4	0.5	2.16228
<input checked="" type="checkbox"/>		GRMZM2G175867_P01	NP_001130422 hypothetical ...	-1.60206	0.025	1		1		2.12 %	66514.1	0.0454545	0.110336
<input checked="" type="checkbox"/>		GRMZM2G175867_P02	NP_001130422 hypothetical ...	-1.60206	0.025	1		1		3.61 %	38959.6	0.0588235	0.145048
<input checked="" type="checkbox"/>		GRMZM2G153969_P01	NP_001149564 OB-fold nucleic ...	-2.52288	0.003	2		1		8.43 %	18248.5	0.142857	0.389495
<input checked="" type="checkbox"/>		GRMZM2G174479_P01	B6TM56 Chloroplast outer ...	-3.85387	0.00014	2		1		11.40 %	12394.1	0.166667	0.467799
<input checked="" type="checkbox"/>		GRMZM2G167698_P01	seq=translation; ...	-2.5376	0.0029	2		1		2.48 %	58821.2	0.04	0.0964782
<input checked="" type="checkbox"/>		GRMZM2G009232_P01	NP_001141257 hypothetical ...	-2.5376	0.0029	2		1		2.49 %	58693.1	0.0416667	0.100694
<input checked="" type="checkbox"/>		GRMZM2G009232_P02	NP_001141257 hypothetical ...	-2.5376	0.0029	2		1		3.95 %	37105.2	0.0769231	0.193777
<input checked="" type="checkbox"/>		GRMZM2G009232_P03	NP_001141257 hypothetical ...	-2.5376	0.0029	2		1		3.94 %	37454.5	0.0769231	0.193777
<input checked="" type="checkbox"/>	c141.a1.a1	GRMZM2G318635_P01	seq=translation; ...	-66.2764	5.29139e-67	16	16	10	10	9.69 %	190929	0.136986	0.370839
<input checked="" type="checkbox"/>	c530.a1.a1	GRMZM2G157462_P01	NP_001152484 dynamin-2A ...	-20.5331	2.93026e-21	7	7	4	4	4.93 %	99589.6	0.0851064	0.216484

search accession



proteins all:6814 valid:3895 valid&checked:3895 grouped:2481 displayed:6814

The protein identifications list window displays the proteins assembled into groups. A number of metadata about the identifications are shown in a number of columns, the content of which is described in detail in the text.

FIGURE 4.1:

The contents of the protein identifications list window are detailed below:

- *Checked*: if checked, the identified protein sitting on the table row is set to an “accepted” state. By default, all proteins are set to this accepted state. Unchecking a protein determines the protein inference reprocessing, because disregarding a protein modifies the whole protein identifications results set.
- *group*: the protein group the protein belongs to.
- *accession*: the accession number field of the protein database.
- *description*: the description field in the protein database.
- *log(E-value)*: the Log10 of the protein E-value;
- *E-value*: the protein E-value;
- *spectra*: the number of spectra that identified the protein.
- *specific spectra*: the number of spectra that identified *only* this protein.

- *sequences*: the number of peptidic sequences that can be assigned to this protein.
- *specific sequences*: the number of peptidic sequences that can be assigned *only* to this protein.
- *coverage*: the percentage of the protein sequence covered by the peptides that identified it.
- *MW*: the molecular weight of the protein (M_r).
- PAI: “Protein abundance index”. This index was defined as the “number of peptides identified divided by the number of theoretically observable tryptic peptides”. See [HTTPS://WWW.NCBI.NLM.NIH.GOV/PMC/ARTICLES/PMC186633/](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC186633/) .
- emPAI: “Exponentially modified protein abundance index”. This index was defined as $\text{emPAI} = 10^{\text{PAI}} - 1$. See [HTTPS://PUBMED.NCBI.NLM.NIH.GOV/15958392/](https://pubmed.ncbi.nlm.nih.gov/15958392/) .

It is possible to select the columns that must be displayed in the table by checking or unchecking the corresponding item in the *Columns* menu.

The *Show only* menu allows one to select the kind of protein items to be shown:

- *Valid proteins*: when check the program only shows valid proteins, that is protein identifications that full-fill the restriction parameters, like protein E-value, for example. These parameters were set at protein identification results file loading time but can be modified later.
- *Checked proteins*: show only the proteins that were checked. This setting is useful when the user has unchecked a number of proteins and that they want to regularly keep an eye on them. When proteins are unchecked, the protein inference process is run anew to compute a new grouping by taking *not* into account the proteins that were disregarded.
- *Grouped proteins*: only show the proteins that belong to a group.

The protein identifications list window picture in the figure show greyed protein identities. These are proteins that, by current filter parameters (E-value threshold, for example), are considered not valid.

