SIRIUS Documentation

Release 3.1

Kai Dürhrkop

November 03, 2015
SIRIUS 3 is a Java library for analyzing metabolites from tandem mass spectrometry data. It combines the analysis of isotope patterns in MS spectra with the analysis of fragmentation patterns in MS/MS spectra.

SIRIUS 3 requires high mass accuracy data. The mass deviation of your MS and MS/MS spectra should be within 20 ppm. Mass Spectrometry instruments like TOF, Orbitrap and FTICR usually provide high mass accuracy data, as well as coupled instruments like Q-TOF, IT-TOF or IT-Orbitrap. However, spectra measured with a quadrupole do not provide the high mass accuracy that is necessary for our method.

SIRIUS expects MS and MS/MS spectra as input. Although it is possible to omit the MS data, it will make the analysis much more time consuming and might give you worse results.

SIRIUS expects processed peak lists. It does not contain routines for peak picking from profiled spectra nor routines for merging spectra in an LC/MS run. There are several tools specialized for this task, e.g. XCMS or MZmine.

The main purpose of SIRIUS is to identify the molecular formula of the measured ion. Beside this, the software also annotates the spectrum providing a molecular formula for each fragment peak as well as detecting noise peaks. A fragmentation tree is predicted. This tree contains the predicted fragmentation reaction leading to the fragment peaks.

SIRIUS does not identify the (2D or 3D) structure of compounds, nor does it look up compounds in databases. There are other tools for this purpose, e.g. CSI:FingerId, MetFrag, CFM, and MAGMa.

SIRIUS can be used within an analysis pipeline. For example you can identify the molecular formula of the ion and the fragment peaks and use this information as input for other tools like FingerID or MAGMa to identify the 2D structure of the measured compound. For this purpose you can also use the SIRIUS library directly, instead of the command line interface. See SIRIUS Java Library.

Since Sirius 3.1 our software ships with an user interface. The user interface, however, have to be downloaded separately. If you want to use an user interface, use the file SiriusGUI.exe instead of sirius.exe.

Modelling the fragmentation process as tree comes with some flaws: Namely pull-ups and parallelograms. A pull-up is a fragment which is inserted too deep into the trees. Due to our combinatorial model SIRIUS will always try to generate very deep trees, claiming that there are many small fragmentation steps instead of few larger ones. SIRIUS will for example prefer three consecuting C2H2 losses to a single C6H6 loss. This does not affect the quality of the molecular formula identification. But when interpreting fragmentation trees you should keep this side-effect of the optimization in mind. Parallelograms are consecutive fragmentation processes that might happen in different orders. SIRIUS will always decide for one order of this fragmentation reactions, as this is the only valid way to model the fragmentation as tree.
1.1 Literature

1.1.1 Mass Decomposition

- Faster mass decomposition.
  Kai Dührkop, Marcus Ludwig, Marvin Meusel and Sebastian Böcker

- DECOMP – from interpreting Mass Spectrometry peaks to solving the Money Changing Problem
  Sebastian Böcker, Zsuzsanna Lipták, Marcel Martin, Anton Pervukhin, and Henner Sudek

- A Fast and Simple Algorithm for the Money Changing Problem
  Sebastian Böcker and Zsuzsanna Lipták

1.1.2 Isotope Pattern Analysis

- SIRIUS: decomposing isotope patterns for metabolite identification
  Sebastian Böcker, Matthias C. Letzel, Zsuzsanna Lipták and Anton Pervukhin
  Bioinformatics (2009) 25 (2): 218-224

1.1.3 Fragmentation Tree Computation

- Fragmentation trees reloaded.
  Kai Dührkop and Sebastian Böcker

- Speedy Colorful Subtrees.
  W. Timothy J. White, Stephan Beyer, Kai Dührkop, Markus Chimani and Sebastian Böcker

- Finding Maximum Colorful Subtrees in practice.
  Imran Rauf, Florian Rasche, François Nicolas and Sebastian Böcker

- Computing Fragmentation Trees from Tandem Mass Spectrometry Data
  Florian Rasche, Aleš Svatôš, Ravi Kumar Maddula, Christoph Böttcher, and Sebastian Böcker
  Analytical Chemistry (2011) 83 (4): 1243–1251

- Towards de novo identification of metabolites by analyzing tandem mass spectra
  Sebastian Böcker and Florian Rasche
2.1 User Interface

The SiriusGUI.exe should hopefully work out of the box. Just run the SiriusGUI.exe on Windows or Sirius.sh on Linux.

2.2 Windows

The sirius.exe as well as SiriusGUI.exe should hopefully work out of the box. To execute SIRIUS from every location you have to add the location of the sirius.exe to your PATH environment variable. This is not necessary for the user interface SiriusGUI.exe.

