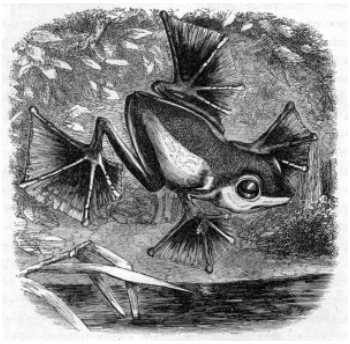


mineXpert is part of the msXpertSuite software package

Modelling, simulating and analyzing ionized flying species

mineXpert User Manual

Visualization, analysis and mining of mass spectrometric data

mineXpert 5.8.6

mineXpert User Manual: Visualization, analysis and mining of mass spectrometric data

by Filippo Rusconi

March 24, 2020 , 5.8.6

Copyright 2009,...,2019 Filippo Rusconi

msXpertSuite - mass spectrometry software suite

http://www.msxpertsuite.org/

This book is part of the msXpertSuite project.

The msXpertSuite project is the successor of the massXpert project. This project now includes various independent modules:

- massXpert, program to model polymer chemistries and simulate mass spectrometric data;
- mineXpert, program to visualize and mine mass spectral data (mass spectrum, drift spectrum, XIC chromatograms) starting from the TIC chromatogram.

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

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

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

The flying frog picture is courtesy http://www.papuaweb.org. The specific license as of 20190104 is: *Please acknowledge the use of Papuaweb resources in your publications. To do this include the complete item URL (for example "http://www.papuaweb.org/gb/ref/hinton-1974/63.html") or a general reference to "http://www.papuaweb.org" in your citation/bibliography. This will improve recognition of these resources by Google Scholar and similar search engines.*

5.8.6

Revision History

Revision 5.8.2	february 2019	fr

Rework the IsoSpec-based isotopic clust calculations to better describe the three available processes. Also, describe the new FWHM calculation feature in the PeakShaperDlg.

Revision 5.8.0	end january 2019	fr

Add section to described the newly developed feature about isotopic cluster calculations based on the IsoSpec library;

Revision 5.7.0	beginning january 2019	fr

Finished porting of the historical LaTeX-based documentation to the DocBook/DAPS/FOP publishing system. Please, see Colophon for details;

Revision not_set	september 2018	fr

Update the document to include new features and to make fixes. Insist on the fact that mineXpert reads mzML data files.

Revision not_set	march 2018	fr

Update the document to include new m/z~integration features and scripting capabilities.

Revision not_set	december 2016	fr

Major rewriting of the document to incorporate all the new features. Large chapter on the scripting of mineXpert.

Revision not_set	december 2017	fr

Refactored document. First version documenting almost all the of current features of the software program.

Revision not_set	may 2017	fr

Refactored document. First version documenting almost all the of current features of the mineXpert software program.

Revision not_set	november 2016	fr

Resume writing, with new program name: mineXpert.

Revision not_set	september 2016	fr

Start of writing.

Dedication

To Marìa Cecilia

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

To all involved in the development of the K Desktop Environment (KDE)

To all the readers who helped me with this manual.

Contents

Preface iv

1 Generalities 1

1.1 General concepts and terminologies 1

1.2 Acquiring Mass Data Along Time: To Profile or Not To Profile? 1

1.3 Mass Data Visualisation: To Combine or Not To Combine? 2

1.4 Examples of Various Mass Spectral Data Integrations 4

TIC -> MZ integration 4 • TIC -> DT integration 5

2 Visualizing mass spectral data 6

2.1 Opening mass spectrum files 6

2.2 The Window Layout 6

2.3 The Main Program Window Menu 8

2.4 The main data windows 9

The TIC chromatogram window 9 • The mass spectrum window 11 • The drift spectrum window 12 • The color map window 14

2.5 General structure of the windows containing plot widgets 15

2.6 The data-displaying widgets 15

2.7 Data integrations featured by mineXpert 18

2.8 Irks and Quirks of Data Integrations 22

Integrations to a mass spectrum 22 • Integrations to a drift spectrum 30 • Integrations to a TIC intensity value 30

2.9 Chained Integrations 31

2.10 Mass spectral feature analysis 32

Mass spectral deconvolution based on charge state 33 • Mass spectral deconvolution based on isotopic cluster peaks 35 • Reading the resolving power based on mass spectral data 36

2.11 Calculating Isotopic Clusters with IsoSpec 36

The IsoSpec Graphical User Interface in mineXpert 38

2.12 Shaping mass peak centroids into well-shaped peaks 46

2.13 Recording the data mining work 48

Feature labelling to the console window 48 • Recording the data mining discoveries 49

2.14 Splitting files into smaller chunks 53

2.15 Converting mzML to SQLite3 55

3 JavaScript-based Scripting 57

3.1 Using the scripting console 57

3.2 The available objects list 58

3.3 The scripting history list 59

3.4 Exploring the available objects' features 59

3.5 The JavaScript reference 61

3.6 Creation of plot objects as a result of data mining 63

3.7 JavaScript-oriented classes 64

Creating a `DataPoint` object 64 • Creating `Trace` objects 65 • Exporting a `Trace` object to a file 69 • Plotting a JavaScript `Trace` object to a plot widget window 70 • Printing a JavaScript `Trace` object to the console 70 • Running a JavaScript script from file 71

4 JavaScript Reference 72

A GNU General Public License version 3 140

Preface

1 Software feature offerings and intended audience

This manual is about the msXpertSuite mass spectrometric software suite, a software environment that contains two modules:

- *massXpert* module: Allows users to define brand new polymer chemistries and use these polymer chemistry definitions to model linear polymer sequence. Once modelled, a polymer sequence can undergo chemical reactions (enzymatic or chemical cleavages, gas-phase fragmentations...). The obtained results are a model of what a mass spectrum would look like if the modelled experiment had actually been carried over up to the mass spectrometry analysis;

- *mineXpert* module: Allows users to load mass spectrometry data from mzML files, visualize and mine the data throughout all their depth. The mass data visualization starts at the TIC chromatogram level and deepens to the mass spectra, the drift spectra, the XIC chromatograms.

As such, this manual is intended for people willing to learn how to use the comprehensive msXpertSuite software package.

Mass spectrometry has gained popularity across the past twenty years or so. Indeed, developments in polymer mass spectrometry have made this technique appropriate to accurately measure masses of polymers as heavy as many hundreds of kDa, and of any chemical type.

There are a number of utilities—sold by mass spectrometer constructors with their machines, usually as a marketing "plus"—that allow predicting/analyzing mass spectrometric data obtained on polymers. These programs are usually different from a constructor to another. Also, there are as many mass spectrometric data prediction/analysis computer programs as there are different polymer types. You will get a program for oligonucleotides, another one for proteins, maybe there is one program for saccharides, and so on. Thus, the biochemist/massist, for example, who happens to work on different biopolymer types will have to learn to use several different software packages. Also, if the software user does not own a mass spectrometer, chances are he will need to buy all these software packages.

The msXpertSuite mass spectrometric software is designed to provide *free* solutions to all these problems by providing the following features:

- massXpert:
  - Model *ex nihilo* polymer chemistry definitions (in the XpertDef module that is part of the massXpert program);
  - Peform simple yet powerful mass computations to be made in a mass desktop calculator that is both polymer chemistry definition-aware and fully programmable (that's the XpertCalc module also part of the massXpert program);
  - Edit polymer sequences on a polymer chemistry definition-specific basis, along with chemical reaction simulations, finely configured mass spectrometric computations... (all taking place in the XpertEdit module that is the main module of the massXpert program);
  - Customize the way each monomer will show up graphically during the program operation (in the XpertEdit module);
  - Edit polymer sequences with immediate visualization of the mass changes elicited by the editing activity (in the XpertEdit module);
  - Open an unlimited number of polymer sequences at any given time and of any given polymer chemistry definition type (in the XpertEdit module);
- mineXpert:
  - Load mass spectometry data files in the mzML format, thanks to the excellent libpwiz library of ProteoWizard[1]fame;
  - Display the data in powerful ways in a unified graphical user interface. The interface was designed to integrate all the most useful characteristics of the various proprietary environments known by the author, thanks to the excellent libqcustomplot [2]library;
  - Configure the way mass spectrometry data integrations are performed and optionally configure and apply a Savitzky-Golay smoothing;
  - Perform data mining by performing data integrations in various ways;
  - Ion mobility mass spectrometry data are supported with an automatic m/z = f(dt) color map plot calculation;

1 http://proteowizard.sourceforge.net/.

2 http://qcustomplot.com/.

- A specific data integration mode allows easy quantitation of spectral data at any level (TIC chromatogram, mass spectrum, drift spectrum);
- Innovative data analysis recording allows to store the features mined during the data mining sessions in flexible ways that allow further data processing, like injection in databases;
- A JavaScript scripting environment allows taking the control of the software and of all the widgets from a script file;
- Convert data from mzML to the private (albeit open) database file format that allows to load data much faster. mineXpert can slice big data files into smaller chunks retaining all the data selected by the user in the most flexible ways.

2 Feedback from the users

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

The msXpertSuite software team might be contacted *via* the following addresses

```
msxpertsuite@msxpertsuite.org  = general mailing list about msXpertSuite
bug-reports@msxpertsuite.org   = report bugs found in msXpertSuite software
```

FIGURE 1: **ADDRESSES TO REPORT FEEDBACK TO**

3 Project History

This is a brief history of msXpertSuite.

- *1998–2000*

  The name massXpert comes from a project I started while I was a post-doctoral fellow at the École Polytechnique (Institut Européen de Chimie et Biologie, Université Bordeaux 1, Pessac, France);

  The massXpert program was published in *Bioinformatics* (Rusconi, F. and Belghazi, M. Desktop prediction/analysis of mass spectrometric data in proteomic projects by using massXpert. Bioinformatics, 2002, 644–655 (https://academic.oup.com/bioinformatics/article/18/4/644/243311) ↗).

  At that time, MS-Windows was at the Windows NT 4.0 version and the next big release was going to be *"you'll see what you'll see"*: MS-Windows 2000.

When I tried massXpert on that new version (one colleague had it with a new machine), I discovered that my software would not run normally (the editor was broken). The Microsoft technical staff would advise to *"buy a new version of the compiler environment and rebuild"*. This was a no-go: I did not want to continue paying for using something I had already produced with legitimate software.

- *2001–2006*

  During fall 1999, I decided that I would stop using Microsoft products for my development. At the beginning of 2000 I started as a CNRS research staff in a new laboratory and decided to start fresh: I switched to GNU/Linux (I never looked back). After some months of learning, I felt mature to start a new development project that would eventually become an official GNU package: GNU polyxmass.

  The GNU polyxmass software, much more powerful than what the initial massXpert software used to be, was published in *BMC Bioinformatics* in 2006 (Rusconi, F. GNU polyxmass: a software framework for mass spectrometric simulations of linear (bio-)polymeric analytes. BMC Bioinformatics, 225– (https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-7-226) ↗).

  Following that publication I got a lot of feedback (very positive, in a way) along the lines: — *"Hey, your software looks very interesting; only it's a pity we cannot use it because it runs on GNU/Linux, and we only use MS-Windows and MacOSX!"*.

- *2007–2016*

  In december 2006, I decided to make a full rewrite of GNU polyxmass. The software of which you are reading the user manual is the result of that rewrite. I decided to "recycle" the massXpert name because this software is written in C++, as was the first massXpert software. Also, because the first MS-Windows-based massXpert project is not developped anymore, taking that name was kind of a "revival" which I enjoyed. However, the toolkit I used this time is not the Microsoft Foundation Classes (first massXpert version) but the Trolltech Qt framework (see the About Qt menu in the *Help* menu in massXpert).

  Coding with the Qt libraries has one big advantage: it allows the developer to code once and to compile on the three main platforms available today: GNU/Linux, MacOSX, MS-Windows. Another advantage is that the Qt libraries are wonderful software, technically and philosophically (Free Software).

  The rewritten software was published in 2009 (Rusconi, F. massXpert 2: a cross-platform software environment for polymer chemistry modelling and simulation/analysis of mass spectrometric data.Bioinformatics, 2009, 2741–2742 (https://academic.oup.com/bioinformatics/article/25/20/2741/194220) ↗).

- *2016–*

  In 2016, I started a new project about visualization of mass spectrometric data. The project developed pretty quickly, as we needed at the mass spectrometry facility a software that would allow to cope efficiently with ion mobility mass spectrometric experimental data. mineXpert was thus started.

  To bundle both massXpert and mineXpert in a single software suite, I bought the msXpertSuite website http://msxpertsuite.org ↗ and created that new name.

4 Program and Documentation Availability and License

The programs and all the documentation that are shipped along with the msXpertSuite software suite are available at http://www.msxpertsuite.org. Most of the time, a new version is published as source, as binary install packages for MacOSX and MS-Windows. No GNU/Linux are created outside of the autbuilder of the various distributions. As a Debian Developer, Filippo Rusconi creates Debian[3] packages that are uploaded on the distribution servers. These packages are available using the system's software management infrastructure (like apt, for example).

The software and all the documentation are all provided under the Free Software license *GNU General Public License, Version 3, or later, at your option*. For an in-depth study of the *Free Software* philosphy I kindly urge the reader to visit http://www.gnu.org/philosophy.

3 http://www.debian.org/

1 Generalities

In this chapter, I wish to introduce some general concepts around the mineXpert program and the way data elements are named in this manual and in the program.

A mass spectrometry experiment generally involves monitoring the m/z value of analytes injected in the mass spectrometer along a certain time duration. The m/z value of each detected analyte is recorded along with the corresponding signal intensity i, so that a mass spectrum is nothing but a series of (m/z,i) pairs recorded along the acquisition duration. All along the acquisition, the precise moment at which a given analyte is detected (and its (m/z,i) pair is recorded), is called the retention time of that analyte (rt). This retention time is not to be misunderstood as the drift time of that analyte in an ion mobility mass spectrometry experiment.

1.1 General concepts and terminologies

Most generally, the mass spectrometer acquires an important number of spectra in, say, one second. But all these spectra are *combined* together, and, on the surface, the massist only sees a "slow" acquisition of 1 spectrum per second. This apparent slow acquisition rate is configurable. At the time of writing, generally 1 spectrum per second is recorded on disk. So, say we record mass spectra for 5 minutes, we would have recorded (5*60) spectra.

1.2 Acquiring Mass Data Along Time: To Profile or Not To Profile?

As a mass spectrometry user, the reader of this manual certainly has used mass spectrometers where mass spectra are acquired and stored in different ways:

- Mass spectra are acquired and summed—the next to the previous—in such a manner that one is left, at the end of the acquisition, with a single spectrum of which the various peak intensities have been increasing all along the acquisition. Indeed, in this mode, each new spectrum is actually *"combined"* to the previously acquired ones. The resulting mass spectrum that is displayed on screen and that gets ultimately stored on

disk is called a *combined spectrum*. This is typically the way MALDI-TOF mass spectrometers are used when acquiring data from samples deposited onto sample plates. We refer to this kind of acquisition as an "accumulation" mode acquisition;

- Mass spectra are acquired and stored on disk as a single file containing all the spectra, appended one after the other. There is no combination of the spectra: each time a new spectrum is displayed on screen, that spectrum is appended to the file. [1] This is typically the case when mass spectra are acquired all along a chromatography run and is generally called a *"profile"* mode acquisition.

1.3 Mass Data Visualisation: To Combine or Not To Combine?

In the previous section, we mentioned *spectrum combination* a number of times. What does that mean, that spectra are *"combined"* together into a single *"combined spectrum"*? Say we have 200 spectra that need to be combined together into a *single* spectrum that summatively represents the data of these 200 spectra.

First, a new spectrum would be allocated *(result spectrum)*, entirely empty at first. Then, the very first spectrum of the 200 spectra is literally copied into that result spectrum. At this point the combination occurs, according to an iterative process that has the following steps:

- Pick the next spectrum of the 200-spectra dataset;
    1. Pick the first (m/z,i) pair of the currently iterated spectrum;
    2. Look up in the *result spectrum* if a m/z value identical to the m/z value of the current (m/z,i) pair is already present;
    3. If the m/z value is found, increment its intensity by the intensity of the (m/z,i) pair;
    4. Else, if the m/z value is not found, add the current (m/z,i) pair to the result spectrum;
    5. Iterate over all the remaining (m/z,i) pairs of the current spectrum and redo these steps.
- Iterate over all the 198 remaining spectra of the dataset and do the steps above for each single iterated spectrum.

At the end of the two nested loops above, the combined spectrum is still a single spectrum that represents---summatively---all the 200 spectra. This whole process is very computing-intensive, in particular if:

1 Although there certainly is spectrum combination going on in the guts of the software, because the system actually acquires much more spectra than is visible on screen and each newly displayed spectrum is actually the combination of many spectra acquired under the surface.

- The m/z range is large: there are lots of points in each spectrum, which means that for each new (m/z,i) pair we need to iterate in the long list of m/z values that make the result spectrum;

- The resolving power of the mass spectrometer is high: there are many points per m/z range unit.

When a profile mode acquisition is performed, the user gets an innumerable number of distinct spectra, all appended to a single file. These unitary spectra are virtually unusable if an initial processing is not performed. This initial processing of the spectra is called *"total ion current chromatogram calculation"*. What is it? Let's say that the user has performed a profile mode mass spectrometry acquisition on the eluate of a chromatography column. Now, imagine that the spectrometer stores the mass data at a rate of one spectrum per second and that the chromatography gradient develops over 45 min: there would be a total of (45 * 60) spectra in that file. The question is: — *"How can we provide the user with a data representation that might be both meaningful and useful to start mining the data?"* The conventional way of doing so is to load all the mass spectra and compute the *"total ion current chromatogram"* (the TIC chromatogram). The analogy with chromatography is evident: the TIC chromatogram is the same as the UV chromatogram unless optical density is not the physical property that is measured over time; instead, the amount of ions that are detected in the mass spectrometer is measured over time. That amount is actually the sum of the intensities of all the (m/z,i) pairs detected in each spectrum. When mass data are acquired during a chromatography run, often, the total ion current chromatogram mirrors (mimicks) the UV chromatogram[2]. For each retention time (RT) a TIC value is computed by summing the intensities of all the (m/z,i) pairs detected at that specific RT.

How is this total ion current chromatogram computed? This is an iterative process: from the first spectrum (retention time 0 s), to the second spectrum (retention time 1 s) up to the last spectrum (retention time 45 min), the program computes the sum of the intensities of all the spectrum's (m/z,i) pairs. That computation ends up with a map that relates each RT value with the corresponding TIC value. The TIC chromatogram is nothing but a plot of the TIC values as a function of RT values. In that sense, it is indeed a chromatogram.

mineXpert works exactly in this way. When mass spectrometry data are loaded from a file, the TIC chromatogram is computed and displayed. This TIC chromatogram serves as the basis for the mass data mining, as described in this manual. The TIC chromatogram serves as the basis for spectral combinations that can be performed in various ways, and not all formally *combinations*, which is why I prefer the term *"integrations"*. Some of these integrations are described below:

- Integrating data from the TIC chromatogram to a single mass spectrum;

- Integrating data from the TIC chromatogram to a single drift spectrum;

2 Unless eluted analytes do absorb UV light but do not either desorb/desolvate or ionize, or both.

Note that the reverse actions are possible (and indeed necessary for a thorough data mining): selecting a region of a mass spectrum and asking that the TIC chromatogram be reconstituted from there; or selecting a region of a drift spectrum and asking that the TIC chromatogram be reconstituted from there also. Finally, integrations may, of course, be performed from a mass spectrum to a drift spectrum, and reverse.

1.4 Examples of Various Mass Spectral Data Integrations

In the sections below, the inner workings of mineXpert are described for some exemplary mass data integrations. For example, when doing ion mobility mass spectrometry data mining, it is essential to be able to characterize most finely the drift time of each and any analyte. Since each analyte is actually defined as one or more (m/z,i) pairs, it is essential to be able to ask questions like the following:

- What is the drift time of the ions below this mass peak?
- What are all the drift times of all the analytes going through the mobility cell for a given retention time range?
- What are all the ions that are responsible for this shoulder in the drift spectrum?

1.4.1 TIC -> MZ integration

What computation does actually mineXpert do when a mass spectrum is computed starting from a TIC chromatogram region, say between retention time RT minute 7 and RT minute 8.5?

1. List all the mass spectra that were acquired between RT 7 and RT 8.5. In this *spectral set*, there might be many hundreds of spectra that match this criterion, if we think that, in ion mobility mass spectrometry, ≈ 200 spectra are acquired and stored individually every second (I mean it, every 1 s time lapse);
2. Allocate a new empty spectrum—the *"combined spectrum"*—and copy into it without modification the first spectrum of the spectral set;
3. Go to the next spectrum of the spectral set and iterate into each (m/z,i) pair:
    - Check if the m/z value of the iterated pair is already present in the combined spectrum. If so, increment the combined spectrum's (m/z,i) pair's intensity value by the intensity of the iterated (m/z,i) pair's intensity. If not, simply copy the iterated (m/z,i) pair in the combined spectrum;
    - Iterate over all the remaining (m/z,i) pairs and perform the same action as above.
4. Iterate over all the remaining spectra of the spectral set and perform step number 3.

mineXpert then displays the combined spectrum.

1.4.2 TIC -> DT integration

What computation does mineXpert actually do when a drift spectrum is computed starting from a given TIC chromatogram region, say between retention time RT minute 7 and RT minute 8.5?

What is a drift spectrum? A drift spectrum (mobilogram) is a plot where the cumulated ion current of the detected ions is plotted against the drift time at which they were detected. Let's see how that computation is handled in mineXpert:

1. Create a map to store all the (drift time, intensity) pairs that are to be computed below, the (dt,i) map;
2. List all the mass spectra that were acquired between RT 7 and RT 8.5. The obtained list of mass spectra is called the *"spectral set"*;
3. Go to the first spectrum of the spectral set and compute its TIC value (sum of all the intensities of all the (m/z,i) pairs of that spectrum). Get the drift time value at which this mass spectrum was acquired. We thus have a value pair: (dt, i), that is, for drift time *dt*, the intensity of the total ion current is *i*;
   At this point, we need to do a short digression: we saw earlier that, at the time of this writing, one of the commercial instruments on which the author of these lines does his experiments stores 200 spectra each second. These 200 spectra actually correspond to the way the drift cycle is divided into 200 bin (time bins). That means that in the retention time range [7–8.5], there are (1.5*60) complete drift cycles. And thus there are (1.5*60) spectra with drift time x, the same amount of spectra with drift time y, and so on for the reminaing 198 time bins. Of course, a large number of these spectra might be almost empty, but these spectra are there and we need to cope with them.
   The paragraph above must thus lead to one interrogation about the current (dt,i) pair: — *"Has the current dt value be seen before, during the previous iterations in this loop?"*. If not, then create the (dt, i) pair and add it to the (dt,i) map; if yes, get the *dt* element in the map and increment its intensity value by the TIC value computed above;
4. Iterate over all the remaining spectra of the spectral set and perform step number 3.

At the end of the loop above, we get a map in which each item relates a given drift time with a TIC value. This can be understood this way: — *"For each drift time value, what is the accumulated ion current of all the ions having that specific drift time?"*.

At this point, mineXpert displays the drift spectrum (mobilogram).

2 Visualizing mass spectral data

Data mining, in mass spectrometry, entails, for a large part, the relentless scrutinization of the mass spectra by an expert eye. Without a powerful mass spectrum viewer, capable of numerous data display modes, the expert eye remains powerless.

After having completed this chapter you will be able to perform mass spectrum visualization and analysis, optionally reporting all the analysed peaks to a file on disk.

2.1 Opening mass spectrum files

To start a mineXpert session, open one or more mass spectra using the menu Open full mass spectrum file(s) from the *File* menu.

mineXpert understands the mzML format and the file loading procedures are delegated to the excellent libpwiz library from the ProteoWizard project[1]. Simple txt,asc data where *m/z* and *i* values are separated by any character that is neither a newline nor a dot nor a digit (loading is handled by a private parser) can be loaded either from file or directly from the clipboard. A third format is SQLite3, a private open/documented database format (private parser) that is used in mineXpert to allow slicing very big datasets into smaller chunks. Incidentally, the SQLite3 format allows for faster data loads.

There are two variants of the mass spectrometry file opening menu, one for which *all* the mass data are read from file and stored in memory and one for which the mass data are read from file in streamed mode, used to compute the TIC chromatogram and discarded. The latter mode is useful when the mass data are so large that they cannot fit in memory. The TIC chromatogram that is computed in streamed mode is then used to access the mass data in the file according to criteria set by the user (retention time range, for example).

2.2 The Window Layout

The graphical interface of mineXpert comprises a number of windows where data and informations are displayed. These windows are described below (see Figure 2.1, "General view of the graphical user interface"):

- *mineXpert* main program window: this is an unintrusive window sporting the main menu and a status bar where messages are displayed;
- The *Loaded mass spectrum files* window, that lists all the mass spectrometry data files that are currently loaded in the program;

1 Please, see http://proteowizard.sourceforge.net/

- The *TIC chromatogram* window[2] where the various TIC chromatograms are displayed for the various mass spectrometry data files that have been loaded. There is, by definition, a single TIC chromatogram per data file currently loaded in the program. However, this window will also display TIC chromatograms that are computed as an integration step from the other windows, like from the *Mass spectrum* window or from the *Drift spectrum* window. In this case, the chromatogram is an extracted ion current chromatogram (XIC chromatogram);

- The *Mass spectrum* window, where the various mass spectra are displayed. A given mass spectrum may originate from a TIC chromatogram or from a drift spectrum, or even from a color map. A given originating chromatogram or drift spectrum or color map may be the origin of more than one derived mass spectrum;

- The *Drift spectrum* window, where the various drift spectra are displayed. Drift spectra can originate from the TIC chromatograms, from the mass spectra or from the color map;

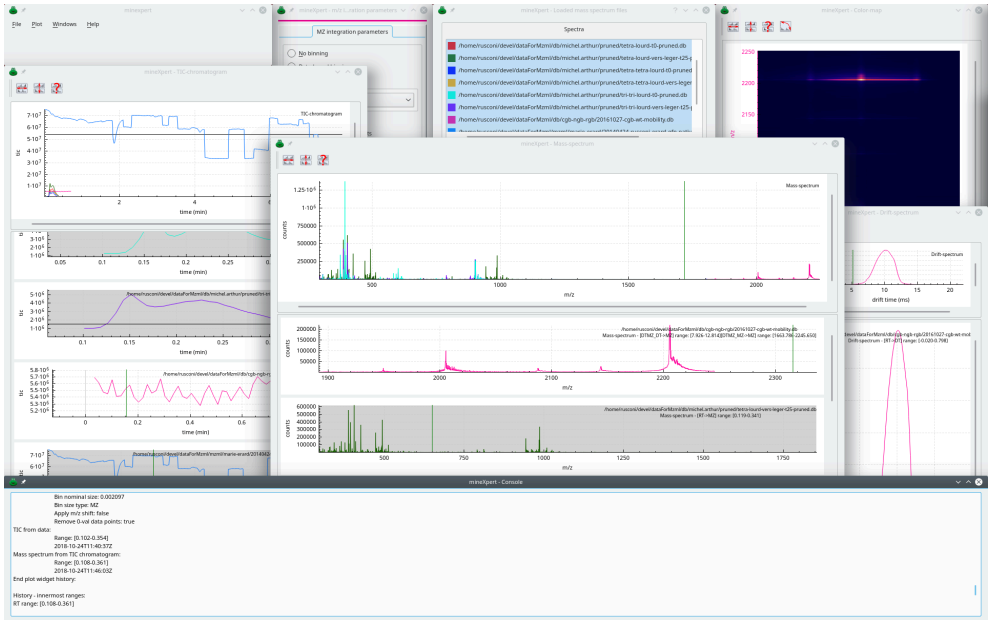
- The *Color map* window, that contains a single color map for each loaded mass data file. At the time of this writing, there is no way to produce a color map from any other window;

- The *m/z integration parameters* window, where the parameters governing the mass data integrations to a mass spectrum are set;

- The *XIC extraction parameters* window, where the parameters governing the XIC extractions to a XIC chromatogram are set;

- The *Console* window, where the various messages or analysis data elements are displayed for the user to select, copy and paste in an electronic lab-book;

2 TIC stands for "total ion current."



The position and size of all the windows can be stored for a later session

FIGURE 2.1: GENERAL VIEW OF THE GRAPHICAL USER INTERFACE

2.3 The Main Program Window Menu

The menu bar in the main program window displays a number of menu items, reviewed below:

- *File*
    - *File* -> Open full mass spectrum file(s) Choose the mass spectrum file(s) to load. Note that the "full" descriptor indicates that the user wants to actually load the full data set in memory. This means that she explicitely knows that the system's memory will cope will all the data in the file;
    - *File* -> Open streamed mass spectrum file(s) Choose the mass spectrum file(s) to load. Note that the "streamed" descriptor indicates that the user wants *not* to actually load the full data set in memory. This is typically the case when the data file is so large that its data cannot fit in memory. The program then only "looks" at the data in the file and crafts, piecemeal, the TIC chromatogram and the color map. In this context, any other data integration will be performed by looking into the same mass data file since no data are available in memory;

- *File* -> Mass spectrum from clipboard creates a mass spectrum from a textual representation of (m/z,i) pairs in the same format as described above for the txt,asc file format;

- *File* -> Analysis preferences Define the analysis preferences. The analysis preferences govern how the data about scrutinized

- *Plot* -> Clear plots Clears all the plots currently displayed in the program. The plot items in the *Loaded mass spectrum files* window are all removed. Note that this releases all the memory that was used by the data. This menu is equivalent to "closing all files";

- *Windows*

  The menus are self-explanatory, as they explicitely explain which window is to be shown. The *Save workspace* menu records on disk the position and size of all the windows, so that upon reopening the program, the windows all position themselves at the recorded position and size;

- *Help*

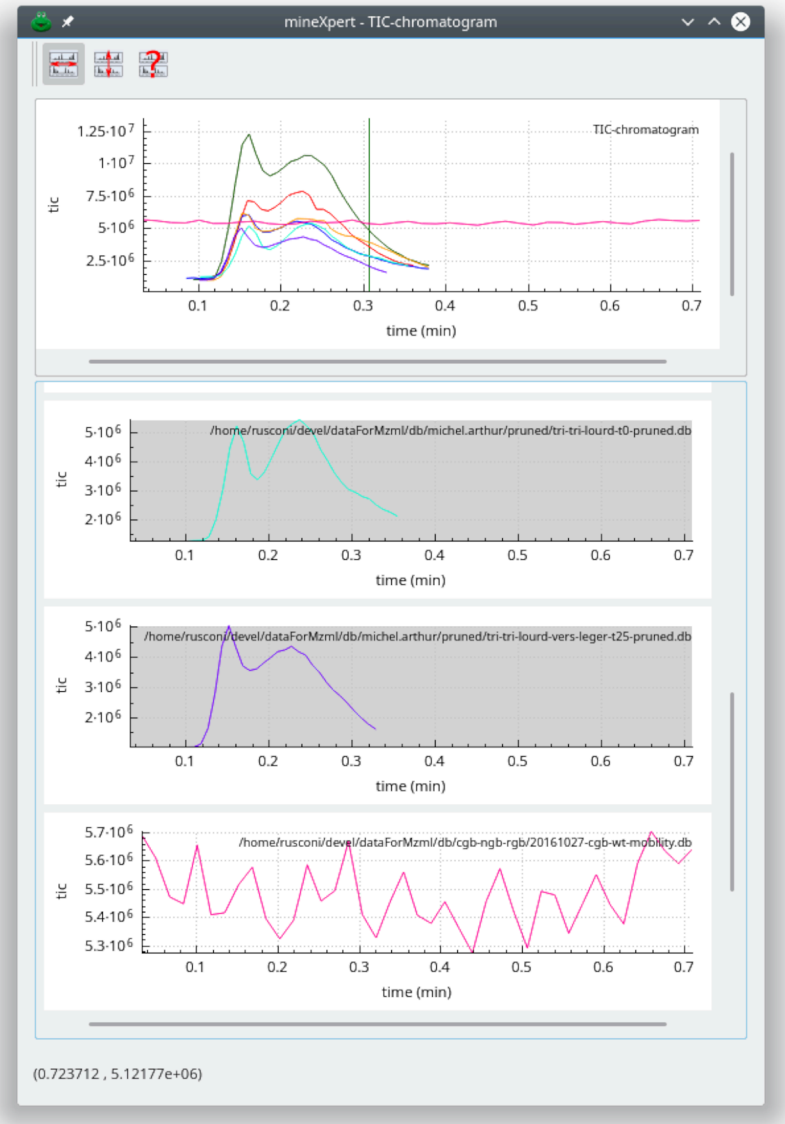
  This menu's items show help about the program itself and also about the Qt libraries that were used to build it. These informations are essential in case the user wants to make a bug report.

