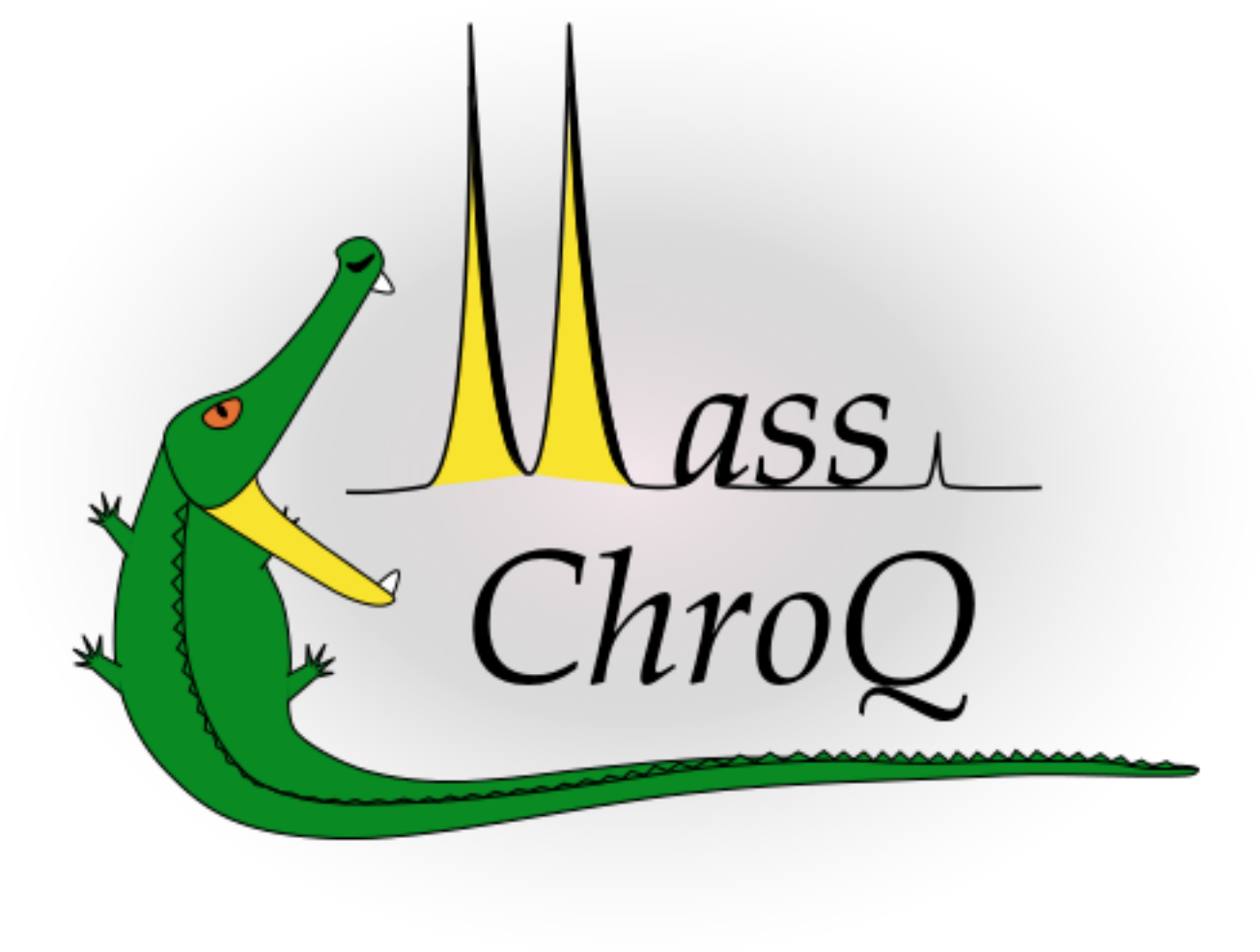
MassChroQ manual

Mass Chromatogram Quantification software



Plateforme d'Analyse Protéomique de Paris Sud-Ouest

MassChroQ manual

Third edition for MassChroQ version 2.0 *Spectacled Caiman*

Author: Edlira Nano
Contributors: Olivier Langella, Benoît Valot, Michel Zivy

Contents

Preface

MassChroQ is an open-source software released under the Gnu General Public License version 3. It is developed at the PAPPSO team (Plateforme d'Analyse Protéomique de Paris Sud-Ouest) by:

Benoît Valot benoit.valot@moulon.inra.fr

Olivier Langella olivier.langella@moulon.inra.fr

Edlira Nano edlira.nano@moulon.inra.fr

Michel Zivy michel.zivy@moulon.inra.fr

This manual contains detailed information about the concepts and features of MassChroQ v.2.0, called *Spectacled Caiman*. It is intended to provide all the technical information needed for an advanced comprehension and use.

In chapter 1 you will be introduced to the main features of MassChroQ.

In chapter 2 you will learn how to install and run MassChroQ with a quick overview of its available command-line options.

In chapter 3 you will learn how MassChroQ works in depth: every analysis step is detailed and the most important algorithms (peak detection, alignment) are explained step-by-step.

In chapter 4 the masschroqML format, format of the input xml file to MassChroQ, is explained in details. Explanation is illustrated with an example file, line by line.

In chapter 5 you will find the complete specification of the *csv* file format containing the identified peptides.

In chapter 6 you will find a cheat-sheet of all the parameters in MassChroQ, with the corresponding recommended values.

Finally in the appendixes you will find the complete files used in this manual.

Chapter 1

Introduction

1.1 What is MassChroQ?

MassChroQ (Mass Chromatogram Quantification) software performs alignment, XIC extraction, peak detection and quantification on data obtained from LC-MS (Liquid Chromatography-Mass Spectrometry) techniques.

A full description and evaluation of MassChroQ can be found in [1].

MassChroQ can analyze:

- label-free data as well as isotopic labeled ones (e.g. SILAC, ICAT N-terminal, C-terminal labels);
- data obtained from high-resolution systems (HR) as well as low-resolution ones (LR).
- It is able to take into account complex peptide or protein experiments on data (e.g. peptide or protein separations prior to LC with SDS-PAGE, SCX fractionations, etc.).
- It is fully configurable and every single step of its analysis is fully traceable.
- Its modular implementation facilitates integration of third-part libraries as well as MassChroQ's integration into them.
- It is platform-independent and uses or produces only open format data.

MassChroQ is developed in the C++ language using the Qt framework. Version 2.0 is its latest public release.

MassChroQ v.2.0 comes with :

- `masschroq`, a stand-alone command-line program which performs all of the essential tasks. This is the core module of MassChroQ;

- `libmasschroq`, a library for integration in other software or proteomic pipelines;

- `MassChroQ Studio`, a graphical interface allowing the user to perform extraction, filtering detection and alignment on runs via masschroq, in order for him to perfectly adjust the final `masschroq` parameters (first added in version 2.0);

- `MassChroQ GUI`, a graphical interface allowing the user to launch `masschroq` graphically, not from the command line (first added in version 2.0).

On the MassChroQ homepage (http://pappso.inra.fr/bioinfo/masschroq/) you can find download and install instructions, various documentation files and the latest news about this project.

On the MassChroQ development page hosted by SourceSup at http://sourcesup.renater.fr/projects/masschroq/, you will find a subversion repository, a bug tracker and forums.

The source code is anonymously available via direct access to the subversion repository from https://subversion.renater.fr/masschroq/.

Feel free to contribute to the MassChroQ project by directly contacting one of its authors.

1.2 What is new in MassChroQ v.2.0?

1.2.1 The `MassChroQ Studio`

This is the main novelty in MassChroQ v.2.0. `MassChroQ Studio` is a graphical, user friendly tool, allowing to load runs or masschroqML files and to perform XIC extraction, filtering, detection and alignment on them. The results of these operations can be instantly visualized on graphical plots. The goal of the Studio is to help the user find the masschroqML parameters that best fit his data and experiment. The final parameters can be exported into masschroqML format, the user has only to integrate them in his masschroqML file.

1.2.2 The `MassChroQ Graphical User Interface (GUI)`

Another novelty in MassChroQ v.2.0 is the `MassChroQ GUI`, a simple graphical interface that allows the user to launch `masschroq` graphically, without having to do strange things on the command-line. A graphical console allows to see the runtime messages of MassChroQ.

1.2.3 The `.time` and `.trace` alignment files

In the `masschroq` core module, we have added the possibility for the user to decide whether he wants to load previous `.time` files or not, and if yes to directly indicate the directory where to find these files. This way, one can use previous alignment results withut having to repeat the alignment. In the same way, the user has to indicate in the alignment methods the directory on disk where to put the .time and .trace alignment files if he wants them. If no directory is specified, these files will not be written to disk by default. In previous versions of MassChroQ the .time files were always loaded from the current directory if any, and they were always written to disk in the current directory. This implicit behavior could easily induce errors. Indeed, the user had to be careful and to remember to delete the .time files that he did not wanted to be loaded. He does not have to delete anything now.

As a consequence, the masschroqML schema has changed in MassChroQ v.2.0, the current schema version being 2.0 (the same version number as MassChroQ). Older masschroqML files stay functional with the new schema, but new masschroqML files containing the .time directories attributes, will not work with older schema versions.

1.3 MassChroQ features overview

MassChroQ has been designed to perform quantification on a wide range of LC-MS data: label-free or isotopic labeled ones, high-resolution (HR) or low-resolution (LR) ones. To achieve this MassChroQ can combine and perform the following features :

- Determination of items of interest to be quantified. These items can be:
  - the identified peptides,
  - the identified isotopes,
  - a list of mass over charge (m/z) ratios,
  - a list of couples of m/z and retention time (rt) values.
- Alignment of samples within each group (two different alignment methods are proposed).
- Extraction of the XICs (Extracted Ion Chromatogram) of the predetermined items of interest.
- XIC filtering (several filters are provided for signal noise removal, spike removal, signal smoothing, etc.).

- Detection of peaks on these XICs.
- Quantification of the predefined items of interest (two different quantification methods are proposed).
- Peak matching during and after quantification.
- Grouping of LC-MS data that present similarities (for example grouping of the same LC fractions in an SCX fractionated analysis in order to perform alignment on them).

MassChroQ accepts mzXML as well as mzML LC-MS data formats.

To include the identified peptides/isotopes in a MassChroQ analysis there are two possibilities :

- by using the freely available open-source tool *X!Tandem pipeline* developed at our team;
- by directly providing to MassChroQ spreadsheet text files (*tsv* or *csv*) that contain the identified peptides for each sample (see section 5 for details on the peptides files format);

MassChroQ offers two alignment methods: the OBI-Warp alignment method which uses MS level one retention times only, and the MS2 alignment method which uses MS level 2 retention times.

The quantification results in MassChroQ are sorted by group and sample, allowing comparisons and automatic statistics to be performed easily. Several results file formats are available: gnumeric, tsv, xhtml and masschroqML format.

1.4 Help

On the MassChroQ homepage (http://pappso.inra.fr/bioinfo/masschroq/) you can find :

- a FAQ;
- masschroq's manpage;
- masschroq's schema;
- Dataset examples of masschroqML input files to MassChroQ for various ordinary situations (fractionated sample, isotopic labeled ones, etc);
- this user manual, frequently updated;

- the latest news and the upcoming features on this project;
- BibTeX and text entries for MassChroQ citation.

On the MassChroQ project page hosted on SourceSup you can find :

- a subversion repository;
- a bug tracker;
- several user and developer forums.

The source code of MassChroQ contains C++ *Doxygen* documentation, which you can generate and use for development needs.

Chapter 2

Installing and running MassChroQ

2.1 Installation

2.1.1 Linux platforms (32 and 64 bytes)

All the download possibilities of MassChroQ can be found on
http://pappso.inra.fr/bioinfo/masschroq//download.php

- Debian and Ubuntu: Precompiled binary packages for all MassChroQ versions are available for *32-bit* and *64-bit* systems. Ubuntu's software center automatically installs the package and the dependencies by double-clicking on it.

- Other distributions: to build `masschroq` on Linux you need:

  - Qt 4.5.2 or higher. Most Linux distributions have packages available. In any case, be sure to get the -dev package for Qt in addition to any other libraries.
  - Cmake 2.6 or higher.