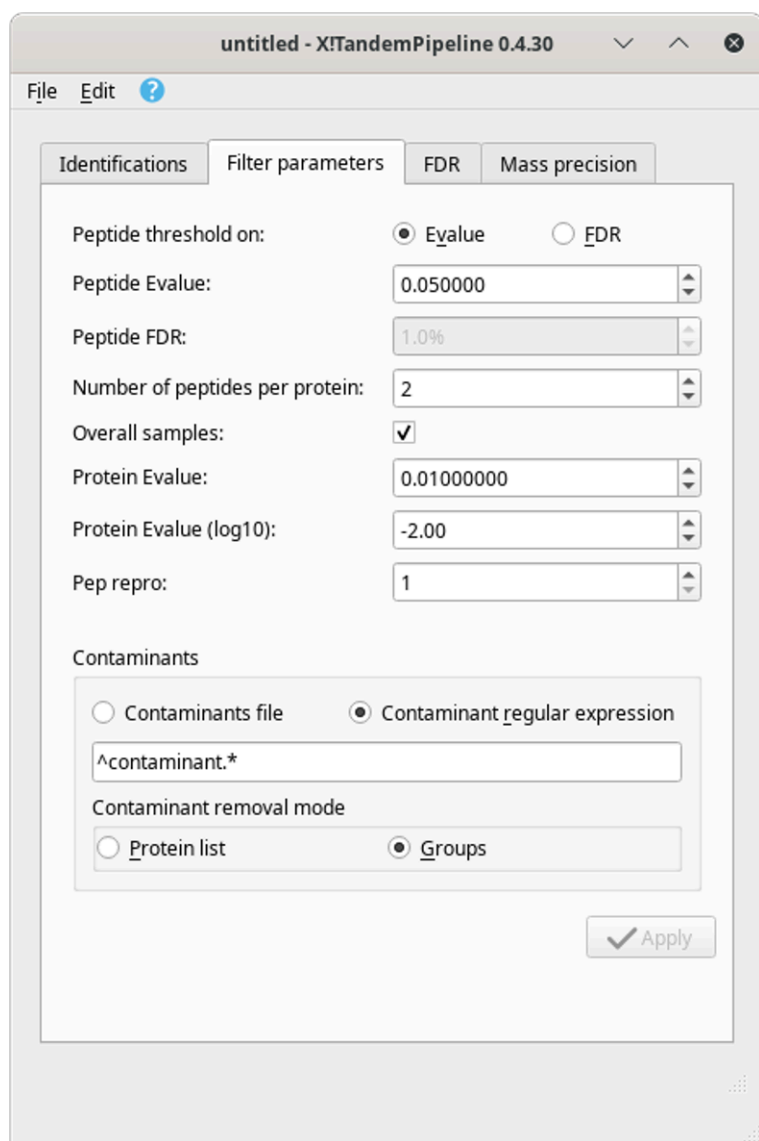
4.1.2 OPERATIONS IN THE PROTEIN IDENTIFICATION LIST WINDOW

The protein identifications list window houses a number of pretty interesting features that let the user scrutinize the protein identifications and also modify the results to suit either more or less stringent filtering parameters.

Searching data in the table view. One interesting feature of the protein identifications list window is the ability to search through the table's contents using the *Search* item at the bottom of the window. A number of fields of the protein record, that is, columns in the table view might be searched.

Dynamic setting of the filter parameters. *X!TandemPipeline* provides a rather high level of flexibility: once a protein identification results set of files has been loaded and that the protein inference process is achieved,

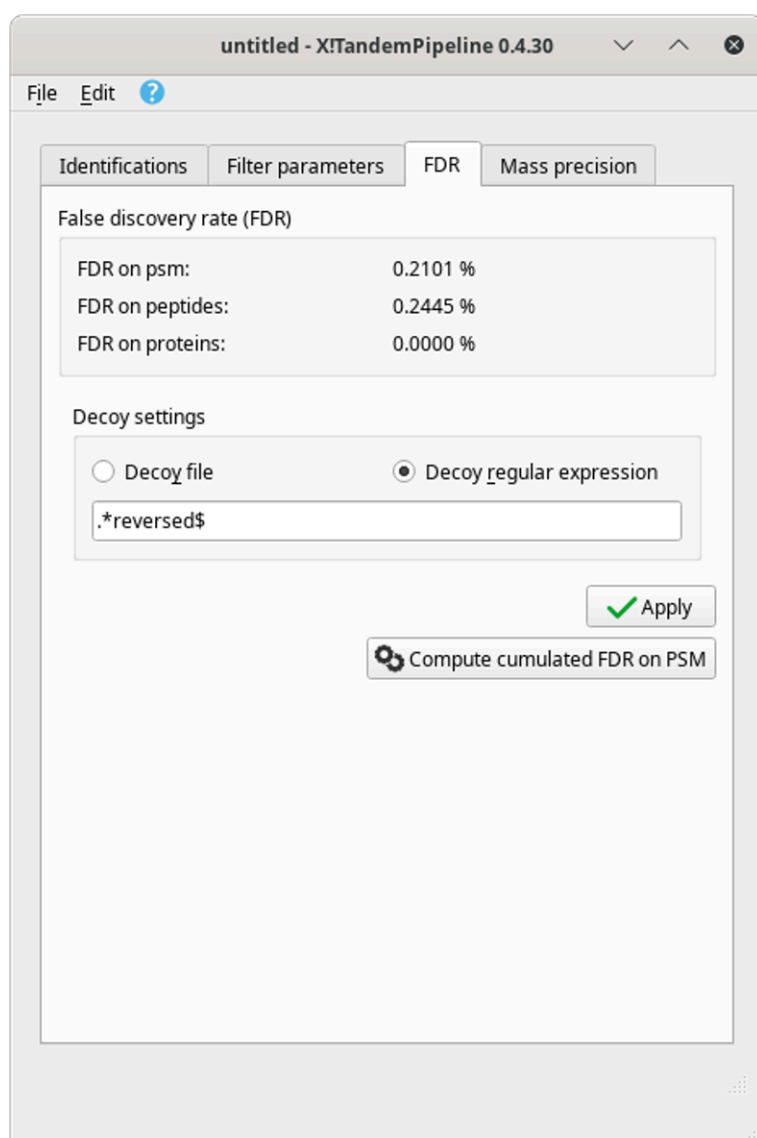
the resulting protein groups are displayed in the protein identification list window. At this time, the grouping was performed using the parameters set as pictured in [SECTION 3.4.1, “CONFIGURATION OF THE PARAMETERS”](#). It is nonetheless possible to modify these parameters on the fly using the main program window's *Filter parameters* tab, as pictured in [FIGURE 4.2, “PROTEIN IDENTIFICATION FILTER PARAMETERS TAB OF THE MAIN WINDOW”](#).



The filter parameters in this dialog box window do mirror the ones that one can set prior to loading protein identification results files. When modified, these parameters elicit a complete run of the protein inference process.