2.4 The main data windows

This section will succinctly describe the main data windows of mineXpert. Each window will be described in greater detail when the features of the program will be described.

2.4.1 The TIC chromatogram window

Each time a new mass spectrum file is loaded, its corresponding TIC chromatogram is computed and then displayed in a new plot widget in the TIC chromatrogram window (Figure 2.2, "The total ion current (TIC) chromatogram window"). Each new TIC chromatogram plot generated as a result of the loading of a mass spectrometry data file is plotted using a new color. That color encodes the filiation of the whole set of plots that are generated starting from that initial TIC chromatogram plot. For example, a red TIC chromatogram plot that serves as the starting point for a mass spectrum integration will trigger the creation of a mass spectrum plot widget that will have a red graph in it. Same is true for the color map widget that has its axis and tick labels of the same color as that of the TIC chromatogram plot.



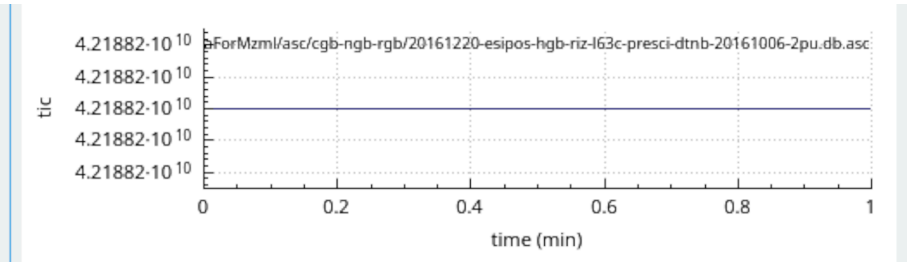
As many traces as necessary can be shown in the window. Each plot has its own set of markers.

FIGURE 2.2: THE TOTAL ION CURRENT (TIC) CHROMATOGRAM WINDOW

NOTE: CREATING A MASS SPECTRUM FROM NON-PROFILE ACQUISITIONS

The attention of the reader is drawn on the specific situation whereby the user loads mass spectrometric data from a non-profile acquisition data file or from clipboard data. For example, when a mass spectrum is opened from a txt,asc,xy text-based format file where the data correspond to a *single* spectrum, *not* a sequence of spectra, the TIC chromatogram really has a single (rt,i) pair denoting the TIC intensity a the single retention time of that very unique spectrum. The TIC chromatogram window thus artificially

creates and displays a TIC chromatogram that is a simple line, like shown in Figure 2.3, "Loading a single-spectrum data file or a mass spectrum from clipboard data". Because the is a single point, the user has nothing to do than "integrate" these data to a mass spectrum. This is the reason why this integration is actually performed automatically and the spectrum is thus shown in the *Mass spectrum* window.

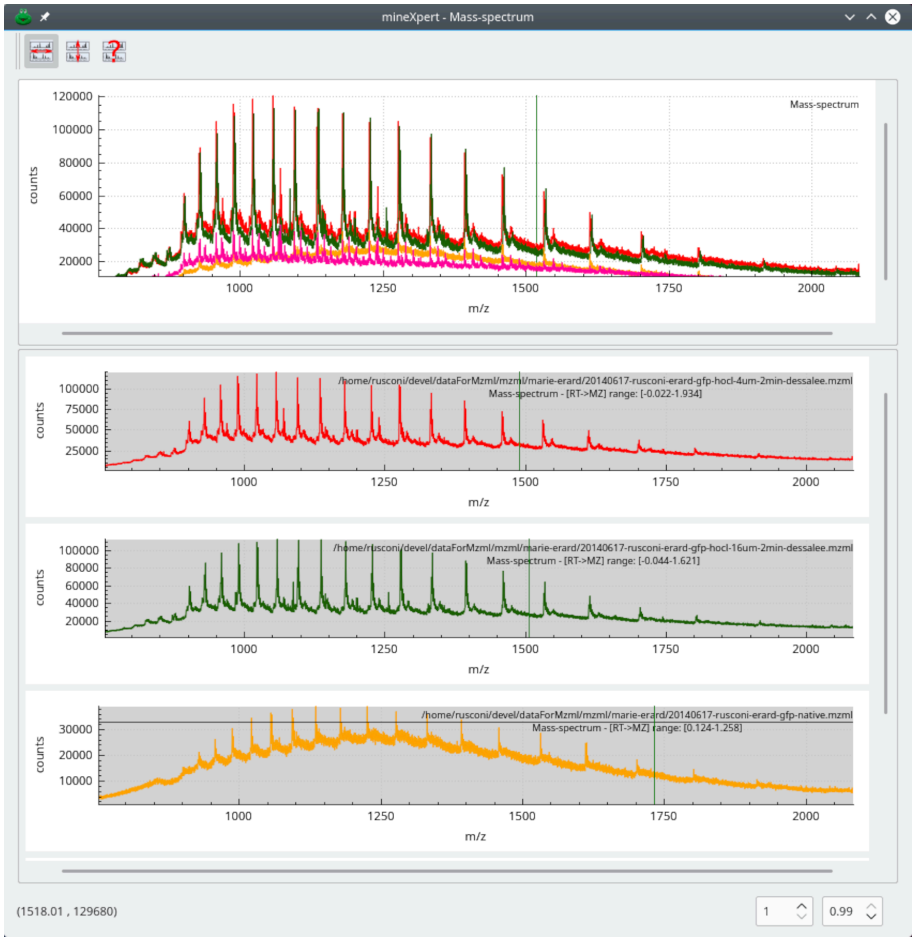


When data are loaded from a file containing a single spectrum or from the clipboard, the TIC chromatogram only contains a single data point. The mass spectrum is automatically displayed in the *Mass spectrum* window.

FIGURE 2.3: LOADING A SINGLE-SPECTRUM DATA FILE OR A MASS SPECTRUM FROM CLIPBOARD DATA

2.4.2 The mass spectrum window

The mass spectrum window contains all the plot widgets that display mass spectra that originated in other windows. For example, the user might select a region in a TIC chromatogram and then ask that a mass spectrum integration be computed. In this case, the resulting mass spectrum is displayed in a new plot widget that is located in the mass spectrum window (Figure 2.4, "The mass spectrum window").

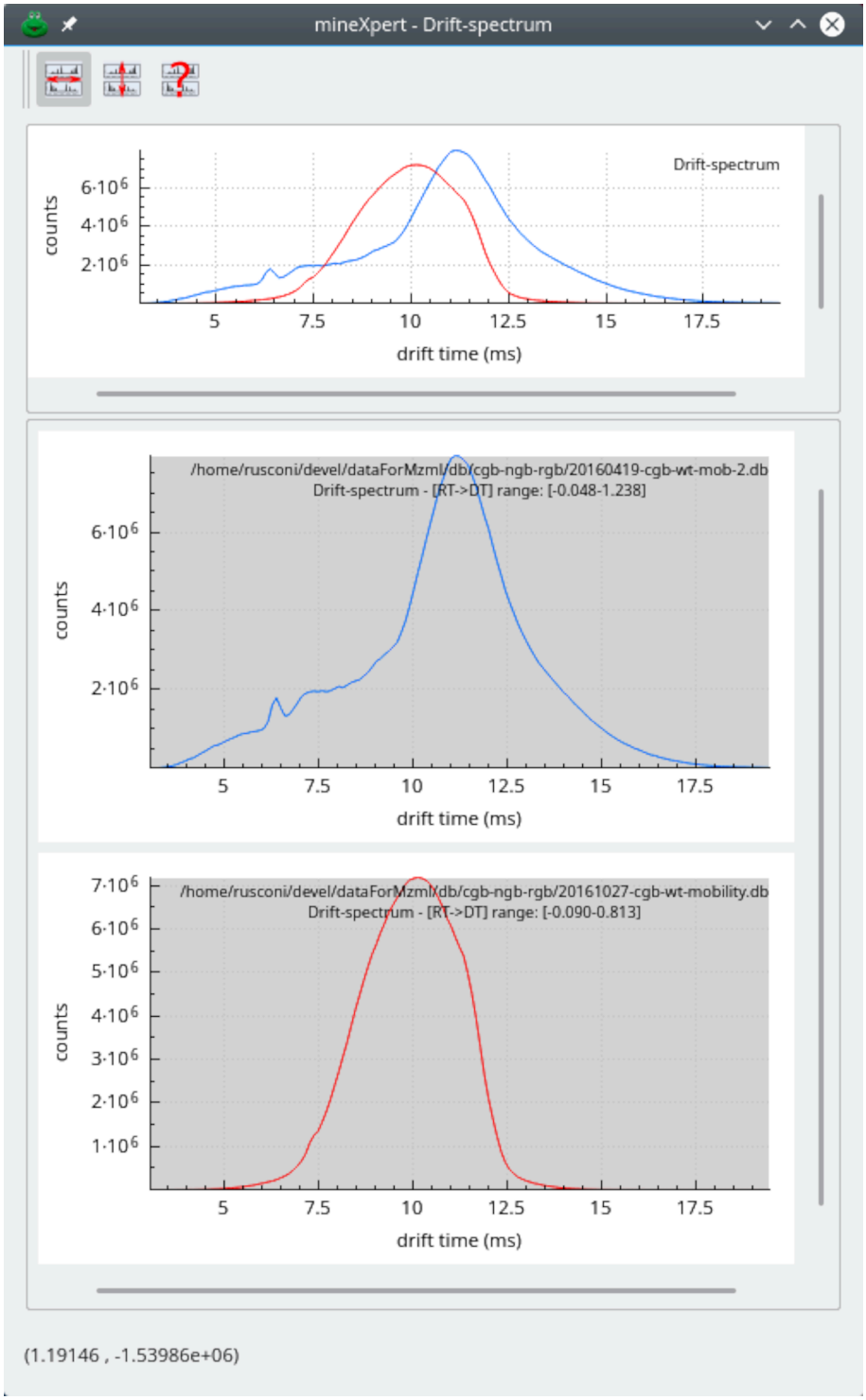


This mass spectrum window shows multiple mass spectra.

FIGURE 2.4: **THE MASS SPECTRUM WINDOW**

2.4.3 The drift spectrum window

As described for the mass spectrum window, the drift spectrum window contains all the plot widgets that display drift spectra (Figure 2.5, "The drift spectrum window").

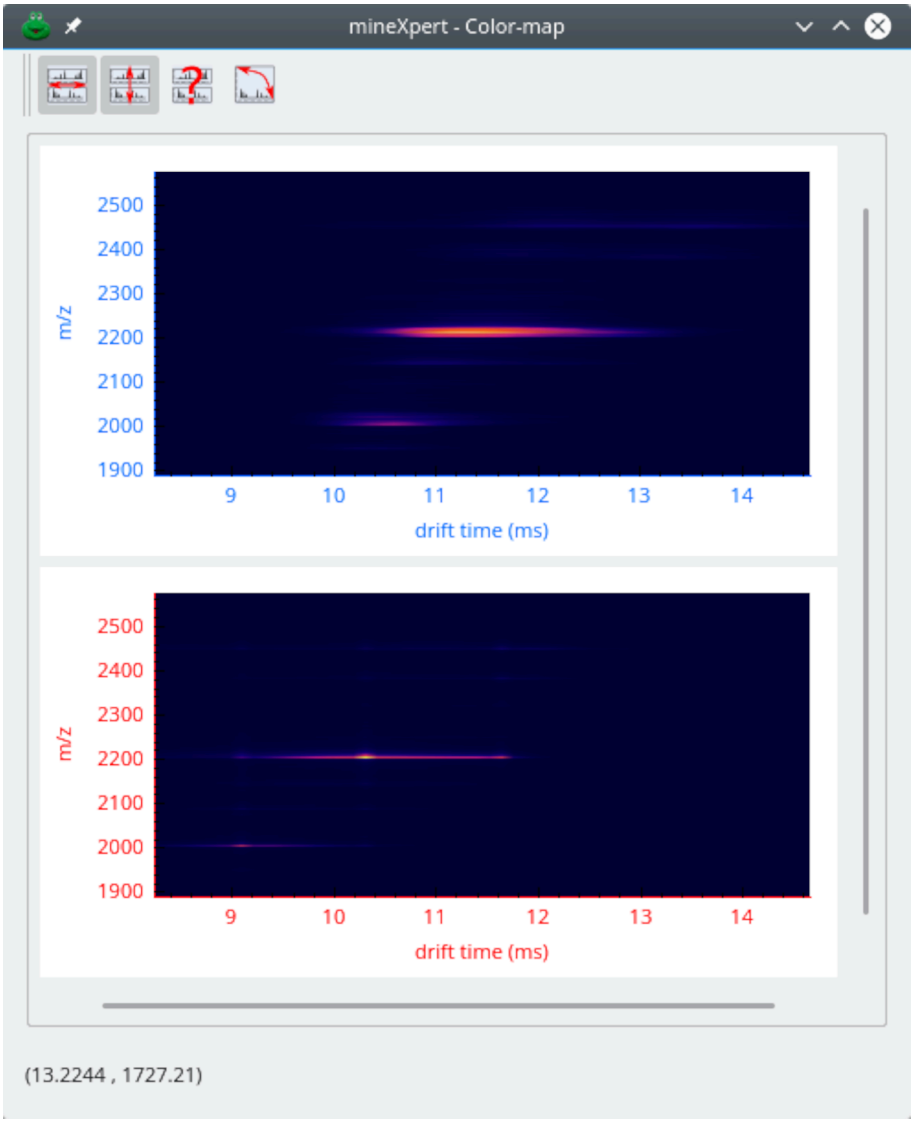


This drift spectrum window shows multiple mass spectra.

FIGURE 2.5: THE DRIFT SPECTRUM WINDOW

2.4.4 The color map window

The color map window displays a color map view of the drift data in the form of m/z *vs* drift time (dt). The intensity of the m/z values is coded in colour. The axes can be switched, such that either the m/z *vs* dt or the dt *vs* m/z representation can be obtained (see toolbar button on Figure 2.6, "The color map window").



The color map window relates mass spectra to drift times.

FIGURE 2.6: THE COLOR MAP WINDOW

2.5 General structure of the windows containing plot widgets

The TIC chromatogram, mass spectrum and drift spectrum windows are all structured in a similar way. The window is divided vertically in two compartments. The bottom compartment will host all the plot widgets stacked vertically. The top compartment hosts a single plot widget where all the graphs that are displayed unitarily in the lower compartment are shown superimposed.



Note: The multi-graph plot widget at the top of the windows

The plot widget that is packed in the top compartment of the window is called the *"multi-graph"* plot widget because it can hold more than one graph. The plot widget(s) that is(are) packed in the bottom compartment of the windows is(are) called *"single-graph"* plot widget(s) because each plot contains only one graph.

The two vertical compartments of the window are resizable by dragging the sliding horizontal bar that separates them. It is possible to totally occlude one of the compartments by dragging that sliding bar all the way up (or down) to the window side.

The behaviour described above does not apply to the Color map window that has no upper compartment with all the color maps superimposed.

2.6 The data-displaying widgets

The TIC chromatogram, color map, mass spectrum and drift spectrum windows all contain plot widgets (or color map widgets) that have a general working scheme as to how the data can be visualized. The main visualization operations are succintly described below. The following convention will be used to describe the mouse buttons:

- : left mouse button;
- : middle mouse button;
- : right mouse button.

The different plot or colormap graph visualization methods are detailed below:

- Zooming in and zooming out:

  - Zoom in: 🖱-click-drag to draw a selection rectangle. When the mouse button is released, the new plot view contains the data contained in the selection rectangle;

  - Zoom in: 🖱-click-drag along the X-axis over the region to zoom and release the mouse button. The new zoomed view does not automatically scale to full scale in the Y-axis direction. To ensure that the new view automatically scales on the Y-axis, press `Shift` while releasing the mouse button. When the zoom in operation is performed using `Shift` in the multi-graph widget, the Y-axis is set to full scale with respect to the point having the maximum intensity of all the graphs being shown at that moment;

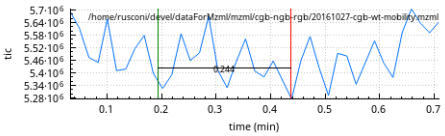


FIGURE 2.7:

As seen in the figure above, the region defined by the 🖱-click-dragging operation is delimited by green and red markers, respectively at the start and at the end of the selection. The distance between the start and end points is updated along the mouse move operation.

  - Zoom in/out: 🖱-click-drag on the X- or Y-axis to interactively zoom in or out along the selected axis. In this mode, the zoom operates by contracting/expanding the data in such a manner that the left/bottom part of the graph (the origin of the graph) is anchored and does not move. When the drag occurs towards larger values on the clicked axis, the view is zoomed in along that axis. Conversely, it is possible to zoom out by dragging the mouse towards lower axis values. When the number of points in the plot is so large that the zoom operation is sluggish, pressing `Ctrl` will fluidify the zoom operation;

- Zoom in/out: The 🖱-wheel-rotation can be used to zoom in or out the whole plot on both the X- and Y-axis simultaneously. Note that the position of the mouse cursor when the wheel is rolled defines the new view of the plot. Practising a bit allows to make that zooming in/out mode very powerful.

- Zoom out: To reset the zoom along one axis, 🖱-double-click that axis. In this case, only the clicked axis will be full-scale, the other axis remains unchanged. To reset the zoom such that the full scale is calculated on the data set displayed after the zoom, maintain the `Shift` key pressed when double-clicking. To reset the zoom on both axes in one go, 🖱-double-click one of the axes maintaining the `Ctrl` key pressed;

- Panning:
  🖱-click-drag on one of the axes to pan the plot view along that axis;

- History:
  Each time a new zoomed in/out view of the plot is triggered, a history element is stored in the plot widget. To back-replay the various steps of the zoom in/out operations in sequence, from pre-last to first, hit the `Backspace` key. The exceptions to this mechanics is when the plot view is panned or when the mouse wheel is used.



Note: Locking the x and y axes of all the plots

The tool bar located at the top of the windows described above contains two buttons that allow to lock the x axis (the button icon has the horizontal red line) and/or the y axis (red line is vertical) range throughout all the graphs displayed in the window. This of great use when the user wants to compare a number of graphs that have been obtained on comparable samples. The movements and zooming-in or zooming-out operations in one graph are then synchronized to all the other graphs.

The third button performs a transpose operation. When the color map is initially created, the horizontal axis (the keys of the map) is the drift time axis and the vertical axis (the values of the map) is the m/z axis. The transpose operation switches the representation of the map such that the axes are inverted.

2.7 Data integrations featured by mineXpert

Analyzing mass spectrometric data (with or without drift data) usually involves performing various data integrations in sequence. We saw earlier that the first data that are plotted upon loading a mass spectrometry data file are the TIC chromatograms along with (if applicable) the m/z *vs* dt color maps. These two graphed data sets are the starting points for the mass spectrometric data mining, that may involve the following integration operations:

- *TIC chromatogram to mass spectrum* This kind of operation is triggered upon 🖱-click-dragging the mouse over the region of interest and maintaining the S key pressed. mineXpert integrates all the spectra that have been acquired at all the retention times between the start and the end of the selected region. A new mass spectrum is then plotted in a new plot widget in the mass spectrum window;

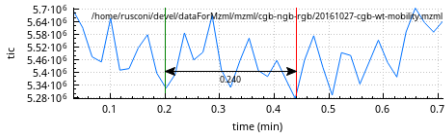


FIGURE 2.8:

  As seen on the figure above, the region defined by the 🖱-click-dragging operation is delimited by arrows, a green marker at the start and a red marker at the end.

- *TIC chromatogram to drift spectrum* This kind of operation is similar to the one described above, unless the D key must be pressed. As above, a new drift spectrum is appended to the drift spectrum window.

- *Color map to mass spectrum* This operation involves 🖱-selecting a rectangular region of interest on the color map and by maintaining the S key pressed. A new mass spectrum is then plotted in a new plot widget in the mass spectrum window. Note that, due to a bug in the plotting library, the rectangle is not currently drawn on top of the color map.

- *Color map to drift spectrum* Same as above, but with the S key pressed. As above, a new drift spectrum is appended to the drift spectrum window. Same remark as above.

- The same mechanics is at work in the other plot widget windows. For example, to trigger the integration of a mass spectrum starting from a drift spectrum, simply drag the mouse over the drift spectrum and maintain the S key pressed.



Note

Rule of thumb: when a ma S s spectrum is to be generated, use the S key, when a D rift spectrum is to be generated, use the D key and, finally, when a R etention time TIC chromatogram is to be generated, use the R key.

- One of the most interesting features for detailed mass data mining is the integration to a TIC intensity (-click-drag with I pressed). That integration can be triggered from any of the data windows (any *single-graph*plot widget in any of these windows, that is). No plot is created, the data are simply displayed in the status bar of the window and in the *Console window*.

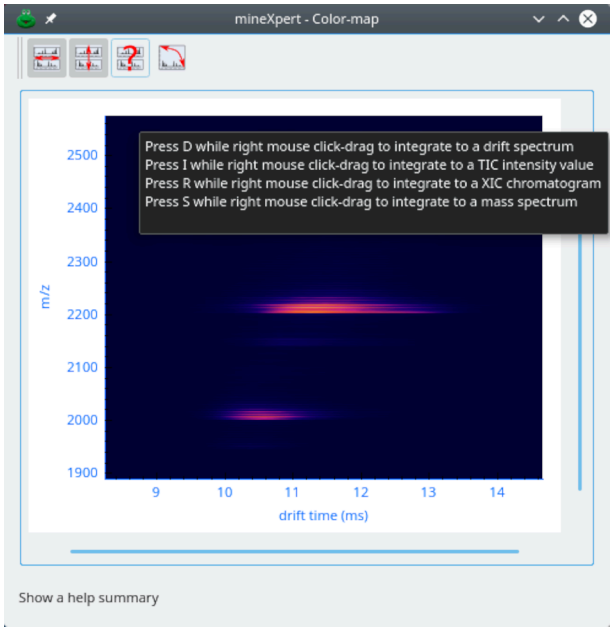


Important

It is important that the mouse is maintained still right after having triggered the intensity calculation because otherwise the status bar message displaying the result vanishes.

This integration will be discussed later.

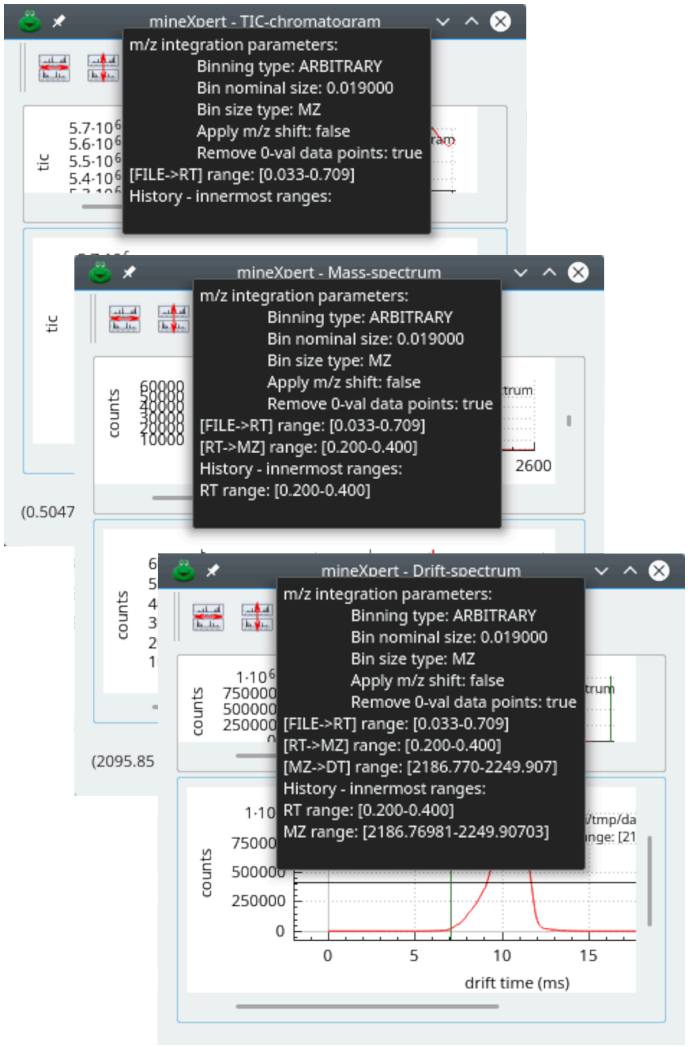
Figure 2.9, "The tool bar and its buttons" shows the various buttons of a plot window. The button with the `?' character will show a tooltip describing the various keyboard/mouse combinations to use to trigger the various data combinations described ealier.



The ? button shows a tool tip helping the user to select the proper keyboard/mouse combination to perform a given integration task.

FIGURE 2.9: THE TOOL BAR AND ITS BUTTONS

The "filiation" of the plots is maintained using identifying colors. However, color is not enough to unambiguously identify the "filiation" of any given plot. Indeed, the same TIC chromatogram or Color map plot can be used multiple times to perform integrations. The newly created plots will have the same color as the originating plot, but it will not be possible to distinguish between all the "child" plots. This is why the plots maintain a "history" of the way they have derived from the initial TIC chromatogram/Color map plot. This history is shown in a small widget that shows up when the 0 key is pressed while the mouse cursor hovers over the widget at hand. One example of plot "history" is shown in Figure 2.10, "The plot filiation history widget".



Each plot has a filiation history that can be displayed by keying 0 .