  Archives for 32 and 64 bytes systems containing the source code and precompiled Linux binaries for MassChroQ are also available. Decompress the archive and if necessary rebuild the binaries with the commands :

```
cd path_to_masschroq_archive/masschroq-[version_number]
cmake .
make
sudo make install
```

where *[version_ number]* is the MassChroQ version you have downloaded (for example 1.2) This will install masschroq and its schema into your system; you will then be able to run the `masschroq` command from your console. You will also have access to masschroq's manpage by typing `man masschroq`.

2.1.2 Windows platforms

An archive named `masschroq_win.tar.gz` is available for download. You just need extract archive on your system. This archive contains MassChroQ executable, Qt dll and documentation.

On windows platform, only MassChroQ Gui version are provided. You want to try it, download example, open `masschroq_gui.exe`, select an example of masschroqML and run it.

Note 1. *The* masschroqML *format is an XML format. On Windows systems, xml files are by default opened with Internet Explorer or Notepad, and are impossible/difficult to edit. When you need to edit these files, we strongly recommend you to open them with text editors (other than Notepad, try it, you will see why), for example the free open source* Notepad++ *text editor which offers syntax highlighting with a nice look. And if you cannot stand Windows problems anymore, we recommend you the latest Ubuntu Desktop edition (all the advantages of Linux but nice and easy graphical use).*

2.1.3 SVN repository

The subversion repository located at https://subversion.renater.fr/masschroq/ is available read-only to the public at large. Assuming you have at least version 1.0.0 of Subversion installed, you can checkout MassChroQ sources, including latest developments from `trunk` by using the following command :

```
svn checkout https://subversion.renater.fr/masschroq/ my_masschroq_directory
```

2.2 Running MassChroQ

The command-line executable/binary of MassChroQ is called `masschroq`. Once installation is completed you can type `masschroq --help` on the command-line to get the list of the available options and their usage.

The `MassChroQ GUI` launcher executable is called `masschroq_gui`.

The `MassChroQ Studio` executable is called `masschroq_studio`.

On Linux platforms a manpage for `masschroq` is automatically installed with the program.

2.2.1 MassChroQ's XML input file

The most important input parameter for `masschroq` is an XML file. This is where the user defines the LC-MS data to be analyzed and all the different treatments and parameters to be performed on them. This input XML file follows the *masschroqML* format, whose annotated schema can be found on the MassChroQ homepage and also in the `doc/schema/` directory of your installation directory.

A masschroqML input file can be generated automatically from this schema by any XML editor. You can of course use a standard text editor to manually write such a file, MassChroQ performs verification of these files towards its schema and will guide you to the line where syntax errors have been introduced if any.

You can also use the masschroqML example templates we provide on MassChroQ's website and adjust them to your analysis. These templates together with the *masschroq_ complete_ input_ example.xml* of appendix A can also be found in the `doc/xml_examples/` directory of your MassChroQ installation directory. They are also available on the MassChroQ homepage.

To produce a masschroqML input file that also includes all the identified peptides of your data, you can use the freely-available *X!Tandem pipeline*. Given a set of LC-MS runs, this pipeline performs, in one shot, peptide identification using the *X!Tandem* software, filtering of the identification results and export to several formats, one of them being a ready-to-use masschroqML input file.

If you want to perform identification on your data by using other engines, you should use `masschroq`'s `-parse-peptides` option to integrate these identified peptides in your masschroqML input file. For this, you will have to provide one peptide *csv* text file per data. For details on how to use this option see section 2.2.2. For details on the peptide *csv* text file format see section 5.

In section 4 the masschroqML format and the precise impact of every parameter it contains is explained in details.

2.2.2 Peptide identification file parsing option

The *X!Tandem pipeline* way of processing your LC-MS data has a double advantage :

- the identification, filtering and quantification processes are automated and linked through an intuitive graphical user interface;
- `X!Tandem` and the *X!Tandem pipeline* are open-source software, freely available, very reliable and easy to install.

However, if you do not need to use the full pipeline, you can provide your own spreadsheet text files containing the identified peptides for each sample. These

text files can be in *tsv* or *csv* format (tabulation, comma or semi-colon separated values). One peptide text file per data has to be provided. The `-parse-peptides` command-line option makes MassChroQ parse them, combine the same peptides appearing in several data and integrate them in the XML input file.

To achieve this you will have to put in your masschroqML input file the references to the peptide text files to be parsed in the following way :

```
<masschroq>
...
<peptide_files_list>
<peptide_file data="sample0" path="peptides_sample0.txt"/>
<peptide_file data="sample1" path="peptides_sample1.txt"/>
</peptide_files_list>
...
</masschroq>
```

One peptide text file corresponds to a sample data. In the above example the `peptide_sample0.txt` file contains only the peptides identified in `sample0` data. Running `masschroq` on this input file will produce a new XML input file named `parsed-peptides_input_file.xml` containing the original `input_file.xml` with all the identified peptides integrated and organized in the masschroqML format.

Note 2. *At this point MassChroQ automatically continues execution on the newly generated* `parsed-peptides_input_file.xml` *performing the analysis instructions it contains. If you do not want MassChroQ to continue the analysis, but only parse the peptide files, you should use the the* `-parse-peptides` *option as follows :*

```
masschroq --parse-peptides input_file.xml
```

This will produce a new XML input file named `parsed-peptides_input_file.xml` *containing the original* `input_file.xml` *with all the identified peptides integrated and organized in the masschroqML format but will not continue analysis on it.*

2.2.3 Temporary working directory option

While it parses the mzXML/mzML data files, MassChroQ writes the spectra information they contain into temporary files that it accesses during execution time each time it needs them. By default `masschroq` puts these temporary files on its currently working directory. These files are named `masschroq_tmp_file` followed by a randomly generated extension. You should not delete or alter these files while masschroq is working. They will be automatically deleted when MassChroQ finishes working. If you stop MassChroQ abruptly during work, make sure you delete them manually so that they do not overstock on your computer.

To change the directory where MassChroQ puts its temporary working files you can specify one of your choice by using the `-tmp-dir` option on the command-line as follows :

```
masschroq --tmp-dir DIRECTORY input_file.xml
```

MassChroQ will then perform analysis on `input_file.xml` and will put the temporary files in `DIRECTORY` instead of its current working directory.

Note 3. *The total size of temporary files produced during an analysis, is very close to the total size of the data files (mzXML/mzML files) being analyzed . Be careful when specifying another working directory not to choose one that has size limitations.*

Chapter 3

How MassChroQ works

In this section we give an in-depth explanation of MassChroQ's operation. In the following section (4) we give you detailed practical usage information.

3.1 Quantified items

In MassChroQ the user determines operating items of interest on which XIC extraction, peak detection and quantification processes will be performed. These items can be :

- all the identified peptides in the data being analyzed;
- all the identified isotopes in the data being analyzed;
- a given list of mass over charge (m/z) values;
- a given list of couples of mass over charge and retention times values (mz-rt).

The user can choose one or several of the above items. MassChroQ will extract the corresponding XICs in every sample data being analyzed and will perform peak detection and quantification on each of them.

3.1.1 Identified peptides

A peptide in MassChroQ is defined as :

- a unique amino-acid sequence;
- a unique MH value : mass of the peptide plus mass of an H^+ ion).

MassChroQ does not perform identification of peptides/proteins. This has to be performed upstream using the identification tool of your choice. It is up to the user to provide the identified peptides (in case he wants to operate on them) in *tsv* or *csv* (tab, comma or semi-colon separated values) format files (for details on this format see section 5). MassChroQ parses these files and puts the information they contain in its input masschroqML file.

Most of the identification tools (Mascot, X!Tandem, Phenyx) offer the possibility to export identification results in `tsv` files. Our *X!Tandem pipeline* offers the possibility to directly export X!Tandem identification results in a masschroqML input file (no parsing is necessary).

3.1.2 Identified isotopes

If isotopic labeling has been performed and the user needs to quantify all the identified isotopes, he simply describes the different isotopic labelings performed on the different data being analyzed in the masschroqML file. During analysis MassChroQ automatically computes isotopic masses (using these descriptions) and quantifies the desired isotopes.

For a precise explanation of how to describe and quantify isotopes see section 4.5.

3.2 Parsing of LC-MS/MS files in mzXML or mzML formats

MassChroQ can parse mzXML as well as mzML LC-MS files. Due to the simpler, far more robust and stable nature of the mzXML format, we highly recommend its use in MassChroQ instead of the mzML one.

MassChroQ does not validate mzXML and mzML files against their respective schema (it is not its goal), the user has to provide valid files. Usually, most of the proteomic pipeline tools that convert raw data to mzXML/mzML format produce valid files.

In both formats, MassChroQ parses all the MS levels it finds (1, 2 and greater). If the items to be quantified are the identified peptides, LC-MS run files should contain MS levels 1 and 2 in order for MassChroQ to work. Indeed, MassChroQ uses the MS/MS information to compute the real observed retention times of these peptides in each run. This retention time will be used later during peak matching to assign the computed quantitative value to the right corresponding observed peptide and to avoid false assignments.

MassChroQ automatically decodes the base64 encoded spectra in both mzXML and mzML formats, in both 32 or 64 bytes precision. MassChroQ processes nei-

ther compressed spectra, nor compressed mzXML/mzML files. If this feature is important to you, please let us know and we will try to implement it sooner than scheduled.

3.3 Grouping of LC-MS runs

MassChroQ uses the notion of *groups* of LC-MS data according to their technical similarities : for instance, in case of fractionation, samples of the same fraction will be grouped together in order to be aligned together, or a group of samples obtained from an LTQ low resolution spectrometer can be grouped together so that the appropriate XIC extraction range and XIC filtering can be applied on them, etc. It is up to the user to define the different groups of LC-MS runs in its analysis.

Here are the main grouping possibilities that can be performed in MassChroQ, according to the analysis needs.

Fractions grouping

If peptide or protein pre-fractionation has been performed on your samples, samples of the same fractions should be grouped together. For example, in a peptide SCX fractionation experiment, suppose you have 3 samples A, B and C, each fractionated in 10 fractions: A_1, A_2 ... , A_{10}, B_1, ..., B_{10}. You should define 10 groups, the first one containing the samples A_1, B_1 and C_1, the second one containing samples A_2, B_2 and C_2 and so on. This way, only the same fractions will be aligned to each other.

Alignment grouping

All runs of the same group will be aligned together with the same alignment method. Thus, only samples presenting technical similarities should be grouped and aligned together. In the fractionation example above, only samples of the same fractions can be aligned one to another, it has no sense to align fractions in 2 with those in 3 or 1.

Another case of use of the alignment grouping is when you have two differently obtained experiments, for instance some samples obtained from a low resolution spectrometer, and some others obtained from a high resolution one. The alignment parameters can be adjusted to each of these experiments: for example the high resolution samples have much more MS level 2 acquisition points, thus an MS2 alignment method with bigger smoothing window parameters than for the low resolution experiment, can be more appropriate and give better aligned retention

times. You can run an analysis with MassChroQ on both experiments in one shot by simply defining two different groups and two alignment methods, one for each group.

Quantification grouping

XIC extraction, XIC filtering, peak detection and quantification will be performed in all the runs of the same group with the same quantification method.

So, in the example of the high and low resolution experiments above, the XIC extraction range parameter, which depends on the spectrometer's range, should be different for each experiment. Also, some low resolution spectrometers generate an important baseline noise, whereas some high resolution ones generate spikes. Thus the XIC filtering should be different for each experiment. By grouping the low resolution samples together and the high resolution samples in a different group, by defining two different quantification methods, one for each group of samples, we can perform specialized analysis on our two groups in one shot.

Extra features of the grouping

Groups give the user the possibility to perform specialized analysis on several different sets of data in one shot. But technically speaking, they also allow the following extra features in MassChroQ:

Efficient XIC extraction: XICs for a given identified peptide will only be extracted in groups where the MS/MS allowed its identification, no unnecessary extractions will be performed.