2.3 Linux and MacOSX

To execute SIRIUS from every location you have to add the location of the sirius executable to your PATH variable. Open the file ~/.bashrc in an editor and add the following line (replacing the placeholder path):

```bash
export PATH=$PATH:/path/to/sirius
```

SIRIUS need an ilp solver to analyze MS/MS data. You can install the free available GLPK solver, e.g. for Ubuntu:

```bash
sudo apt-get install libglpk libglpk-java
```

Alternatively, SIRIUS ships with the necessary binaries. You might have to add the location of sirius to your LD_LIBRARY_PATH variable (in linux) or to your DYLIB_LIBRARY_PATH variable (MacOsx). For example:

```bash
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/path/to/sirius
```

However, it might be that libglpk needs further dependencies, so installing GLPK via package manager is recommended.

2.4 Gurobi

SIRIUS ships with the GLPK solver which is fast enough in most cases. However, if you want to analyze large molecules and spectra with lot of peaks, you can greatly improve the running time by using a more efficient solver. Next go GLPK we also support the Gurobi solver. This is a commercial solver which offers a free academic licence

---

for university members. You can find the installation instruction for Gurobi on their website. SIRIUS will automatically use Gurobi als solver if the environment variables for the library path (PATH on windows, LD_LIBRARY_PATH on linux, DYLIB_LIBRARY_PATH on MacOSX) are set to the lib directory of your gurobi installation and if the environment variable GUROBI_HOME is set to your gurobi installation location. Gurobi will greatly improve the speed of the computation. Beside this there will be no differences in using Gurobi or GLPK.
CHAPTER
THREE

USER INTERFACE

With Sirius 3.1 our software ships with an user interface. Please be aware that the user interface have to be downloaded separately from the commandline tool. Take sure that you downloaded a file named SiriusGUI.exe.

3.1 Overview

On top of the screen you find the toolbar(1). On the left side is the experiment list (2) displaying all imported experiments. An experiment describes the set of MS and MS/MS spectra corresponding to a single measured compound. If an experiment is successfully computed you will see a tick mark on the right (3); if something goes wrong during computation you will see a cross symbol (4). The output of a computation is an ordered list of suggested molecular formula candidates. After selecting an experiment in the experiment list the candidate list (5) should display all molecular formula candidates together with their score. When selecting a candidate, the tree viewer (6) displays the computed fragmentation tree. Nodes in the tree are colored according to their score.
3.2 Data Import

Sirius offers two modes for data import: **Single Import** and **Batch Import**. The Single Import is triggered when clicking on the *Import* button in the toolbar. It allows you to import **one** experiment. We will use the term experiment as a description of MS and MS/MS spectra belonging to a single compound. The Single Import mode is recommended if your data consists of several csv (comma separated values) files (like the data from the CASMI challenges). First press on *Import* to start the import dialog.

For each spectrum you have to select the MS level (either MS 1 or MS 2). If you have MSn spectra you can just import them as MS 2 spectra. You can select a name for the experiment as well as an ionization mode. The collision energy is an optional attribute as it does not affect the computation.

You can import * .ms and .mgf* files using the **Batch Import**. In this mode Sirius will read all attributes (MS level, ionization, parentmass) directly from the file. You can, however, change these attributes afterwards by selecting the imported experiment and clicking on the *Edit* button.

See section *Supported Input Formats* for a description of the file formats * .ms and .mgf*.

3.2.1 Drag and Drop

Sirius supports Drag and Drop: Just move your input files into the application window. This is usually the easiest way to import data into Sirius. Supported file formats for Drag and Drop are *.csv, .ms, .sirius and .mgf*.  

3.2. Data Import
3.3 Computation

As for importing data Sirius offers two computation modes: **Single Computation** and **Batch Computation**. The Single Computation allows you to setup different parameters for each experiment. You can trigger it by right-clicking on an experiment and choosing **Compute** in the context menu. The Batch Computation will run in background and compute all experiments in the workspace.

3.3.1 parent mass

The exact m/z of the parent peak. If MS 1 data is present, the m/z of the monoisotopic peak is presented as default. Otherwise, an autocompletion offers a list of high intensive peaks from the MS 2 spectra.

3.3.2 elements besides CHNOPS

Sirius will use the elements carbon (C), hydrogen (H), nitrogen (N), oxygen (O), phosphorus (P) and sulfur (S) by default. Additional elements can be selected in the checkboxes. Adding elements that do not occur in the correct molecular formula of the compound will usually not lead to wrong results but to an increased running time of the algorithm.

3.3.3 other

The ionization mode determines the polarity of the measurement (positive or negative) as well as the adduct (e.g. protonation or sodium adduct). If you choose **Unknown Positive** or **Unknown Negative** Sirius will not care about the adduct, but report the molecular formula of the ion in the candidate list. Otherwise, Sirius will subtract the adducts formula from the ions formula and report neutral molecular formulas in the candidate list as well as in the fragmentation trees.

Choose either **Q-TOF**, **Orbitrap** or **FT-ICR** in the instrument field. The chosen instrument affects only very few parameters of the method (mainly the allowed mass deviation). If your instrument is not one of these three then just select the Q-TOF instrument.

You can change the maximal allowed mass deviation in the ppm field. Sirius will only consider molecular formulas which mass deviations are either below the chosen ppm or smaller than \( \frac{100 \cdot \text{ppm}_{\text{max}}}{10^6} \).
Finally, you can select the number of molecular formula candidates that should be given in the output. But Sirius will search through the space of all possible molecular formulas, anyways.