FIGURE 4.2: PROTEIN IDENTIFICATION FILTER PARAMETERS TAB OF THE MAIN WINDOW

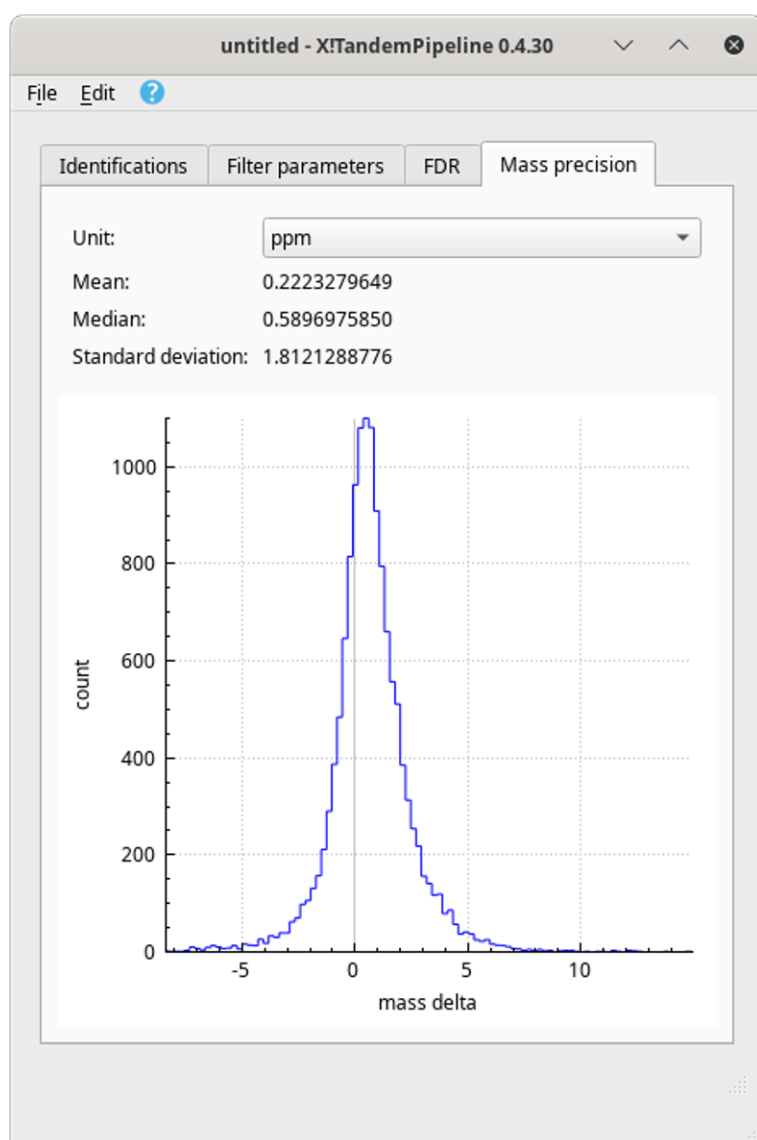
Real time update of the false discovery rate. The false discovery rate (FDR) is recalculated at each protein inference process. The data regarding this quality assessment criterion are shown in [FIGURE 4.3, “FALSE DISCOVERY RATE \(FDR\) DATA AFTER A PROTEIN INFERENCE PROCESS IS RUN”](#).



The various data bits about the false discovery rate that is computed each time a protein inference process is run. Note that it is possible to modify the *Decoy settings*, after which the *Apply Compute cumulated FDR on PSM* button triggers the recalculation of the FDR. FIXME.

FIGURE 4.3: FALSE DISCOVERY RATE (FDR) DATA AFTER A PROTEIN INFERENCE PROCESS IS RUN

Distribution of mass errors on PSMs plotted in a histogram. It is possible to visualize the distribution of the mass errors over the whole dataset, as pictured in [FIGURE 4.4, “MASS PRECISION QUALITY ASSESSMENT”](#). The histogram plots the number of mass spectra that could achieve a PSM against the mass error (mass delta), that is, the difference between the experimental peptide mass and the calculated peptide mass. [FIGURE 4.4, “MASS PRECISION QUALITY ASSESSMENT”](#).



The histogram plots the number of PSMs against the mass error calculated between the experimental mass of the peptide and the calculated mass.

FIGURE 4.4: MASS PRECISION QUALITY ASSESSMENT

The mass delta calculation involves only the peptides that successfully identified proteins that are currently checked in the protein identification list and that satisfy the filter parameters. The proteins identified in the decoy database are not processed. The unit of the mass delta may be selected using the *Unit* drop-down list. Two units are available: ppm (for part-per-million) or Dalton.

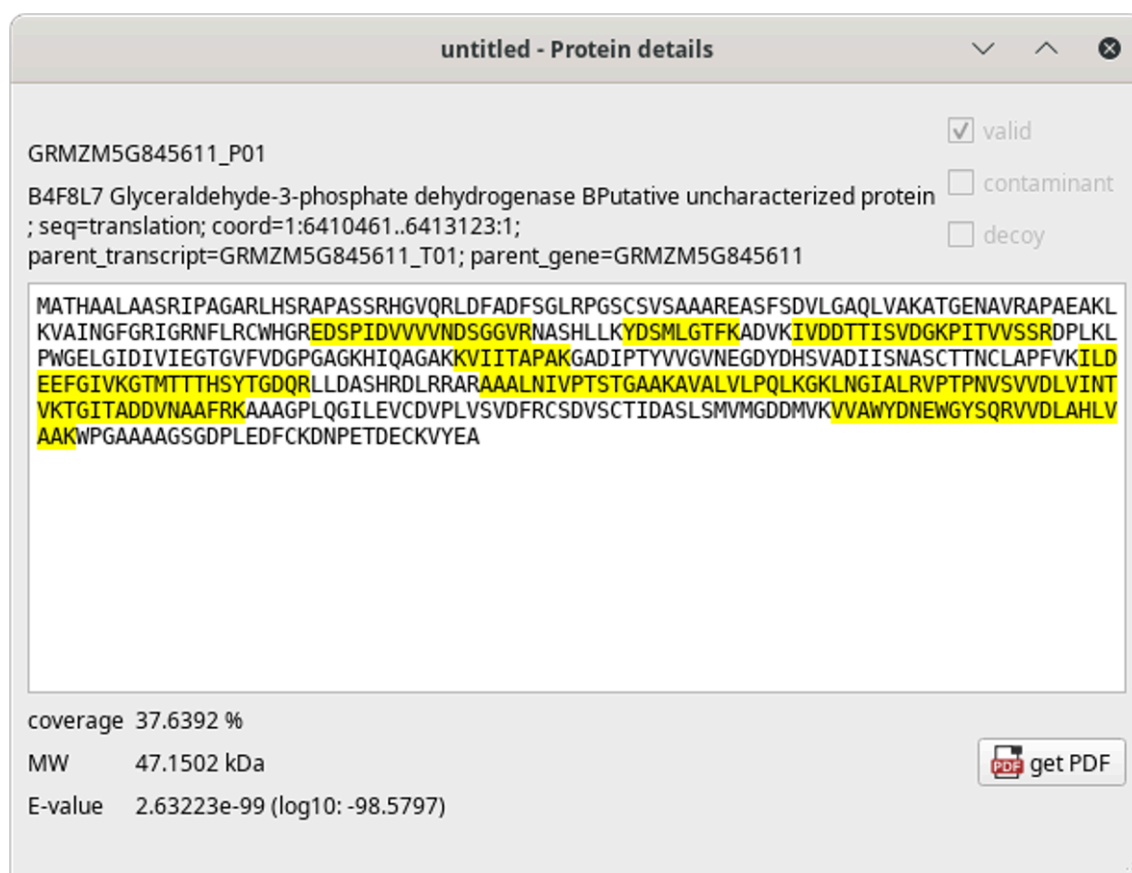
Exporting the final protein identifications list to a spread sheet. Once all the proteins in the identifications list have been properly checked, the user might export the data set to an OpenDocumentFormat (ODF) spread sheet file using the *As ODS file* menu item of the main window's *Export* menu.