Smart quantification: peptides identified in at least one run of the group, will be quantified in every run of this group, including those where they have not been identified. See section 3.8.2 for more details on how this is done in MassChroQ.

The final quantification results in MassChroQ are sorted by group and by run, associating to each identified peptide (or other chosen entity) its quantitative value in every group and in every run of the analysis. They allow easy statistical analysis without ambiguity.

3.4 XIC extraction

The underlying operating items in MassChroQ are the XICs. Whatever operating item the user has chosen to analyze (identified peptides, isotopes, mz or mz-rt

values), MassChroQ extracts and analyzes the XIC corresponding to the mz of this item.

More precisely, in the XIC of a given item of interest, the mz of this item is extracted from the entire LC-MS run. The intensity curve (within a mass tolerance range centered on the given mz) is plotted for every chromatographic retention time in the analysis.

The size of the mass tolerance window depends on the mass accuracy and the mass resolution of the spectrometer. In MassChroQ the user can define it as a XIC extraction parameter (`mz_range` and `ppm_range` parameters).

The intensity of XICs in MassChroQ can be represented in two different ways:

- as the summed intensity across the range of masses (also called TIC : Total Ion Current chromatogram);
- as the most intense (maximal) peak in the range of masses (base peak chromatogram).

The user can choose one of this representations in the XIC extraction parameters.

Note 4. *In MassChroQ we have purposefully chosen to perform quantification on the extracted XICs as explained above, rather than on feature detection on the 2D virtual image which many other software use. Indeed, the latter needs high resolution in MS mode in order to be able to identify isotopic profiles. By contrast, quantification based on XICs can be used with low-resolution as well as with high-resolution mass spectrometers by simply adapting the window size of XIC extraction.*

3.5 XIC filtering

The following XIC filters are implemented in MassChroQ :

- The background filter : this is a median filter by default. If the user wants he can add an open (max/min) morphological filter. Both of this filters are widely used for baseline signal noise removal.
- The smoothing filter : a moving window average filter that smooths the signal. This filter can be useful in rare cases of very noisy signals, but as the Zivy detection method already performs a temporary smoothing filter before peak detection, this filter is usually unnecessary.
- The anti-spike filter : removes artifact spikes that some HR spectrometers (e.g. Orbitrap) introduce in the signal.

3.6 Peak detection

Once the XICs are extracted, MassChroQ detects all the peaks on them. He then computes the peak boundaries and integrates the peak area, the latter being the quantitative value.

The following peak detection methods are possible in MassChroQ :

- the *Moulon peak-detection* method : a simple threshold peak detection method;
- the *Zivy peak-detection* method : a combination of open-close morphological filtering of the signal with a local maxima detection algorithm using thresholds.

3.6.1 The Moulon peak detection algorithm

The Moulon peak detection is the first historical detection method in MassChroQ. In practice this method has been widely replaced by the Zivy one, and is likely to disappear in the future. This method looks for local maxima on the XIC beginning from a threshold parameter defined by the user. This threshold should be superior to the global background noise, so to avoid detection of maxima in the background noise. In details, the algorithm proceeds as follows:

- it applies an optional moving average filter to smooth XIC intensities (the moving window being a user-defined parameter);
- browses intensity signal in ascending retention time order and when it reaches the `tic start` intensity value parameter it begins detection of a maximal local intensity;
- ends detection of the maximal local intensity when reaching the `tic stop` intensity value threshold parameter.

The `tic start` and `tic stop` threshold parameters should be slightly superior to the background noise of your data.

3.6.2 The Zivy peak detection algorithm

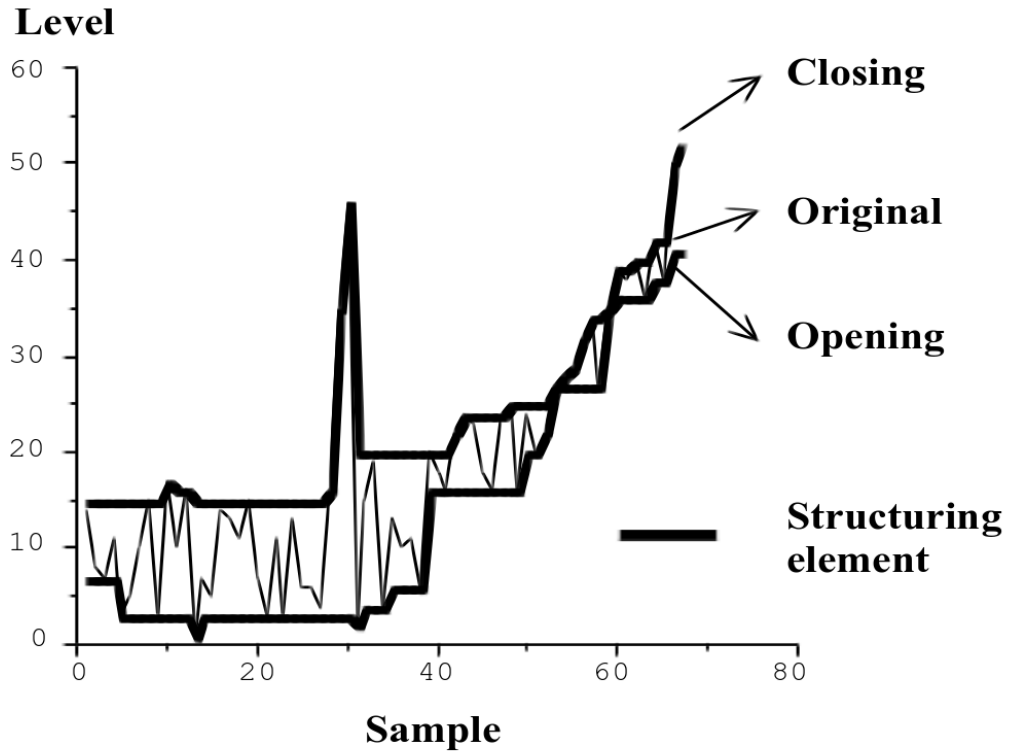
The Zivy peak detection method has widely replaced the Moulon one in practice in our laboratory, giving much more accurate and precise results.

Note 5. *The Zivy peak detection method is a peak localization method: its purpose is to determine the peak positions and the peak boundaries on the signal. Peak intensities an peak area are then computed on the original unaltered signal.*

This method uses morphological opening and closing signal transforms with small flat linear structural elements (also known as respectively max/min and min/max transforms). Mathematical morphology was born in 1964 from the collaborative work of Georges Matheron and John Serra at the *Ecole des Mines de Paris* (see [2]). Since then, morphological transforms have been widely used in image processing to remove noise and to detect peaks or edges showing their efficiency in particular in noisy signals (see [3] and [4]).

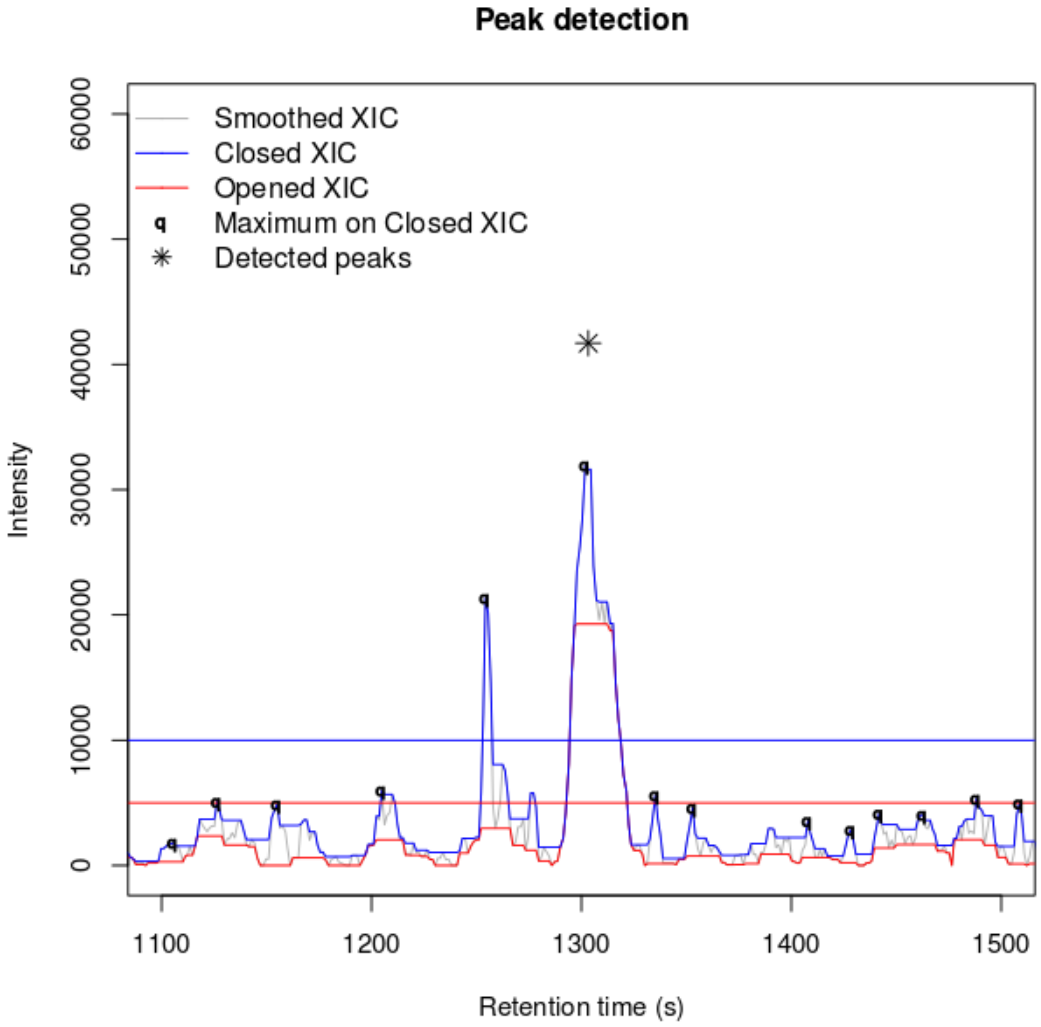
Note 6. *On a one-dimensional signal, the open (resp. close) transform with a flat linear structural element (i.e. a segment) of size R is equivalent to replacing the signal values at every point by the maximum of the minimum (resp. minimum of the maximum) of all the points in a neighborhood of radius R (see [5] and [6]). This is what we do in MassChroQ.*

Schematically, as illustrated in the figure below, opening and closing transforms with flat linear structural elements both smooth and simplify the original signal: opening removes small peaks by flattening them from the top, and closing fills small holes by filling them from below.



The opening eliminates peaks that are thinner than the structural element. By choosing an appropriate size of radius R the opening separates background signal from relevant one. Indeed, thin background peaks are eliminated, but in return

intense relevant peaks are flattened, which makes the opening not suited for peak detection. In contrast, the closing eliminates valleys thinner than the structural element and not only it perfectly preserves peaks, but it also smooths them and clarifies its boundaries. You can see it on the following figure which comes from a real trace file produced by MassChroQ.



Here is how the Zivy peak detection algorithm works:

- Peak localization in MassChroQ is performed on the closing signal, which preserves peaks but also clarifies its boundaries. More precisely, local maxima intensity positions are detected on the closed signal.
- These maxima are then filtered using two intensity thresholds; only the ones that overpass the thresholds are retained.

Threshold on the open signal: If a local maxima position reported on the open signal does not exceed in intensity the open threshold it is eliminated. This eliminates intense and thin noise peaks (almost spikes) that are flattened by the open transform given their thinness.

Threshold on the closed signal: If a local maxima position reported on the close signal does not exceed in intensity the threshold on the close signal, it is eliminated. This eliminates very wide noise peaks (hence not flattened by the open transform) but not intense.

- The final retained local maxima are the peak positions in MassChroQ. We use this positions to compute peak boundaries on the closed signal and also peak intensity, and peak area on the original signal. Indeed, morphological transforms in MassChroQ are solely used to detect peak positions: the peak intensity and peak area are computed on the original signal using these positions.