FIGURE 2.10: **THE PLOT FILIATION HISTORY WIDGET**

This figure shows the filiation history of plots. As shown, the first integration was performed when loading the mass data file in order to produce the TIC chromatogram. The first history item is thus [File -> RT], indicating that the plot widget graph originates from loading a file and computing the TIC chromatogram (RT stands for retention time).

The computed TIC chromatogram served as a starting point for an integration to a mass spectrum. The second history item shows just this, in addition to the first item that is still shown: [RT -> MZ]. The range shown indicates that the integration was performed over the specified range of RT values.

The final integration step was to compute a drift spectrum starting from the mass spectrum and that is denoted using the [MZ -> DT] expression. The concerned range is also shown.

As can be seen in the various filiation history items of Figure 2.10, "The plot filiation history widget", there is always a *History - innermost ranges* section that lists the three ranges for RT, MZ and DT. What is that "innermost range" concept? The idea is that, when chaining integrations, each new plot reflects an always smaller

subset of the initial dataset that was loaded from file. The RT, MZ, DT ranges may thus be reduced progressively. For each of these RT, MZ, DT properties, the innermost ranges is just that: the smallest range that is currently plotted in the plot widget at hand. An example is worth a thousand words:

1. An [RT -> MZ] integration starting from a TIC chromatogram;
2. The m/z range obtained is [500-2500]. From that mass spectrum, the user integrates to a drift spectrum [MZ -> DT];
3. Then, from the drift spectrum, a peak seems interesting and the user back-integrates to a mass spectrum [DT -> MZ];
4. In the mass spectrum, a mass peak is of interest and the user want to see at which retention times the m/z value elutes: she does an integration [MZ -> RT].

The innermost MZ range in the XIC chromatogram obtained at step 4 will be the last m/z range selected for the mass peak of interest, not the range at step 2.

2.8 Irks and Quirks of Data Integrations

Depending on the integrations that are triggered in the various data plot/map widgets, the computations vary significantly. This section will describe the general computation algorithms in such a manner that the mineXpert user can grasp what is actually going on in the guts of the software. The integrations to a mass spectrum are particularly sensitive to some parameters that will be described in detail in the following section.

2.8.1 Integrations to a mass spectrum

This integration occurs when the user 🖱-selects a range in a given plot while pressing the S key. Integrations to a mass spectrum can be elicited from a TIC chromatogram plot, a color map plot or a drift spectrum plot. In all these cases, the integration computation (that is, a mass spectral combination) needs to be aware of the kind of data at hand.

In order to clarify what integration means in the context of the creation of a mass spectrum, that is, the summative integration (also known as "combination") of any number of mass spectra, the following describes a combination in detail.

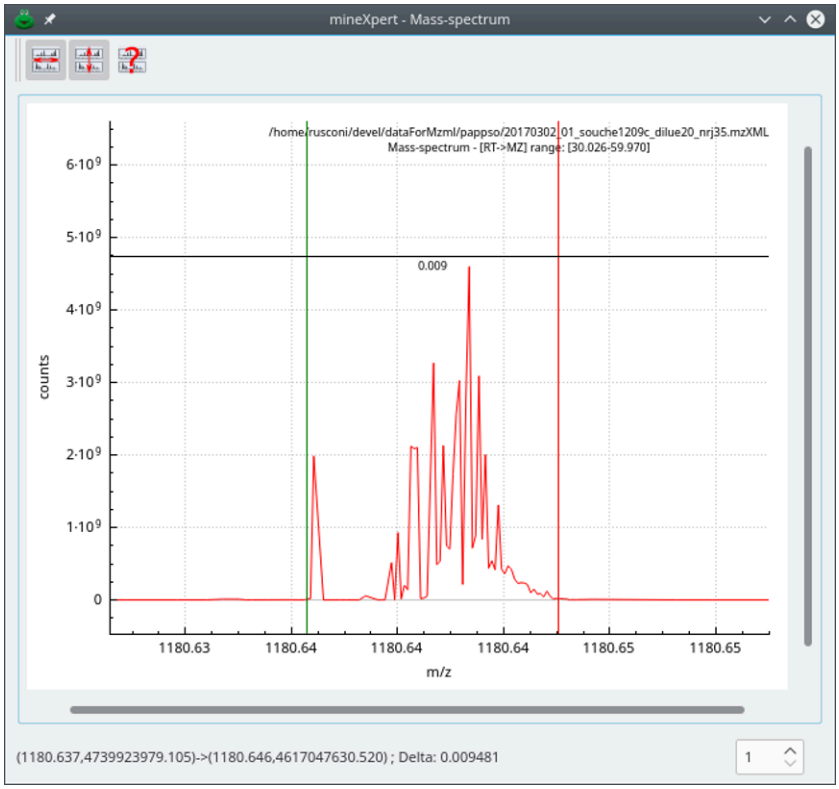
In this example, the user has loaded a mass data file obtained after an acquistion of mass data in profile mode. mineXpert calculates the TIC chromatogram right after having loaded the mass data. The user performs an integration for a given retention time range in the TIC chromatogram. If we consider an integration range [0–15] min, this is what would occur in the guts of mineXpert. In this example, we omit any step corresponding to any binning.

- First of all, create a new mass spectrum (let's call it *newMS*, also known as the "combination spectrum", that is, the result spectrum);
- Extract from the mass spectrometry data all the spectra that have their internal rt value (retention time) contained in the [0–15] min interval. The list of extracted mass spectra (let's call that list *msL*) is then processed as follows:
- Iterate in *msL* and for each iterated mass spectrum (*iterMS*):
    - Iterate in all the (m/z,i) pairs of *iterMS* and for each one check if the m/z value was already found in any of the previous mass spectra, that is, if a (m/z,i) pair in *newMS* has that m/z value. If:
        - the m/z was not found, copy the (m/z,i) pair in *newMs*;
        - else if the m/z value was already encountered in previously iterated mass spectra, increment the intensity of the corresponding (m/z,i) pair of *newMs* by the value of the iterated (m/z,i) pair. This is where the summative combination of mass spectra is at work.

At the end of this process, *newMS* will correspond to the summation of all the spectra contained in the *msL* list. The *newMS* mass spectrum is then plotted in the mass spectrum window as a new plot. The color of the *newMS* plot is the same as the color of the initial TIC chromatogram plot.

The process described above can only work in very limited circumstances, with data files generated with particular instruments. In general, this process does not lead to a usable mass spectrum, as described in Figure 2.11, "Unusable combination spectrum without binning". In this combination mass spectrum, computed from Lumos Orbitrap-originating data, the plot shows what should have been a high resolution monoisotopic peak (the m/z delta of the whole signal is 0.009). As can be seen, the signal in this mass spectrum is totally useless and the integration to a mass spectrum requires binning to overcome the presence of so many peaks in that 0.009 m/z interval.

mineXpert provides a number of ways to configure mass spectral combinations such that the obtained mass spectrum is usable. The m/z integration parameters that might be set are described in the following sections.



The mass data used to compute this combination spectrum originate from a Lumos Orbitrap analyzer. The visible signal should have been a single high-resolution monoisotopic peak. The width of signal is 0.009, as shown by the ruler.

FIGURE 2.11: **UNUSABLE COMBINATION SPECTRUM WITHOUT BINNING**

2.8.1.1 Considerations on the diversity of mass data contents

Loading data from mass data files in mzML format does not guarantee that the data will be of the same kind when they originate from different mass spectrometers. For example, data from Orbitrap mass spectrometers have the following characteristics:

- All spectra do not start at the same m/z value;
- All spectra do not have the same number of data points (they do not have the same size);
- A large number of data points might have 0 values (intensity at a given m/z value is 0);
- The m/z delta between two consecutive m/z values is not constant, and this is the major difficulty for data integration to a mass spectrum.

This is the output of the statistical analysis of the data loaded from a Lumos Orbitrap-originating file:

```
Spectral data set statistics:
Total number of spectra: 6203
Average of spectrum size: 391.311946
StdDev of spectrum size: 168.062934
Mininum m/z value: 400.007111
Average of first m/z value: 401.448935
StdDev of first m/z value: 1.590049
Maximum m/z value: 1999.928589
Average of last m/z value: 1901.852315
StdDev of last m/z value: 45.864131
Minimum m/z shift: -0.344452
Maximum m/z shift: 0.000000
Average of m/z shift: 1.097372
StdDev of m/z shift: 1.590049
Smallest Delta of m/z (step): 0.006195
Average of smallest Delta of m/z (step): 0.023757
StdDev of smallest Delta of m/z (step): 0.013179
Greatest Delta of m/z (step): 405.356934
Average of greatest Delta of m/z (step): 163.112057
StdDev of greatest Delta of m/z (step): 75.947334
```

As mentioned earlier, the most interesting bit of information is in the line reproduced below:

```
Smallest Delta of m/z (step): 0.006195
```

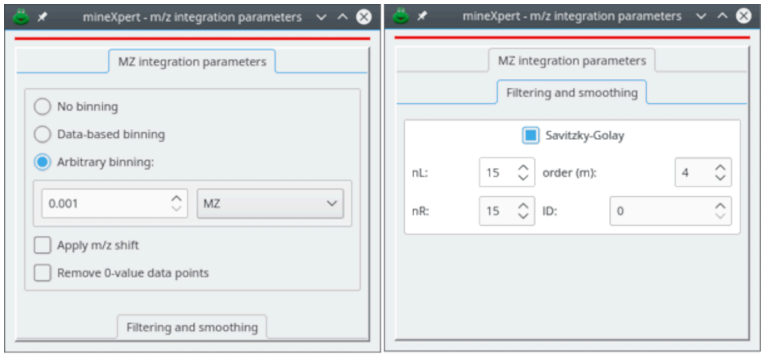
That 0.0062 value somehow gives an indication of the "definition" of the spectrum, that is, the smallest distance possible between two consecutive points in the m/z axis.

In general, the fact that the spectra of an acquisition do not all have the same m/z vector as the m/z axis is a great difficulty for mass spectral integration because it requires setting up binning prior to performing the mass spectral combination. That binning is nothing else than crafting a m/z value vector able to receive the intensities of all the m/z data points in the spectra to be combined. These concepts are developed in the following paragraphs.

2.8.1.2 Statistical analysis of mass data

At the end of the data file loading, mineXpert performs a rudimentary statistical analysis of the data. The main datum of interest is the smallest m/z step that is observed in the whole set of mass data loaded from disk (the mass spectrum list, that can hold mass spectra in the thousands). For each mass spectrum in the list, the smallest m/z delta between any two consecutive data points is recorded. Then, the smallest ever m/z delta value is sought

amidst all the recorded values. Intuitively, that smallest m/z delta value provides an idea of the resolution power of the instrument that generated the mass spectra. Interestingly, this is not the proper value to configure binning. The best value is the median value of the smallest m/z delta values encountered over all the mass spectra of the data file. It is the value that is suggested by default to arbitrarily construct the bins during an integration to a mass spectrum, as described in Figure 2.12, "The m/z integration parameters window" (*Arbitrary binning* value with bin size unit *MZ*).



Integrations to a mass spectrum (whatever the source, TIC chromatogram, drift colormap or drift spectrum) can be configured to ensure the best results depending on the kind of mass data. Proper binning configuration is key to getting best results.

FIGURE 2.12: THE M/Z INTEGRATION PARAMETERS WINDOW

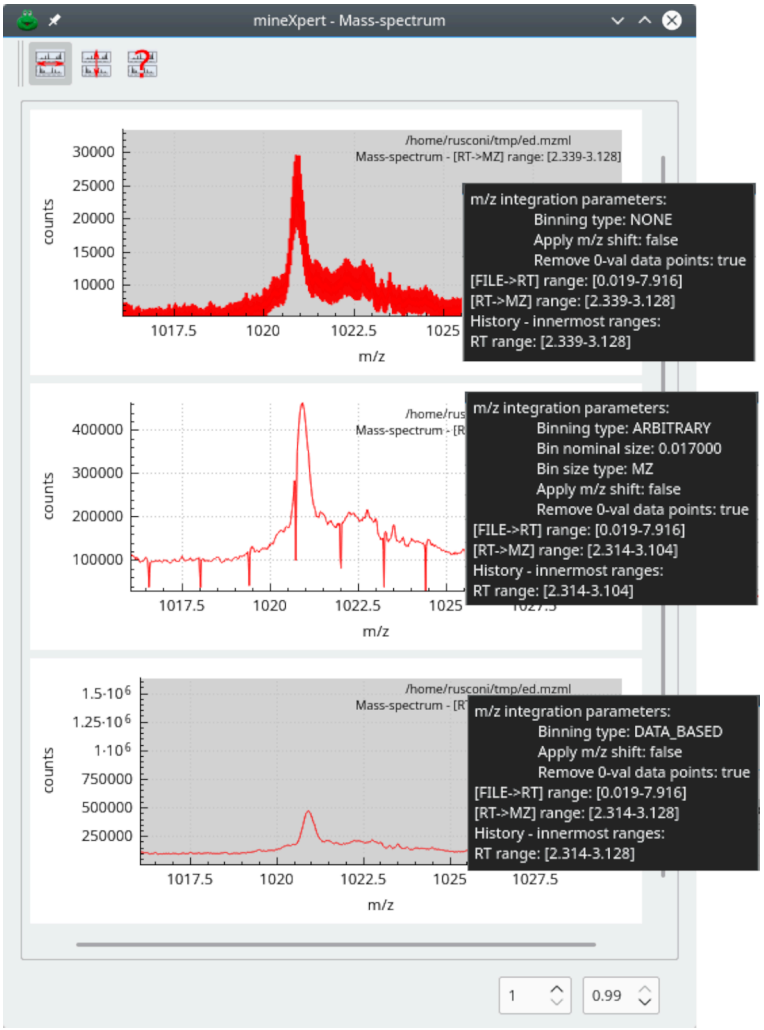
The bin size units, when using *Arbitrary binning*, might be *MZ*, *PPM* or *RES*. In the two latter cases, the bin size changes along the m/z axis. It increases along with increasing m/z values. For example, if the bin size unit is *PPM* and the bin size is *10*, then at m/z 300, the bin size would be m/z 0.003, while at m/z 2000, the bin size would be m/z 0.02. If the bin size unit is *RES* and the bin size is *10000*, then at m/z 300, the bin size would be m/z 0.003, while at m/z 2000, the bin size would be m/z 0.02.

Once the *Arbitrary binning* is selected in the m/z integration parameters window, the bin size and bins size unit have been set, the program creates the bins in the combination mass spectrum according to these settings. The first bin and the last bin are simply the smallest and greatest m/z values found in all the spectra to be combined. The program fills in the void in between these two values in steps matching the bin size with or without PPM/RES ponderation: In case the bin size unit is *MZ*, there is no specific calculation; if the bin size unit is either *PPM* or *RES*, then the bins are calculated accordingly, as shown in the examples above. Once the bins have been set up in the combination spectrum, the actual combination of all the mass spectra can take place.

2.8.1.3 Effects of the m/z integration parameters

This section provides some examples of how the integration parameters might impact the mass spectrum resulting from combination of mass spectra. Also, this section details the general guidelines for ensuring the best combination calculation.

When *Data-based binning* is recommended. Data-based binning means that the bins in the combination spectrum are nothing but the m/z values of the first spectrum of the mass spectral set to be combined. This is the simplest integration mechanism and is recommended when the mass data are perfectly coherent, that is, when all the mass spectra are rooted in the (roughly) same m/z value and the vector of m/z values along the m/z axis is reproduced over all the mass spectra of the combination set. This situation is exemplified in FIGURE 2.13, "BRUKER MICROQTOF ACQUISITION OF A PROTEIN MASS DATA".



In this example, not all the m/z integration parameters produce acceptable results for the combination mass spectrum. The integration with no binning produces an unusable spectrum. Note that the best results are for *Data-based binning*. This is because the m/z data from the Bruker microQTof instrument are very reproducible from a spectrum to the other. Setting the bins exactly as they are in the first spectrum of the mass spectrum list to be combined is thus efficient.

FIGURE 2.13: **BRUKER MICROQTOF ACQUISITION OF A PROTEIN MASS DATA**

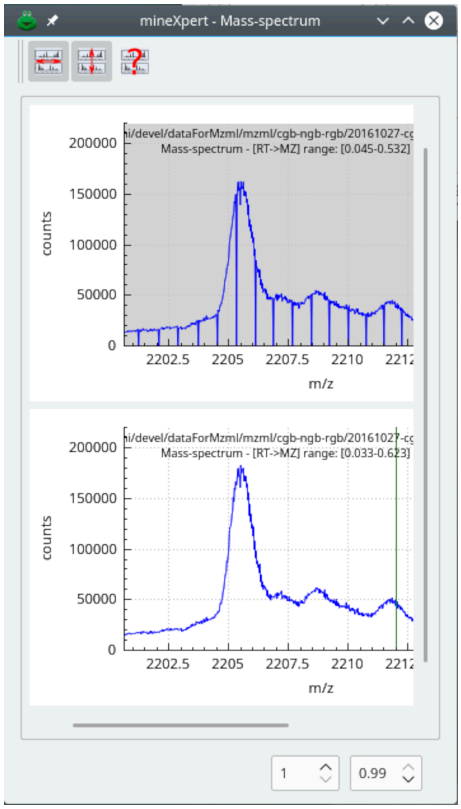
These combination spectra were obtained by performing a mass spectral integration of mass data acquired for a protein solution in a microQTof Bruker instrument. The mass acquisition settings were characteristic of a protein analysis in the 25–35 kDa range. The top spectrum was obtained by performing an integration with no binning at all. The spectrum is useless.

The statistical analysis of the mass data calculated after loading of the mass data had shown that the median smallest m/z delta value was of 0.017. The middle spectrum was obtained after an integration with an arbitrary binning of size *0.017* and with bin size unit *MZ* (constant bin size throughout the m/z vector). The result is much better than the one obtained earlier. Some glitches are still visible, but the data are eminently usable. The bottom spectrum was obtained by performing a combination with *Data-based binning*. This result is the best.



Note: When removing 0-intensity m/z data points is useful

The setting up of bins ultimately consists in creating a mass spectrum out of preexisting data (the first mass spectrum of the set in the case of *Data-based binning*) or out of arbitrary values (the smallest and greatest m/z values of the spectral set, the bin size and finally the bin size unit). In the latter case, the data points making the newly created mass spectrum have their m/z value calculated and their intensity set to 0. Because the m/z value is calculated starting from an arbitrary bin size value, it might be possible that not a single data point in the whole set of mass spectra has a m/z value matching that bin m/z value. In that case, the m/z data point still has a 0-intensity value at the end of the mass spectral combination. This is illustrated in Figure 2.14, "Removing 0-intensity data points". When the 0-intensity data points are not removed (upper spectrum), the signal is deteriorated by these inverted spikes. Removal of the 0-intensity data points, cleans the trace perfectly.



When arbitrary binning is performed, residual 0-intensity data points might survive in the combination spectrum, which deteriorates the resulting mass spectrum. Removing these data points from the combined mass spectrum cleans the trace.

FIGURE 2.14: **REMOVING 0-INTENSITY DATA POINTS**



Note: When Savitzky-Golay filtering of the data might be useful

The Savitzky-Golay filtering method is widely known for its effectiveness in removing noise from mass spectral data. It is possible to apply that filter at the end of a mass spectral combination. The m/z integration parameters window allows setting the Savitzky-Golay parameters:

- *nL*: specifies the number of data points to the left of the point being filtered;
- *nR*: specifies the number of data points to the right of the point being filtered. The total number of points in the window that is considered for the regression is thus nL + nR + 1.
- *m*: specifies the order of the polynomial to use in the regression analysis leading to the Savitzky-Golay coefficients (typically between 2 and 6);
- *lD*: specifies the order of the derivative to extract from the Savitzky-Golay smoothing algorithm (for regular smoothing, use 0);

2.8.2 Integrations to a drift spectrum

This integration occurs when the user 🖱-selects a range in a given plot while pressing the D key. In the detailed example below, the integration occurs for a given retention time range in the TIC chromatogram (integration range [0–15] min):

- First of all, create a <dt,tic> map to store all the drift time values encountered below, along with the cumulated total ion current intensity value of the spectra acquired at the corresponding dt drift time;
- Extract from the mass spectrometry data all the spectra that have their internal rt value (retention time) contained in the [0–15] min interval. The list of extracted mass spectra (*msL*) is then processed in such a manner that each mass spectrum (*ms*) it contains is iterated over:
  - Get the dt at which the *ms* was acquired;
  - Calculate the total ion current (tic) for the *ms*;
  - In the <dt,tic> map, check if the dt value was already found. If:
    - The dt was not already found, create one (dt,tic) pair and insert it in the map;
    - Else if the dt was already encountered in previously iterated mass spectra, increment the tic value of the corresponding (dt,tic) pair in the map by the tic value calculated above for *ms*.

At the end of this process, the <dt,tic> map will correspond to the drift spectrum. That spectrum is then plotted in the drift spectrum window as a new plot with the same color as that of the initial TIC chromatogram plot.

2.8.3 Integrations to a TIC intensity value

This integration occurs when the user 🖱-selects a range in the TIC chromatogram plot while pressing the I key. The integration is performed by looking into the mass data for (m/z,i) pairs that match the current integration history of the current data plot and sums all the intensities to yield a final TIC intensity value. This value is printed in the status bar of the window. Be aware that if you move the cursor right after having performed the computation, the message in the status bar of the window will be erased. In this case, that same value is printed in the console window in the same color as the color of the plot from which it was computed.



Note

Worthy of note is the fact that this kind of integration can be performed in the exact same way in the various data plots (TIC chromatogram, mass spectrum, drift spectrum, mz=f(dt) color map).

2.9 Chained Integrations

The user, in the process of mining the data, will inevitably chain integrations to pinpoint a specific feature of interest. For example, let's say that the user is mining ion mobility mass spectrometry data. After having loaded the data file, the colormap is computed and displayed (see Figure 2.15, "Example of chained integrations"))

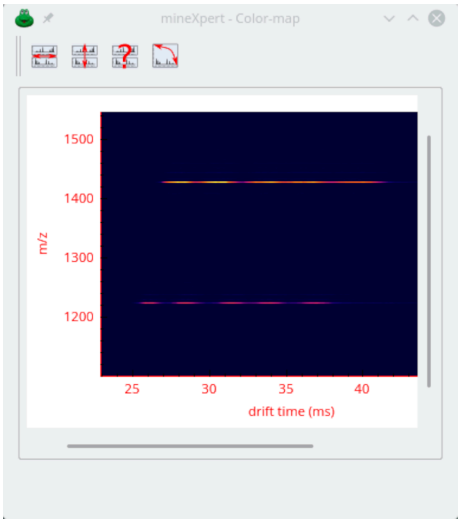


FIGURE 2.15: **EXAMPLE OF CHAINED INTEGRATIONS**

There starts the exploration. The user sees that there are a number of species having discrete drift times at the m/z ratio around 1220 (lower region of the colormap). She thus integrates to a single drift spectrum (🖱-click-drag with D pressed) that horizontal lower region of the colormap. The obtained drift spectrum is shown at the right hand side of Figure 2.16, "One example of chained integrations".

Because there are multiple drift peaks in the drift spectrum, the user perform individual mass data integrations to a mass spectrum for each drift peak (🖱-click-drag with S pressed). The mass spectra obtained are all shown in the middle window of the same figure.

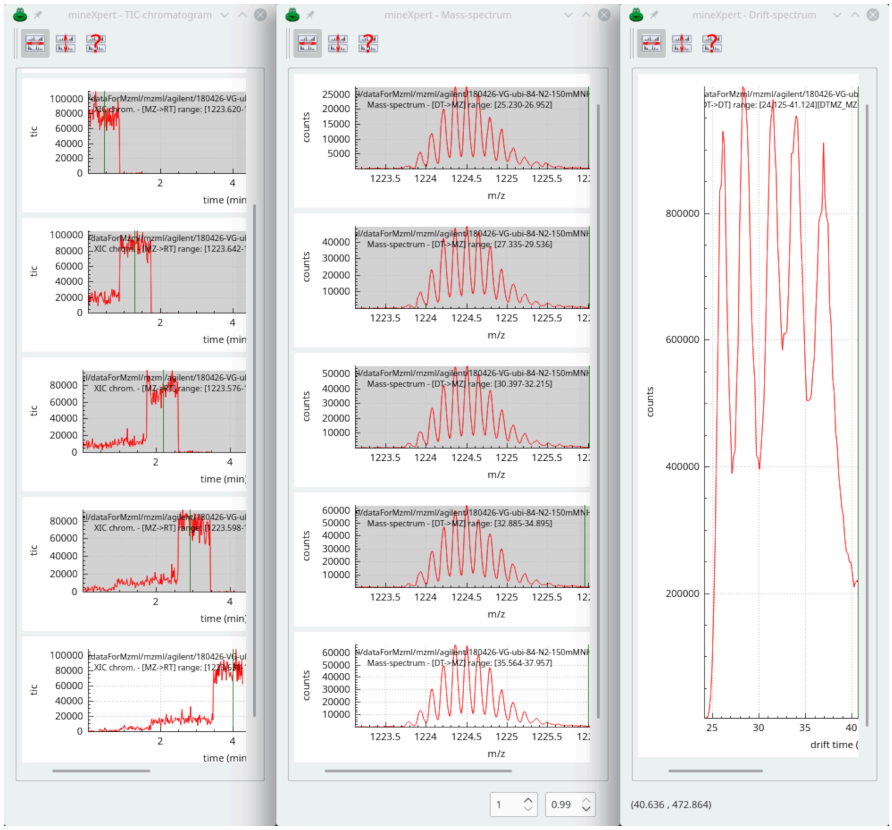


FIGURE 2.16: **ONE EXAMPLE OF CHAINED INTEGRATIONS**

Most interestingly, the various drift regions are integrated to almost identical m/z values in their respective mass spectrum. In order to know when the various molecular species eluted in the chromatogram, the user performs for each mass spectrum an integration to a XIC chromatogram (-click-drag with R pressed). It is then visible that each molecular species was eluting from the TIC chromatogram at discrete retention times (this was clearly not a *true* chromatography but instead an infusion with instrument parameters changed during the acquisition).



Note

The dataset used for the Section 2.9, "Chained Integrations" section are kind courtesy of Dr. Valérie Gabelica and correspond to a work entitled *Optimizing Native Ion Mobility Q-TOF in Helium and Nitrogen for Very Fragile Noncovalent Structures* published in *JASMS* with DOI: 10.1007/s13361-018-2029-4.

2.10 Mass spectral feature analysis

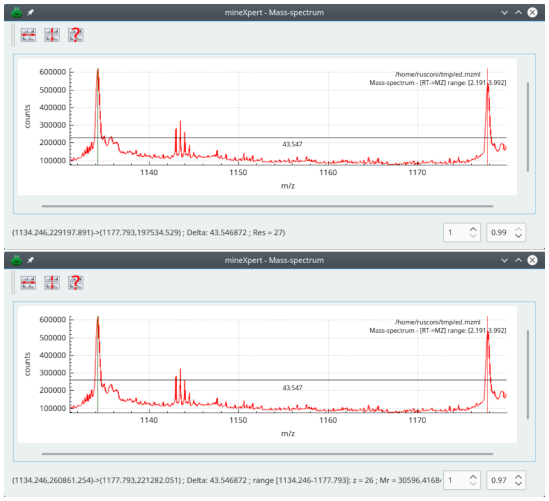
When analysing a mass spectrum, two major deconvolutions are performed to get back to the M_r mass of the analyte while reading m/z values: the charge-based deconvolution and the monoisotopic cluster-based deconvolution. In the following sections, both deconvolutions are described.

2.10.1 Mass spectral deconvolution based on charge state

In this kind of deconvolution, at the present time, the software assumes that the ionization agent is the proton and that the ionization is positive.

The deconvolution is based on the determination of the distance between consecutive (or not) peaks of a given charge state envelope. When the user ⇧-click-drags the cursor from one peak to another, the program tries to calculate if the distance between two peaks matches a charge difference. If so, it computes the molecular (M_r) mass of the analyte whose mass peak is located *under the cursor*. Figure 2.17, "Charge-based mass deconvolution (consecutive peaks)" shows that precise state for two *consecutive* peaks of a charge state envelope.

Note that the charge calculation almost never produces an integer value with no fractional part (say, charge z=15.0) because it is almost impossible to drag the mouse cursor the exact m/z range that would lead to such an integral charge value. Almost always, the charge that is calculated looks like 14.995 or 15.001, for example. Why is it impossible to drag the mouse cursor exactly the interval that would produce an integral charge value? Simply because the mouse moves at discrete positions on the screen that might be more or less far apart, depending on the mouse capabilities. The zoom state over the two peaks also plays a role. It is advised to zoom in as much as possible over the peaks at hand so as to minimize the difficulties above. It may happen however that even zoomed in peaks are not sufficiently distant to allow a charge calculation (this is the case in the upper spectrum of the Figure 2.17, "Charge-based mass deconvolution (consecutive peaks)", where the computation could not be performed). In this case, reduce the stringency over the fractional part that is allowed in the charge. By default, the stringency is set at 0.99, that is, any calculated value that has a fractional part either superior or equal to 0.99 or inferior or equal to 0.01 would lead to a successful round-up/round-down to the nearest integer value. Outside of the]0.01-0.99[interval, no charge calculation is performed and thus no deconvolution is performed. When the stringency is too high, reducing it will allow the deconvolution to be carried-over (see bottom spectrum of Figure 2.17, "Charge-based mass deconvolution (consecutive peaks)").