4.1.3 DELVING INSIDE THE PROTEIN IDENTIFICATION DATA

The protein identifications list table view, as pictured in [FIGURE 4.1](#), “” is actually an active matrix where the user can easily trigger the exposition of the data that yielded any protein identification element of the table. This is simply done by clicking onto any cell of the table at the row matching the protein for which scrutiny of the data is requested.

Depending on the column at which the mouse click happens, there might be two different windows showing up:

- The *Protein details* window, showing the sequence of the protein, the matching peptides, as pictured below:



When one cell in the *Accession*, *Description* or *Coverage* column is clicked, this window shows up and displays the sequence of the protein, the coverage of the peptides and other useful data.

FIGURE 4.5: PROTEIN DETAILS AS DISPLAYED WHEN CLICKING ONE ROW OF THE TABLE VIEW

- When one cell in any one of the remaining columns is clicked, the window that shows up is the *Peptide list* window showing all the peptide identifications list, to be described in the next section.



TIP

When clicking one cell in one column and one given row, the corresponding window shows up, if one was not already opened. If one window was already opened, no other window shows up, but the existing window has its data updated to match the new protein row being clicked on.

It is possible to have multiple windows opened at a time by clicking a new row while maintaining the **Ctrl** key pressed.

4.2 THE PEPTIDE IDENTIFICATIONS LIST WINDOW

The peptide identifications list window displays all the data in a table view similar to the one used to display the protein identifications list described in the previous sections.

4.2.1 THE PEPTIDE IDENTIFICATIONS LIST TABLE VIEW

The peptide identifications list table view has a pretty large number of columns to display all the data about each peptide that identified a given protein. These columns are described in the following figures.

Checked	Peptide ID	Engine	Sample	Scan	Index	RT (min)	RT (s)	Charge	Obs. m/z	N-term	Sequence
<input checked="" type="checkbox"/>	pepc407a1	X!Tandem	20120906_balliau_extract_1_A01_urmb-1	15581		46.16"	2769.40"	2	880.42310	K	AMSIMNSFINDIFEK
<input checked="" type="checkbox"/>	pepc407a1	X!Tandem	20120906_balliau_extract_1_A02_urzb-1	16007		46.19"	2771.48"	2	880.42383	K	AMSIMNSFINDIFEK
<input checked="" type="checkbox"/>	pepc407a2	X!Tandem	20120906_balliau_extract_1_A01_urmb-1	9412		30.56"	1833.81"	2	470.29733	R	LVLPGELAK
<input checked="" type="checkbox"/>	pepc407a2	X!Tandem	20120906_balliau_extract_1_A02_urzb-1	9642		30.21"	1812.83"	2	470.29752	R	LVLPGELAK
<input checked="" type="checkbox"/>	pepc407a3	X!Tandem	20120906_balliau_extract_1_A01_urmb-1	2296		12.48"	748.85"	3	579.31262	K	KPAAKKPAEEEPAAEK
<input checked="" type="checkbox"/>	pepc407a4	X!Tandem	20120906_balliau_extract_1_A01_urmb-1	2038		11.43"	685.96"	3	541.61798	K	KPAEEEPAAEKAPAGK
<input checked="" type="checkbox"/>	pepc407a4	X!Tandem	20120906_balliau_extract_1_A02_urzb-1	1971		10.86"	651.38"	3	541.61755	K	KPAEEEPAAEKAPAGK
<input checked="" type="checkbox"/>	pepc407a6	X!Tandem	20120906_balliau_extract_1_A02_urzb-1	4898		19.07"	1144.02"	2	590.81934	K	QVHPDIGISSK

The peptide identifications list table view has many columns, this one is the first part over two.

FIGURE 4.6: PEPTIDE IDENTIFICATIONS LIST TABLE VIEW (1)

untitled - Peptide list

Export Columns Show only

GRMZM2G119071_P01

P30756 Histone H2B.2 seq=translation; coord=4:63092571..63093826:-1; parent_transcript=GRMZM2G119071_T01; parent_gene=GRMZM2G119071