Note 7. *In the figure above one can see different interesting cases: for example the very intense peak on the left of the unique detected peak is eliminated because its intensity in the open signal does not exceed the open threshold (too thin to be a relevant peak). This peak is indeed a noisy pulsing spectrometer effect.*

Algorithm 3.1: The Zivy peak detection

```
1  input: X a XIC (i.e. a set of (retention time, intensity) values);
2         detection_threshold_on_max, detection_threshold_on_min : integers;
3  output: the set 𝒫 of the detected peak positions on X
4
5  apply a moving average filter on the X intensities (optional);
6  compute 𝒪_X: the open transform of X;
7  compute 𝒞_X: the close transform of X;
8  compute Max_{𝒞_X}: the set of (retentiontime, intensity) points
9  corresponding to the local maximum intensities on the close
10 𝒞_X curve signal;
11 forall point P in Max_{𝒞_X} do
12   let P_{𝒪_X} be the point having the same coordinates that
13   P in the open curve signal
14   if ( intensity(P_{𝒪_X}) > detection_threshold_on_min and
15        intensity(P_{𝒪_X}) > detection_threshold_on_max )
16          put P_{𝒪_X} in the set 𝒫 of detected peak positions
17   endif
18 endfor
19 return 𝒫.
```

Here follows a line-by-line description of the Zivy peak detection algorithm:

Line 5: optionally smooth the signal with a moving average filter;

Lines 6-7: compute the open and close transforms of the signal;

Line 8: perform a local maximum detection on the intensities of the closed signal; we obtain $Max_{\mathscr{C}_X}$, a preliminary set of (retention time, intensity) potential peak positions.

Lines 11-18: for every such (retention time, intensity) point:

- check that the corresponding retention time point in the opened signal curve $\mathscr{O}_X$ has a greater intensity than $detection\_threshold\_on\_min$;
- check that the corresponding retention time point in the closed signal $\mathscr{C}_X$ has a greater intensity than $detection\_threshold\_on\_max$;

The final peak positions are the points that verify both of these conditions.

The Zivy peak detection method is inspired in part by the morphological *top-hat* peak detection method first introduced in [7].

3.6.3 Peak boundaries and area

In the Zivy peak detection case

Peak boundaries are computed during the detection algorithm, just after having selected the final peak positions. The boundary positions are computed on the closed signal because this signal does not preserve intensity values, but preserves retention time positions while making their detection more accurate. These boundary positions are then reported on the original unaltered signal where peak area integration is performed.

In the Moulon peak detection case

The peak boundaries for each peak are the `tic start` and `tic stop` positions. Peak area integration is computed between them as above.

3.7 Alignment

LC-MS instruments do not trigger MS/MS at exactly the same retention time in every sample, and chromatographic retention time distortions occur naturally between runs. Thus, RTs must often be aligned between similar samples before peak matching.

MassChroQ performs alignment of samples of the same group. For each group of runs to be aligned the user chooses a *reference alignment run* : the reference run against which all the other runs of the group will be aligned. It stands to reason that the user should choose a representative run as reference alignment run, otherwise all the runs of the group will present abnormal retention time deviations affecting peak matching and quantification values.

Two different alignment algorithms are implemented in MassChroQ : the OBI-Warp alignment method and the MS2 alignment method. Each alignment method can produce a `.time` file for each MS run it has aligned, if the user wants to. The MS2 alignment method can also produce a `.trace` file. For example, if the MS run file BSA1.mzXML has been aligned, the alignment can create a BSA1.time file containing the old and the aligned retention times and a BSA1.trace file containing all the time values operations. These .time files can be used for the next MassChroQ analysis of the same MS run: the user does not have to perform alignment the next time, he provides the older .time file, so the aligned retention times it contains will automatically be taken in account by MassChroQ.

3.7.1 The OBI-Warp alignment method

The OBI-Warp (Ordered Bijective Interpolated Warping) is an external integrated library developed by John T.Prince in the University of Texas at Austin. This method is based on the MS level 1 data solely. It considers the spectra as matrixes that it aligns along the (MS level 1) retention time axis by using dynamic time warping and a one-to-one interpolated warp function.

For more information on the OBI-Warp alignment library see http://obi-warp.sourceforge.net/ and [8].

3.7.2 The MS/MS alignment method

The in-house developed MS/MS alignment method is based on MS/MS (MS level 2) data, hence it is not possible to use this method with data not containing MS/MS acquisition.

This method uses MS/MS identifications as landmarks to evaluate time deviation along the chromatography. More precisely, suppose we want to align two runs. We first calculate the retention time deviation of the MS/MS identified peptides these two runs have in common. Then, by linear interpolation, we use this deviation curve to calculate a tendency deviation curve of the MS level one retention times. Of course this deviation curve is accurate only if there are enough points that allowed its shape, i.e. if there are enough common identified peptides in the two runs. For each sample to be aligned, MassChroQ will output the number of shared peptides. Also, the MS2 alignment method can output a `.trace` file for each run aligned. These files contain the traces of every alignment step and can be used for precise checking and plotting of the alignment procedure and parameter adjustment.

Here follows a precise explanation step by step of the MS/MS alignment algorithm. Let Run_1 be the run whose MS level one retention times are going to be aligned towards the MS level one retention times of the reference run Run_{ref}. We will use the following notation conventions :

- RT_{ms1} denotes a set of MS level one retention times and RT_{ms2} a set of MS/MS retention times.
- $RT_{ms2}(Run_1)$ denotes the set of the retention times of the identified peptides during MS/MS acquisition in Run_1.
- rt_{ms1} denotes a retention time member of the RT_{ms1} set.

Algorithm 3.2: MS/MS alignment of $RT_{ms1}(Run_1)$ towards $RT_{ms1}(Run_{ref})$

```
1  input: RT_ms1(Run_1); RT_ms1(Run_ref);
2         RT_ms2(Run_1) ≠ ∅; RT_ms2(Run_ref) ≠ ∅;
3  output: alignedRT_ms1(Run_1) set of values
4
5  get the list Peps of peptides identified in both Run_1 and Run_ref;
6  forall pep in Peps do
7    Δrt_ms2(Run_1) = rt_ms2(Run_1) − rt_ms2(Run_ref)
8  endfor
9  to each rt_ms2 in Run_1, associate its Δrt_ms2(Run_1) as computed above;
10 let (RT_ms2, ΔRT_ms2)(Run_1) be the set of all such pairs;
11 apply a moving median filter on the ΔRT_ms2 set (optional);
12 apply a moving average filter on the ΔRT_ms2 set (optional);
13 add first and last elements to the (RT_ms2, ΔRT_ms2)(Run_1) set (by
14 extrapolation of MS1 retention time values) as follows:
15    first−rt_ms2 = first−rt_ms1 − 1 and Δfirst−rt_ms2 = Δfirst−rt_ms1
16    last−rt_ms2 = last−rt_ms1 + 1 and Δlast−rt_ms2 = Δlast−rt_ms1
17 forall rt_ms1 in RT_ms1(Run_1) do
18   compute a corresponding Δrt_ms1(Run_1) value by linear interpolation on
19   the set of (RT_ms2, ΔRT_ms2)(Run_1), i.e.:
20   Δrt_ms1 ← linear interpolant of (rt^1_ms2, Δrt^1_ms2) and (rt^2_ms2, Δrt^2_ms2)
21   where rt^1_ms2 and rt^2_ms2 are the two nearest surrounding points to rt_ms1
22 endfor
23 apply a moving average filter on the such obtained ΔRT_ms1 set (optional);
24 forall rt_ms1 in RT_ms1(Run_1) do
25   aligned_rt_ms1(Run_1) = rt_ms1 − Δrt_ms1
26 endfor
27 automatically check the alignedRT_ms1(Run_1) curve slope and
28 if necessary, correct it;
29 return the set of aligned_rt_ms1(Run_1) values.
```

Step 1 (line 5): here we get the retention times of all the common peptides identified in both runs. The computation of peptide retention times follows the *real_ or_ mean best Rt* method explained in section 3.8.

Step 2 (lines 9-10): to each peptide MS2 retention time in Run_1 we associate its deviation from the peptide's MS2 retention time in Run_{ref}.

Step 3 (lines 11-12): smoothing of this deviation curve (optional): we first apply a moving median filter followed by a moving average filter.

Step 4 (lines 13-16): compute the first and last rt_{ms2} and $\Delta rt_{ms2}(Run_1)$ elements by linear extrapolation on the first and last $rt_{ms1}(Run_1)$ points.

Step 5 (lines 17-22): for each rt_{ms1} level one retention time in Run_1 to be aligned, we compute a corresponding Δrt_{ms1} value by linear interpolation on the set of $(RT_{ms2}, \Delta RT_{ms2})(Run_1)$ level 2 retention times and associated deviations(computed in Step 2). The linear interpolation is done on the two rt_{ms2} values that surround this rt_{ms1}.

Step 6 (line 23): smooth the such obtained Δrt_{ms1} points by applying a moving average filter (optional);

Step 7 (lines 24-26): for each original MS1 retention time in Run_1 we compute its corresponding aligned MS level one retention time.

Step 8 (line 27-28): the above mathematical computations do not necessarily preserve the ascending order of time values. But the computed aligned MS1 retention times must be in ascending order (as the original retention times are). Despite the smoothings, sometimes this is not the case at this point. Hence, the algorithm always checks the ascending order of the aligned values and automatically applies a correction to ensure it if needed. This correction is based on the slope of the original retention time curve. The correction is performed as follows:

- compute the correction value as the slope of the original retention time curve ($rt_{ms1}(Run_1)$) (divided by 4 for a finer correction);
- whenever two consecutive aligned retention time values are not in increasing order, we replace the smaller one with the biggest one plus the correction value.

3.7.3 Alignment reuse

Reuse of previous alignments

For each LC-MS run being aligned, you can indicate in the masschroqML file a directory where to create a text *tsv* `run_filename.time` file containing the original retention times of the run and the new corresponding ones after alignment. You can

reuse these .time files in a future run of MassChroQ (by indicating their directory in the masschroqML files). This way you do not have to repeat an alignment previously performed. Also, this allows you to provide alignment values generated by other external alignment tools.

Note 8. *For this to work, the masschroqML input file should not contain an alignment instruction (the <align> tag in the masschroqML file) that concerns the runs whose .time file have been provided. Indeed, if an alignment instruction asks MassChroQ to align a sample whose* `.time` *file has been loaded before, he destroys these* `.time` *values and performs the asked alignment instead. Thus, the alignment instructions in the masschroqML input files have the priority over the preloaded* `.time` *alignment files.*

3.8 Peak matching

After alignment, for each peptide being quantified, MassChroQ performs XIC extraction and peak detection on it. Remember that a XIC extracted for a given peptide, is the intensity curve of the peptide's m/z during the whole chromatographic retention time. So, not all the peaks detected on this XIC correspond to the retention time this peptide has been identified in the MS-run being quantified. Moreover, after retention time alignment, the retention time of the peptide can change.

That is why MassChroQ performs peak matching on each detected peak: the detected peaks for a peptide are assigned/matched to the peptide they belong to. This peak matching is based on retention times and it is performed as follows: the peak is assigned to a peptide if and only if the RT of this peptide (after alignment if any) is within the boundaries of this peak.

What is the RT of a peptide in a given MS-run?

In a given run a peptide can be identified or not. In case it has been identified, it can be identified at several distinct chromatographic retention times: in that case MassChroQ computes its *real RT* following the *best RT* method explained below. In case the peptide has not been identified in the run, its *mean RT* is computed following the *smart quantification* method which allows quantification of peptides even in the runs where they have not been identified (provided that they have been identified in another run of the same group).

The user can choose between three different modes in MassChroQ to perform peak matching :

- the *real_ or_ mean* mode;
- the *mean* mode;

- the *post_ matching* mode.

The two first modes correspond to the computation mode of the RT of the peptide whose peaks are being matched. The last mode is more complex. Here follows an explanation of each of this peak matching modes.