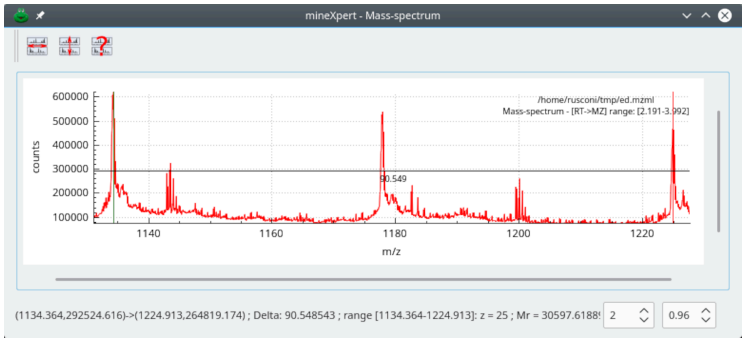


Approach using two *consecutive* mass peaks. Note that the 0.99 stringency set in the spin box in the status bar could not allow any deconvolution to be carried-over, while reducing that stringency to 0.97 allowed it to proceed successfully.

FIGURE 2.17: **CHARGE-BASED MASS DECONVOLUTION (CONSECUTIVE PEAKS)**

The status bar of the window documents the current inter-peak distance measurement operation that is performed by ⇧-click-drag of the cursor starting at the left peak towards the right peak. The start peak is marked with a green marker and the end peak is marked with a red marker. Start and end positions are documented in the form (m/z start, i) -> (m/z end, i). Then, the m/z delta, that is, the distance between both positions is provided. When the end position matches a theoretically expected distance corresponding to a charge difference of 1, then the charge *z* of the peak under the cursor is provided and the molecular mass (M_r) is provided for the analyte whose peak is under the cursor.

It might happen that two *consecutive* peaks of the charge state envelope are not of a good shape enough to point and click precisely in the center of the peaks. In that case, the software allows indicating the number of intervals that run between two ⇧-click-drag-connected peaks. This is illustrated in Figure 2.18, “Charge-based mass deconvolution (non-consecutive peaks)”. The user knew that she had to measure the distance between two peaks that were separated by two intervals. She thus incremented the interval value in the status bar to *2* and performed the measurement. The M_r value that is displayed is different than the previous one because without enlarging the window, it is more difficult to click right at the center of the gaussian shape of each peak. Theoretically, the M_r values should be identical, and actually are when the measurements are performed cleanly in widely-laid mass spectra.



Approach using two *non*-consecutive mass peaks. Note the *z* interval value in the status bar of the window.

FIGURE 2.18: **CHARGE-BASED MASS DECONVOLUTION (NON-CONSECUTIVE PEAKS)**

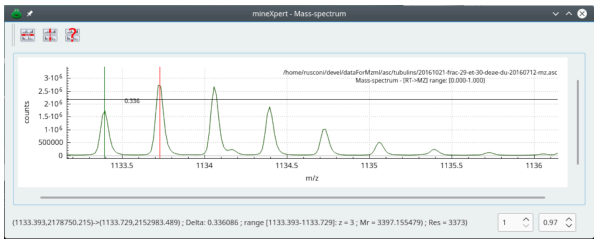


Note: The mouse drag position is significant

Note that the ⌥-click-dragging direction (left -> right or right -> left) has an impact on the value of the charge (*z*) that is obtained, since that charge value is relative to the peak *under* the cursor at the moment of the deconvolution. Conversely, the mouse-dragging direction has no effect on the M_r (molecular mass) of the analyte obtained as a result of the deconvolution process.

2.10.2 Mass spectral deconvolution based on isotopic cluster peaks

In this kind of deconvolution, the user ⌥-click-drags the cursor between the first two peaks (when possible) of the isotopic cluster. The charge state of the ion is the inverse of the m/z delta value that is the distance between the two consecutive peaks. Figure 2.19, "Isotopic cluster-based mass deconvolution" shows that deconvolution process at work.



The user has performed a ⌥-click-drag between the peak under the green marker and the peak under the red marker. The m/z distance between the two markers is computed and the inverse is the charge of the analyte under this isotopic cluster.

FIGURE 2.19: **ISOTOPIC CLUSTER-BASED MASS DECONVOLUTION**

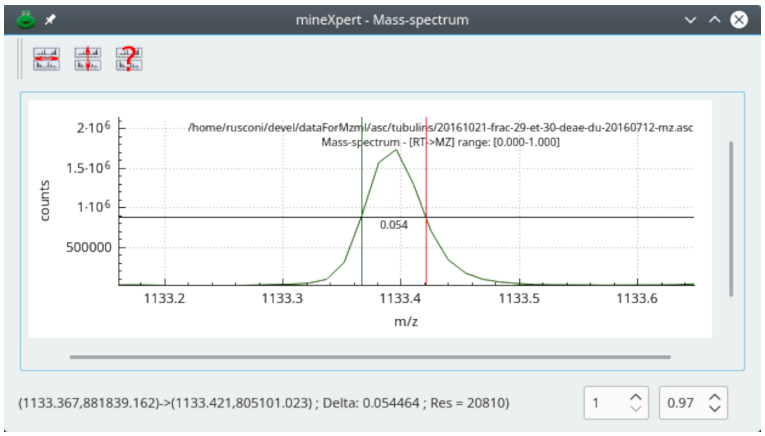


Note: The mouse drag position is not significant

Note that the ⌥-click-dragging direction (left -> right or right -> left) has no impact on the value of monoisotopic mass computed because the software postulates that the lightest ion is the peak on the left.

2.10.3 Reading the resolving power based on mass spectral data

When ⌥-click-dragging the mouse cursor between two mass spectrum locations of interest, the program computes the apparent resolving power. This process is shown on Figure 2.20, "Calculation of the resolving power", where the resolving power is calculated by dragging the mouse cursor from one edge of a peak to the other at half maximum height (this is called *full width at half maximum* [FWHM] resolution).



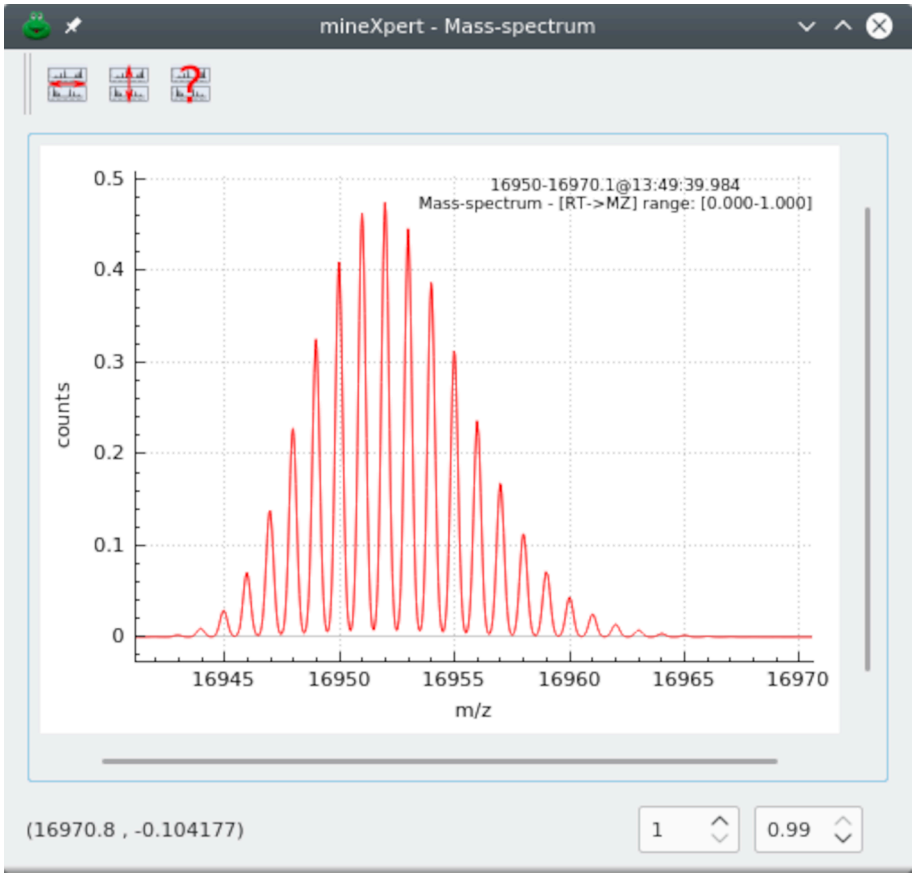
⌥-click-dragging the mouse cursor will trigger the calculation of the resolving power of the instrument. That value is printed in the status bar.

FIGURE 2.20: CALCULATION OF THE RESOLVING POWER

2.11 Calculating Isotopic Clusters with IsoSpec

When the resolution of the mass spectrometer is good, zooming-in on a mass peak may reveal that a given ion has given rise not to one peak but to a set of peaks. This set of peaks is called a "*isotopic cluster*".

It is possible to predict how a given ion (of a given chemical formula) is supposed to be revealed in a mass spectrum, in the form of such an isotopic cluster. One such cluster is shown in Figure 2.21, "Calculation of the isotopic cluster of an analyte", for the horse apomyoglobin protein, of elemental composition C769H1213N210O218S2 (this formula is typeset like this intentionally, to show how the formulæ need to be entered in the IsoSpec module).



Calculated isotopic cluster for the apomyoglobin protein.

FIGURE 2.21: **CALCULATION OF THE ISOTOPIC CLUSTER OF AN ANALYTE**

As of version 5.8.0, mineXpert provides an interface to the `libIsoSpec++` library.

> IsoSpec: Hyperfast Fine Structure Calculator
>
> Mateusz K. Łącki, Michał Startek, Dirk Valkenborg, and Anna Gambin
>
> *Analytical Chemistry*, 2017, 89, 3272−3277
>
> DOI: 10.1021/acs.analchem.6b01459

This library performs high-resolution isotopic cluster calculations. In order to run the calculations, it is necessary to have the following items ready:

- An elemental composition formula of the analyte (for example, H_2O_1). This formula needs to account for the ionization agent that is involved in the ionization of the analyte prior to its detection in the mass spectrometer.



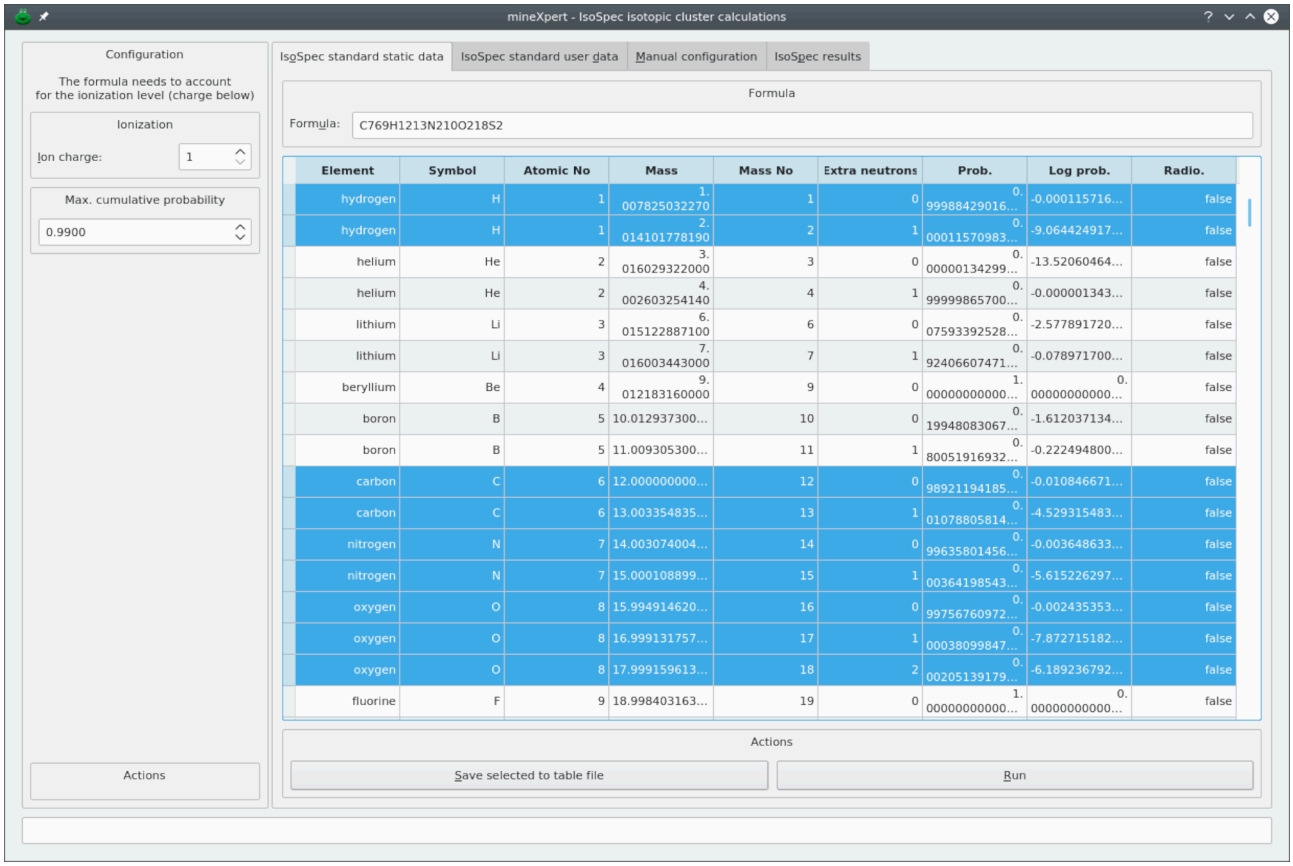
Tip

The IsoSpec software requires that all the chemical elements of a chemical formula be indexed. This means that, for water, for example, the formula should be H2O1 (notice the index 1 after the O element symbol).

- A detailed isotopic configuration of all the chemical elements that are used in the elemental composition formula. mineXpert provides two interfaces to define the isotopic characteristics of the chemical elements. These will be described in the following sections.

2.11.1 The IsoSpec Graphical User Interface in mineXpert

Generating iosotopic clusters using the IsoSpec software package is not easily carried over, in particular because this remarkable library is designed to be highly performant. The authors rightfully put their energy into optimizations for accuracy and speed instead of investing in a graphical user interface. mineXpert provides that graphical user interface, shown in Figure 2.22, "Isotopic cluster calculation dialog window", that shows up upon selection of the "*Utilities*→Isotopic cluster calculator"menu.



The dialog window contains two panels. The left hand side panel configures the charge for which the calculation is to be carried over and the maximum cumulative isotopic presence probability that IsoSpec must reach during the calculation. The right hand side panel contains a tab widget that contains the configuration tabs and the results tab.

FIGURE 2.22: **ISOTOPIC CLUSTER CALCULATION DIALOG WINDOW**

An isotopic cluster calculation is most probably performed with the aim of simulating an expected isotopic cluster for an analyte that is being analyzed by mass spectrometry. It is thus logical that the analyte be in an ionized form. The way that the analyte has been ionized needs to be taken into account in the chemical formula that describes the ion for which the isotopic cluster is being calculated. For example, when determining the chemical formula of a protein in the positive ion mode, the number of protons used to ionize the protein need to be included in the analyte elemental composition formula.



Warning

The IsoSpec software is "charge-agnostic" in the sense that it does not know what element in the chemical formula is responsible for the ionization of the analyte. Therefore, IsoSpec does not know of (and does not care about) the charge of the analyte. The ionization level of the analyte can be handled by mineXpert if that information is set to the *Ion charge* spin box widget. By default, the charge state of the analyte is 1.

The *Max. cumulative probability* spin box widget serves to configure the extent to which IsoSpec simulates the theoretically expected isotopic cluster. A value of *0.99* tells the software to simulate enough combinations of the analyte isotopes to represent 99 % of the theoretically expected combinations.



Tip

For large biopolymers, it might be prudent to start with a relatively low value for *Max. cumulative probability*, because setting this value too high near 1 would increase notably the calculation duration.

To perform isotopic cluster calculations, the simulation software needs to be aware of all the isotopes of all the chemical elements that enter in the composition of the ionized analyte. An isotope is defined by its mass and by the probability that it is found in nature. Carbon has two major isotopes that can be found in nature: the ^{12}C most abundant isotope and the ^{13}C least abundant isotope.

There are three ways that the user might use the IsoSpec software package. Two of them involve a configuration preparation on the part of the user. The third one, that we'll describe first, does not involve any chemical element configuration. The other ways are reviewed in the next sections.

2.11.1.1 Static Standard Element Tables Shipped within the IsoSpec Library

In order to document all the chemical elements' isotopes' characteristics, the IsoSpec library has in its own headers a number of arrays that mineXpert automatically loads up when opening the Figure 2.22, "Isotopic cluster calculation dialog window" dialog window. These data are displayed in the *IsoSpec standard static data* tab. The table view widget is not editable, hence the *static* qualifier in the tab name.

In this tab, all the user has to do is enter the formula for which the isotopic cluster calculation is to be performed. The formula needs to be pasted in the *Formula* line edit widget.

Once the formula has been set to its line edit widget, configure the charge of the ion (*Ion charge* spin box widget) and the *Max. cumulative probability*. Now, click *Run*.

2.11.1.2 Modified Standard Element Tables Shipped within the IsoSpec Library

While the previous section showed how to use the static element tables from the IsoSpec library, this section shows a way to configure these elemental data. For this, start from the standard IsoSpec-based data and modify them according to one's needs. To proceed, select all the element rows of interest from the *IsoSpec standard static data* tab and click *Save selected to table file*.



Tip

The order with which the rows are selected is respected in the export, so make sure to select the rows in the oders that makes sense for your. Also be sure to select rows and not only some cells; for this click in the left margin of the table view widget.

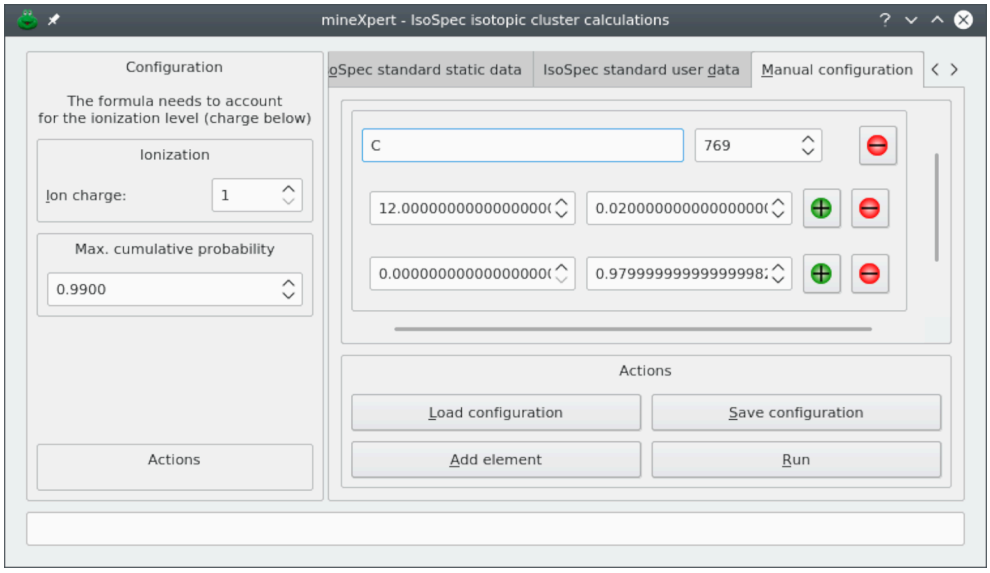
The export is performed as a text TSV (tab-separated value format) file in a layout that closely mimicks the data visible in the table view. Once the file has been saved, open it up in LibreOffice and modify the data to suit your needs. For example, in case of a labelling with ^{13}C, with an efficiency of almost 98 %, lets's change the ^{12}C abundance (probability) to 0.02 and the ^{13}C abundance to 0.98.

Now that the modifications were performed, save the file under the same format and load it in the *IsoSpec standard user data* tab of the dialog window by clicking *Load table from file*. The table view now shows the new carbon abundance values. From now on, use this tab exactly like you did for the standard isospec static data work described above.

Here also, it is possible to select determinate rows and then save them to a file for later use.

2.11.1.3 User hand-made configuration of the element data within mineXpert

The last way the user might configure the chemical elements to be used during an isotopic cluster calculation is based on the fully manual description of the elements and of the isotopes. That configuration is performed in the *Manual configuration* tab of the dialog window. This method is slightly more involved than the previous one but provides also for a much greater flexibility: it allows one to create "new chemical elements" that might be required in specific labelling experiments. The manual configuration is carried over in the *Manual configuration* tab of the dialog, as shown in Figure 2.23, "Hand-made user configuration of the chemical elements and formula"



When the dialog is created, the tab is empty. To start creating element definitions, you click *Add element*.

FIGURE 2.23: **HAND-MADE USER CONFIGURATION OF THE CHEMICAL ELEMENTS AND FORMULA**

Upon creation of the dialog window, the *Manual configuration* tab is empty, with only two rows of buttons at the bottom of the tab. To start configuring chemical elements, you click *Add element* to create an "element group box" that contains a number of widgets organized in two rows:

- Top row, a line edit widget to receive the chemical element symbol, *C* in the example;
- A spin box widget in which to set the number of such atoms in the formula for which the isotopic cluster is being calculated. In the example, we set this value to *769*;
- A button with a "minus" image that removes all the "element group box" in one go;
- The bottom row contains an "isotope frame widget" with two spin boxes for the mass of the isotope being configured (left) and its corresponding abundance (right);
- In addition to the spin boxes, two buttons, with a "plus" or a "minus" figure, allow one to respectively add or remove isotope frames.



Note

It is not possible to remove all the isotope frames from an element group box, otherwise that group box would become useless.

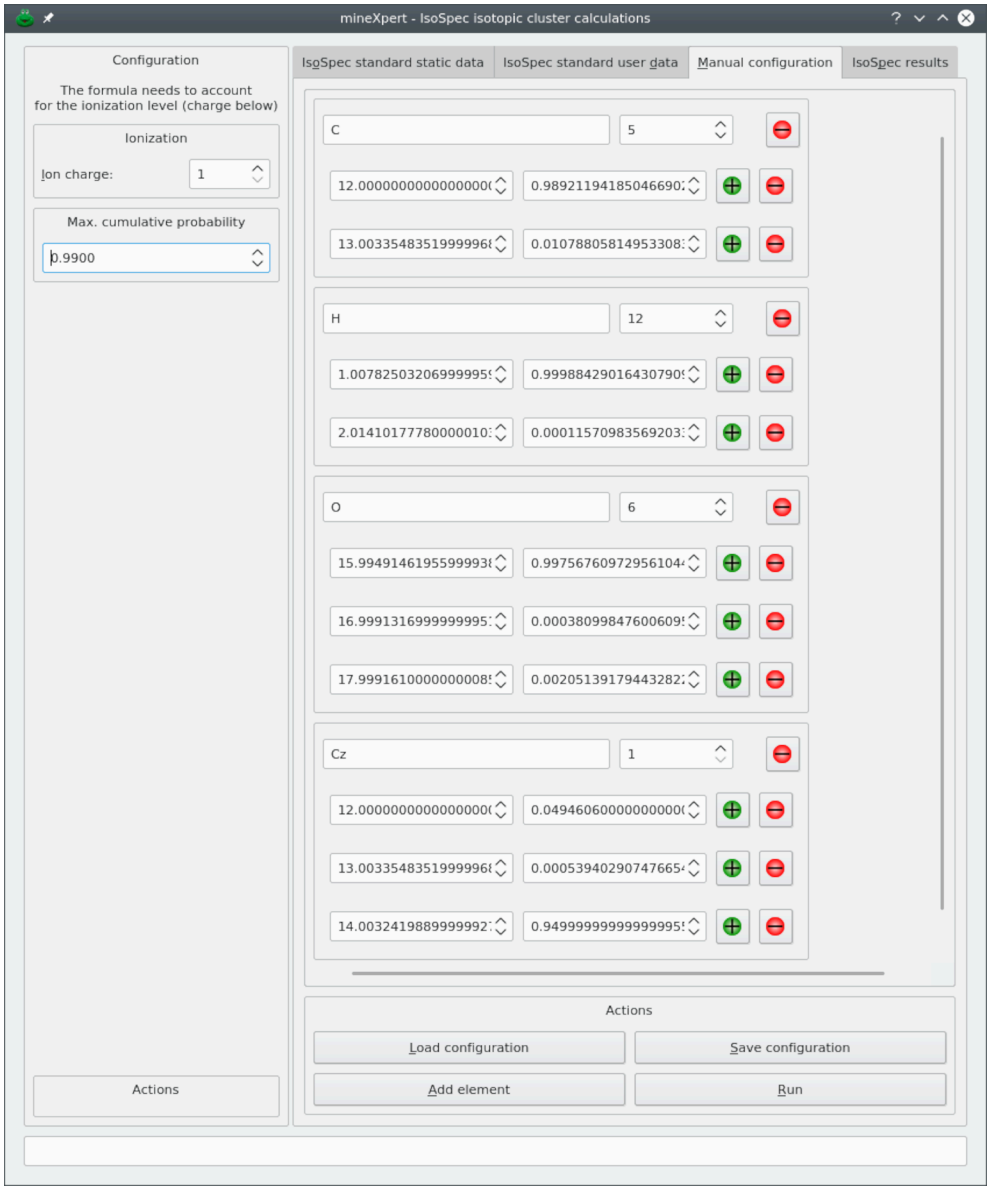
Once an isotope frame has been filled-up, a new line might be required. To create a new isotope frame widget, click any "plus"-labelled button in any of the isotope frames. Once a new frame is created, the spin box widgets that it contains are set to *0.00000*. Fill-in these spin boxes with mass and abundance and go on along this path to create as many isotopes as required.

Once all the isotopes for a given chemical element have been defined, a new element might be needed. For this, click *Add element* and start the configuration of the new element as described above.

The manual isotopic configuration of the chemical elements required to perform an isotopic cluster calculation for a given formula is tedious. The user may want to save a given configuration to a file (click *Save configuration*) so that it is easier to recreate automatically all the widgets upon loading of that saved configuration (click *Load configuration*).

The final configuration is shown in Figure 2.24, "Typical manual configuration of the isotopic characteristics of the chemical elements". The experiment that was configured above is a labelling of a glucose molecule with *Cz*, an imaginary chemical element that is like carbon but that has a ^{14}C isotope. The glucose molecule (normal formula: C6H12O6) is labelled on one single carbon atom with an efficieny of 95 %. This means that, when the labelling fails (in 5 % of the cases) the carbon atom has its isotopes with usual probabilities (compounded by the fact that the normal atom is found at that position only in 5 % of the cases). The isotopic abundances for the Cz element are thus:

- For the ^{12}C isotope: 0.05 * normal ^{12}C abundance;
- For the ^{13}C isotope: 0.05 * normal ^{13}C abundance;
- For the ^{14}C isotope: 0.95;



The user has configured a labelling experiment where the glucose molecule is labelled at a single carbon position with a ^{14}C atom (the efficiency of the labelling is 95 %).

FIGURE 2.24: TYPICAL MANUAL CONFIGURATION OF THE ISOTOPIC CHARACTERISTICS OF THE CHEMICAL ELEMENTS



Note

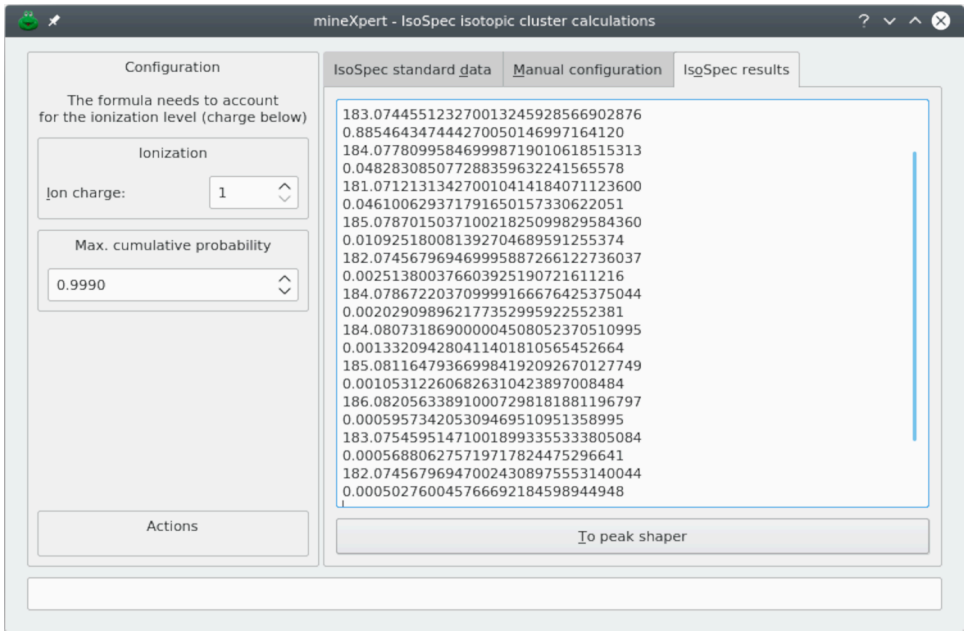
Note that the "normal" carbon count is 5 (and not 6), that the hydrogen count is 13 (and not 12, because the glucose is protonated) and the labelling carbon is present only once.