Sequence	C-term	Modifs	Start	Length	Used	Subgroups	E-value	Cumulated FDR	Obs. [M+H] ⁺	Theor. [M+H] ⁺	Delta [M+H] ⁺	Delta (ppm)	HyperScore
AMSIMNSFINDIFEK	L		84	15	2	c407.a1 c407.a2	4.2e-07		1759.83892	1759.83936	-0.000440704	-0.250423	41.1
AMSIMNSFINDIFEK	L		84	15	2	c407.a1 c407.a2	4.6e-07		1759.84038	1759.83936	0.0010243	0.582039	41.2
LVLPGELAK	H		126	9	2	c407.a1 c407.a2	0.021		939.58739	939.58734	4.55356e-05	0.0484634	27.5
LVLPGELAK	H		126	9	2	c407.a1 c407.a2	0.031		939.58776	939.58734	0.000411536	0.437996	27.5
KPAAKKPAEEEPAAEK	A	1K42.01	8	16	1	c407.a1	4.4e-08		1735.92331	1735.92249	0.000823266	0.474253	47.3
KPAEEEPAAEKAPAGK	K		13	16	1	c407.a1	1.8e-05		1622.83939	1622.83843	0.000964245	0.594172	35.1
KPAEEEPAAEKAPAGK	K		13	16	1	c407.a1	3.4e-05		1622.83811	1622.83843	-0.000317755	-0.195802	35.5
QVHPDIGISSK	A		73	11	2	c407.a1 c407.a2	5.1e-05		1180.63140	1180.63206	-0.000666812	-0.564792	44.9

search peptide

The peptide identifications list table view has many columns, this one is the second part over two.

FIGURE 4.7: PEPTIDE IDENTIFICATIONS LIST WINDOW (2)

The table's contents are well described by the column headers that are self-explanatory. When hovering over a column header with the mouse cursor, a tool-tip explanatory text is displayed.

It must be noted that more columns might make the table view depending on the protein identification data that were loaded. Indeed, depending on the database searching engine that was used for the protein identification, the data to be displayed vary. The whole list of columns that might be displayed in the table view are pictured in **FIGURE 4.8, "COLUMNS THAT POPULATE THE PEPTIDE IDENTIFICATIONS LIST TABLE VIEW"**

<input checked="" type="checkbox"/> Checked	<input checked="" type="checkbox"/> Inter prophet prob.
<input checked="" type="checkbox"/> Peptide ID	<input checked="" type="checkbox"/> HyperScore
<input checked="" type="checkbox"/> Engine	<input checked="" type="checkbox"/> Mascot score
<input checked="" type="checkbox"/> Sample	<input checked="" type="checkbox"/> Mascot E-value
<input checked="" type="checkbox"/> Scan	<input checked="" type="checkbox"/> OMSSA E-value
<input checked="" type="checkbox"/> Index	<input checked="" type="checkbox"/> OMSSA p-value
<input checked="" type="checkbox"/> RT (min)	<input checked="" type="checkbox"/> MS-GF raw score
<input checked="" type="checkbox"/> RT (s)	<input checked="" type="checkbox"/> MS-GF de novo
<input checked="" type="checkbox"/> Charge	<input checked="" type="checkbox"/> MS-GF energy
<input checked="" type="checkbox"/> Obs. m/z	<input checked="" type="checkbox"/> MS-GF spectral E-value
<input checked="" type="checkbox"/> N-term	<input checked="" type="checkbox"/> MS-GF E-value
<input checked="" type="checkbox"/> Sequence	<input checked="" type="checkbox"/> MS-GF isotope error
<input checked="" type="checkbox"/> C-term	<input checked="" type="checkbox"/> Comet XCorr
<input checked="" type="checkbox"/> Modifs	<input checked="" type="checkbox"/> Comet DeltaCn
<input checked="" type="checkbox"/> Label	<input checked="" type="checkbox"/> Comet DeltaCnStar
<input checked="" type="checkbox"/> Start	<input checked="" type="checkbox"/> Comet SpScore
<input checked="" type="checkbox"/> Length	<input checked="" type="checkbox"/> Comet SpRank
<input checked="" type="checkbox"/> Used	<input checked="" type="checkbox"/> Comet E-value
<input checked="" type="checkbox"/> Subgroups	<input checked="" type="checkbox"/> DeepProt matched peaks
<input checked="" type="checkbox"/> E-value	<input checked="" type="checkbox"/> DeepProt fitted peaks
<input checked="" type="checkbox"/> Cumulated FDR	<input checked="" type="checkbox"/> DeepProt match type
<input checked="" type="checkbox"/> Obs. [M+H] ⁺	<input checked="" type="checkbox"/> DeepProt status
<input checked="" type="checkbox"/> Theor. [M+H] ⁺	<input checked="" type="checkbox"/> DeepProt mass delta
<input checked="" type="checkbox"/> Delta [M+H] ⁺	<input checked="" type="checkbox"/> DeepProt mass delta pos.
<input checked="" type="checkbox"/> Delta (ppm)	
<input checked="" type="checkbox"/> Prophet prob.	

Depending on the provenience of the protein identifications (the database search engine), the columns that are part of the table view differ. This full list is displayed when selecting the *Columns* menu.

FIGURE 4.8: COLUMNS THAT POPULATE THE PEPTIDE IDENTIFICATIONS LIST TABLE VIEW

4.2.2 OPERATIONS IN THE PEPTIDE IDENTIFICATION LIST WINDOW

The peptide identifications list window houses a number of pretty interesting features that let the user scrutinize the peptide details.

Searching data in the table view. One interesting feature of the peptide identifications list window is the ability to search through the table's contents using the *Search* item at the bottom of the window. A number of fields of the protein record, that is, columns in the table view might be searched.