3.8.1 The *real RT* mode and the best RT method

A given peptide can be observed/identified at several different retention times in a given run, with different intensities or charge states. During parsing, MassChroQ will get from the LC-MS mzXML or mzML file, all the retention times the peptide has been identified in and the corresponding precursor intensities. He will then retain only the retention time corresponding to the most intense occurrence of this peptide. This will be the retention time of this peptide for this run during the rest of the analysis. We refer to this retention time as the *real retention time* of the peptide in the MS-run, meaning the peptide has been really observed in this run. We refer to the computation method above as the *best RT* method.

3.8.2 The *mean RT* mode and smart quantification

If a given peptide has never been observed/identified in a given run, MassChroQ nevertheless computes a retention time for this peptide in this run as the mean of its *real retention times* in the other runs of the same group. To be more precise, MassChroQ computes the mean of all the best RTs of the peptide in the runs of the group where the peptide was observed/identified in.

This way, MassChroQ is able to align and quantify a given peptide even in runs where it has not been identified. It suffices that this peptide has been identified in at least one run of the same group. In the same way, during peak matching in the samples, where a peptide has not been identified, MassChroQ will nevertheless try to assign the detected peaks to the *mean RT* of this peptide. We call this feature *smart quantification*.

3.8.3 The *real_ or_ mean RT* peak matching mode

In the *real_ or_ mean RT* mode the RT of a peptide in a given MS run is its *real RT* if the peptide has been identified in this run, or its *mean RT* if not. This RT is then used to assign the detected peaks to this peptide as follows : for each peptide in each MS run, for each detected peak on the XIC of this peptide, if the peptide's *real_ or_ mean RT* in this run is within the peak's rt boundaries, than the peak is assigned to this peptide.

3.8.4 Peak post-matching mode

In both previous peak matching modes (*real_ or_ mean* and *mean* mode), peak matching is performed peak after peak during quantification, by matching the peptide retention times (MS/MS level retention times) to the peak's retention time interval (MS level retention times). A flaw in this method is that sometimes the peptide retention times (MS/MS) are not necessarily close to the corresponding peak's retention time interval(MS), which can cause match failure. This could be avoided if, When matching a new peak to a peptide, we considered the previously matched peaks for that same peptide in the same group. The previously matched peaks of this peptide can indeed give us better peptide retention times. Whenever a peak is matched to a peptide, we could assign to this peptide the retention time corresponding to the maximum intensity of this peak. Let us call it the *best matched RT*. In comparison to the *real_ or_ mean RT* which is given by the triggered MS/MS of our spectrometer, the *best matched RT* should be closer to the peak's center. But we could do even better! If we could have all the matched peaks for a peptide in every run of a given group, we could compute the mean of their *best matched RTs* over all the runs of the group, before trying to rematch the previously unmatched peaks.

That is what the post-matching mode does. More precisely, here is how it is performed:

- A first peak matching pass is performed during quantification, peak after peak, as in the previous versions of MassChroQ, by matching the peptide retention time with the peak's retention time interval. The retention time of the peptide being matched is its *real RT* in every MS-run it has been identified in. This means that no peak matching is performed during this step in the runs where the peptide has not been identified.

- During the previous first pass, if a peak is matched, we compute the retention time corresponding to the maximum intensity of this peak. We will call it the *best matched RT* of the peptide in this MS run. If a peak is not matched, we put it beside and keep it for the second peak matching pass.

- After quantification is finished in the current group, we perform another peak matching pass (followed by another quantification round of the newly matched peaks) on the previously unmatched peaks. For every peptide in every run of the group, we try to rematch the unmatched peaks of this run to the *best matched RT* of this peptide in the group. The *best matched RT* of a peptide in a group is the mean value, over all the runs of the group, of the previously defined *best matched RTs* of this peptide in each run of the group, if any. (If there is no *best matched RT* for a peptide in a given run, we keep its *real RT* if any for the computation of the mean.)

During the second peak matching pass, by taking into account the maximum intensity RTs of the previously matched peaks, the matched retention time is often more accurate, more close to the center of the peak, so that the probability of missing a relevant peak is lower.

3.8.5 What peak matching mode should I choose?

The *mean* peak-matching mode is present in MassChroQ for historical reasons mainly, and is no more used in practice in our laboratory. It was the first peak matching mode to have been implemented and used with the Moulon peak detection method. This mode is likely to disappear in the future.

The peak *post-matching* feature allows matching of thin relevant peaks, usually in samples where the peptide has not been identified (because of this thinness), and has not been matched during the first matching pass. In the post-matching pass, the chances to be closer to the center of the peak are higher ans so are the chances to match thin peaks. But there is always a risk of peak misassignments. According to numerous tests performed in our laboratory, the peak post-matching feature is interesting in cases of samples acquired in high resolution systems. Indeed, in these cases, the peaks are well defined and the signal noise is less important, so the probability to find previously missed peaks is higher, and the risk to misassign background noise peaks to peptides is as low as in the previous matching modes.

On the other side, its use in case of samples acquired with low resolution systems is not recommended. Indeed in low resolution samples the background noise peaks are much more present. This increases the risk of peak misassignments during the peak post-matching process. In these cases, the *real_ or_ mean* mode is recommended.<

Chapter 4

The masschroqML format

In this section we give a detailed practical description of `masschroqML`, MassChroQ's XML input file format. It is in this file that the user describes to MassChroQ all the relevant information about the analysis he wants to perform: the data file paths, the groups of data files, the alignment and detection methods to use, the peak matching method and other options.

To illustrate the masschroqML format we will use the complete example file called *masschroq_complete_input_example.xml* included in appendix A. This file is also located in the `doc/xml_examples` directory of your MassChroQ installation directory.

Here follows an explanation line by line of this file.

4.1 Data files

In the `<rawdata>` block the user defines the mzXML/mzML data files to analyze. The `time_values_dir` attribute is optional, when present it tells MassChroQ the directory where to find the .time files to load. The retention time values in these directory will be the ones that MassChroQ will use in the rest of its analysis.

```
<rawdata time_values_dir="masschroq_tests/time">
<data_file id="samp0" format="mzxml" path="bsa1.mzXML" type="centroid"/>
<data_file id="samp1" format="mzxml" path="bsa2.mzXML" type="profile"/>
<data_file id="samp2" format="mzml" path="/home/user/bsa3.mzml"
   type="profile"/>
<data_file id="samp3" format="mzml" path="/home/user/bsa4.mzml"
   type="profile"/>
</rawdata>
```

For each file to analyze a `<data_file>` tag line has to be present. The `<data_file>` tag gives MassChroQ all the information needed to identify and

find a file on the system. It contains the following attributes:

- The `id` attribute is a unique name assigned to the data file in order to reference it later in the document. It is mandatory.
- The `format` attribute indicates the format of the data file. It can be `mzxml` or `mzml`. It is mandatory.
- The `path` attribute is the absolute or relative path to the physical file on your local system.
- The `type` attribute informs the user about the type of data acquisition of this file. It can be `profile` or `centroid`. This attribute is optional, MassChroQ does not use it.

In the above example four data files have been defined : `samp0` in mzxml format, `samp1` in mzxml format, `samp2` and `samp3` in mzml format.

The `<rawdata>` block is mandatory in a masschroqML file.

4.2 Groups

In the `<groups>` block the user defines groups of data files.

```
<groups>
<group data_ids="samp0 samp1" id="G1"/>
<group data_ids="samp2 samp3" id="G2"/>
</groups>
```

- The `data_ids` attribute is the list of space separated data files identifiers that form this group. It is mandatory.
- The `id` attribute is a unique name assigned to the group in order to reference it later in the document. It is mandatory.

In the above example we have defined group `G1` containing the previously defined data `samp0` and `samp1` and the group `G2` containing `samp1` and `samp2`.

The `<groups>` block is mandatory in a masschroqML file.

4.3 Peptide text files

In the <peptide_files_list> block the user defines the path to the peptide csv text files containing information about the identified peptides and proteins in the data being analyzed. These peptide files will be parsed and analyzed by MassChroQ.

```
<peptide_files_list>
<peptide_file data="samp0" path="bsa1_peptides.txt"/>
<peptide_file data="samp1" path="bsa2_peptides.txt"/>
<peptide_file data="samp2" path="bsa3_peptides.txt"/>
<peptide_file data="samp3" path="bsa4_peptides.txt"/>
</peptide_files_list>
```

- The data attribute contains the data id this peptide file corresponds to. One peptide text file per data file should be provided. It is mandatory.
- The path attribute is the path of the peptide file in the local system. It is mandatory.

In this example, we are telling MassChroQ that the file bsa1_peptides.txt contains the identified peptides in data file bsa1.mzxml called samp0; the file bsa2_peptides.txt the ones identified in sample samp1.

The <peptide_files_list> block is not mandatory in a masschroqML file.

4.4 Identified peptides and proteins

Once it has parsed the given peptide identification files, MassChroQ automatically puts the results in the following form :

```
<protein_list>
<protein desc="conta|P02769|ALBU_BOVIN SERUM ALBUMIN PRECURSOR."
   id="P1.1"/>
<protein desc="conta|P02770|ALBU_RAT SERUM ALBUMIN PRECURSOR."
   id="P1.2"/>
</protein_list>
<peptide_list>
<peptide id="pep0" mh="1463.626" mods="114.08" prot_ids="P1.1"
   seq="TCVADESHAGCEK">
<observed_in data="samp0" scan="655" z="2"/>
<observed_in data="samp1" scan="798" z="2"/>
</peptide>
<peptide id="pep1" mh="1103.461" mods="57.04" prot_ids="P1.1"
   seq="ADESHAGCEK">
<observed_in data="samp3" scan="663" z="2"/>
</peptide>
</peptide_list>
```

It creates two blocks :

- the `protein_list` block containing the description of each identified protein;
- the `peptide_list` block containing for each identified peptide :
  - a unique id (mandatory),
  - the amino-acid sequence (mandatory),
  - the MH value (mass of the peptide plus mass of an $H+$ ion); the MH already takes into account the amino-acids modifications, (mandatory);
  - the `mods` attribute giving optional information about the amino-acid modifications on this peptide (here the total mass of the modification) (optional);
  - the id-s of the identified proteins this peptide belongs to (mandatory);
  - a list of `observed_in` elements that indicate the data file, the scan number and the charge state this peptide has been observed in (mandatory).

Note 9. *The user does not have to fill the* `<protein_list>` *and* `<peptide_list>` *blocks:*

- *The* X!Tandem pipeline*, when asked to export its results in masschroqML format, automatically creates these two blocks in this file.*
- *When you ask MassChroQ to parse peptide text files with an input file containing a* `peptide_files_list` *block as described in the previous section, it will create a new file named* `parsed-peptides_input_file.xml` *already containing the above peptide and protein blocks.*

The peptide and protein list blocks are not mandatory in a masschroqML file.

4.5 Isotope labels

The `isotope_label_list` block contains the list of the isotopic labels performed on the data being analyzed if any.

```
<isotope_label_list>
<isotope_label id="iso1">
<mod at="Nter" value="28.0"/>
<mod at="K" value="28.0"/>
</isotope_label>
<isotope_label id="iso2">
<mod at="Nter" value="32.0"/>
<mod at="K" value="32.0"/>
</isotope_label>
</isotope_label_list>
```

- The `id` attribute uniquely identifies an isotope label (mandatory).
- The `at` attribute indicates where the label has been performed (mandatory).
- The `value` attribute indicates the mass modification value for this label (mandatory).

In the above example we have defined two isotope labels : label `iso1` consisting in two modifications at Nter and K both with mass modification value of 28; and label `iso2` label containing Nter and K modifications each of value 32.

Note 10. *Different `isotope_label` elements can be defined, for example to perform quantification on different groups of differently labeled samples.*

Note 11. *The identified peptides in the `peptide_list` block do not take into account the isotopic labelings (their MH value is not the modified one). MassChroQ computes their isotopic masses during quantification (if quantification on isotopes has been asked) using the information contained in the `isotope_label_list` block if any.*

The `isotope_label_list` block is not mandatory in a masschroqML file.