2.11.1.4 The IsoSpec results are not shaped mass peaks

Once the configurations have been terminated, the calculations can finally be performed by the IsoSpec library. In the manual configuration setting, the formula is automatically handled since each chemical element that is defined goes along with the count of the correponding atoms. In the case of the standard IsoSpec configuration (either modified or not), the user has to enter the chemical formula of the analyte in the *Formula* line edit widget.

Click *Run*. If the configuration was correct and IsoSpec could run the calculation properly, then the dialog window switches to the *IsoSpec results* tab (Figure 2.25, "Results from the isotopic cluster calculation"). That tab contains a text edit widget in which the results are displayed.

Note that the m/z values calculated by IsoSpec are "corrected" for the charge level that was specified in the left panel of the dialog window prior to their display in the results tab (Figure 2.22, "Isotopic cluster calculation dialog window").



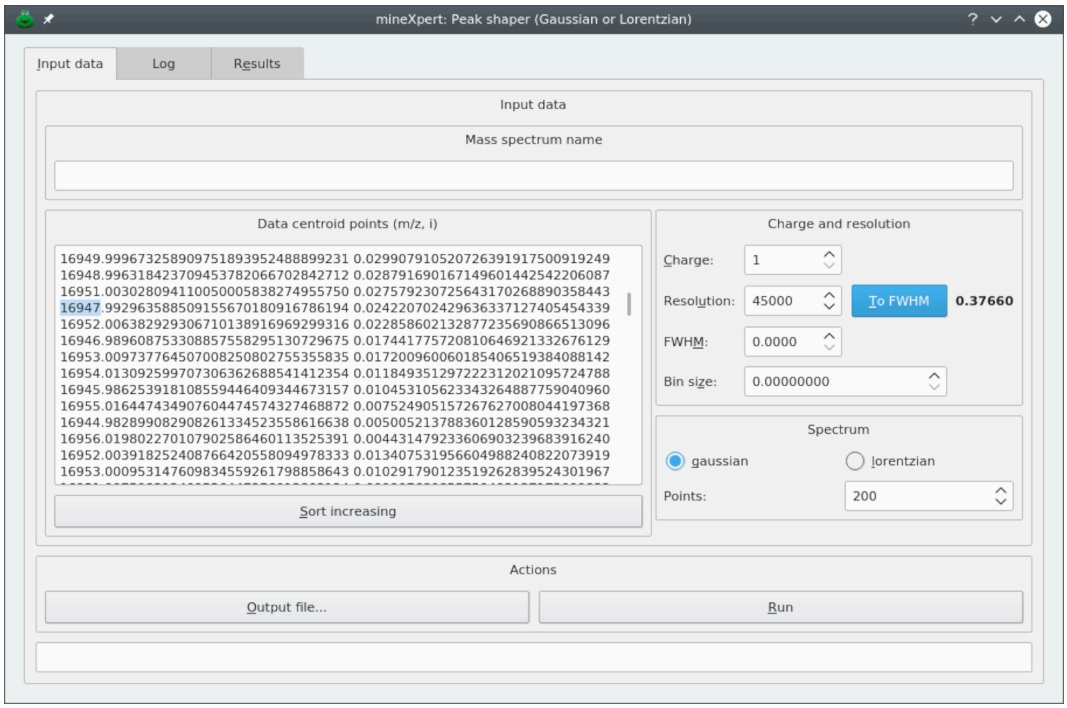
The IsoSpec library computes the relative probability of the various combinations of all the isotopes that make the chemical formula submitted to it. The results are in the form of peak centroid values along with corresponding probabilities. The sum of the probabilities corresponds to the *Max. cumulative probability* value that was set by the user.

FIGURE 2.25: RESULTS FROM THE ISOTOPIC CLUSTER CALCULATION

The results that are produced by IsoSpec represent the peak centroids of the isotopic cluster. The results are thus a set of (m/z,i) pairs that have not the characteristic shape (the profile) that is found in mass spectra. mineXpert features the ability to give a shape to the centroids peaks. For that, click *To peak shaper* to open the *Peak Shaper* dialog window preloaded with the IsoSpec-generated peak centroids. The workings of this peak shaping feature is described in Section 2.12, "Shaping mass peak centroids into well-shaped peaks".

2.12 Shaping mass peak centroids into well-shaped peaks

The shape of mass peaks is typically Gaussian or Lorentzian (or a mix thereof). There are some data simulation or analysis processes that lead to having mass peaks characterized by a single centroid m/z value and a corresponding intensity. Plotted to a graph, a centroid mass peak yields a bar. In order to convert mass peaks centroid into something that resembles a real "profile" mass spectrum, a mathematical formula can be applied (with some parameters) to configure the shapes generated. mineXpert now includes that feature accessible *via* the menu "*Utilities*→Peak centroid shaper". The window that is opened is shown in Figure 2.26, "Setting-up of the centroid mass peak shaping process"



The dialog window allows one to configure the shaping of mass centroid peaks. Setting a spectrum name in the *Mass spectrum name* line edit widget will help recognize the result mass spectrum once displayed in the *Mass spectrum* window (see below).

FIGURE 2.26: SETTING-UP OF THE CENTROID MASS PEAK SHAPING PROCESS

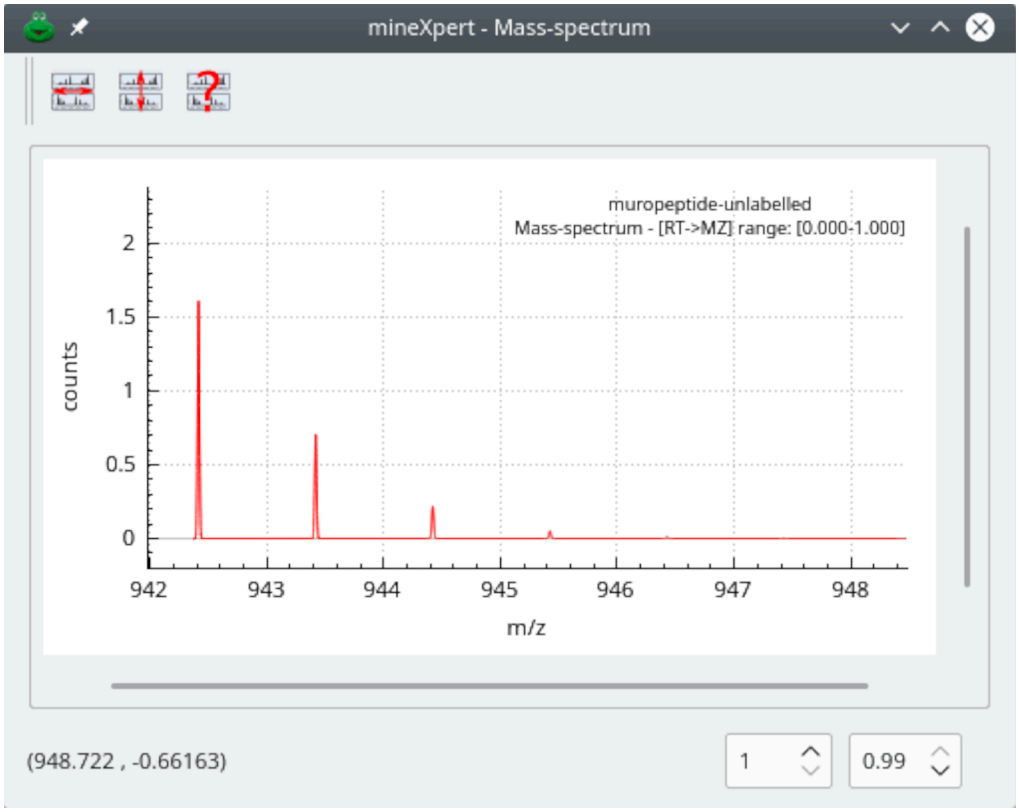
The mass centroid peaks are listed in the *Data centroid points (m/z,i)* text edit widget. These values are pasted there by the user or copied automatically from the isotopic cluster calculation dialog window (see Section 2.11.1.4, "The IsoSpec results are not shaped mass peaks"). The width of the "profile" mass peak is determined either by setting the resolution of the instrument (in the example that is set to 45000) or by setting the width of the peak at half maximum of its height (FWHM).



Tip

It might be useful to have and idea of the FWHM value for a given pair of m/z and resolution values when defining the parameters for the peak shaping process. Double-click-select a single mass peak centroid m/z value from the text edit widget, set the resolving power of your instrument and then click *To FWHM*. The computed FWHM value will be displayed next to the button. The text is selectable so as to copy it to the clipboard and then in the *FWHM* spin box widget.

The spectrum that is generated can be of a Gaussian or a Lorentzian shape. That parameter is configured by selecting the corresponding radio button widget. The number of points used to actually craft the shape of the peak is configurable. In the example, that parameter is set to 150. When the calculation is performed by clicking the *Execute* button, the mass spectrum that is calculated is displayed as a list of (m/z,i) pairs in the *Results* tab of the dialog window. In that tab widget, the *Display mass spectrum* button make the spectrum available in the *Mass-spectrum* window (see Figure 2.27, "Spectrum created using the peak shaping feature").



The spectrum corresponds to a combination of each individual spectrum obtained by shaping each individual mass peak centroid in the input data list.

FIGURE 2.27: **SPECTRUM CREATED USING THE PEAK SHAPING FEATURE**

Note that if the resolution asked is very high, the resulting shaped mass peaks might appear a bit "hairy". By tweaking the *Bin size* value, the binning of the spectra might improve the situation. Otherwise, using the contextual menu in the mass spectrum graph to apply a Savitzky-Golay filter, described at Section 2.8.1.3, "Effects of the m/z integration parameters" (page 29), will certainly improve things. To achieve such a filtering process, right-click on the mass spectrum trace and choose the according menu item.



Tip

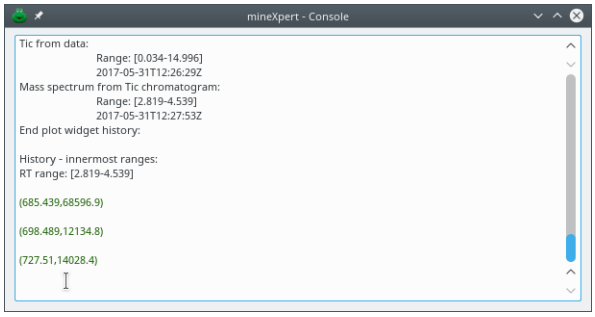
If the peak centroids were not "corrected" for their charge in the previous generation step (as in the case of the isotopic cluster calculation, it is still time to apply this "correction" by setting the charge in the *Charge* spin box widget. If the charge was already accounted for, as described in Section 2.11.1.4, "The IsoSpec results are not shaped mass peaks" (page 45), then leave the charge to *1* and the results will be correct.

2.13 Recording the data mining work

When doing mass analysis work it is often desirable to store the painstakingly manually picked m/z or M_r values for later use. mineXpert provides a number of solutions to record the data mining work.

2.13.1 Feature labelling to the console window

The simplest way to record any graph feature is to point that feature with the mouse and press the L key. That key shortcut prints to the console window the coordinates of the current mouse cursor location. To be able to trace back the graph source of that (x,y) pair, the text is printed in the console using the same color as the graph whence the labelling action came. The console is actually a rich text format editor in which it is possible to edit the text contents so as to copy/paste them in the lab-book or an email to a colleague, for example. This is shown in Figure 2.28, "Recording the peak feature coordinates to the console". The label operation described here does not require any previous integration operation. This is in contrast to the requirements of the mass spectral data analysis recording described below.



The text color identifies the graph being analyzed.

FIGURE 2.28: **RECORDING THE PEAK FEATURE COORDINATES TO THE CONSOLE**



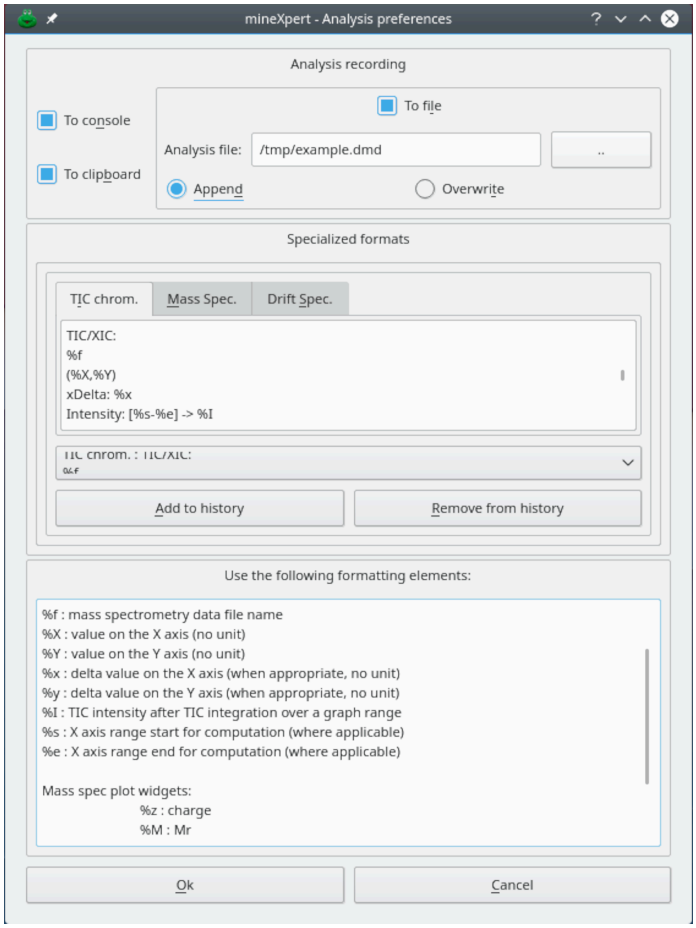
NOTE: SINGLE-GRAPH VS MULTI-GRAPH

The label recording process works without ambiguity when the cursor is located in the single-graph plot widgets. However, when the cursor is located in the multi-graph plot widget (top part of the window displaying TIC chromatograms, mass spectra or drift spectra) then, only the graph(s) currently selected in the *Loaded mass spectrum files* window is(are) concerned by the label operation.

2.13.2 RECORDING THE DATA MINING DISCOVERIES

In order to record the innumerous analysis steps that make a data mining session, the *File* -> Analysis preferences menu might be called to display the window shown in FIGURE 2.29, "SETTING-UP OF THE RECORDING OF THE DATA MINING DISCOVERIES". In that window, the user can select the destination of the data analysis recording system: console, clipboard, file or any combination of the three. When selecting file recording, the user might specify if the recording whould overwrite any preexisting file or, instead, append to that file. Depending on the kind of graph where data mining occurs, the format of the data to be recorded needs to change. Indeed, it would make no sense to record the charge *z* when mining data in the *Drift spectrum* window. This is why the text format of the data export needs to be defined for each one of the three kinds of graphs: TIC/XIC chromatogram, mass spectrum or drift spectrum.

Before delving into the configuration intricacies, let us tell immediately how to trigger the recording of the mining discoveries: using the `Space` bar.



It is possible to configure the recording system to record to either the console, the clipboard, a file (in append mode or in overwrite mode) or any combination thereof. The format of the string is defined using special characters (see text) and might be defined specifically for the three main graphs: TIC/XIC chromatogram, mass spectrum and drift spectrum.

FIGURE 2.29: SETTING-UP OF THE RECORDING OF THE DATA MINING DISCOVERIES

The format used to define the text string to be stored on console and/or in file can contain particular tokens as described below:

- *%f* : mass spectrometry data file name
- *%X* : value on the X axis of the graph (no unit). For a drift spectrum, that would be drift times in milliseconds, for a mass spectrum, that would be m/z values, for a TIC/XIC chromatogram, that would be retention times in minutes;
- *%Y* : value on the Y axis of the graph (no unit). In all the graph plots, that would be intensities in any unit provided by the mass spectrometer (typically, counts);
- *%x* : delta value on the X axis (when appropriate, no unit)

- *%y* : delta value on the Y axis (when appropriate, no unit)
- *%I* : TIC intensity after TIC integration over a graph range
- *%s* : X axis range *start* for computation (where applicable, for example for the TIC integration to a single value;)
- *%e* : X axis range *end* for computation (where applicable);
- *For mass spec plot widgets:*
  - *%z* : charge
  - *%M* : M_r as computed during deconvolution (see Section 2.10, "Mass spectral feature analysis").



Note

It is important to keep in mind that the *%z* and *%M* format strings can only work if the user is actually analyzing a mass spectrum and if the user has *effectively* performed a deconvolution operation that has allowed computing these two values. If the values are not available, the program shows *nan* ("not a number") in the textual output upon hitting the space bar (see below).

In the drift spectrum window, the data recording processes data matching the cursor position at the last Ctrl-single-click. The program tries to define the intensity by looking at the graph ordinate (y axis) matching the nearest abscissa point (x axis) to the last Ctrl-clicked location.

Also, as stated above for the simple labelling of cursor location points (see Section 2.13.1, "Feature labelling to the console window"), the recording of data analysis steps work both in the multi-graph plot widgets (those at the top of the plot windows) and in the single-graph plot widgets (those at the bottom of the windows). When doing data analysis in the top multi-graph plot widget, it is necessary to select the traces to be analyzed in the *Loaded mass spectrum files* window, otherwise no data will be recorded. This is of course not necessary when working in bottom plot widgets because, in that case, there is no ambiguity on what data to record.

Once configured, the format strings might be stored in a drop down box for later use. To that end, click onto the *Add to history* button while having the format text displayed in the text editor and it will be appended to the drop-down list. The list gets stored when the dialog window is closed and will be filled-up again when the program is restarted.

As an example, if the user defined the following format string for a mass spectrum graph:

```
Mass spec. :
mz = (%X, %Y) z = %z
filename = %f
date = 20161021
session = 20161021
mslevel = 1 msion = esi msanal = tof
chrom = DEAE fraction = 25
seq =  pos =    oxlevel = 0 pos =
intensity =
comment =
```

Then, a resulting data mining stanza that would be recorded will look like this:

```
Mass Spec. :
mz = (1051.8, 50863) z = 1
filename = 20161017-rusconi-frac-25-deae-20160712.db
date = 20161021
session = 20161021
mslevel = 1 msion = esi msanal = tof
chrom = DEAE fraction = 25
seq =  pos =    oxlevel = 0 pos =
intensity =
comment =
```

Interestingly, the user can define any kind of format, leaving fields available for later filling-in. This feature is of immense value when the analysis file is used later to fill-in a database for easy storage and interrogation of the mining discoveries. In this case, it would be useful to have the file opened in an editor and at each new stanza edit the *comment* field if something needs to be commented, like the shape/intensity of a mass peak, for example.

Note that the program closes the file each time a new stanza has been written. This makes it possible to edit that file safely in between each stanza record. Remember to force the editor to reload the file from disk after each mining discovery recording.

When the recording involves sending the analysis data to the console, the data are sent to it as text colored the same as the spectrum that was under scrutiny.

When the mouse cursor has been placed at the proper location on the graph (with or without ⇧-click-dragging, depending on the situation), the user hits the space bar and the data analysis stanza is recorded to the selected destination(s): console, clipboard, file.

2.14 Splitting files into smaller chunks

At the moment, this feature is only available for mass data files in the SQLite3 format, as obtained via the mineXpert conversion feature described at Section 2.15, "Converting mzML to SQLite3".

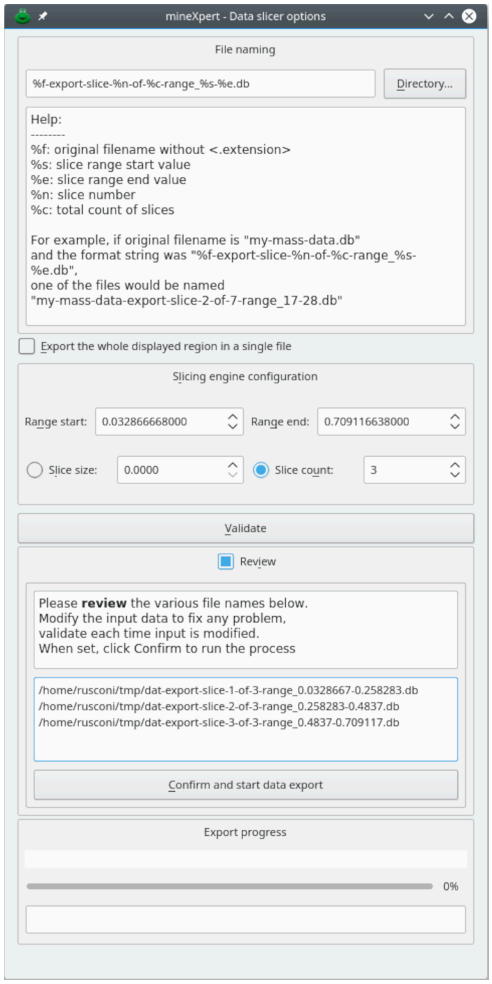
Mass spectrometry data acquisitions performed in line with a chromatography setup generally yields massive data files holding mass data acquired all along a chromatographic development. Depending on the application, the data in the obtained file might not be of interest throughout all the acquisition duration. For example, a size exclusion chromatography might have resolved the molecular species of interest only in a very short retention time range. In this case, it might be useful to extract the data corresponding to that retention time range of interest from the initial very large file and store them in another file that will be sufficiently light to be loaded and analyzed quickly and easily.



Note

The data file slicing feature is called by using the *Export ->* Data contextual menu (right-click the plot widget of interest) either from the *TIC chromatogram* plot widget corresponding to the file to be sliced or from the *Drift spectrum* plot widget of interest.

mineXpert can export data according to various modes, as illustrated in Figure 2.30, "Setting-up of the data export".



It is possible to split very large files into smaller chunk files. In this example, the number of slices to be created is *3*. The program computes automatically the size of the chunks by looking at the *rangeStart* and *rangeEnd* values.

FIGURE 2.30: **SETTING-UP OF THE DATA EXPORT**

The easiest operation is to first zoom in the relevant data in the *TIC chromatogram* plot widget and leave the default *Export the whole displayed region in a single file* checkbox checked. This process basically prunes all the data outside of the currently zoomed-in region. It is possible to export a TIC chromatogram range different than the currently displayed one by entering the *Range start* and *Range end* values in the respective spin boxes. For this, uncheck the *Export the whole displayed region in a single file* checkbox and set the *Slice count* value to `1`.

Another way to operate the slicer is to uncheck the *Export the whole displayed region in a single file* checkbox and define either the number of slices to be generated or their size. Depending on the "slicing" configuration, the program will calculate the missing configuration bits to perform the required action. If the user specifies the number of slices, the size of the slices is automatically calculated. Conversely, if the user specifies the size of the slices, the number of slices is deduced.



Note

Note that the *Range start* and *Range end* values, corresponding to the limits of the currently zoomed-in data range, are always honored when performing the slice number or slice size computations mentioned above.

The *file naming* pattern used for the various data files generated in the process of the data export is governed by the format string displayed at the top of the window. By default, the generated files are located in the same directory as the source data file. If a new directory is to be used, it can be selected by pressing the *Directory...* push button.

Once the configuration is done, press the *Validate* push button to have a preview of the various file names that have been choosen for the various data slices to be written to. If the configuration is correct, click the *Confirm and start data export* button.

Note that the data export feature can only be used with the following two data ranges:

- Retention time ranges: the data export configuration window must be activated from one of the TIC chromatogram plot widgets;
- Ion mobility drift time ranges: the data export must be triggered from one of the Drift spectrum plot widgets;

This is because it does not make sense to split up a given data file into smaller files on the basis of m/z data ranges.

2.15 Converting mzML to SQLite3

The mzML format is very verbose and parsing it causes a notable delay during loading of mass spectrometry files. mineXpert allows one to convert mzML files to a private open file format based on the SQLite3 database software.

Using the SQLite3 format also allows to slice very large data files into smaller files on the basis of user-selected criteria (see Section 2.14, "Splitting files into smaller chunks").

For this feature, mineXpert must be run in a system console window. To show a detailed help, type the following:

```
minexpert --help
```

Use the following parameters (or flags) to perform a data file conversion:

```
minexpert -x -o <db file name> <mzML file name>
```

For example, to convert file `test-file.mzml` into `test-file.db`, the command line would be:

```
minexpert -x -o test-file.db test-file.mzml
```

Alternatively, the `-o` flag can specify a directory, in which case the new file name is crafted from the mzML file and written into that directory. In that case, the extension of the mzML file needs to be either mzML or mzml for the automatic renaming to occur.

Note that *in batch* conversion is possible using this kind of command line: **`minexpert -x -o /tmp /home/<user>/lab/mzml/*.mzml`**

In this case, automatic file renaming happens and the new db files are all stored in the `tmp` directory.

3 JavaScript-based Scripting

When mining data, often, the user finds herself repeating tasks time and time again. Scripting allows to craft text files in which the user writes commands that are then run by the software program one after the other.

mineXpert makes a scripting environment available to the user. That environment is materialized by a window, shown on Figure 3.1, "Scripting console, scripting tab".

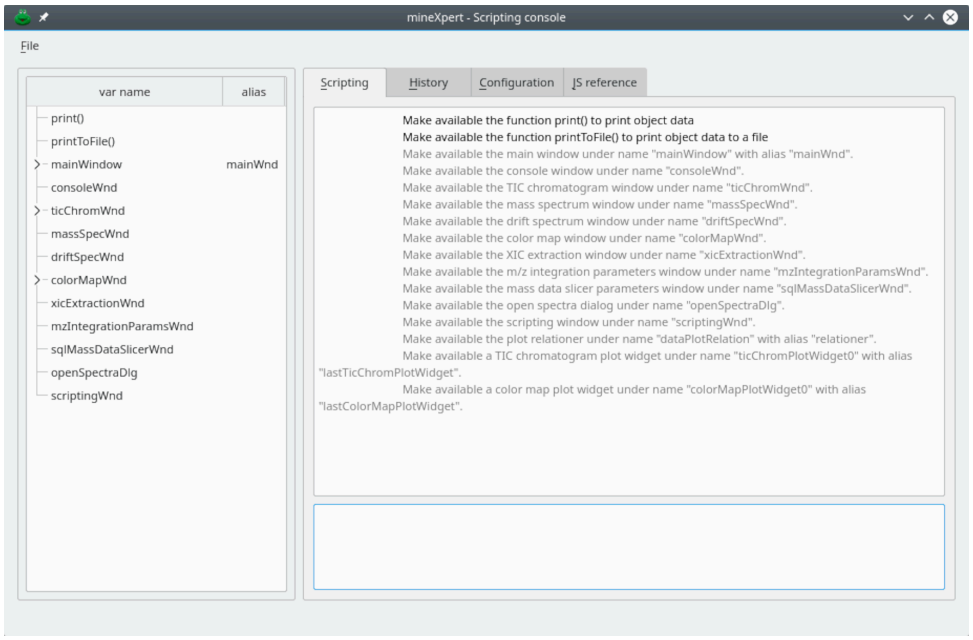


FIGURE 3.1: SCRIPTING CONSOLE, SCRIPTING TAB

The left panel is a tree view listing all the objects or function names that are made available to the scriptor. The right panel has three tabs. The *Scripting* tab has two sections; the upper section is where the scripting engine output is recorded; the lower section (empty in the figure) is the actual console where the user enters the script text.

3.1 Using the scripting console

The scripting console is a multiline text edit widget. The user may enter multiple lines in that widget, using the carriage `Return` key. Once all the lines have been entered to make one or more complete JavaScript statements, the whole set of lines is executed by pressing the `Ctrl`–`Return` key combination.

Upon execution of the script statements, any output is recorded in the upper section. When the output obtained from the engine evaluates to true, it is recorded in green color, if it evaluates to undefined, its color is black, and if it evaluates to error, it is colored in red.

Looking at the script engine output section, it is possible to read indented text, colored in grey. These are comments that mineXpert provides when it runs script statements itself as part of making available to the user globally accessible objects. For example *Make available the console window under name "consoleWnd"* is an explanatory text for the script statement[1] that creates a globally accessible object for the scriptor to use if necessary. The *consoleWnd* object that is created is listed in the tree view in the left panel of the window.

3.2 The available objects list

The left panel of the scripting window lists objects that are made available by mineXpert to the scriptor. The main objects that mineXpert makes available are shown in Figure 3.1, "Scripting console, scripting tab"

- *mainWindow*: the window of the program, where the *File* menu resides;
- *consoleWnd*: the console window where the various feedback messages to the user are displayed;
- *ticChromWnd*: the window where all the total ion current chromatograms are shown;
- *massSpecWnd*: the window displaying all the mass spectra;
- *driftSpecWnd*: the window displaying all the drift spectra;
- *colorMapWnd*: the window showing all the mz=f(dt) color maps;
- *xicExtractionWnd*: the window displaying the XIC integration parameters;
- *mzIntegrationParamsWnd*: the window displaying the m/z integration parameters;
- *sqlMassDataSlicerWnd*: the window displaying the mass data file slicer configuration;
- *openSpectraDlg*: the window listing all the currently opened mass spectrometry data files;
- *scriptingWnd*: the window where all the scripting occurs.