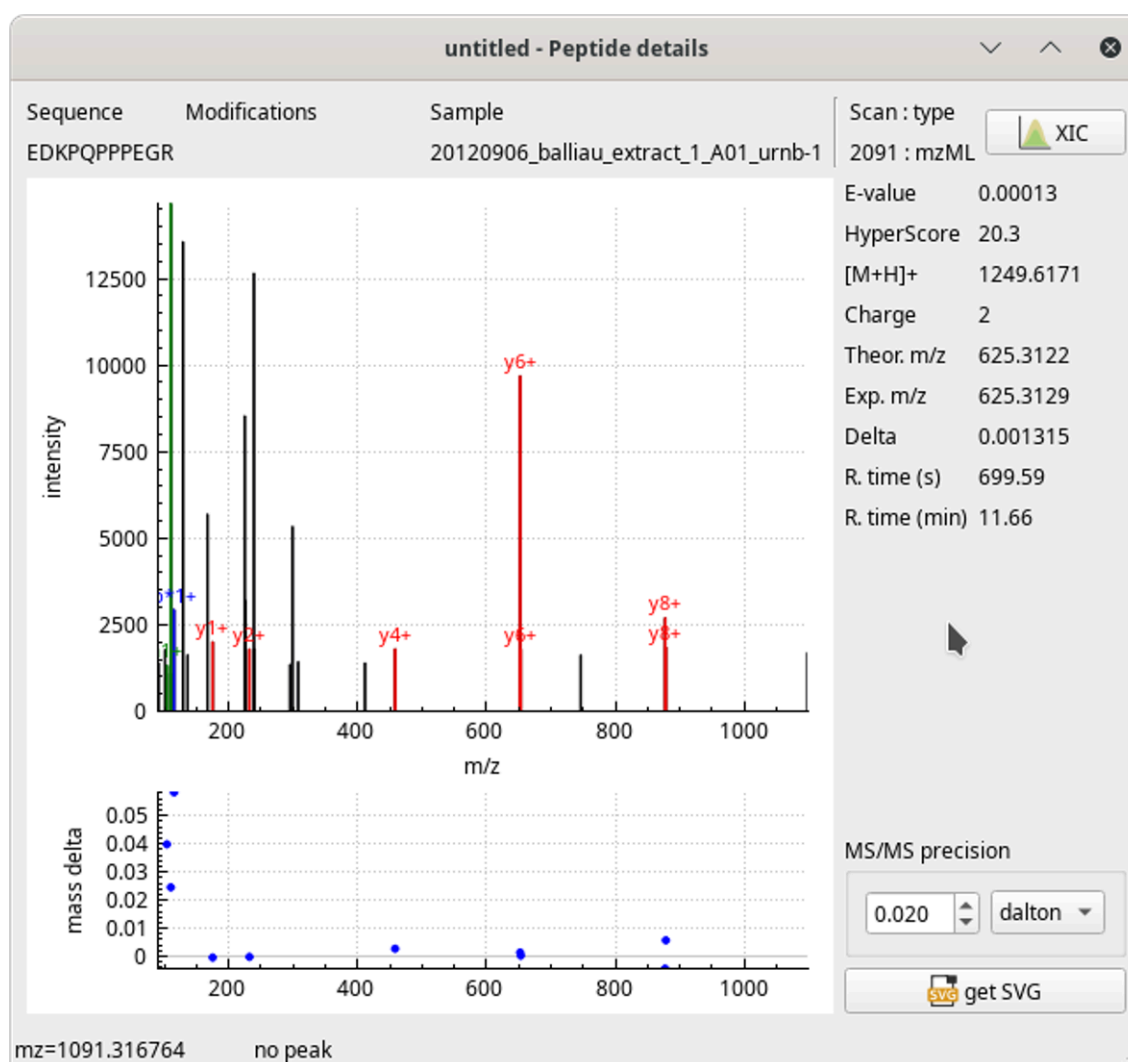
Exporting the final protein identifications list to a spread sheet. Once all the peptides in the identifications list have been properly checked, the user might export the data set to an OpenDocumentFormat (ODF) spread sheet file using the *As ODS file* menu item of the main window's *Export* menu.

4.2.3 DELVING INSIDE THE PEPTIDE IDENTIFICATION DATA

The peptide identifications list table view, as pictured in [FIGURE 4.6, “PEPTIDE IDENTIFICATIONS LIST TABLE VIEW \(1\)”](#) is actually an active matrix where the user can easily trigger the exposition of the data that yielded any peptide identification element of the table. This is simply done by clicking onto any cell of the table at the row matching the peptide for which scrutiny of the data is requested.

4.2.3.1 THE PEPTIDE DETAILS WINDOW

When clicking on any cell of the peptide identifications list table view, one window shows up that details the various data elements for the peptide documented in the table row. The window is pictured in [FIGURE 4.9, “PEPTIDE VS MASS SPECTRUM DETAILS”](#).



Each row of the peptide identifications list table view holds data about one of the peptides that identified a given protein. When clicking onto any cell of the table view, the window in this figure shows up, describing all the details about that peptide *vs* mass spectrum match.

FIGURE 4.9: PEPTIDE VS MASS SPECTRUM DETAILS

In **FIGURE 4.9**, “PEPTIDE VS MASS SPECTRUM DETAILS”, the two graphs show the following:

- The top graph displays the mass spectrum of this PSM. This MS/MS spectrum has its recognized peaks labelled in the y and bion series.
- The bottom graph plots, for each matching MS/MS peak (that is, b or y series ions), the error (mass delta) compared to the theoretical ion mass. In this example, we see that the y ion series is almost perfectly matched (low error and also all the errors in the same value range).



TIP

It is possible to zoom in on a region of the graphs by positioning the mouse cursor on the region of interest and then rotating the mouse wheel. To unzoom, simply rotate the mouse wheel in the reverse direction.

The right hand side margin provides a number of data about the PSM, like the peptide E-value, the HyperScore, the ion charge, the theoretical and experimental masses, the retention time at which this ion was detected... The data bits are self-explanatory.

4.2.3.2 THE XIC VIEWER WINDOW FOR THE PEPTIDE DETAILS

One interesting feature of the *Peptide details* window, is the *XIC* button (top right) that triggers the calculation of an extracted ion current chromatogram, as pictured in [FIGURE 4.10](#), “THE EXTRACTED ION CURRENT (XIC) CHROMATOGRAM VIEWER WINDOW”.



TIP: WHAT IS A XIC CHROMATOGRAM?

The notion of *extracted ion current* chromatogram is best explained by describing the computation that yields that chromatogram.

The user defines the m/z value for which the chromatogram is to be determined. The program iterates in each MS (that is, full scan) spectrum and looks if an ion by that m/z value was encountered. If so, a variable holding the cumulated intensity of that ion is incremented for the retention time at which the mass spectrum was acquired. For example, if m/z value 1254.25 is searched for, and an ion of that m/z value is found in the mass spectrum acquired at retention time 2.5 min, then a tuple variable is stored like this: (2.5, intensity). Then, another mass peak by that m/z value is found in mass spectrum acquired at retention time 47 min, for which another tuple is created: (47, intensity).