4.6 Alignments

```
<alignments>
<alignment_methods>
<alignment_method id="my_ms2">
<ms2 write_time_values_output_dir="masschroq_tests/time">
<ms2_tendency_halfwindow>10</ms2_tendency_halfwindow>
<ms2_smoothing_halfwindow>5</ms2_smoothing_halfwindow>
<ms1_smoothing_halfwindow>3</ms1_smoothing_halfwindow>
</ms2>
</alignment_method>
<alignment_method id="my_obiwarp">
<obiwarp write_time_values_output_dir="masschroq_tests/time">
<lmat_precision>1</lmat_precision>
<mz_start>500</mz_start>
<mz_stop>1200</mz_stop>
</obiwarp>
</alignment_method>
</alignment_methods>
<align group_id="G1" method_id="my_ms2" reference_data_id="samp0"/>
<align group_id="G2" method_id="my_obiwarp" reference_data_id="samp2"/>
</alignments>
```

In the `alignments` block we define the different alignment methods that we will use to align our different groups of data. In this example we have defined two alignment methods:

- an `ms2` type alignment method with id `my_ms2` (mandatory);
- an `obiwarp` type alignment with id `my_obiwarp` (mandatory).

For each alignment method we have set the appropriate parameters, for example for the `my_ms2` method we have set the half moving window used to smooth the MS/MS retention times to 5. For details on every alignment parameter see the parameters cheat-sheet on section 6.

For both alignment methods, the `write_time_values_output_dir` attribute is optional, when present it indicates to MassChroQ that the user wants, for each run aligned, a .time and a .trace file written to the given directory.

Note 12. *In the `align` element (lines 57 and 58), we give MassChroQ the order to perform alignment on groups G1 and G2 using respectively the `my_ms2` method and the `my_obiwarp` method. All the attributes here are mandatory.*

Note 13. *The* `reference_data_id` *attribute (lines* 57 *and* 58*) indicates the data file in the group that will be used as a reference for the alignment : the other samples of the group will be aligned towards this reference sample. Thus, the choice of the reference sample in alignments is very important.*

The `alignments` block is not mandatory in a masschroqML file.

4.7 Quantification methods

In the `quantification_methods` block we define the different quantification methods we will use in this analysis.

```
<quantification_methods>
<quantification_method id="my_qzivy">
  <xic_extraction xic_type="sum">
  <mz_range max="0.3" min="0.3"/>
  </xic_extraction>
  <xic_filters>
  <anti_spike half="5"/>
  <background half_mediane="5" half_min_max="15"/>
  <smoothing half="3"/>
  </xic_filters>
  <peak_detection>
    <detection_zivy>
    <mean_filter_half_edge>1</mean_filter_half_edge>
    <minmax_half_edge>3</minmax_half_edge>
    <maxmin_half_edge>2</maxmin_half_edge>
    <detection_threshold_on_max>5000 </detection_threshold_on_max>
    <detection_threshold_on_min>3000 </detection_threshold_on_min>
    </detection_zivy>
  </peak_detection>
</quantification_method>
```

As we can see in the lines above, a quantification method consists in a XIC extraction method, XIC filters and a peak detection method. For each quantification method we define:

- A unique quantification `id` used to reference this method (mandatory).
- The `xic_type` : the type of the XIC extraction. It can be `sum` if we want the XICs to be extracted by computing the sum of the intensities, or `max` for the maximum intensity (mandatory).

- The size of the mass tolerance window of extraction in `ppm_range` or `mz_range` (mandatory choice).
- A list of XIC filters to apply (optional).
- A peak detection method among `detection_zivy` and `detection_moulon`.

In the above example we have defined a quantification method called `my_qzivy` that :

- extracts XICs by computing the sum of the intensities;
- uses an mz range for extraction of 0.5 to 1.5;
- applies an anti-spike filter on the XICs, followed by a background noise removal filter and a smoothing moving-window filter;
- uses a `detection_zivy` peak detection method.

In the next lines of our example file :

```
<quantification_method id="my_qmoulon">
  <xic_extraction xic_type="max">
  <ppm_range max="10" min="10"/>
  </xic_extraction>
  <xic_filters>
  <background half_mediane="5" half_min_max="15"/>
  </xic_filters>
  <peak_detection>
    <detection_moulon>
    <smoothing_point>3</smoothing_point>
    <TIC_start>5000</TIC_start>
    <TIC_stop>3000</TIC_stop>
    </detection_moulon>
  </peak_detection>
</quantification_method>
</quantification_methods>
```

we define the `my_qmoulon` quantification method which extracts XICs based on the max intensity, using a ppm (parts per million) range of $0.5 - 1.5$. It applies a background removal filter to them and uses the `detection_moulon` peak detection method.

The `quantification_methods` block is not mandatory in a masschroqML file.

4.8 Quantifying

In the `quantify` block we describe the items to be quantified in each group of data (peptides, isotopes, m/z values or $(m/z, rt)$ values) and the quantification methods to be used.

4.8.1 Quantifying peptides

```
<quantify id="q1" withingroup="G1" quantification_method_id="my_qzivy">
<peptides_in_peptide_list mode="real_or_mean"/>
</quantify>
```

As we can see in the lines above, we have chosen to:

- quantify all the peptides (`<peptides_in_peptide_list>` element) identified
- in group $G1$ (`withingroup` attribute)
- by using the *myqzivy* quantification method (`quantification_method_id` attribute.
- The `mode` attribute indicates the way we compute this peptide's retention time in each run sample during quantification. This rt value is the one used for peak matching after peak detection during the quantification process. Two RT computation modes are available:

  real_or_mean: if the peptide has been identified in a run sample, its retained RT is the observed one (more exactly the best RT one). If it has not been identified in this sample, its RT is the mean of the RTs in the run samples of the same group where this peptide has been identified. This way we can quantify a peptide in a sample even if it has not been identified in it. For details on this method see section 3.8.

  mean: if this mode is selected, the RT of the peptide in a sample is the mean of the best RTs of this peptide in the other samples of the group the peptide was identified in. No difference is made whether the peptide was identified in this sample or not.

4.8.2 Quantifying isotopes

```
<quantify id="q2" withingroup="G2" quantification_method_id="my_moulon">
<peptides_in_peptide_list mode="post_matching" isotope_label_refs="iso1
   iso2"/>
```

As you can see above, to quantify isotopes it suffices to add the attribute `isotope_label_refs` containing the references to the isotope labels we want to quantify. These labels have been previously defined in the masschroqML file (see section 4.5). MassChroQ automatically computes the isotope masses after modification during quantification.

4.8.3 Quantifying m/z values

The list of the desired m/z values to be quantified can be given as follows:

```
<quantify id="q2" withingroup="G2" quantification_method_id="my_moulon">
<peptides_in_peptide_list mode="post_matching" isotope_label_refs="iso1
   iso2"/>
<mz_list>732.317 449.754 552.234 464.251 381.577 569.771
   575.256</mz_list>
```

4.8.4 Quantifying $(m/z, rt)$ values

The list of the desired $(m/z, rt)$ values to be quantified can be given as follows:

```
<quantify id="q2" withingroup="G2" quantification_method_id="my_moulon">
<peptides_in_peptide_list mode="post_matching" isotope_label_refs="iso1
   iso2"/>
<mz_list>732.317 449.754 552.234 464.251 381.577 569.771
   575.256</mz_list>
<mzrt_list>
<mzrt mz="732.317" rt="230.712"/>
<mzrt mz="575.256" rt="254.788"/>
</mzrt_list>
</quantify>
```

4.9 Result files

Quantification result files

```
<quantification>
<quantification_results>
<quantification_result output_file="result1" format ="tsv"/>
<quantification_result output_file="result2" format="gnumeric"/>
<quantification_result output_file="result3" format ="xhtmltable"/>
<quantification_result output_file="result4" format ="masschroqml"
   xic_traces="true"/>
</quantification_results>
```

In the `quantification_results` block we define the format and name of the files that will contain the final quantification results. The format can be :

- `tsv` : tab-separated values text format; this format is ready to use for automatic or manual statistical analysis ;
- `gnumeric` : Gnome spreadsheet format to use with Gnumeric software;
- `xhtmltable` : xhtml (eXtensible HyperText Markup Language) format; all the results are in an xhtml table and you can visualize them via an Internet browser for example;
- `maschroqml` : masschroqML format, XML format, used as an input but also output format to MassChroQ; it is intended for development use, for example to integrate masschroq results in databases.

The user can choose multiple output formats for the same analysis as we have done in the example above. We have chosen to export results into a tsv file that will be called `result1.tsv`, a gnumeric file that will be called `result2.gnumeric`, etc.

Note 14. *The `tsv` and `xhtmltable` outputs will create two files : one with extension `_pep` containing the quantification results, and a second one with extension `_prot` resuming for each peptide, the corresponding proteins and descriptions. Both these files are well suited for automatical statistical analysis. The `gnumeric` format will contain two separate sheets for each of them. The `tsv` output will also create a third file with the extension `_copmpar`, containing the quantification results sorted in a different way, well suited for a first direct visual verification of the results.*

In the example above three files named `results_pep.tsv`, `results_prot.tsv` and `results_compar.tsv` will be created for the tsv format and a unique file named `results.gnumeric` containing two sheets for the `gnumeric` format.

Alignment result (`.time`) files

When MassChroQ reads the `align` tag in its input masschroqML file (lines 57 and 58 in appendix A), it launches the alignment of each sample in the indicated group towards the reference sample, using the indicated alignment method. For each LC-MS run it aligns, if the `write_time_values_output_dir` attribute is present, MassChroQ creates in the given directory a `run_filename.time` file containing two tab-separated columns of values :

- the first column (with `old_rt` header) contains the original retention times of this sample as they appear in the raw data file;
- the second column (called `new_rt`) contains the corresponding computed aligned retention times.

Note 15. *The `.time` files are useful for analysis and alignment checking, but they can also be used to avoid realigning samples or to inject external alignment values. Indeed, MassChroQ can load previously generated `.time` files if the user puts the `time_values_dir` attribute in the `rawdata` element of the masschroqML file. This way, the user does not have to repeat alignment on previously aligned files. Or he can use an external alignment tool and inject its results via `.time` files in masschroq's analysis.*

4.10 Trace files

Quantification traces

You can tell MassChroQ to produce detailed quantification traces in the following way :

```
<quantification_traces>
<peptide_traces peptide_ids="pep0 pep1" output_dir="pep_traces"
   format="tsv"/>
<all_xics_traces output_dir="all_xics_traces" format="tsv"/>
<mz_traces mz_values="634.635 449.754 552.234" output_dir="mz_traces"
   format="tsv"/>
<mzrt_traces output_dir="mzrt_traces" format="tsv">
<mzrt_values>
<mzrt_value mz="732.317" rt="230.712"/>
<mzrt_value mz="575.256" rt="254.788"/>
</mzrt_values>
</mzrt_traces>
</quantification_traces>
```

Three types of traces can be produced :

- `peptide_traces` : traces for a given list of space separated peptide ids;
- `all_xics_traces` : traces for all the quantified items of the current analysis (i.e. all the extracted XICs);
- `mz_traces` : traces for a given list of space separated mz values;
- `mzrt_traces` : traces for a given list of mz, rt couple of values;

The `output_dir` attribute (which is mandatory) indicates the directory name where the traces should be put in the local system. For each traced item (peptide, mz value or mz-rt value) a file is created in this directory. The traces file names contain information about the group, the MS run, the peptide id, the mz, and the rt of the traced item.

A trace file is always and by default in `tsv` format (the `format` attribute is optional). It contains from left to right order:

- a header giving the precise mz, rt and peptide id information.
- an `rt` column containing the retention time values of the XIC this traced item corresponds to;

- an `intensity` column containing the retention time values of the XIC this traced item corresponds to;
- for each filter applied to this XIC, an `filtered_intensity` column containing the intensity values after filtering;
- an `all_peaks` column containing the peak intensity values of all the detected peaks on this XIC (if a peak has been detected, the corresponding rt and intensity line will contain the peak's intensity value);
- a `matched_peaks` column containing the retained peaks after peak matching, i.e. the peaks that really correspond to the quantified item.

Trace files are very useful for analysis checking, but also for parameter refining. Indeed, one can fastly trace a small list of peptides with different quantification method parameters and compare the traces (for example to see what detection threshold detects more peaks or what XIC filter better removes background noise).