All the items listed above are systematically made available upon launching of the application. For some, they are of no use until a new mass spectrometry file is loaded. Upon opening of a mass spectrometric data file, new objects are created that are made available and listed in this same treeview. For example, upon loading of a mass spectrometry data file, a total ion current chromatogram is computed by mineXpert and displayed in a `TicChromPlotWidget` object that is made available under the *ticChromWnd* object in the tree view. If the data file were for ion mobility mass spectrometry, a mz = f(dt) color map would be computed also and made available as a new plot widget object under the *colorMapWnd* object in the tree view.

When a given object is used for scripting, a shortcut to writing its name in the scripting editor is to simply double-click its item in the treeview and the name will be copied into the scripting editor.

1 That statement was not entered by the user because it is run in the guts of mineXpert.

3.3 The scripting history list

Each time a JavaScript statement (single- or multi-line) is executed, that statement gets stored in the *History* tab that contains a list, as shown in Figure 3.2, "Scripting console, history tab".



FIGURE 3.2: **SCRIPTING CONSOLE, HISTORY TAB**

The items in the history list are reusable by double-clicking on them (or selecting them and hitting `Return`). By doing so, the selected items are copied to the *Scripting* tab, in the script input widget. It is then only required to hit `Ctrl`–`Return` to execute the script line(s).

Note that the history list might become very long, and the user might want to filter the entries of the list. By typing a regular expression in the edit widget below the list and pressing `Return`, the user can select history items matching that expression.

3.4 Exploring the available objects' features

There are three kinds of objects that are available in the scripting environment:

- High-level graphical user interface objects that are semi-automatically made available to the JavaScript environment by the Qt libraries. These objects are not fully scriptable and comprise the highest-level windows that are automatically made available upon running the program as top level branches of the tree view in the left panel of the scripting window;
- Middle-level graphical user interface objects that are created "on demand", like the various plot widgets that are created to display data. These objects have a useful range of features, typically listed in the upper pane of the scripting window by double-clicking their item in the tree view (see below);
- Low-level C++ classes that are mapped to first-class JavaScript objects (with support for the JavaScript `new` constructor operator, for example) such that they expose their full functionality to the scripting environment. The methods of these classes are described in detail in the JavaScript reference section number Chapter 4, JavaScript Reference.

The functional capabilities of the various objects listed in the tree view can be explored by using the following combination: `Ctrl`–*left mouse button* . The list of functional capabilities is printed in the script output widget, as shown in Figure 3.3, "Scripting console, checking object methods".

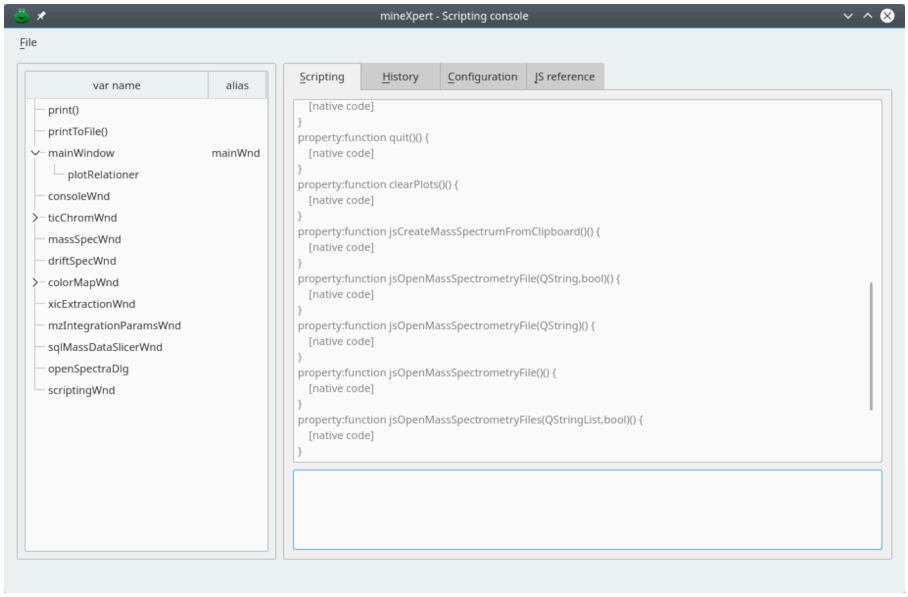


FIGURE 3.3: SCRIPTING CONSOLE, CHECKING OBJECT METHODS

As shown in the figure, the mainWindow has a number of member functions that can be called according to the conventional <object>.<function> notation. For example, entering **`mainWindow.quit()`** in the script input text widget will trigger the **`quit()`** function of the mainWindow object, effectively quitting the application (the mainWindow is actually the main mineXpert window).

Calling **`mainWindow.jsCreateMassSpectrumFromClipboard()`**, for example, will create a new mass spectrum on the basis of data presently in the clipboard, if the data are in the conventional x<sep>y format, with x, the m/z value; y, the intensity and <sep> any character that is neither a cipher nor a dot (decimal separator).

Caution

It is essential to understand that not all functions are of use in the scripting context. This is because the listing of the functional capabilities of the objects also comprise functions that are not aimed at the scripting environment but are for internal Qt or mineXpert use.

Each time a new plot widget is created during a data mining session, the matching JavaScript object is add to the tree view shown in Figure 3.3, "Scripting console, checking object methods". The name matches the kind of plot widget, like *ticChromPlotWidgeto*, *massSpecPlotWidgeto* or *driftSpecPlotWidgeto*. The creation of these plot objects is discussed in the section below.

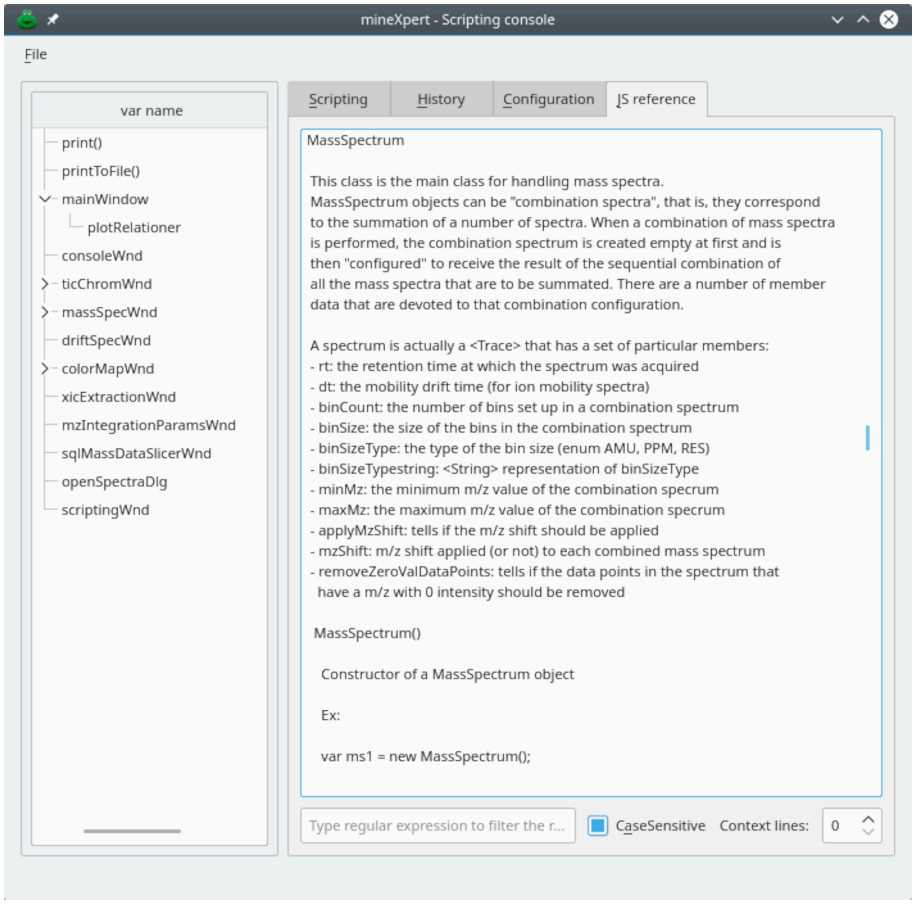
Another way to iterate in the JavaScript properties of the objects made available in the JavaScript environment is by using the following JavaScript code (using *massSpecPlotWidgeto* as an example object):

```
for (var prop in massSpecPlotWidgeto)
print("prop: " + prop)
```

This bit of JavaScript code is particularly useful to explore the various functions that are available for any given C++ object that is "exported" to the JavaScript environment (the GlobalObject, really).

3.5 The JavaScript reference

The JavaScript reference material related to all the classes available in the JavaScript environment is located in the form of formatted plain text in the *JS reference* tab of the scripting window (Figure 3.4, "Scripting console, JavaScript reference tab").



The reference tab displays all the JavaScript-related documentation that is extracted automatically from the source code. This documentation describes all the JavaScript classes available to the scripting environment. The description details the members and the methods of the class.

FIGURE 3.4: **SCRIPTING CONSOLE, JAVASCRIPT REFERENCE TAB**

The line edit widget at the bottom of the tab provides an easy means to filter the documentation such that only the matching lines are displayed in the text widget. To filter the documentation, enter a regular expression (or a simple text) and press the `Return` key. To displays the original contents anew, remove the regular expression and press the `Return` key.



Tip

Sometimes, some context around the matching lines might be desirable. The number of lines upstream and downstream of the matching lines that need to be output can be defined with the *Context lines* spin box. A value of 0 indicates that only the matching lines are output. A value of 5 indicates that five lines above the matching line and below that matching line are output.

That JavaScript reference is also available in identical form in the JavaScript reference chapter Chapter 4, JavaScript Reference.

3.6 Creation of plot objects as a result of data mining

When new plots are created as a result of the user making data mining operations, new JavaScript objects are created. The JavaScript objects mirror the C++ objects that live in the guts of mineXpert. The convention is to name the plot objects made available to the scripting environment using an index, like in *ticChromPlotWidget0*, *ticChromPlotWidget1*... For example, when a new mass spectrometry data file is loaded, mineXpert automatically computes the TIC chromatogram and makes it available to the scripting environment under object name *ticChromPlotWidget0*. That name is listed as a child of the *ticChromWnd* object in the tree view of the available objects. This object will remain available as long as the plot lives in the *ticChromWnd*. Note that simulateanously, the *lastTicChromPlotWidget* alias is provided to the scriptor to capture that newly created object. The scriptor willing to store the name of a newly created plot can thus make an assignment like in the following code:

```
mainWindow.openMassSpectrometryFile("massSpecFile1.mzml");

// The last TIC chromatogram plot is made available under the
// temporary name lastTicChromPlotWidget, take advantage to store it
// under a meaningful name for later use

msFile1TicPlot = lastTicChromPlotWidget

// Open a new file so that comparisons can be made
// against the previous one

mainWindow.openMassSpectrometryFile("massSpecFile2.mzml");

// The lastTicChromPlotWidget is made available, take advantage to store
// it under a meaningful name for later use

msFile2TicPlot = lastTicChromPlotWidget

// Now it is possible to reference the two TIC chromatograms
// with their new names:

msFile1TicPlot.integrateToMz(4.5, 7);
msFile2TicPlot.integrateToMz(4.5, 7);
```

3.7 JavaScript-oriented classes

At the time of writing, there are a number of C++ classes that have a full JavaScript counterpart: `DataPoint`, `Trace`, `MassSpectrum`, `MzIntegrationParams`, `SavGolFilter`, for example. In the following sections, a number of use cases are detailed to give the reader a flavor of what is scriptable in mineXpert. Note that the classes are detailed in the scripting reference section Chapter 4, JavaScript Reference.

3.7.1 Creating a `DataPoint` object

The C++ class `DataPoint` has two member data: `m_key` and `m_val`. Both are high precision double values. The JavaScript counter part is also named `DataPoint` and can be initialized in two ways:

```
// create a new DataPoint object, initialized using double values
var dp1 = new DataPoint(123.321,456.654);

// check that this worked fine:
dp1.key;
// --> returns 123.321 (same as dp1["key"])
dp1.val;
// --> returns 456.654 (same as dp1["val"])

dp1.isValid();
// --> returns true

// create a new DataPoint object, initialized using an xy-formatted string
var dp2 = new DataPoint("147.741 258.852");

// check that this worked fine:
dp2.key;
// --> returns 147.741
dp2.val;
// --> returns 258.852
```

```
dp2.isValid();
// --> returns true

// create an empty DataPoint object
var dp3 = new DataPoint()

// an empty DataPoint object is invalid
dp3.isValid();
// --> returns false

// a DataPoint can be initialized a posteriori
dp1.initialize(159.951, 357.753)

// or using a string
dp2.initialize("258.852 741.147");
```

The `DataPoint` class is the main building block `Trace` objects are made of. A `Trace` object is nothing but a list of `DataPoint` instances.

3.7.2 Creating `Trace` objects

A JavaScript `Trace` object is nothing but an array of `DataPoint` objects. One useful application of scripting is to perform a variety of calculations on a given `Trace` or on a set of `Trace`_s.

A JavaScript `Trace` object can be created and initialized in a number of ways.

3.7.2.1 Creating an empty `Trace` and initializing it

```
// create a new Trace using an xy-formatted string
var t1 = new Trace("This is the title of the trace object");
t1.title
// --> returns “This is the title of the trace object”
```

```
t1.initialize("123.321 456.654\n789.987 147.741\n258.852 369.963");
// --> returns 3 // the number of created DataPoint objects

// check that the trace has effectively 3 items:
t1.length
// --> returns 3

// check that the initialization was ok
print("(" + t1[0 ].key + "," + t1[0].val + ")")
// --> returns (123.321,456.654)

// create a new Trace using two arrays of numbers (key list and value list)
var t2 = new Trace();
t2.initialize([123.321,456.654,789.987],[147.741,258.852,369.963])
// --> returns 3 (the number of created DataPoint objects)

// check actual length
t2.length
// --> returns 3

// check that the initialization went fine
t2[0].key
// --> returns 123.321
t2[0].val
// --> returns 147.741

// get the key and value lists
t2.keyArray()
// --> returns 123.321,456.654,789.987
t2.valArray()
// --> returns 147.741,258.852,369.963

// once a Trace has been initialized, it can undergo interesting actions:

// calculate the sum of all the values:
t2.valSum()
```

```
// --> returns 776.556

// a Trace object can combine into itself another Trace object
t2.combine(t1)

// now check if the t2 values have increased accordingly
t2.valArray()
// --> returns 604.395,369.963,258.852,517.704 // correct combination

// now subtract t2 from t2:
t2.subtract(t2)

// now check that subtraction worked fine
t2.valArray()
// --> returns 0,0,0

// it is possible to initialize a trace with limitation to a given key range
var t3 = new Trace()
t3.initializeRange([123.321,456.654,789.987], [147.741,258.852,369.963], 200, 500);

// there should be only one DataPoint in t3
t3.length
// --> returns 1
t3.keyArray();
// --> returns 456.654
```

3.7.2.2 Creating a `Trace` object starting from a plot

A `Trace` object can be easily created by using data already obtained from using mineXpert. For example, a `Trace` object can be made so that it reproduces a plot dispalyed in the graphical user interface. The process is shown below:

```
// load mobility data file
```

```
mainWnd.openMassSpectrometryFile(``mobility.mzml'')

// integrate over the full TIC chromatogram range
// creates the  massSpecPlotWidget0 object

ticChromPlotWidget0.integrateToMz()

// create a new Trace JavaScript object
var t0 = new Trace()

// now initialize t0 with the plot data of ticChromPlotWidget0
t0.initialize(massSpecPlotWidget0.keys(),massSpecPlotWidget0.values());

// check that the initialization actually worked
t0.length // should return the number of DataPoint objects in that Trace

// load a blank data file that acts as a baseline data file
mainWnd.openMassSpectrometryFile(``mobility-blank.mzml'')

// integrate over the full TIC chromatogram range
// creates the massSpecPlotWidget1 object

ticChromPlotWidget1.integrateToMz()

// create Trace JavaScript object
var t1 = new Trace()

// now initialize t1 with the plot data of ticChromPlotWidget1
t1.initialize(massSpecPlotWidget1.keys(),massSpecPlotWidget1.values());

// check that the initialization actually worked
t1.length // should return the number of DataPoint objects in that Trace

// finally perform some useful task: subtract noise from t0
t0.subtract(trace1)

// Trace object t0 now has the baseline removed.
```

3.7.3 Exporting a `Trace` object to a file

It is possible at each moment to export a given `Trace` object (or plot) to a file. This is possible either at the JavaScript `Trace` object level:

```
mainWnd.openMassSpectrometryFile("mobility.mzml")
ticChromPlotWidgeto.integrateToMz()
var to = new Trace()
to.initialize(massSpecPlotWidgeto.keys(),massSpecPlotWidgeto.values());
to.exportToFile("/home/rusconi/demo.xy");
```

Or, at the plot level:

```
mainWnd.openMassSpectrometryFile("mobility.mzml")
ticChromPlotWidgeto.integrateToMz()

// note how the export function name has “Plot” in it.
massSpecPlotWidgeto.exportPlotToFile("/home/rusconi/demobis.xy");
```

At this point both files demo.xy and demobis.xy are bit-by-bit identical.

3.7.4 Plotting a JavaScript `Trace` object to a plot widget window

It is necessary for the data miner to be able to look at her data graphically. Nobody wants to scrutinize data in ASCII files. Anytime the scriptor has a `Trace` object, she can plot it in the window that makes sense for the data. For example, if a plot derives from mass spectral data, then logically, the new plot should derive for these mass spectral data, as exemplified below:

```
mainWnd.openMassSpectrometryFile("mobility.mzml");
ticChromPlotWidget0.integrateToMz();

var t1 = new Trace();
t1.initialize(massSpecPlotWidget0.keys(), massSpecPlotWidget0.values());

// and now plot that Trace object as a descendant of massSpecPlotWidget0.
// if massSpecPlotWidget0 is removed, then the descendant will be removed also
massSpecPlotWidget0.newPlot(t1);
```

Displaying a new plot using the **`<existingPlot Object>.newPlot(<Trace>)`** call occurs in the window that matches the type of <existingPlot>. If <existingPlot> is a mass spectrum, then the descendant plot will be displayed in the Mass~Spectrum~Window.

Any of the plot widget windows can receive a new plot according to the mechanics described above.

3.7.5 Printing a JavaScript `Trace` object to the console

It is possible to have a look at the numerical data of a `Trace` object like so:

```
mainWnd.openMassSpectrometryFile("mobility.mzml");
ticChromPlotWidget0.integrateToMz();

var t1 = new Trace();
t1.initialize(massSpecPlotWidget0.keys(), massSpecPlotWidget0.values());
```

```
// print the data to the console
t1.asText()

// outputs a series of xy-formatted lines like so:
// 2504.3185084749 1262.0000000000
// 2504.3503266026 1277.0000000000
// ...

// doing this is also possible at the plot widget level
// note the function name change
massSpecPlotWidget0.asXyText()
```

3.7.6 Running a JavaScript script from file

It is possible to run a script directly from file by using either the following flag to the mineXpert command line:

```
minexpert -j <scriptFile>
```

or by using the *File* -> Run a JavaScript from file menu of the scripting window. In that case the script is immediately run and the results are printed in the upper part of the *Scripting* tab of the scripting window.

DataPoint

The DataPoint class describes a data point. Its members are:
- key: the X-axis coordinate
- val: the Y-axis coordinate

DataPoint()

Constructor of a DataPoint object

Ex:
var dp = new DataPoint;
dp.key; // --> NaN
dp.val; // --> NaN
dp.isValid(); // --> false

DataPoint(xyLine)

Constructor of a DataPoint object initialized using the xy-formatted line.

xyLine: <String> holding the key,val pair in the format key<sep>val, with
<sep> being any character not a cipher nor a decimal point.

Ex:
var dp = new DataPoint("123.321,456.654");
dp.key; // --> 123.321
dp.val; // --> 456.654
dp.isValid(); // --> true

DataPoint(dataPoint)

Constructor of a DataPoint object initialized with another DataPoint object

dataPoint: <DataPoint> object used to initialize this object.

Ex:
var dp1 = new DataPoint(dp);

DataPoint(key, val)

Constructor of a DataPoint object initialized with the numerical parameters.

key: key (X-value)
val: value (Y-value)

Ex:
var dp = DataPoint(123.321, 456.654);

<DataPoint>.key

Return the key <Number> property.

Ex:
var dp = new DataPoint(123.321, 456.654);
var mz = dp.key;
mz // --> 123.321

<DataPoint>.val

Return the val <Number> property.

Ex:
var dp = new DataPoint(123.321, 456.654);
var i = dp.val;
i // --> 456.654

DataPoint.initialize(other)

Initialize this DataPoint object with a <DataPoint> object

other: <DataPoint> object

Ex:

```
var newDp = new DataPoint(123.321,456.654);
var otherDp = new DataPoint();
otherDp.initialize(newDp);
otherDp.key; // --> 123.321
otherDp.val; // --> 456.654
otherDp.isValid(); // --> true
```

DataPoint.initialize(key, val)

Initialize this DataPoint object with two <Number> entities

key: <Number> holding the key of this DataPoint
val: <Number> holding the value of this DataPoint

Ex:

```
var dp = new DataPoint();
dp.initialize(123.321,456.654);
dp.key; // --> 123.321
dp.val; // --> 456.654
dp.isValid(); // --> true
```

DataPoint.isValid()

Return if this <DataPoint> object is valid.

Ex:

```
var dp = new DataPoint();
dp.initialize(123.321,456.654);
dp.key; // --> 123.321
dp.val; // --> 456.654
dp.isValid(); // --> true
```

SavGolParams

This class handles the parameters of a Savitzy-Golay filter.

A SavGolParams object contains the following member data:
- nl: <Number> of points on the left of the filtered point;
- nR: <Number of points on the right of the filtered point;
- m: <Number> (order) of the polynomial to use in the regression analysis leading to the Savitzky-Golay coefficients (typically between 2 and 6);
- lD: <Number> specifying the order of the derivative to extract from the Savitzky-Golay smoothing algorithm (for regular smoothing, use 0)
- convolveWithNr <Boolean> Set to false for best results

SavGolParams()

Constructor of a SavGolParams object

Ex:

```
var sgp1 = new SavGolParams();
sgp1.nL; // --> default 15
sgp1.nR; // --> default 15
sgp1.m; // --> default 4
sgp1.lD; // --> default 0
sgp1.convolveWithNr; // --> default false
```

SavGolParams(savGolParams)

Constructor of a SavGolParams object using a <SavGolParams> object as template

Ex:

```
var sgp1 = new SavGolParams();
sgp1.nL = 10;
sgp1.nR = 10;
sgp1.m; // --> 4
sgp1.lD; // --> 0
sgp1.convolveWithNr; // --> false

var sgp2 = new SavGolParams(sgp1);

sgp2.nL; // --> 10
sgp2.nR; // --> 10
sgp2.m; // --> 4
sgp2.lD; // --> 0
sgp2.convolveWithNr; // --> false
```

SavGolParams(savGolFilter)

Constructor of a SavGolParams object using a <SavGolFilter> object as template

Ex:

```
var sgf1 = new SavGolFilter();
sgf1.nL = 10;
sgf1.nR = 10;
sgf1.m; // --> 4
sgf1.lD; // --> 0
sgf1.convolveWithNr; // --> false

var sgp2 = new SavGolParams(sgf1);
```

```
sgp2.nL; // --> 10
sgp2.nR; // --> 10
sgp2.m; // --> 4
sgp2.lD; // --> 0
sgp2.convolveWithNr; // --> false
```

```
SavGolParams(nL, nR, m, lD, convolveWithNr)

  Constructor of a SavGolParams object using the various configurations
  <Number> and <Boolean> parameters.

  Ex:

  var sgf1 = new SavGolParams(10, 10, 6, 0, true);
  sgf1.nL; // --> 10
  sgf1.nR; // --> 10;
  sgf1.m; // --> 6
  sgf1.lD; // --> 0
  sgf1.convolveWithNr; // --> true
```

```
SavGolFilter

 This class handles the parameterization of the Savitzky-Golay
 filter. This filter is an efficient way to remove noise from noisy mass
 spectra and can be automatically applied after any mass spectrum
 combination processing.

 A SavGolFilter object contains the following member data as members of a
 <SavGolParams> object (see documentation for SavGolParams):
 - nl: <Number> of points on the left of the filtered point;
 - nR: <Number of points on the right of the filtered point;
 - m:  <Number> (order) of the polynomial to use in the regression analysis
   leading to the Savitzky-Golay coefficients (typically between 2 and 6);
 - lD: <Number> specifying the order of the derivative to extract from the
   Savitzky-Golay smoothing algorithm (for regular smoothing, use 0)
```

- convolveWithNr <Boolean> Set to false for best results

SavGolFilter()

Constructor of a SavGolFilter object

Ex:
var sgf1 = new SavGolFilter();
sgf1.nL; // --> 15
sgf1.nR; // --> 15
sgf1.m; // --> 4
sgf1.lD; // --> 0
sgf1.convolveWithNr; // --> false

SavGolFilter(savGolParams)

Constructor of a SavGolFilter object initialized using a <SavGolParams>
object

Ex:

var sgp1 = new SavGolParams();

sgp1.nL = 10; --> 10
sgp1.nR = 10; --> 10
sgp1.lD = 0; --> 0
sgp1.m = 5; --> 5

var sgf1 = new SavGolFilter(sgp1);
sgf1.asText(); --> output is:

Savitzy-Golay filter parameters:
nL: 10 ; nR: 10 ; m: 5 ; lD: 0 ; convolveWithNr : false
savGolParams: <SavGolParams> object to be used to initialize this object's SavGolParams

SavGolFilter(nL, nR, m, lD, convolveWithNr)

Constructor of a SavGolFilter object initialized using parameters that
initialize the SavGolParams member object

nL: number of data points on the left of the point being filtered (default * 15)

nR: number of data points on the right of the point being filtered (default * 15)

m: order of the polynomial to use in the regression analysis
leading to the Savitzky-Golay coefficients (typicall [2-6], default 4)

lD: order of the derivative to extract from the Savitzky-Golay
smoothing algorithm (for regular smoothing, use 0, the default);

Ex:

var sgf1 = new SavGolFilter(10, 10, 6, 0, true);

sgf1.asText(); --> output is:

Savitzy-Golay filter parameters:
nL: 10 ; nR: 10 ; m: 6 ; lD: 0 ; convolveWithNr : true

convolveWithNr: set to false for best results (default false)

SavGolFilter(savGolFilter)

Constructor of a SavGolFilter object

Ex:

var sgf1 = new SavGolFilter();
sgf1.asText(); --> output is:

```
  Savitzy-Golay filter parameters:
  nL: 15 ; nR: 15 ; m: 4 ; lD: 0 ; convolveWithNr : false*

  sgf1.nL = 10;
  sgf1.nR = 10;
  sgf1.m= 6;

  var sgf2 = new SavGolFilter(sgf1);
  sgf2.asText(); --> output is:

  Savitzy-Golay filter parameters:
  nL: 10 ; nR: 10 ; m: 6 ; lD: 0 ; convolveWithNr : false

  savGolFilter: <SavGolFilter> object to be used to initialize this object


SavGolFilter.initialize(other)

  Initialize this SavGolFilter object with another
  <SavGolFilter> object

  other: <SavGolFilter> object used to initialize this object


SavGolFilter.initialize(nL, nR, m, lD, convolveWithNr)

  Initialize this SavGolFilter object with parameters to configure the
  SavGolParams member object

  nL: number of data points on the left of the point being filtered
  nR: number of data points on the right of the point being filtered
  m: order of the polynomial to use in the regression analysis
  leading to the Savitzky-Golay coefficients (typicall [2-6])
  lD: order of the derivative to extract from the Savitzky-Golay
  smoothing algorithm (for regular smoothing, use 0);
  convolveWithNr: set to false for best results
```

SavGolFilter.asText()

Return a <String> object containing a textual representation of all the member data of this SavGolFilter object.

MzIntegrationParams

The MzIntegrationParams class provides an interface to all the parameters that are required to configure all the integrations of mass spectral data that combine into a MassSpectrum object.

When combining multiple mass spectra into a single one, the process might involve the creation of bins according to specific parameters measured on the whole set of the spectra to be combined into the combination spectrum. These parameters are configured using a MzIntegrationParams object. Its members are:

-binningType: <Number> type of the binning. The type of binning specifies if binning is arbitrary or data-based (refer to msXpS::BinningType for details). It might also specify that no binning is required.

-binSize: <Number> size of the bins. A typical bin size would be 0.005 for a high resolution mass spectrometer with unit m/z.