If the data are from ion mobility—mass spectrometry (IM-MS) experiments, then there might be a large number of spectra acquired at a given retention time. For example, data from the Waters Synapt2 instrument have 200 spectra acquired for any given retention time value (the spectra are drift-related spectra). In Bruker timsTOF data, there are more than 700 spectra acquired at any given retention time. Thus, the searched m/z value might be found more than once for a retention time value. In this case, the tuple's intensity value is incremented by the intensity of the new peak of the m/z value at that specific retention time value.

When the program has finished iterating in all the mass spectra of the acquisition, it plots the XIC chromatogram as $\text{intensity} = f(\text{retention time})$. This is the reason why it is a chromatogram.

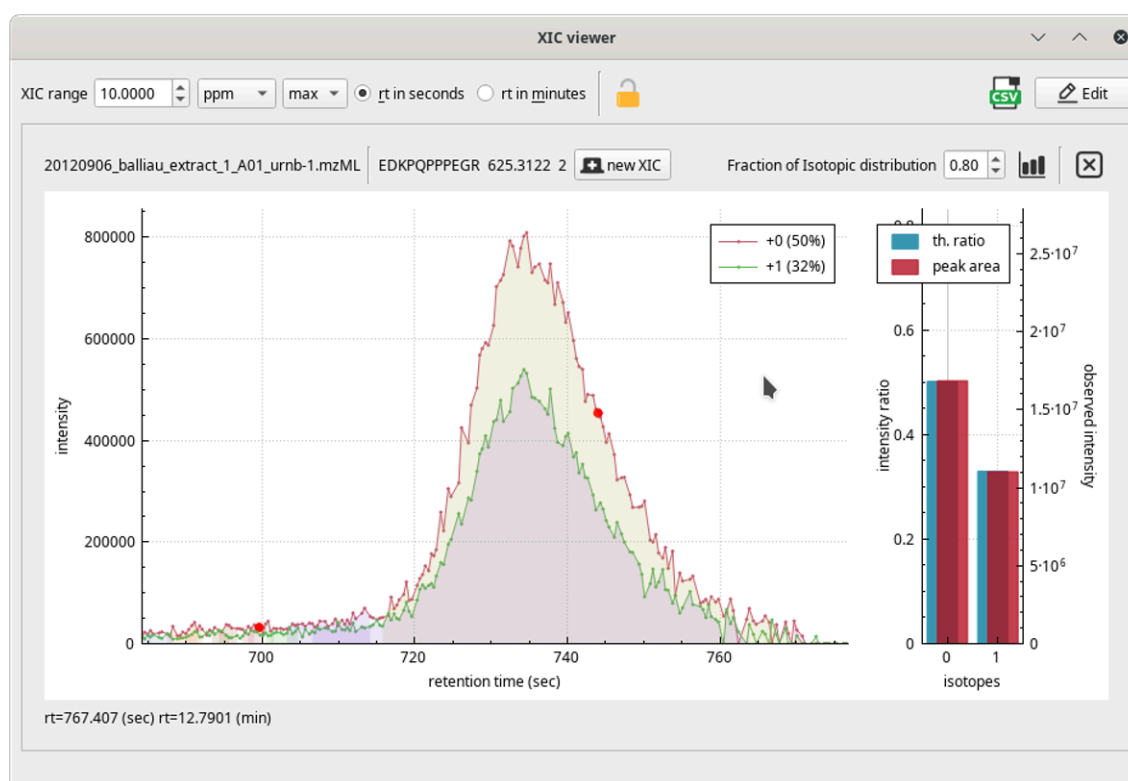


The extracted ion current (XIC) chromatogram viewer is useful to scrutinize the mass data at the very origin of a PSM. It is routinely used to ensure that the PSM is faithful. If not, the corresponding peptide can be unchecked from the peptide identifications list table view, which triggers the running anew of the protein inference process.

FIGURE 4.10: THE EXTRACTED ION CURRENT (XIC) CHROMATOGRAM VIEWER WINDOW

The *XIC viewer* window displays the “guts” of the of MS spectrum of the precursor ion that was fragmented and that yielded a PSM. The XIC chromatogram (left plot panel) is actually a set of XIC chromatograms that are superimposed in the plot widget (see [FIGURE 4.11, “THE EXTRACTED ION CURRENT \(XIC\) CHROMATOGRAM VIEWER WINDOW \(ZOOMED VIEW\)”](#)). One of the traces (legend +0) is for the first peak of the isotopic cluster of the searched ion in the MS data of the acquisition; the second trace (legend +1) is for the second peak of the isotopic cluster. In the typical informatics-oriented style of numbering, the first isotopic peak (only light isotopes enter in the composition of the peptidic ion), is “isotope 0”; the the second isotopic peak (one light isotope is substituted with a heavy one) is “isotope 1”.

The right panel is a bar plot showing the theoretical isotopic ratio between the first and the second peak of the isotopic cluster (blue) with, superimposed, the experimental ratio. In the example, the match between the experimental and the theoretical cluster shape is perfect.



Zoomed view over the two XIC chromatogram plots in the left hand side plot widget.

FIGURE 4.11: THE EXTRACTED ION CURRENT (XIC) CHROMATOGRAM VIEWER WINDOW (ZOOMED VIEW)

Another interesting bit of information is the *Fraction of Isotopic distribution* number that reflects the ratio between the plotted isotopic cluster peaks over the whole theoretically calculated isotopic cluster (in which more than one light isotope is substituted with a heavy isotope (with a mass increment of +2 and not +1). In the example that ratio is 80 %.

To zoom in/out regions of the XIC chromatogram plot widget, hover the mouse cursor over the region of interest and rotate the mouse wheel.