Alignment traces : (`.trace` files)

For each LC-MS run it aligns, if the `write_time_values_output_dir` attribute is present, the MS2 alignment method in MassChroQ creates in the given directory a `sample_name.trace` file containing the retention time values at each step of the alignment, from the original state to the final aligned one, including the MS/MS alignment values before and after smoothing. These files are not used by masschroq in any way, they are solely intended to help the users check and analyze the alignment.

An alignment trace file is always and by default in `tsv` format. It contains from left to right order:

- an `rt_MS2` column containing the retention time values of the shared peptides in the run being aligned;
- a `deltaRT_MS2` column containing the differences between the retention time value of the shared peptides in the reference run and their retention time value in the run being aligned;
- a `post-median-deltaRT_MS2` column containing the previous delta_RT values after a moving window median filter;
- a `post-mean-deltaRT_MS2` column containing the previous post-median-delta_RT values after a moving window average filter;

- an `rt_MS1` column containing the level 1 retention time values of the run being aligned, before their alignment;
- a `smoothed-deltaRT_MS1` column containing the computed retention time deviations for the previous rt_MS1 values (after linear interpolation). These values will be added to the original retention times to obtain the final aligned values. If a smoothing moving filter has been defined by the user, these deviation values are already smoothed.

Note 16. *Alignment traces are only available for the in-house developed MS2 alignment method. The third-party OBi-Warp alignment method that we have integrated in MassChroQ does not support trace files.*

Chapter 5

Specification of the identified peptides files

MassChroQ can parse peptide identification results from text files in Tab, Comma or Semi-Colon Separated Values formats (*tsv* or *csv*).

5.1 masschroqML peptide files instructions

To make MassChroQ parse identified peptides and protein descriptions from text files you should put instruction in the masschroqML file as follows:

```
<peptide_files_list>
<peptide_file data="samp0" path="bsa1_peptides.txt"/>
<peptide_file data="samp1" path="bsa2_peptides.txt"/>
<peptide_file data="samp2" path="bsa3_peptides.txt"/>
<peptide_file data="samp3" path="bsa4_peptides.txt"/>
</peptide_files_list>
```

The `peptide_files_list` group should immediately follow the `</groups>` tag.

At most one peptide file per LC-MS run has to be provided. The `data` attribute references the run id the peptide file corresponds to; the `path` attribute is the path to the peptide file in your system.

5.1.1 masschroq command-line `-parse-peptides` option

Running `masschroq` with an input file containing `peptide_files_list` instructions, will make it parse the indicated peptide files and produce a new masschroqML input file named `parsed-peptides_input_file.xml` containing the original `input_file.xml` plus all the identified peptides integrated and organized in the masschroqML format.

Moreover, MassChroQ will automatically continue analysis on this newly produced file. If you want not to continue analysis but only to parse the peptide files and put them in the `parsed-peptides_input_file.xml` run MassChroQ with the `-parse-peptides or -p` option as follows:

```
masschroq --parse-peptides input_file.xml
```

5.1.2 Peptide text file format specification

Here are the first lines of the peptide identification example file presented in appendix B :

```
scan sequence mh z proteins mods
778 CCTKPESER 1166.4934 2 "P02769|ALBU_BOVIN SERUM ALBUMIN PRECURSOR."
    114.08
839 NYQEAK 752.3585 2 "P02769|ALBU_BOVIN SERUM ALBUMIN PRECURSOR."  Ymod
1136 TCVADESHAGCEK 1463.5852 2 "P02769|ALBU_BOVIN SERUM ALBUMIN
    PRECURSOR." 57.04
1585 SHCIAEVEK 1072.5111 2 "P02769|ALBU_BOVIN SERUM ALBUMIN PRECURSOR."
1935 NECFLSHK 1034.4729 2 "P02769|ALBU_BOVIN SERUM ALBUMIN PRECURSOR."
1960 ECCDKPLLEK 1291.6019 3 "P02769|ALBU_BOVIN SERUM ALBUMIN PRECURSOR."
1980 LCVLHEK 898.48236 2 "P02769|ALBU_BOVIN SERUM ALBUMIN PRECURSOR."
```

Separation character

In this example values are separated by tabulations. Other correct separation characters are comma "," and semi-colon ";".

Note 17. *In a given peptide file, separation characters should not be melted: one has to exclusively use the tab, the comma or the semi-colon separator everywhere in the file. Melting them will cause masschroq to exit with a parsing error.*

Header specification

As shown above, the header for the peptide files is:

```
scan sequence mh z proteins mods
```

Note 18. *The first five columns are mandatory. The sixth column (`mods`) is optional.*

Note 19. *The header (the first five columns exactly) is mandatory: it must be present in all of your peptide files. If another header is used masschroq will exit with a parse error.*

The other correct headers, depending on the chosen separation character are :

```
scan,sequence,mh,z,proteins,mods
```

and for the semi-colon :

```
scan;sequence;mh;z;proteins;mods
```

Values specification

The accepted values for a peptide text file are :

- **scan** (mandatory): integer representing the scan number in the mzXML or mzML file this peptide belongs to.
- **sequence**: (mandatory) string representing the amino-acid sequence of a peptide.
- **mh** (mandatory): real number representing the mass of a peptide plus the H^+ mass.
- **z** (mandatory): integer representing the charge state of this peptide.
- **proteins** (mandatory): string in free text representing the description of one protein this peptide belongs to. At most one protein per line is allowed.
- **mods** (optional): string representing the mass modification on this peptide if any. This is optional, it can help facilitate user's analysis but is not used by MassChroQ to compute the peptide mass modifications (which are computed by MassChroQ using isotopes and MH values).

Note 20. *One line represents a given peptide sequence in a given charge state in the given scan number of the corresponding run data, identified in the given protein. Hence, more than one line can be found for the same peptide sequence, with different scan numbers, charge states, or protein description values. This also means that for the same peptide sequence, with same MH, same scan number and same charge state but belonging to two different proteins, two lines should be put in the file, one per each protein (as in lines 5 and 6 above).*

Chapter 6

Parameters cheat-sheet

In this chapter you will find the list of all MassChroQ parameters with an explanation, recommended values for them and some practical advice.

Note 21. *An archive containing several ready-to-use examples illustrating different alignment and quantification methods is available on the MassChroQ homepage.*

Note 22. *In all the following moving-window filters or transforms, the user is asked to enter a half window size parameter. The corresponding window in MassChroQ will then be of size:* $2 * half\_window + 1$.

6.1 Alignment parameters

- **write_time_values_output_dir (directory path)**: the directory where MassChroQ will put the .time and .trace files.

The *ObiWarp* alignment parameters

- **lmat_precision (real number)**: matrix precision in Thompson. A good value is often 1.
- **mass_start (mz value)**: beginning of the mass window used to compute the matrix. This value depends on your data and spectrometer but we advise you (from our experience) to always exclude the spectra masses range containing contaminants present during the whole run (for example the methylxyloxane contaminant).
- **mass_stop (mz value)**: end of the mass window used to compute the matrix.

The *MS2* alignment parameters

- **ms2_tendency_halfwindow (natural integer)**: half size of the window used to apply a moving median on the MS/MS retention time deviation curve. Used to create the tendency deviation curve. Of course the appropriate value for this window depends on the number of identified peptides that the two runs (reference run and run being aligned) have in common. Usually a good value is 10. While aligning, MassChroQ outputs on the console the number of peptides in common which you can use to readjust this parameter if necessary.

- **ms2_smoothing_halfwindow (natural integer**: half size of the window used to apply a moving average on the MS/MS retention time deviation curve. Smooths the deviation curve. Same as the above parameter, usually a good value is 10.

- **ms1_smoothing_halfwindow (natural integer)**: half size of the window used to apply a moving median on the MS level 1 retention time corrections curve. This smoothing parameter is optional, and it is not necessary most of the time. It could be used in place of the ms2 smoothing parameter in cases of a small number of shared identified peptides (< 100), in which case a good value is 20.

6.2 Quantification parameters

XIC extraction parameters

These parameters depend on your spectrometer.

- **xic_type ("sum" or "max")**: type of the XIC intensity representation. It can be:
  - **sum** for the sum of intensities across the range of XIC masses;
  - **max** for the maximal intensity across the range of XIC masses;

- **mz_range** or **ppm_range** parameters: mass tolerance XIC window in mz or ppm resolution. For each of them you have to define:
  - **min**: the minimum value of the window range;
  - **max**: the maximal value of the window range.

XIC filters parameters

- Background filter: corrects baseline noise.
  - **half_mediane (natural integer)**: half size of the moving window used to apply a median on the XIC intensity values. Usually a value of 5 is sufficient, but depending on your spectrometer and the level of background noise intensity it generates, it can be greater.
  - **half_min_max (natural integer)**: half size of the moving window used to apply an open transform (a min then a max) on the XIC intensities. This filter is sometimes useful in some LR signals presenting baseline noise; it smoothes the signal from below, so a good half-window value would be the peak width (in scan points number) divided by 2. For example for a peak of 30 seconds with one scan per second, this value would be 15. For security make this value a little greater, in this example 20.
- Smoothing filter:
  - **half (natural integer)**: half size of the moving window used to apply an average on the XIC intensities. Use this filter in rare cases of very noisy signal.
- Anti-spike filter: removes spikes on HR analysis.
  - **half (natural integer)**: half size of the moving window (in scan points) used to eliminate a spike. An intensity value is considered as a spike if it is the only value greater than 0 among the other values of the window. A good value is usually 5.

The Moulon peak-detection parameters

- **smoothing_point (natural integer)**: the half window size used to compute the average intensity in the filter preceding peak detection;
- **TIC_start**: intensity value representing the starting threshold point of peak detection, i.e. at this intensity value we start the "chronometer" and begin searching forward (in increasing retention time order) on the signal for local maxima.
- **TIC_stop**: intensity value representing the ending threshold of peak detection, i.e. at this intensity value we stop the "chronometer" and consider the local maximum found since TIC_start as a peak.

The Zivy peak-detection parameters

- **mean_filter_half_edge (natural integer)** : half window size used to compute the average intensity in the smoothing filter preceding peak detection; this filter is used only for detection purpose, the original signal is not altered. A good value is 1 or 2.

- **minmax_half_edge (natural integer)**: the half window size used to apply the close (min/max) transform on the XIC intensities. This window determines the number of scan points over which two peaks will be considered separately, otherwise they would have been merged. A good half window value is usually 3 (which makes a window of 7).

- **maxmin_half_edge (natural integer)**: same as above but for the close (max/min) transform. This window determines the minimum peak width (in scan points number) below which the peak would not be detected. A good half window value is usually 2 (which makes a window of 5).

- **detection_threshold_on_max (intensity value)**: threshold on the close signal: a minimum intensity value below which peaks are not detected on the closing signal. This threshold is usually two or three times the background noise intensity level (this latter depends on your mass spectrometer).

- **detection_threshold_on_min (intensity value)**: threshold on the open signal: a minimum intensity value below which peaks are not detected. It corresponds to the opening signal upper limit and it represents the background signal upper level. A good value would thus be slightly bigger than your background noise intensity level.

The *mode* parameter

The *mode* parameter in the *quantify* element indicates the peak matching mode and the computation mode of the retention time of peptides and isotopes used to match these peptides to the detected peaks. For more details on peak matching see section 3.8. It can be:

- **post_matching**: use this mode only in HR experiments. For details on how it works see section 3.8.4.

- **real_or_mean**: use this mode in all other experiments (LR for example). In this mode, if the peptide is identified in a run its retention time is the observed one; if not it is the mean of its retention times in the runs of the current group, in which this peptide was identified.

- **mean** : This mode is obsolete and likely to disappear in the future. In this mode the retention time of the peptide in a run is always the mean retention time of its retention times in all the runs of the current group where this peptide was identified (whether the peptide is identified in this run or not).