-binSizeType: <Number> type of the bin size. The bin size type specifies if the binSize value is to be considered a m/z value or a atomic mass unit value or a resolution value or a part-per-million value (refer to msXpS::MassToleranceType for details)

-binSizeTypeString: <String> type of the bin size. This member datum hold a textual representation of the binSizeType member datum, like "PPM" or "RES" or "AMU" or "MZ".

-applyMzShift: <Boolean> value that tells if the m/z shift correction should be applied.

-removeZeroValDataPoints: <Boolean> that tells if the 0-val m/z data points should be removed from the combination spectrum once all the source mass spectra have been effectively combined into the combination spectrum.

- applySavGolFilter: <Boolean> that tells if a Savitzky-Golay filtering is to be applied to the combination spectrum after all the source mass spectra have been combined to the combination spectrum.

- savGolParams: <SavGolParams> object holding the configuration of a Savitzky-Golay filtering process.

MzIntegrationParams()

Construct a MzIntegrationParams instance with default values.

Ex:

```
var mzip1 = new MzIntegrationParams();

mzip1.binningType; // --> 0
mzip1.binningTypeString; // --> NONE
```

MzIntegrationParams(binningType, binSize, binSizeType, applyMzShift, removeZeroValDataPoints)

Construct a MzIntegrationParams instance using specified values.

binningType: <Number> type of the binning (refer to msXpS::BinningType for details)
binSize: <Number> size of the bins
binSizeType: <Number> type of the bin size (refer to msXpS::MassToleranceType for details)
applyMzShift: <Boolean> tells if the m/z shift correction should be applied
removeZeroValDataPoints: <Boolean> tells if the 0-val m/z data points should be removed

Ex:

```
var mzip1 = new MzIntegrationParams(1, 0.0055, 2, false, true);

mzip1.binningTypeString; // --> DATA_BASED
mzip1.binSize; // --> 0.0055
mzip1.binSizeType; // --> 2
mzip1.binSizeTypeString; // --> MZ
mzip1.applyMzShift; // --> false
mzip1.removeZeroValDataPoints; // --> true
```

MzIntegrationParams(mzIntegrationParams)

Construct a MzIntegrationParams object using a <MzIntegrationParams> template object.

Ex:

```
var mzip1 = new MzIntegrationParams(1, 0.0055, 2, false, true);

mzip1.binningTypeString; // --> DATA_BASED
mzip1.binSize; // --> 0.0055
mzip1.binSizeType; // --> 2
mzip1.binSizeTypeString; // --> MZ
mzip1.applyMzShift; // --> false
mzip1.removeZeroValDataPoints; // --> true

var mzip2 = new MzIntegrationParams(mzip1);

mzip2.binningTypeString; // --> DATA_BASED
mzip2.binSize; // --> 0.0055
mzip2.binSizeType; // --> 2
mzip2.binSizeTypeString; // --> MZ
mzip2.applyMzShift; // --> false
mzip2.removeZeroValDataPoints; // --> true
```

mzIntegrationParams: <MzIntegrationParams> to be used to initialize this object

MzIntegrationParams(savGolParams)

Construct a MzIntegrationParams object using a SavGolParams template object.

Ex:

```
var sgp1 = new SavGolParams(10, 10, 6, 0, true);
var mzip1 = new MzIntegrationParams(sgp1);

mzip1.savGolParams.nL; // --> 10
mzip1.savGolParams.nR; // --> 10
mzip1.savGolParams.m; // --> 6
mzip1.savGolParams.lD; // --> 0
```

savGolParams: <SavGolParams> object to be used to initialize this object's SavGolParams

<MzintegrationParams>.binningType

Return the binningType property of the object as a <Number>

Ex:

```
var mzip1 = new MzIntegrationParams();

mzip1.binningType; // --> 0
```

Return the binning type property as a <Number>.

<MzintegrationParams>.binningTypeString

Return the binningType property of the object as a <String>

Ex:

var mzip1 = new MzIntegrationParams();

mzip1.binningTypeString; // --> NONE

Return the binning type property as a <String>.

<MzIntegrationParams>.binSize

Return the bin size property of the object.

Ex:

var mzip1 = new MzIntegrationParams();

mzip1.binSize; // --> NaN (uninitialized value)

Return the bin size <Number> property (size of the bins in the mass spectrum).

<MzintegrationParams>.binSizeType

Return the binSizeType property of the object as a <Number>

Ex:

var mzip1 = new MzIntegrationParams();

mzip1.binSizeType; // --> 0

Return the bin size type property as a <Number>.

<MzintegrationParams>.binSizeTypeString

Return the binSizeType property of the object as a <String>

Ex:

```
var mzip1 = new MzIntegrationParams();

mzip1.binSizeTypeString; // --> NONE
```

Return the bin size type property as a <String>.

<MzIntegrationParams>.applyMzShift

Ex:

```
var mzip1 = new MzIntegrationParams();

mzip1.applyMzShift; // --> false
mzip1.applyMzShift = true;
mzip1.applyMzShift; // --> true
```

Return the <Boolean> value telling if the m/z shift should be applied.

<MzIntegrationParams>.removeZeroValDataPoints

Ex:

```
var mzip1 = new MzIntegrationParams();

mzip1.applyMzShift; // --> false
```

Return the <Boolean> value telling if the <DataPoint> objects having m/z keys of a 0-intensity value should be removed from the <MassSpectrum> object.

<MzIntegrationParams>.applySavGolFilter

Ex:

```
var mzip1 = new MzIntegrationParams();

mzip1.applySavGolFilter // --> false
```

Return the <Boolean> value telling if the Savitzky-Golay filter should be applied to the <MassSpectrum> object.

<MzIntegrationParams>.savGolParams

Ex:

```
var mzip1 = new MzIntegrationParams();

var sgp1 = mzip1.savGolParams;

sgp1.nL; // --> 15
sgp1.nR; // --> 15
sgp1.m; // --> 4
sgp1.lD; // --> 0
sgp1.convolveWithNr;  // --> false
```

Return the <SavGolParams> object from this <MzIntegrationParams> object.

MzIntegrationParams.binSize = <Number>

Set the bin size.

Ex:

```
var mzip1 = new MzIntegrationParams();
mzip1.binSize; // --> NaN (uninitialized)
mzip1.binSize = 0.0055;
mzip1.binSize; // --> 0.0055
```

MzIntegrationParams.binSizeType = <Number>

Set the bin size type as a <Number>

Ex:

```
var mzip1 = new MzIntegrationParams();

mzip1.binSizeType; // --> 0
mzip1.binSizeTypeString; // --> NONE
mzip1.binSizeType=1;
mzip1.binSizeType; // --> 1
mzip1.binSizeTypeString; // --> PPM
```

MzIntegrationParams.binSizeTypeString = <String>

Set the bin size type as a <String>

Ex:

```
var mzip1 = new MzIntegrationParams();
mzip1.binSizeTypeString; // --> NONE
mzip1.binSizeTypeString = "RES";
mzip1.binSizeTypeString; // --> RES
mzip1.binSizeType; // --> 4
```

MzIntegrationParams.applyMzShift = <Boolean>

Tell if the m/z shift between spectra must be applied

Ex:

```
var mzip1 = new MzIntegrationParams();

mzip1.applyMzShift; // --> false
mzip1.applyMzShift = true;
mzip1.applyMzShift; // --> true
```

MzIntegrationParams.removeZeroValDataPoints = <Boolean>

Tell if the DataPoint objects having a value of 0 need to be removed from the spectrum.

Ex:

```
var mzip1 = new MzIntegrationParams();

mzip1.applyMzShift; // --> false
mzip1.applyMzShift = true;
mzip1.applyMzShift; // --> true
```

MzIntegrationParams.removeZeroValDataPoints = <Boolean>

Tell if the Savitzky-Golay filter needs to be applied to the combination spectrum.

Ex:

```
var mzip1 = new MzIntegrationParams();

mzip1.applySavGolFilter; // --> false
mzip1.applySavGolFilter = true;
mzip1.applySavGolFilter; // --> true;
```

MzIntegrationParams.savGolParams = <SavGolParams>

Initialize the member <SavGolParams> object using the parameter as template.

Ex:

```
var sgp1 = new SavGolParams(10, 10, 6, 0, true);
var mzip1 = new MzIntegrationParams();

mzip1.savGolParams.nL; // --> 15
mzip1.savGolParams.nR; // --> 15
mzip1.savGolParams = sgp1;
mzip1.savGolParams.nL; // --> 10
mzip1.savGolParams.nR; // --> 10
```

MzIntegrationParams.initialize(other)

Initialize this MzIntegrationParams object with a <SavGolParams> object

Ex:

```
var sgp1 = new SavGolParams(10, 10, 6, 0, true);
var mzip1 = new MzIntegrationParams();

mzip1.savGolParams.nL; // --> 15
mzip1.savGolParams.nR; // --> 15
mzip1.savGolParams.m; // --> 4
mzip1.savGolParams.lD; // --> 0

mzip1.initialize(sgp1);

mzip1.savGolParams.nL; // --> 10
mzip1.savGolParams.nR; // --> 10
mzip1.savGolParams.m; // --> 6
```

```
mzip1.savGolParams.lD; // --> 0

other: <SavGolParams> object


MzIntegrationParams.initialize(other)

Initialize this MzIntegrationParams object with a
<MzIntegrationParams> object

Ex:

var sgp1 = new SavGolParams(10, 10, 6, 0, true);
var mzip1 = new MzIntegrationParams(sgp1);
var mzip2 = new MzIntegrationParams(mzip1);

mzip2.savGolParams.nL; // --> 10
mzip2.savGolParams.nR; // --> 10
mzip2.savGolParams.m; // --> 6
mzip2.savGolParams.lD; // --> 0

other: <MzIntegrationParams> object


MzIntegrationParams.initialize(nL, nR, m, lD, convolveWithNr)

Initialize this MzIntegrationParams object with the individual <SavGolParams> parameters

nL: number of data points on the left of the point being filtered

nR: number of data points on the right of the point being filtered

m: order of the polynomial to use in the regression analysis
leading to the Savitzky-Golay coefficients (typicall [2-6])

lD: order of the derivative to extract from the Savitzky-Golay
smoothing algorithm (for regular smoothing, use 0);
```

convolveWithNr: set to false for best results

Ex:

```
var mzip1 = new MzIntegrationParams();
mzip1.initialize(10, 10, 6, 0, false);

mzip1.savGolParams.nL; // --> 10
mzip1.savGolParams.nR; // --> 10
mzip1.savGolParams.m; // --> 6
mzip1.savGolParams.lD; // --> 0
```

MzIntegrationParams.asText()

Return a <String> object containing a textual representation of all the member data of this MzIntegrationParams object.

MzIntegrationParams.asText()

Return a textual representation of this <MzIntegrationParams> object.

Ex:

```
var mzip1 = new MzIntegrationParams();
mzip1.binningType = 2;
mzip1.binningTypeString; // --> ARBITRARY
mzip1.binSize = 0.0055;
mzip1.binSizeTypeString = "MZ";
mzip1.applyMzShift = true;
mzip1.applySavGolFilter= true;
mzip1.asText(); // returns this:

m/z integration parameters:
Binning type: ARBITRARY
```

Bin nominal size: 0.005500
Bin size type: MZ
Apply m/z shift: true
Remove 0-val data points: false
Savitzky-Golay parameters
nL = 15 ; nR = 15 ; m = 4 ; lD = 0 ; convolveWithNr = false

Trace

The Trace class is fundamentally an <Array> of DataPoint objects.

Trace()

Constructor of a Trace object

Ex:
var t = new Trace;
t.length; // --> 0

Trace(title)

Constructor of a Trace object

title: <String> containing the title of the Trace object

Ex:
var t = new Trace("Trace title");
t.title; // --> Trace title

Trace(other)

Constructor of a Trace object

other: <Trace> object to be used to initialize this object

Ex:

```
var t2 = new Trace();
t2.initialize("123.3321 456.654\n147.741 258.852\n");
t2[0].key; // --> 123.3321
t2[0].val; // --> 456.654
t2[1].key; // --> 147.741
t2[1].val; // --> 258.852
var t3 = new Trace(t2);
t3[0].key; // --> 123.3321
t3[0].val; // --> 456.654
t3[1].key; // --> 147.741
t3[1].val; // --> 258.852
```

Trace[index]

Return the <DataPoint> object at index.

Ex:

```
var t1 = new Trace();
t1.initialize([123.321,456.654,789.987],[147.741, 258.852, 369.963]);
t1.asText();

123.3210000000 147.7410000000
456.6540000000 258.8520000000
789.9870000000 369.9630000000

var dpo = t1[0];
dpo.key; // --> 123.321
dpo.val; // --> 147.741

t1[0].key; // --> 123.321
t1[0].val; // --> 147.741
```

Trace.initialize(xyFormatString)

Initialize this Trace object with a <String> object representing the
mass data in the xy text format:

"<key><sep><val>\n<key><sep><val>\n...".

xyFormatString: <String> object representing the <DataPoint> data points of
the <Trace> in the format <key> <value>, with one such key-value pair per
line. The separator between <key> and <value> might be anything
non-numerical and not a dot (decimal separator).

Ex:

```
var t2 = new Trace();
t2.initialize("123.3321 456.654\n147.741 258.852\n");
t2[0].key; // --> 123.3321
t2[0].val; // --> 456.654
t2[1].key; // --> 147.741
t2[1].val; // --> 258.852
```

Trace.initialize(other)

Initialize this Trace object with a <Trace> object

other: <Trace> object

Ex:

```
var t1 = new Trace();
t1.initialize("123.3321 456.654\n147.741 258.852\n");
var t2 = new Trace();
t2.initialize(t1);
t2[0].key; // --> 123.3321
t2[0].val; // --> 456.654
t2[1].key; // --> 147.741
t2[1].val; // --> 258.852
```

Trace.initialize(keyArray, valArray)

Initialize this Trace object with two numerical <Array> entities

keyArray: the <Array> object containing all the keys of the <Trace>
valArray: the <Array> object containing all the values of the <Trace>

Ex:
```
var t1 = new Trace;
t1.initialize([123.321,456.654,789.987],[147.741, 258.852, 369.963]);
t1.length; // --> 3
var dpo = t1[0];
dpo.key; // --> 123.321
dpo.val; // --> 147.741
```

Trace.initialize(keyArray, valArray, keyStart, keyEnd)

Initialize this Trace object with two numerical <Array> entities and two <Number> entities

keyArray: the <Array> object containing all the keys of the <Trace>
valArray: the <Array> object containing all the values of the <Trace>
keyStart: the <Number> value defining the beginning of the acceptable key range
keyEnd: the <Number> value defining the end of the acceptable key range

Ex:
```
var t1 = new Trace;
t1.initialize([123.321,456.654,789.987],[147.741, 258.852, 369.963], 124, 450);
t1.length; // --> 0

var t1 = new Trace;
t1.initialize([123.321,456.654,789.987],[147.741, 258.852, 369.963], 124, 457);
t1.length; // --> 1
var dpo = t1[0];
dpo.key; // --> 456.654
```

```
dpo.val; // --> 258.852
```

Trace.keyArray()

Return an <Array> of <Number> entities corresponding to all the keys of this Trace object

```
Ex:
var t7 = new Trace;
t7.initialize([123.321,456.654,789.987],[147.741, 258.852, 369.963]);
t7.keyArray(); // --> 123.321,456.654,789.987
```

Trace.valArray()

Return an <Array> of <Number> entities corresponding to all the values of this Trace object

```
Ex:
var t7 = new Trace;
t7.initialize([123.321,456.654,789.987],[147.741, 258.852, 369.963]);
t7.valArray(); // --> 147.741,258.852,369.963
```

Trace.asText()

Return the <DataPoint> data points of this Trace object as a <String> formatted according to this schema: <key> <value>, with one such pair per line

```
Ex:
var t7 = new Trace;
t7.initialize([123.321,456.654,789.987],[147.741, 258.852, 369.963]);
t7.asText(); // --> output is:
123.32100000000 147.74100000000
456.65400000000 258.85200000000
```

789.9870000000 369.9630000000

Trace.exportToFile(fileName)

Export the <DataPoint> contents of this Trace object to a file. The format is according to this schema: <key> <value>, with one such pair per line

fileName: <String> containing the path to the file in which to store the data

Ex:
var t7 = new Trace;
t7.initialize([123.321,456.654,789.987],[147.741, 258.852, 369.963]);
t7.exportToFile("/tmp/trial.txt"); // --> file contains:
123.3210000000 147.7410000000
456.6540000000 258.8520000000
789.9870000000 369.9630000000

Trace.valSum()

Return the <Number> value corresponding to the sum of all the values of all the <DataPoint> object in this Trace object

Ex:
var t7 = new Trace;
t7.initialize([123.321,456.654,789.987],[147.741, 258.852, 369.963]);
t7.valSum(); // --> 776.556

Trace.combine(dataPoint)

Combine into this Trace object the <DataPoint> object

dataPoint: <DataPoint> object to combine into this MassSpectrum

Ex:

```
var t7 = new Trace;
t7.initialize([123.321,456.654,789.987],[147.741, 258.852, 369.963]);

t7.asText(); // --> output is:

123.32100000000 147.74100000000
456.65400000000 258.85200000000
789.98700000000 369.96300000000

t7.combine(new DataPoint(200.002,1000)); // --> 1, the number of added DataPoint objects

t7.asText(); // --> output is:

123.32100000000 147.74100000000
200.00200000000 1000.0000000000
456.65400000000 258.85200000000
789.98700000000 369.96300000000

t7.combine(new DataPoint(200.002,1000)); // --> 0, no added DataPoint, only increment the value

t7.asText(); // --> output is:

123.32100000000 147.74100000000
200.00200000000 2000.0000000000 // value increment by the same original amount.
456.65400000000 258.85200000000
789.98700000000 369.96300000000
```

Trace.combine(trace)

Combine into this Trace object the <Trace> object

trace: <Trace> object to combine into this Trace

Ex:

```
var t1 = new Trace;
t1.initialize([123.321,456.654,789.987],[147.741, 258.852,369.963]);
var t2 = new Trace(t1);
t2.length; // --> 3
t2.asText(); // --> output is:

123.3210000000 147.7410000000
456.6540000000 258.8520000000
789.9870000000 369.9630000000

t2.asText(); // --> output is:

123.3210000000 147.7410000000
456.6540000000 258.8520000000
789.9870000000 369.9630000000

t2.combine(t1); // --> 3: 3 DataPoint objects were combined
t2.asText(); // --> output is:

123.3210000000 295.4820000000
456.6540000000 517.7040000000
789.9870000000 739.9260000000
```

Trace.combine(key, val)

Combine into this Trace object a data point in the form of
two <Number> entities, key and val

Ex:

```
var t1 = new Trace;
t1.initialize([123.321,456.654,789.987],[147.741, 258.852,369.963]);

t1.length; // --> 3

t1.asText(); // --> output is:
```

```
123.32100000000 147.74100000000
456.65400000000 258.85200000000
789.98700000000 369.96300000000

t1.combine(456.654, 1000);

t1.asText(); // --> output is:

123.32100000000 147.74100000000
456.65400000000 1258.85200000000
789.98700000000 369.96300000000

t1.combine(800, 2000);

t1.asText(); // --> output is:

123.32100000000 147.74100000000
456.65400000000 1258.85200000000
789.98700000000 369.96300000000
800.00000000000 2000.00000000000


Trace.subtract(dataPoint)

 Subtract from this Trace object the <DataPoint> object

 dataPoint: <DataPoint> object to subtract from this MassSpectrum


Trace.subtract(trace)

 Subtract from this Trace object the <Trace> object

 trace: <Trace> object to subtract from this Trace


Trace.subtract(mz, i)
```

Subtract from this Trace object a data point in the form of
two <Number> values, key and value

MassSpectrum

This class is the main class for handling mass spectra.
MassSpectrum objects can be "combination spectra", that is, they correspond
to the summation of a number of spectra. When a combination of mass spectra
is performed, the combination spectrum is created empty at first and is
then "configured" to receive the result of the sequential combination of
all the mass spectra that are to be summated. There are a number of member
data that are devoted to that combination configuration.

A spectrum is actually a <Trace> that has a set of particular members:
- rt: the retention time at which the spectrum was acquired
- dt: the mobility drift time (for ion mobility spectra)
- binCount: the number of bins set up in a combination spectrum
- binSize: the size of the bins in the combination spectrum
- binSizeType: the type of the bin size (enum AMU, PPM, RES)
- binSizeTypestring: <String> representation of binSizeType
- minMz: the minimum m/z value of the combination specrum
- maxMz: the maximum m/z value of the combination specrum
- applyMzShift: tells if the m/z shift should be applied
- mzShift: m/z shift applied (or not) to each combined mass spectrum
- removeZeroValDataPoints: tells if the data points in the spectrum that
have a m/z with 0 intensity should be removed

MassSpectrum()

Constructor of a MassSpectrum object

Ex:

var ms1 = new MassSpectrum();

MassSpectrum(title)

Constructor of a MassSpectrum object

title: <String> containing the title of the MassSpectrum object

Ex:

```
var ms1 = new MassSpectrum("spectrum title");
ms1.title; // --> spectrum title
```

MassSpectrum(other)

Constructor of a MassSpectrum object

other: <Trace> object to be used to initialize this object

Ex:

```
var t1 = new Trace;
t1.initialize([123.321,456.654,789.987],[147.741, 258.852, 369.963]);
t1.asText();

123.3210000000 147.7410000000
456.6540000000 258.8520000000
789.9870000000 369.9630000000

var ms1 = new MassSpectrum(t1);
ms1.asText();

123.3210000000 147.7410000000
456.6540000000 258.8520000000
789.9870000000 369.9630000000
```

MassSpectrum(other)

Constructor of a MassSpectrum object

other: <MassSpectrum> object to be used to initialize this object

Ex:

```
var ms1 = new MassSpectrum();
ms1.initialize([123.321,456.654,789.987],[147.741, 258.852, 369.963]);
ms1.asText();

123.32100000000 147.74100000000
456.65400000000 258.85200000000
789.98700000000 369.96300000000

ms1.rt = 1.2;
ms1.dt = 0.00325698;

var ms2 = new MassSpectrum(ms1);
ms2.asText();

123.32100000000 147.74100000000
456.65400000000 258.85200000000
789.98700000000 369.96300000000

ms2.rt; // --> 1.2
ms2.dt; // --> 0.00325698
```

MassSpectrum.title

Return the title <String> property.

Ex:

```
var ms1 = new MassSpectrum("spectrum title");
```

```
ms1.title; // --> spectrum title
```

MassSpectrum.length

Return the length <Number> property (number of <DataPoint> objects).

Ex:

```
var ms1 = new MassSpectrum();
ms1.initialize([123.321,456.654,789.987],[147.741, 258.852, 369.963]);
ms1.asText();

123.3210000000 147.7410000000
456.6540000000 258.8520000000
789.9870000000 369.9630000000

ms1.length; // --> 3
```

MassSpectrum.rt

Return the rt <Number> property (retention time at which this mass spectrum was acquired.

Ex:

```
var ms1 = new MassSpectrum();
ms1.rt = 1.2;

ms1.rt; // --> 1.2
```

MassSpectrum.dt

Return the dt <Number> property (ion mobility drift time at which this mass spectrum was acquired.

Ex:

```
var ms1 = new MassSpectrum();
ms1.dt = 0.00325698;

ms1.dt; // --> 0.00325698
```

MassSpectrum.binCount

Return the bin count <Number> property (number of bins in this mass spectrum).

MassSpectrum.binSize

Return the bin size <Number> property (size of the bins in this mass spectrum).

MassSpectrum.binSizeType

Return the bin size type property as a <Number>.

Ex:

```
var ms1 = new MassSpectrum();
ms1.binSizeType = 2;
ms1.binSizeType; // --> 2
ms1.binSizeTypeString = "RES";
ms1.binSizeType; // --> 4
```

MassSpectrum.binSizeTypeString

Return the bin size type property as a <String>.

Ex:

```
var ms1 = new MassSpectrum();
ms1.binSizeType = 2;
ms1.binSizeTypeString; // -> MZ;
```

MassSpectrum.minMz

Return the <Number> minimum m/z value to be used to craft the bins.

MassSpectrum.maxMz

Return the <Number> maximum m/z value to be used to craft the bins.

MassSpectrum.applyMzShift

Return the <Boolean> value telling if the m/z shift should be applied.

MassSpectrum.mzShift

Return the <Number> m/z shift value to apply when combining spectra.

MassSpectrum.removeZeroValDataPoints

Return the <Boolean> value telling if the <DataPoint> objects having m/z keys of a 0-intensity value should be removed from the spectrum.

MassSpectrum[index]

Return the <DataPoint> object at index.

Ex:

```
var ms1 = new MassSpectrum();
ms1.initialize([123.321,456.654,789.987],[147.741, 258.852, 369.963]);
ms1.asText();

123.3210000000 147.7410000000
456.6540000000 258.8520000000
789.9870000000 369.9630000000

var dpo = ms1[0];
dpo.key; // --> 123.321
dpo.val; // --> 147.741

ms1[0].key; // --> 123.321
ms1[0].val; // --> 147.741
```

MassSpectrum.initialize(other)

Initialize this MassSpectrum object with a <Trace> object

other: <Trace> object

Ex:

```
var t1 = new Trace;
t1.initialize([123.321,456.654,789.987],[147.741, 258.852, 369.963]);
t1.asText();

123.3210000000 147.7410000000
456.6540000000 258.8520000000
789.9870000000 369.9630000000

var ms1 = new MassSpectrum();
```

```
ms1.initialize(t1);
ms1.asText();

123.3210000000 147.7410000000
456.6540000000 258.8520000000
789.9870000000 369.9630000000
```

MassSpectrum.initialize(other)

Initialize this MassSpectrum object with a <MassSpectrum> object

other: <MassSpectrum> object

Ex:

```
var ms1 = new MassSpectrum();
ms1.initialize([123.321,456.654,789.987],[147.741, 258.852, 369.963]);
ms1.asText();

123.3210000000 147.7410000000
456.6540000000 258.8520000000
789.9870000000 369.9630000000

var ms2 = new MassSpectrum();
ms2.initialize(ms1);
ms2.asText();

123.3210000000 147.7410000000
456.6540000000 258.8520000000
789.9870000000 369.9630000000
```

MassSpectrum.initialize(mzArray, iArray)

Initialize this MassSpectrum object with two <Array> of numerical

values

mzArray: <Array> object containing the m/z values of the spectrum
iArray: <Array> object containing the intensity values of the spectrum

Ex:

```
var ms1 = new MassSpectrum;

var mzArray = [123.321,456.654,789.987];
var iArray = [147.741, 258.852, 369.963];

ms1.initialize(mzArray, iArray);

ms1.asText(); // --> output is:

123.3210000000 147.7410000000
456.6540000000 258.8520000000
789.9870000000 369.9630000000
```

MassSpectrum.initialize(mzArray, iArray, mzStart, mzEnd)

Initialize this MassSpectrum object with two <Array> of numerical values and two <Number> entities delimiting a range of acceptable m/z values.

mzArray: <Array> object containing the m/z values of the spectrum
iArray: <Array> object containing the intensity values of the spectrum
mzStart: m/z value used to filter the data points to be used for the initialization (the start of the acceptable m/z range)
mzEnd: m/z value used to filter the data points to be crafted for the initialization (the end of the acceptable m/z range)

Ex:

```
var ms1 = new MassSpectrum;
```

```
var mzArray = [123.321,456.654,789.987];
var iArray = [147.741, 258.852, 369.963];

ms1.initialize(mzArray, iArray, 122.231, 789.986); // range: [122.231-789.986]

ms1.asText(); // --> output is:

123.32100000000 147.74100000000
456.65400000000 258.85200000000
```

MassSpectrum.mzArray()

Return a list of <Number> entities corresponding to all the m/z values of the spectrum

Ex:

```
var ms1 = new MassSpectrum;

var mzArray = [123.321,456.654,789.987];
var iArray = [147.741, 258.852, 369.963];

ms1.initialize(mzArray, iArray, 122.231, 789.986); // range: [122.231-789.986]

ms1.asText(); // --> output is:

123.32100000000 147.74100000000
456.65400000000 258.85200000000

ms1.mzArray(); --> output is:
123.321,456.654
```

MassSpectrum.iArray()

Return a list of <Number> entities corresponding to all the intensity values of the spectrum

Ex:

```
var ms1 = new MassSpectrum;

var mzArray = [123.321,456.654,789.987];
var iArray = [147.741, 258.852, 369.963];

ms1.initialize(mzArray, iArray, 122.231, 789.986); // range: [122.231-789.986]

ms1.asText(); // --> output is:

123.32100000000 147.74100000000
456.65400000000 258.85200000000

ms1.iArray(); --> output is:
147.741,258.852
```

MassSpectrum.asText()

Return the data points of the spectrum in <String> object. The format of the string is of the kind <m/z>,<intensity>, one data point per line.