4.3 HANDLING PHOSPHO-PROTEOMICS DATA

X!TandemPipeline is able to cope with phospho-peptides. The mass spectrometric data are acquired exactly as usual with the mass spectrometer, but the sample preparation goes along these steps:

- Separate digestion of the samples (when there are more than one);
- Labeling of the peptides, each sample gets a different label;
- Pool of the whole set of peptides into a single mixture;
- Separation of the peptides on a strong cation exchange (SCX) resin, collection of the fractions;

- Phospho-peptide enrichment using IMAC¹ for each SCX fraction. The SCX fraction is loaded onto the IMAC resin and, following a wash step, the phospho-peptides are eluted (pH-based elution). There is thus a one-to-one relation between a SCX fraction and an IMAC-based purification fraction.
- Mass spectrometric analysis of each IMAC-based phospho-peptide-enriched fraction.

X!Tandem needs to be configured in such a manner that it can generate all the theoretical peptides (and fragments) that might bear the phosphoryl group. This process is described in the section below.

¹ Immobilized-metal affinity chromatography.

5 STUFF AWAITING INCLUSION

Empty.

A GNU GENERAL PUBLIC LICENSE VERSION 3

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. [HTTPS://FSF.ORG/](https://fsf.org/) 

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

PREAMBLE

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program—to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals

to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users. Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. DEFINITIONS.

“This License” refers to version 3 of the GNU General Public License.

“Copyright” also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program” refers to any copyrightable work licensed under this License. Each licensee is addressed as “you”.

“Licensees” and “recipients” may be individuals or organizations.

To “modify” a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version” of the earlier work or a work “based on” the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate” a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey” a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

I. SOURCE CODE.

The “source code” for a work means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. BASIC PERMISSIONS.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you

comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. PROTECTING USERS' LEGAL RIGHTS FROM ANTI-CIRCUMVENTION LAW.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. CONVEYING VERBATIM COPIES.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. CONVEYING MODIFIED SOURCE VERSIONS.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a.** The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b.** The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to “keep intact all notices”.
- c.** You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d.** If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an “aggregate” if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation’s users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. CONVEYING NON-SOURCE FORMS.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a.** Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b.** Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable

physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

- c. Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d. Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e. Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized),

the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. ADDITIONAL TERMS.

“Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a.** Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b.** Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c.** Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d.** Limiting the use for publicity purposes of names of licensors or authors of the material; or

- e. Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f. Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered “further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. TERMINATION.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. ACCEPTANCE NOT REQUIRED FOR HAVING COPIES.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. AUTOMATIC LICENSING OF DOWNSTREAM RECIPIENTS.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

II. PATENTS.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's “contributor version”.

A contributor's “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient’s use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. NO SURRENDER OF OTHERS’ FREEDOM.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. USE WITH THE GNU AFFERO GENERAL PUBLIC LICENSE.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. REVISED VERSIONS OF THIS LICENSE.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. DISCLAIMER OF WARRANTY.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. LIMITATION OF LIABILITY.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. INTERPRETATION OF SECTIONS 15 AND 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

HOW TO APPLY THESE TERMS TO YOUR NEW PROGRAMS


If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

```
one line to give the program's name and a brief idea of what it does.
Copyright (C) year name of author
```

```
This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.
```

```
This program is distributed in the hope that it will be useful,  
but WITHOUT ANY WARRANTY; without even the implied warranty of  
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the  
GNU General Public License for more details.
```


You should have received a copy of the GNU General Public License
along with this program. If not, see [HTTPS://WWW.GNU.ORG/LICENSES/](https://www.gnu.org/licenses/) .


Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
program Copyright (C) year name of author  
This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.  
This is free software, and you are welcome to redistribute it  
under certain conditions; type 'show c' for details.
```


The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate parts of the General Public License. Of course, your program’s commands might be different; for a GUI interface, you would use an “about box”.

You should also get your employer (if you work as a programmer) or school, if any, to sign a “copyright disclaimer” for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see [HTTPS://WWW.GNU.ORG/LICENSES/](https://www.gnu.org/licenses/) .


The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read [HTTPS://WWW.GNU.ORG/LICENSES/WHY-NOT-LGPL.HTML](https://www.gnu.org/licenses/why-not-lgpl.html) .


COLOPHON

About the authors. Filippo Rusconi is a senior research scientist at the French national research council (*Centre national de la Recherche scientifique*, CNRS). Filippo has a background in biochemistry and organic chemistry and was trained during his Ph.D. as a bioanalytical chemist. He has extensive knowledge of analytical techniques involved in the study of biopolymers.



Filippo Rusconi is the author of a handbook about mass spectrometry for biochemists (French). The book was published by the French sci/tech publisher **LAVOISIER** ([HTTPS://WWW.LAVOISIER.FR](https://www.lavoisier.fr)) .




Colophon. The look of this book (PDF file) is the result of me having read many books from the O'Reilly publisher.

The frog on the book title page is a frog from Papua. This frog is able to hover when performing downwards leaps. This picture is courtesy [HTTP://WWW.PAPUAWEB.ORG](http://www.papuaweb.org) .

The typesetting of the book has been done on a Debian GNU/Linux computer using only Free Software. Use of the DocBook Authoring and Publishing Suite (**DAPS** ([HTTPS://GITHUB.COM/OPENSUSE/DAPS](https://github.com/opensuse/daps)) ) from SUSE was key in the process.

The layout adopted for this book is an adaptation of the SUSE stylesheets. I would like to thank Frank Sundermeyer <fsundermeyer@opensuse.org> and Stefan Knorr <sknorr@suse.de> for being helpful with all my questions.

The main font used was **EBGARAMOND** ([HTTPS://GITHUB.COM/GEORGD/EB-GARAMOND](https://github.com/georgd/EB-GARAMOND))  and the symbol/mathematical font was from the **STIX PROJECT** ([HTTPS://WWW.STIXFONTS.ORG/](https://www.stixfonts.org/))  (font: STIX2Math).

The screen shots were taken with Spectacle, the screen capture program shipped along with my **KDE** ([HTTPS://WWW.KDE.ORG/](https://www.kde.org/))  desktop environment and resampled using The GNU image manipulation program **THE GIMP** ([HTTPS://WWW.GIMP.ORG/](https://www.gimp.org/)) . Illustrations were done in **INKSCAPE** ([HTTPS://INKSCAPE.ORG/](https://inkscape.org/)) , a vectorial drawing software.