Appendices

Appendix A

masschroqML complete input example file

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<masschroq>
<rawdata time_values_dir="masschroq_tests/time">
<data_file id="samp0" format="mzxml" path="bsa1.mzXML" type="centroid"/>
<data_file id="samp1" format="mzxml" path="bsa2.mzXML" type="profile"/>
<data_file id="samp2" format="mzml" path="/home/user/bsa3.mzml"
   type="profile"/>
<data_file id="samp3" format="mzml" path="/home/user/bsa4.mzml"
   type="profile"/>
</rawdata>
<groups>
<group data_ids="samp0 samp1" id="G1"/>
<group data_ids="samp2 samp3" id="G2"/>
</groups>
<peptide_files_list>
<peptide_file data="samp0" path="bsa1_peptides.txt"/>
<peptide_file data="samp1" path="bsa2_peptides.txt"/>
<peptide_file data="samp2" path="bsa3_peptides.txt"/>
<peptide_file data="samp3" path="bsa4_peptides.txt"/>
</peptide_files_list>
<protein_list>
<protein desc="conta|P02769|ALBU_BOVIN SERUM ALBUMIN PRECURSOR."
   id="P1.1"/>
<protein desc="conta|P02770|ALBU_RAT SERUM ALBUMIN PRECURSOR."
   id="P1.2"/>
</protein_list>
<peptide_list>
```

```
<peptide id="pep0" mh="1463.626" mods="114.08" prot_ids="P1.1"
   seq="TCVADESHAGCEK">
<observed_in data="samp0" scan="655" z="2"/>
<observed_in data="samp1" scan="798" z="2"/>
</peptide>
<peptide id="pep1" mh="1103.461" mods="57.04" prot_ids="P1.1"
   seq="ADESHAGCEK">
<observed_in data="samp3" scan="663" z="2"/>
</peptide>
</peptide_list>
<isotope_label_list>
<isotope_label id="iso1">
<mod at="Nter" value="28.0"/>
<mod at="K" value="28.0"/>
</isotope_label>
<isotope_label id="iso2">
<mod at="Nter" value="32.0"/>
<mod at="K" value="32.0"/>
</isotope_label>
</isotope_label_list>
<alignments>
<alignment_methods>
<alignment_method id="my_ms2">
<ms2 write_time_values_output_dir="masschroq_tests/time">
<ms2_tendency_halfwindow>10</ms2_tendency_halfwindow>
<ms2_smoothing_halfwindow>5</ms2_smoothing_halfwindow>
<ms1_smoothing_halfwindow>3</ms1_smoothing_halfwindow>
</ms2>
</alignment_method>
<alignment_method id="my_obiwarp">
<obiwarp write_time_values_output_dir="masschroq_tests/time">
<lmat_precision>1</lmat_precision>
<mz_start>500</mz_start>
<mz_stop>1200</mz_stop>
</obiwarp>
</alignment_method>
</alignment_methods>
<align group_id="G1" method_id="my_ms2" reference_data_id="samp0"/>
<align group_id="G2" method_id="my_obiwarp" reference_data_id="samp2"/>
</alignments>
<quantification_methods>
<quantification_method id="my_qzivy">
  <xic_extraction xic_type="sum">
```

```
  <mz_range max="0.3" min="0.3"/>
  </xic_extraction>
  <xic_filters>
  <anti_spike half="5"/>
  <background half_mediane="5" half_min_max="15"/>
  <smoothing half="3"/>
  </xic_filters>
  <peak_detection>
    <detection_zivy>
    <mean_filter_half_edge>1</mean_filter_half_edge>
    <minmax_half_edge>3</minmax_half_edge>
    <maxmin_half_edge>2</maxmin_half_edge>
    <detection_threshold_on_max>5000 </detection_threshold_on_max>
    <detection_threshold_on_min>3000 </detection_threshold_on_min>
    </detection_zivy>
  </peak_detection>
</quantification_method>
<quantification_method id="my_qmoulon">
  <xic_extraction xic_type="max">
  <ppm_range max="10" min="10"/>
  </xic_extraction>
  <xic_filters>
  <background half_mediane="5" half_min_max="15"/>
  </xic_filters>
  <peak_detection>
    <detection_moulon>
    <smoothing_point>3</smoothing_point>
    <TIC_start>5000</TIC_start>
    <TIC_stop>3000</TIC_stop>
    </detection_moulon>
  </peak_detection>
</quantification_method>
</quantification_methods>
<quantification>
<quantification_results>
<quantification_result output_file="result1" format ="tsv"/>
<quantification_result output_file="result2" format="gnumeric"/>
<quantification_result output_file="result3" format ="xhtmltable"/>
<quantification_result output_file="result4" format ="masschroqml"
   xic_traces="true"/>
</quantification_results>
<quantification_traces>
```

```
<peptide_traces peptide_ids="pep0 pep1" output_dir="pep_traces"
   format="tsv"/>
<all_xics_traces output_dir="all_xics_traces" format="tsv"/>
<mz_traces mz_values="634.635 449.754 552.234" output_dir="mz_traces"
   format="tsv"/>
<mzrt_traces output_dir="mzrt_traces" format="tsv">
<mzrt_values>
<mzrt_value mz="732.317" rt="230.712"/>
<mzrt_value mz="575.256" rt="254.788"/>
</mzrt_values>
</mzrt_traces>
</quantification_traces>
<quantify id="q1" withingroup="G1" quantification_method_id="my_qzivy">
<peptides_in_peptide_list mode="real_or_mean"/>
</quantify>
<quantify id="q2" withingroup="G2" quantification_method_id="my_moulon">
<peptides_in_peptide_list mode="post_matching" isotope_label_refs="iso1
   iso2"/>
<mz_list>732.317 449.754 552.234 464.251 381.577 569.771
   575.256</mz_list>
<mzrt_list>
<mzrt mz="732.317" rt="230.712"/>
<mzrt mz="575.256" rt="254.788"/>
</mzrt_list>
</quantify>
</quantification>
</masschroq>
```

masschroq_complete_input_example.xml

Appendix B

Peptide identification example tsv file

```
scan sequence mh z proteins mods
778 CCTKPESER 1166.4934 2 "P02769|ALBU_BOVIN SERUM ALBUMIN PRECURSOR."
    114.08
839 NYQEAK 752.3585 2 "P02769|ALBU_BOVIN SERUM ALBUMIN PRECURSOR."  Ymod
1136 TCVADESHAGCEK 1463.5852 2 "P02769|ALBU_BOVIN SERUM ALBUMIN
    PRECURSOR." 57.04
1585 SHCIAEVEK 1072.5111 2 "P02769|ALBU_BOVIN SERUM ALBUMIN PRECURSOR."
1935 NECFLSHK 1034.4729 2 "P02769|ALBU_BOVIN SERUM ALBUMIN PRECURSOR."
1960 ECCDKPLLEK 1291.6019 3 "P02769|ALBU_BOVIN SERUM ALBUMIN PRECURSOR."
1980 LCVLHEK 898.48236 2 "P02769|ALBU_BOVIN SERUM ALBUMIN PRECURSOR."
2089 CCTESLVNR 1138.4973 2 "P02769|ALBU_BOVIN SERUM ALBUMIN PRECURSOR."
2237 YICDNQDTISSK 1443.6373 2 "P02769|ALBU_BOVIN SERUM ALBUMIN
    PRECURSOR."
2241 QNCDQFEK 1051.4155 2 "P02769|ALBU_BOVIN SERUM ALBUMIN PRECURSOR."
2278 ETYGDMADCCEK 1478.5223 2 "P02769|ALBU_BOVIN SERUM ALBUMIN
    PRECURSOR."
2702 QEPERNECFLSHK 1656.7439 3 "P02769|ALBU_BOVIN SERUM ALBUMIN
    PRECURSOR."
2713 EYEATLEECCAK 1502.61 2 "P02769|ALBU_BOVIN SERUM ALBUMIN PRECURSOR."
2753 ECCHGDLLECADDR 1749.6615 3 "P02769|ALBU_BOVIN SERUM ALBUMIN
    PRECURSOR."
2867 CCAADDKEACFAVEGPK 1927.797 3 "P02769|ALBU_BOVIN SERUM ALBUMIN
    PRECURSOR."
2910 EACFAVEGPK 1107.5137 2 "P02769|ALBU_BOVIN SERUM ALBUMIN PRECURSOR."
3267 DDPHACYSTVFDK 1554.6533 3 "P02769|ALBU_BOVIN SERUM ALBUMIN
    PRECURSOR."
3494 YLYEIAR 927.49396 2 "P02769|ALBU_BOVIN SERUM ALBUMIN PRECURSOR."
```

```
3629 RHPEYAVSVLLR 1439.8123 3 "P02769|ALBU_BOVIN SERUM ALBUMIN
   PRECURSOR."
3818 LKPDPNTLCDEFK 1576.7699 3 "P02769|ALBU_BOVIN SERUM ALBUMIN
   PRECURSOR."
3821 KVPQVSTPTLVEVSR 1639.9381 3 "P02769|ALBU_BOVIN SERUM ALBUMIN
   PRECURSOR."
3870 KQTALVELLK 1142.715 2 "P02769|ALBU_BOVIN SERUM ALBUMIN PRECURSOR."
4082 RHPEYAVSVLLR 1439.8177 3 "P02769|ALBU_BOVIN SERUM ALBUMIN
   PRECURSOR."
4236 RPCFSALTPDETYVPK 1880.9225 3 "P02769|ALBU_BOVIN SERUM ALBUMIN
   PRECURSOR."
4253 SLHTLFGDELCK 1419.6945 3 "P02769|ALBU_BOVIN SERUM ALBUMIN
   PRECURSOR."
4460 LVNELTEFAK 1163.6317 2 "P02769|ALBU_BOVIN SERUM ALBUMIN PRECURSOR."
4499 SLHTLFGDELCK 1419.6956 3 "P02769|ALBU_BOVIN SERUM ALBUMIN
   PRECURSOR."
4972 RHPYFYAPELLYYANK 2045.0253 3 "P02769|ALBU_BOVIN SERUM ALBUMIN
   PRECURSOR."
5084 LFTFHADICTLPDTEK 1907.9092 3 "P02769|ALBU_BOVIN SERUM ALBUMIN
   PRECURSOR."
5308 LGEYGFQNALIVR 1479.807 2 "P02769|ALBU_BOVIN SERUM ALBUMIN
   PRECURSOR."
5578 HPYFYAPELLYYANK 1888.9224 3 "P02769|ALBU_BOVIN SERUM ALBUMIN
   PRECURSOR."
6015 TVMENFVAFVDK 1399.6898 2 "P02769|ALBU_BOVIN SERUM ALBUMIN
   PRECURSOR."
6689 MPCTEDYLSLILNR 1724.8433 3 "P02769|ALBU_BOVIN SERUM ALBUMIN
   PRECURSOR."
6788 DAFLGSFLYEYSR 1567.748 2 "P02769|ALBU_BOVIN SERUM ALBUMIN
   PRECURSOR."
```

peptide_example_tsv_file.txt

Bibliography

[1] B. Valot, O. Langella, E. Nano, and M. Zivy, "Masschroq: A versatile tool for mass spectrometry quantification," *Proteomics*, vol. 11, no. 17, pp. 3572–3577, 2011.

[2] J. Serra, *Image Analysis and Mathematical Morphology*, vol. I. Academic Press, London, 1982.

[3] P. Maragos, *Handbook of Image and Video Processing*, ch. 3.3 Morphological filtering for image enhancement and feature detection, pp. 135–156. Elsevier, 2nd ed., 2004.

[4] J. Lee, R. Haralick, and L. Shapiro, "Morphologic edge detection," *IEEE Journal of Robotics and Automation*, 1987.

[5] F. Leymarie and M. D. Levine, "Curvature morphology," Tech. Rep. TR-CIM-88-26, CIM McGill University, Montreal, Canada, December 1988.

[6] J. Goutsias, L.Vincent, and D. Bloomberg, *Mathematical Morphology and its Applications to Image and Signal Processing.* Kluwer Academic Publishers, 2000.

[7] F. Meyer, "Contrast feature extraction," *Special Issues of Practical Metallography*, vol. 8, pp. 374–380, 1978.

[8] J. Prince and E. Marcotte, "Chromatographic alignment of ESI-LC-MS proteomics data sets by ordered bijective interpolated warping," *Analytical Chemistry*, vol. 78, no. 17, pp. 6140–6152, 2006.