Ex:

```
var ms1 = new MassSpectrum;

var mzArray = [123.321,456.654,789.987];
var iArray = [147.741, 258.852, 369.963];

ms1.initialize(mzArray, iArray, 122.231, 789.986); // range: [122.231-789.986]

ms1.asText(); // --> output is:
```

```
123.3210000000 147.7410000000
456.6540000000 258.8520000000
```

MassSpectrum.tic()

Return the total ion current calculated for this MassSpectrum object

Ex:

```
var ms1 = new MassSpectrum;
ms1.initialize([123.321, 456.654, 789.987], [147.741, 258.852, 369.963]);
ms1.asText();
ms1.tic(); // --> 776.556
```

MassSpectrum.combine(dataPoint)

Combine into this MassSpectrum object the <DataPoint> object

dataPoint: <DataPoint> object to combine into this MassSpectrum

Ex:

```
var ms1 = new MassSpectrum;
ms1.initialize([123.321, 456.654, 789.987], [147.741, 258.852,369.963]);
ms1.asText(); // --> output is:

123.3210000000 147.7410000000
456.6540000000 258.8520000000
789.9870000000 369.9630000000

ms1.tic(); // -> 776.556

var dp1 = new DataPoint(123.321, 456.654);
ms1.combine(dp1);
ms1.asText(); // --> output is:
```

```
123.3210000000 604.3950000000 // value incremented
456.6540000000 258.8520000000
789.9870000000 369.9630000000

dp1.key = 800;
dp1.val = 1000;
ms1.combine(dp1);
ms1.asText(); // --> output is:

123.3210000000 604.3950000000
456.6540000000 258.8520000000
789.9870000000 369.9630000000
800.0000000000 1000.0000000000 // creation of new data point


MassSpectrum.combine(trace)

Combine into this MassSpectrum object the <Trace> object

trace: <Trace> object to combine into this MassSpectrum

Ex:

var ms1 = new MassSpectrum;
ms1.initialize([123.321, 456.654, 789.987], [147.741, 258.852, 369.963]);
ms1.asText(); //  --> output is:

123.3210000000 147.7410000000
456.6540000000 258.8520000000
789.9870000000 369.9630000000

ms1.tic(); // --> 776.556
var t1 = new Trace;
t1.initialize([123.321,600.000,789.987],[147.741, 300.000, 369.963]);
t1.asText(); // --> output is:
```

```
123.32100000000 147.74100000000
600.00000000000 300.00000000000
789.98700000000 369.96300000000

t1.valSum(); // --> 817.704
ms1.combine(t1);
ms1.asText(); // --> output is:

123.32100000000 295.48200000000
456.65400000000 258.85200000000
600.00000000000 300.00000000000
789.98700000000 739.92600000000

ms1.tic(); // --> 1594.26
```

MassSpectrum.combine(massSpectrum)

Combine into this MassSpectrum object the <MassSpectrum> object

massSpectrum: <MassSpectrum> object to combine into this MassSpectrum

Ex:

```
var ms1 = new MassSpectrum;
ms1.initialize([123.321, 456.654, 789.987], [147.741, 258.852, 369.963]);

ms1.asText(); // --> output is:

123.32100000000 147.74100000000
456.65400000000 258.85200000000
789.98700000000 369.96300000000

ms1.tic(); // --> 776.556

var ms2 = new MassSpectrum(ms1);
```

```
ms2.combine(1500, 1000);
ms2.asText(); // --> output is:

123.3210000000 147.7410000000
456.6540000000 258.8520000000
789.9870000000 369.9630000000
1500.0000000000 1000.0000000000

ms2.combine(ms1);
ms2.asText(); // --> output is:

123.3210000000 295.4820000000
456.6540000000 517.7040000000
789.9870000000 739.9260000000
1500.0000000000 1000.0000000000

ms2.tic(); // --> 2553.112
```

MassSpectrum.combine(mz, i)

Combine into this MassSpectrum object a data point in the form of two <Number> values, m/z and intensity

Ex:

```
var ms1 = new MassSpectrum;
ms1.initialize([123.321, 456.654, 789.987], [147.741, 258.852, 369.963]);
ms1.asText(); // --> output is:

123.3210000000 147.7410000000
456.6540000000 258.8520000000
789.9870000000 369.9630000000

ms1.tic(); // --> 776.556
var mz1 = 123.321;
var i1 = 300.000
```

```
ms1.combine(mz1, i1);
ms1.asText(); // --> output is:

123.3210000000 447.7410000000 // value incremented
456.6540000000 258.8520000000
789.9870000000 369.9630000000

ms1.combine(800, 1000);
ms1.asText(); // --> output is:

123.3210000000 447.7410000000
456.6540000000 258.8520000000
789.9870000000 369.9630000000
800.0000000000 1000.0000000000 // new data point created


MassSpectrum.subtract(dataPoint)

 Subtract from this MassSpectrum object the <DataPoint> object

 dataPoint: <DataPoint> object to subtract from this MassSpectrum


MassSpectrum.subtract(trace)

 Subtract from this MassSpectrum object the <Trace> object

 trace: <Trace> object to subtract from this MassSpectrum


MassSpectrum.subtract(massSpectrum)

 Subtract from this MassSpectrum object the <MassSpectrum> object

 massSpectrum: <MassSpectrum> object to subtract from this MassSpectrum

 Ex:
```

```
var ms1 = new MassSpectrum;
ms1.initialize([123.321, 456.654, 789.987], [147.741, 258.852, 369.963]);
ms1.asText(); // --> output is:
123.3210000000 147.7410000000
456.6540000000 258.8520000000
789.9870000000 369.9630000000

var ms2 = new MassSpectrum(ms1);
ms2.subtract(ms1);
ms2.asText(); // --> output is:

123.3210000000 0.0000000000
456.6540000000 0.0000000000
789.9870000000 0.0000000000
```

MassSpectrum.subtract(mz, i)

Subtract from this MassSpectrum object a data point in the form of two <Number> values, m/z and intensity

Ex:

```
var ms1 = new MassSpectrum;
ms1.initialize([123.321, 456.654, 789.987], [147.741, 258.852, 369.963]);
ms1.asText(); // --> output is:

123.3210000000 147.7410000000
456.6540000000 258.8520000000
789.9870000000 369.9630000000

ms1.subtract(123.321,100);
ms1.asText(); // --> output is:

123.3210000000 47.7410000000
456.6540000000 258.8520000000
```

789.9870000000 369.9630000000

AbstractPlotWidget

This class is not exported to the JavaScript environment, but it
serves as the base class for all the plot widget subclasses. Its methods
are thus available to all the objects of the various subclasses, like
MassSpecPlotWidget, DriftSpecPlotWidget, ColorMapPlotWidget and
TicChromPlotWidget, collectively referred to as <PlotWidget> below.

When using objects of the various classes above, the object name will thus
be, for example, ticChromPlotWidget0 or driftSpecPlotWidget1. These objects
are made available to the scripting environment in the tree view on the
left part of the scripting window. See the user manual for
details.

<PlotWidget>.showHistory()

Shows a tooltip with the History of this plot widget.

<PlotWidget>.historyAsText()

Return a textual representation of the history of this plot widget.

<PlotWidget>.showHelpSummary()

Show a tooltip with a help string summarizing the various keyboard key
combinations.

<PlotWidget>.shouldDestroyPlotWidget()

Start the destruction of this plot widget.

<PlotWidget>.toggleMultiGraph()

If the multi-graph plot corresponding to this plot widget is visible, make it invisible. And vice versa.

<PlotWidget>.showMultiGraph()

Make the multi-graph plot corresponding to this plot widget visible.

<PlotWidget>.hideMultiGraph()

Make the multi-graph plot corresponding to this plot widget invisible.

<PlotWidget>.exportData()

Open the data export configuration window. Exporting data allows exporting only a subset of the whole data set. It can only be performed if the original data were read from a SQLite3 database-formatted mass data file.

<PlotWidget>.asXyText()

Return a textual representation of \c this plot widget data.

Only the data range currently displayed is processed.

<PlotWidget>.exportPlot()

Open a file selection dialog window to select a file in which to export this plot widget's data.

Unlike the exportData() function, this function only exports the currently

displayed graph data to a comma-separated value text file.

<PlotWidget>.exportPlotToFile(fileName, rangeStart, rangeEnd)

Export plot data to a text file as a simple plot. Limit the data range to export using the numerical parameters.

fileName: <String> holding the file name
rangeStart: <Number> holding the beginning of the range to be exported
rangeEnd <Number> holding the end of the range to be exported

<PlotWidget>.savePlotToGraphicsFile()

Open the graphics export configuration dialog window.

<PlotWidget>.saveToPdfFile(fileName, noCosmeticPen, width, height, pdfCreator, title)

Save this plot widget as a PDF file graphics.

fileName: <String> holding the file name
noCosmeticPen: <Boolean> that tells if pen optimizations should be performed
width: <Number> holding the size of the graphics file in pixels
height: <Number> holding the size of the graphics file in pixels
pdfCreator: <String> holding the name of the creator of the PDF file
title: <String> holding the title of the graphics file

<PlotWidget>.keys()

Return an <Array> of <Number> values representing the keys of the data plotted.

<PlotWidget>.values()

Return an <Array> of <Number> values representing the values of the data plotted.

<PlotWidget>.lastTicIntensity()

Return the last TIC intensity value that was computed.

<PlotWidget>.trace()

Return a Trace object initialized with the data in this plot widget.

<PlotWidget>.replotWithAxisRangeX(lower, upper)

Replot this plot widget with new key (X-axis) data range.

lower: <Number> holding the start value of the range
upper: <Number> holding the end value of the range

<PlotWidget>.replotWithAxisRangeY(lower, upper)

Replot this plot widget with new value (Y-axis) data range.

lower: <Number> holding the start value of the range
upper: <Number> holding the end value of the range

<PlotWidget>.getYatX(x)

get the value at key x.

x: <Number> holding the X-axis key for which the value is returned.

ColorMapPlotWidget

This class cannot be instantiated with the new operator. Objects
of this class are made available to the scripting environment under the
form of object variable names colorMapPlotWidget[index], with [index]
being 0 for the first object created and being incremented each time a new
ColorMapPlotWidget is created.

ColorMapPlotWidget.jsIntegrateToRt(dtRangeStart, dtRangeEnd, mzRangeStart, mzRangeEnd)

Starts an integration of mass spectra data to a XIC chromatogram limiting
the data ranges specified with the numerical arguments.

dtRange(Start | End) drift time range values
mzRange(Start | End) m/z range values

This function creates a new plot widget to display the obtained results.

ColorMapPlotWidget.jsIntegrateToMz(dtRangeStart, dtRangeEnd, mzRangeStart, mzRangeEnd)

Starts an integration of mass spectra data to a mass spectrum limiting the
data ranges specified with the numerical arguments.

dtRange(Start | End) drift time range values
mzRange(Start | End) m/z range values

This function creates a new plot widget to display the obtained results.

ColorMapPlotWidget.jsIntegrateToDt(dtRangeStart, dtRangeEnd, mzRangeStart, mzRangeEnd)

Starts an integration of mass spectra data to a drift spectrum limiting the
data ranges specified with the numerical arguments.

dtRange(Start | End) drift time range values
mzRange(Start | End) m/z range values

This function creates a new plot widget to display the obtained results.

ColorMapPlotWidget.jsIntegrateToTicIntensity(dtRangeStart, dtRangeEnd, mzRangeStart, mzRangeEnd)

Starts an integration of mass spectra data to a TIC intensity single value
limiting the data ranges specified with the numerical arguments.

dtRange(Start | End) drift time range values
mzRange(Start | End) m/z range values

The TIC intensity value is returned

ColorMapPlotWidget.transposeAxes()

Transpose the axes of this color map plot widget

ColorMapPlotWidget.jsSetMzIntegrationParams(mzIntegrationParams)

Set the m/z integration parameters object passed as parameter to this plot
widget so that they are taken into account for the next integrations to a
mass spectrum.

ColorMapPlotWidget.jsMzIntegrationParams()

Return the MzIntegrationParams object currently set for this color map
plot widget.

These parameters are taken into account for the integrations to a mass
spectrum.

TicChromPlotWidget

This class cannot be instantiated with the new operator. Objects of this class are made available to the scripting environment under the form of object variable names ticChromPlotWidget[index], with [index] being o for the first object created and being incremented each time a new TicChromPlotWidget is created.

TicChromPlotWidget.jsIntegrateToMz(lower, upper)

Starts an integration of mass spectra data to a mass spectrum limiting the data range specified with the numerical arguments.

lower, upper: retention time values limiting the integration. If no values are provided, there is no limitation to the integration.

This function creates a new plot widget to display the obtained results.

TicChromPlotWidget.jsIntegrateToDt(lower, upper)

Starts an integration of mass spectra data to a drift spectrum limiting the data range specified with the numerical arguments.

lower, upper: retention time values limiting the integration. If no values are provided, there is no limitation to the integration.

This function creates a new plot widget to display the obtained results.

TicChromPlotWidget.jsIntegrateToTicIntensity(lower, upper)

Starts an integration of mass spectra data to a TIC intensity single value limiting the data range specified with the numerical arguments.

lower, upper: retention time values limiting the integration. If no values are provided, there is no limitation to the integration.

The TIC intensity value is returned

TicChromPlotWidget.jsIntegrateToXic(lower, upper)

Starts an integration of mass spectra data to a XIC chromatogram limiting the data range specified with the numerical arguments.

This process is not equivalent to an integration to RT because we do not take into account the History of the current plot widget. Indeed, the integration is performed without any filtering from the initial mass spectrometry data.

lower, upper: retention time values limiting the integration. If no values are provided, there is no limitation to the integration.

This function creates a new plot widget to display the obtained results.

TicChromPlotWidget.statistics()

Compute the mass spectra statistics on the spectra currently in the range.

TicChromPlotWidget.newPlot(trace)

Create a new TicChromPlotWidget using <Trace> trace for the initialization of the data.

This function creates a new plot widget to display the <Trace> data.

TicChromPlotWidget.jsSetMzIntegrationParams(mzIntegrationParams)

Set the m/z integration parameters object passed as parameter to this plot widget so that they are taken into account for the next integrations to a mass spectrum.

TicChromPlotWidget.jsMzIntegrationParams()

Return the MzIntegrationParams object currently set for this color map plot widget.

These parameters are taken into account for the integrations to a mass spectrum.

MassSpecPlotWidget

This class cannot be instantiated with the new operator. Objects of this class are made available to the scripting environment under the form of object variable names massSpecPlotWidget[index], with [index] being o for the first object created and being incremented each time a new MassSpecPlotWidget is created.

MassSpecPlotWidget.jsIntegrateToDt(lower, upper)

Starts an integration of mass spectra data to a drift spectrum limiting the data range specified with the numerical arguments.

lower, upper: retention time values limiting the integration. If no values are provided, there is no limitation to the integration.

This function creates a new plot widget to display the obtained results.

MassSpecPlotWidget.jsIntegrateToRt(lower, upper)

Starts an integration of mass spectra data to a XIC chromatogram limiting the data range specified with the numerical arguments.

lower, upper: retention time values limiting the integration. If no values are provided, there is no limitation to the integration.

This function creates a new plot widget to display the obtained results.

MassSpecPlotWidget.jsIntegrateToTicIntensity(lower, upper)

Starts an integration of mass spectra data to a TIC intensity single value limiting the data range specified with the numerical arguments.

lower, upper: retention time values limiting the integration. If no values are provided, there is no limitation to the integration.

The TIC intensity value is returned

MassSpecPlotWidget.massSpectrum(lower, upper)

Creates a new <MassSpectrum> object limiting the data to the lower and upper range limits. The data used to craft the new MassSpectrum are this plot widget data. That is, the <MassSpectrum> object is not created by looking into the internal mass spectral data set.

lower, upper: m/z values limiting the integration. If no values are provided, there is no limitation to the integration and all the points in this plot widget are used.

Return a new MassSpectrum object.

MassSpecPlotWidget.newPlot(trace)

Create a new MassSpecPlotWidget using <Trace> trace for the initialization of the data.

This function creates a new plot widget to display the <Trace> data.

DriftSpecPlotWidget

This class cannot be instantiated with the new operator. Objects of this class are made available to the scripting environment under the form of object variable names driftSpecPlotWidget[index], with [index] being o for the first object created and being incremented each time a new DriftSpecPlotWidget is created.

DriftSpecPlotWidget.jsIntegrateToMz(lower, upper)

Starts an integration of mass spectra data to a mass spectrum limiting the data range specified with the numerical arguments.

lower, upper: retention time values limiting the integration. If no values are provided, there is no limitation to the integration.

This function creates a new plot widget to display the obtained results.

DriftSpecPlotWidget.jsIntegrateToRt(lower, upper)

Starts an integration of mass spectra data to a XIC chromatogram limiting the data range specified with the numerical arguments.

lower, upper: retention time values limiting the integration. If no values are provided, there is no limitation to the integration.

This function creates a new plot widget to display the obtained results.

DriftSpecPlotWidget.jsIntegrateToTicIntensity(lower, upper)

Starts an integration of mass spectra data to a TIC intensity single value limiting the data range specified with the numerical arguments.

lower, upper: retention time values limiting the integration. If no values are provided, there is no limitation to the integration.

The TIC intensity value is returned

DriftSpecPlotWidget.newPlot(trace)

Create a new DriftSpecPlotWidget using <Trace> trace for the initialization of the data.

This function creates a new plot widget to display the <Trace> data.

TicChromPlotWidget.jsSetMzIntegrationParams(mzIntegrationParams)

Set the m/z integration parameters object passed as parameter to this plot widget so that they are taken into account for the next integrations to a mass spectrum.

TicChromPlotWidget.jsMzIntegrationParams()

Return the MzIntegrationParams object currently set for this color map plot widget.

These parameters are taken into account for the integrations to a mass spectrum.

MainWindow

This is the main mineXpert program window. It is not instantiable through the JavaScript environment. Some funcitonalities are available to allow highlevel tasks to be automated. The MainWindow single instance of the minexpert software program is made available to the JavaScript

environment under the object name mainWinwod, with an alias
mainWnd.

MainWindow.quit()

Quit the application.

MainWindow.clearPlots()

Clear all plots and give back all memory associated to the mass data.

MainWindow.jsCreateMassSpectrumFromClipboard()

Create a new mass spectrum from data in the clipboard. The data in the clipboard need to be under the format of <number><separator><number>, <separator> being any non-numerical and non-'.' character. The first <number> is the m/z and the second <number> is the intensity.

Upon completing the parsing of the textual data in the clipboard, a pseudo TIC chromatogram is created that needs to be integrated to a mass spectrum using:

lastTicChromPlotWidget.integrateToMz();

Ex:
mainWindow.createMassSpectrumFromClipboard();
lastTicChromPlotWidget.integrateToMz();

MainWindow.jsOpenMassSpectrometryFile(fileName, streamed)

Load mass spectrometry data from fileName.

fileName: <String> holding the mass spectrometry data file name
streamed: <Boolean> telling if the loading should be in streamed mode.

Ex:
mainWindow.jsOpenMassSpectrometryFile("/tmp/20161027-cgb-wt-mobility.db", true);

reads the data in the file in streamed mode.

mainWindow.jsOpenMassSpectrometryFile("/tmp/20161027-cgb-wt-mobility.db", false);

reads the data in the file in full mode.

MainWindow.jsOpenMassSpectrometryFiles(fileNameList, streamed)

Load mass spectrometry data from all the files in the fileNameList.

fileNameList: <Array> of <String> objects holding the mass spectrometry data file names
streamed: <Boolean> telling if the loading should be in streamed mode.

MainWindow.about()

Open an About dialog window.

Ex:
mainWindow.about();

SqlMassDataSlicerWnd

This class cannot be instantiated with the new operator. The only
object of this class that is published to the scripting environment is
named ``sqlMassDataSlicerWnd''.

This class can only be used safely if it is first activated by using the
exportData() function of the <PlotWidget> classes (see <PlotWidget>).
Indeed, the <PlotWidget>.exportData() activates this sqlMassDataSlicerWnd
object and initializes it with the current data range displayed in the

widget.

The following describes the various methods that can be used to automate fully the slicing of very large mass spectrometry data files in a set of smaller files. There are three ways to create new files of a reduced size with respect to the initial mass data file:

- export data from the original file in a single file but with a smaller set of data. For example, export only part of a TIC chromatogram or export only the mass data corresponding to a given drift time range.
- export data from the original file in multiple smaller files (either the whole initial data range or a smaller range). Two possibilities are offered:

  - define the number of slices (their size is dynamically computed);
  - define the size of the slices (their number is dynamically computed);

This window is available for any <PlotWidget> displaying TIC/XIC chromatogram data or drift spectrum data. It is therefore essential to start a data export process from that specific plot widget of interest by calling <PlotWidget>.exportData(). This call will initialize all the widget-specific data into this data export configuration window.

Ex:
mainWnd.openMassSpectrometryFile("/path/to/file/20161027-cgb-wt-mobility.db", false);
// The TIC chromatogram window displays the TIC chromatogram in ticChromPlotWidgeto

// Activate and initialize the SqlMassDataSlicerWnd (its object name is sqlMassDataSlicerWnd)
ticChromPlotWidgeto.exportData();

// Set the number of slices to 2
sqlMassDataSlicerWnd.setSliceCount(2);

```
// Set the start of the range of interest (retention time)
sqlMassDataSlicerWnd.rangeStart(0.2);

// Set the end of the range of interest (retention time)
sqlMassDataSlicerWnd.rangeEnd(12.7);

// Set the format string to use to craft the file names
sqlMassDataSlicerWnd.setFormatString("%f-export-slice-%n-of-%c-range_%s-%e.db");

// Validate the configuration settings (crafts the file names according to
the configuration of the data export and according to the format string)

sqlMassDataSlicerWnd.validate(false);

// Take a peek at the various file names that have been crafted.
sqlMassDataSlicerWnd.reviewFileNames();

// If the file names have no error, then start the data export process.
sqlMassDataSlicerWnd.confirm();

Note that the calls to validate() and reviewFilenames() and confirm() can
be shortened to a single call to run() if it is expected that no error will
occur upon validation of the data export parameters (typically when the
script has been thoroughly tested). In that case, the script runs perfectly
unattended.

 sqlMassDataSlicerWnd.setSrcFileName(fileName)

  Set the source SQLite3 db mass data file name. Only use this function if
  you know the range of the data you are willing to export. Usually this
  function is not used, because when starting the data export process using
  the <PlotWidget>.exportData() function, that source file name is set
  automatically. Thus we recommend you use this function only if you know
  what you are doing.

  fileName : <String> that holds the name of the source mass spectrometry
  data file in the SQLite3 database format.
```

sqlMassDataSlicerWnd.setDestDirectory(dir)

Set the directory (create it if it does not exist) where all the slice files are to be written to. If this function is not called before calling the validate() or run() functions, the destination directory will be the directory where the source file is residing.

dir: <String> holding the name of the directory in which to write all the slice files.

sqlMassDataSlicerWnd.setFormatString(format)

Set the string that configures the naming of the slice files. The following format specification is available:

%f: original filename without <.extension>
%s: slice range start value
%e: slice range end value
%n: slice number
%c: total count of slices

For example, if the original (source) filename were "my-mass-data.db" and the format string were "%f-export-slice-%n-of-%c-range_%s-%e.db", the first file would be named
"my-mass-data-export-slice-1-of-7-range_17-28.db"
the second file
"my-mass-data-export-slice-2-of-7-range_29-40.db"
and so on for another 5 files since the total count of slices is 7.

sqlMassDataSlicerWnd.setRangeStart(start)

Set the start value of the data range to be exported. For example, if the data export process was started in a TIC chromatogram plot widget, and the

whole acquisition run spanned [0-35] minutes, then the user might want to only export a fraction of that span: [17-28]. In that case, the range start value would be 17.

start: <Number> setting the start value of the range to be exported.

sqlMassDataSlicerWnd.rangeStart()

Return the start value of the data range to be exported. For example, if the data export process was started in a TIC chromatogram plot widget, and the whole acquisition run spanned [0-35] minutes, then the user might want to only export a fraction of that span: [17-28]. In that case, the range start value would be 17.

sqlMassDataSlicerWnd.setRangeEnd(end)

Set the end value of the data range to be exported. For example, if the data export process was started in a TIC chromatogram plot widget, and the whole acquisition run spanned [0-35] minutes, then the user might want to only export a fraction of that span: [17-28]. In that case, the range end value would be 28.

end: <Number> setting the end value of the range to be exported.

sqlMassDataSlicerWnd.rangeEnd()

Return the end value of the data range to be exported. For example, if the data export process was started in a TIC chromatogram plot widget, and the whole acquisition run spanned [0-35] minutes, then the user might want to only export a fraction of that span: [17-28]. In that case, the range end value would be 28.

sqlMassDataSlicerWnd.setRange(start, end)

Set the data range to be exported. For example, if the data export process was started in a TIC chromatogram plot widget, and the whole acquisition run spanned [0-35] minutes, then the user might want to only export a fraction of that span: [17-28]. In that case, the range start and end values would be 17 and 28 respectively.

start: <Number> setting the start value of the range to be exported.
end: <Number> setting the end value of the range to be exported.

sqlMassDataSlicerWnd.setSliceCount(count)

Set the number of slices to be created out of the initial data set.

count: <Number> indicating the number of slices.

sqlMassDataSlicerWnd.setSliceSize(size)

Set the size of the slices to be created out of the initial data set.

size: <Number> indicating the size of the slices.

sqlMassDataSlicerWnd.setExportToOneFile()

Tells that the initial mass data set must be exported into a single file. Any [rangeStart-rangeEnd] export range that might have been set is taken into account. Not limiting the range of the exported data makes the data export operation equivalent to copying the file into another file virtually identical to the initial one.

sqlMassDataSlicerWnd.fileNames()

Print the names of the files that will be written during the data export

process. For the list of file names to be created and be available, first set the various data export options (format string, number or size of slices...) and call sqlMassDataSlicerWnd.validate().

sqlMassDataSlicerWnd.reviewFileNames()

Print the names of the files that will be written during the data export process. For the list of file names to be created and available, first set the various data export options (format string, number or size of slices...) and call sqlMassDataSlicerWnd.validate(). In this function, the graphical user interface is modified to allow the user to review the file names and makes available a push button to start the export process.

sqlMassDataSlicerWnd.validate(noconfirm)

Verify the configuration settings for the data export operation. Upon this validation, the file names of the files that will be written during the data export are crafted according to the file format string (see setFormatString()). This function needs to be called prior to calling reviewFileNames() or fileNames(). After reviewing the file names, a call to the function confirm() starts the data export process.

noconfirm: <Boolean> value telling if the data export process can proceed automatically. If true, no call to confirm() needs to be done for the data export to start immediately if there is no error during the validation step.

sqlMassDataSlicerWnd.confirm()

Confirm that the data export process can start. This function must be called after the validation has been performed (see validate()) and the user has checked that the file names have been correcly crafted upon validation according to the format string.

sqlMassDataSlicerWnd.run()

After having set the range, the number (or size) of the slices, let the program compute the size (or number, respectively) of the slices and run the data export. This function calls validate() with noconfirm set to True (see validate()). This way, if no error is encountered during the validation, then the data export process is started right way, which is expected for a script to run unattended.

A GNU General Public License version 3

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. https://fsf.org/

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

Preamble

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

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

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

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

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

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

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

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of

the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

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

TERMS AND CONDITIONS

0. Definitions.

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

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

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

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

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

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

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

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

1. Source Code.

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

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

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

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

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

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

2. Basic Permissions.

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

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

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

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

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

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

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

4. Conveying Verbatim Copies.

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

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

5. Conveying Modified Source Versions.

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

a. The work must carry prominent notices stating that you modified it, and giving a relevant date.

b. The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c. You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d. If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

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

6. Conveying Non-Source Forms.

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

a. Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

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

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

c. Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d. Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e. Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

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

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

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

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

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

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

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

7. Additional Terms.

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

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

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

a. Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b. Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c. Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d. Limiting the use for publicity purposes of names of licensors or authors of the material; or

e. Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f. Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

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

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

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

8. Termination.

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

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

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

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

9. Acceptance Not Required for Having Copies.

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

10. Automatic Licensing of Downstream Recipients.

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

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

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

11. Patents.

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

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

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

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

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

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

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

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

12. No Surrender of Others' Freedom.

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

13. Use with the GNU Affero General Public License.

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

14. Revised Versions of this License.

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

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

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

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

15. Disclaimer of Warranty.

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

16. Limitation of Liability.

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

17. Interpretation of Sections 15 and 16.

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

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

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

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

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

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

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

You should have received a copy of the GNU General Public License
along with this program.  If not, see https://www.gnu.org/licenses/.
```

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

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

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

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

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

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

Colophon

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

Filippo Rusconi is the author of a handbook about mass spectrometry for biochemists (French). The book was published by the French sci/tech publisher Lavoisier (https://www.lavoisier.fr).

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

The frog on the book title page is a frog from Papua. This frog is able to hover when performing downwards leaps. This picture is courtesy http://www.papuaweb.org

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

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

The main font used was EBGaramond (https://github.com/georgd/EB-Garamond) and the symbol/mathematical font was from the STIX project (https://www.stixfonts.org/) (font: STIX2Math).

The screen shots were taken with Spectacle, the screen capture program shipped along with my KDE (https://www.kde.org/) desktop environment and resampled using The GNU image manipulation program The Gimp (https://www.gimp.org/). Illustrations were done in Inkscape (https://inkscape.org/), a vectorial drawing software.