### 3.4 Visualization of the Results

The candidate list displays the best candidate molecular formulas ordered by score. Molecular formulas are always written in neutral form, except for compounds with unknown ionization mode. For every molecular formula the corresponding fragmentation tree is visualized in the „tree view“ tab. Alternatively, the „spectra view“ tab visualizes which peak is assigned to a fragment.

#### 3.4.1 Tree View

The tree view displays the estimated Fragmentation Tree. Each node in this tree assigns a molecular formula to a peak in the (merged) MS 2 spectrum. Each edge is a hypothetical fragmentation reaction. The user has the choice between different node styles and color schemes. The shown tree can be exported as JPEG, GIF, and PNG. Alternatively, the Dot file format contains only a description of the tree. It can be used to render the tree externally. The command-line tool Graphviz can transform dot files into image formats (PDF, SVG, PNG etc). The JSON format yields a machine-readable representation of the tree.
3.4. Visualization of the Results

![Graph showing molecular formulas and mass/charge distribution](image-url)
3.4.2 Spectrum View

In the spectrum view all peaks that are annotated by the Fragmentation Tree are colored in blue. Peaks that are annotated as noise are colored black. Hovering with the mouse over a peak shows its annotation.

3.5 Workspace

All imported experiments together with their results (the candidate molecular formulas and corresponding fragmentation trees) form the workspace. You can save the workspace into a file by clicking on Save Workspace in the toolbar. Analogously, you can load a workspace using Load Workspace. The workspace is saved in the .sirius format, which is simply a Zip Archive containing the input spectra and fragmentation trees. You can share the .sirius files with other people. But be aware that the input spectra are contained in these files!

The Sirius Commandline Tool is able to output .sirius files by using the option -O sirius. You can import these files with the Sirius User Interface to get a visualization of the results.

3.5.1 Export Results

Next to the .sirius format you can also export your results as a simple csv file by clicking on the Export Results button. Such a csv file contains the following fields:

- name of the experiment
- parent mass
- ionization
- for each molecular formula candidate there are two columns: one with the molecular formula and one with the corresponding score

If the number of molecular formula candidates differ between experiments, the number of fields per row might differ, too. However, most software programs do not have a problem with such malformed csv files.

3.6 Example Workflow

3.6.1 Working in Single Mode

1. Move the three files txt/chelidonine_ms.txt, txt/chelidonine_msms1.txt and chelidonine_msms2.txt from the demo data via Drag and Drop into the application window

2. The following dialog offers you to select the columns for mass and intensity values. Just press Ok as the default values are already correct.

3. You see the load dialog with three spectra. The first spectra is wrongly annotated as MS 2 spectrum but should be an MS 1 spectrum instead. Just select MS 1 in the drop down list labeled with ms level.

4. All other options are fine. However, you might want to choose a more memorizable name in the experiment name field.

5. Press the Ok button. The newly imported experiment should now appear in your experiment list on the left side.

6. Choose the experiment, right-click on it and press Compute.
7. In the compute dialog all options should be fine. Just check that the correct parent mass is chosen. You might want to add Chlorine or Fluorine to the set of considered elements. Furthermore, you can change the instrument type to Orbitrap.

8. Just look into the candidate list: The first molecular formula has a quite large score. Furthermore, the second molecular formula has a much lower score. This is a good indication that the identification is correct. However, you can take a look at the fragmentation tree: Do the peak annotation look correct? Take a look at the spectrum view: Are all high intensive peaks are explained?

9. You can now save the result list as csv file (by pressing the Export Results button). Maybe you want save your workspace, too. Just press the Save Workspace button.

1. Move the files Bicuculline.ms and Kaempferol.ms from the demo data via Drag and Drop into the application window.

2. The two experiments are now displayed in the experiment list.

3. Just check if the ionization and parent mass is correctly annotated. You can change this values by clicking on the experiment and then on Edit.

4. Click on the Compute All button.

5. You can now select the allowed elements, the instrument type as well as the maximal allowed mass deviation. Be aware that this settings will be used for all imported experiments.

6. Choose Orbitrap in the instrument field and press Ok.

7. A ... symbol occurs on the lower right corner of each experiment. This means that the experiment will be computed soon. A gear symbol tells you that this experiment is currently computed in background. A check mark appears in all experiments that were successfully computed, a red cross marks experiments which computation fails.

8. Probably you will not see anything than a check mark, as the computation is very fast. However, if you see a compound with a red cross you might want to compute it again in Single Mode. Check if the parent mass and ionization is correct.

9. Sometimes a computation might take a long time (e.g. for experiments with a lot of elements or very high masses). You can cancel the computation of a single experiment by selecting Cancel Computation in the right-click context menu. You can cancel the computation of all experiments by clicking on Cancel Computation in the toolbar.

3.6.2 Identifying a CASMI challenge


2. Move these files via Drag and Drop into the application window.

3. Change the ms level of the first file into Ms 1.

4. Click on Ok.

5. Click on Compute in the right-click context menu of the imported experiment.

6. Choose Q-TOF as instrument and press the Ok button.

7. C23H46NO7P should be suggested as number one hit in the candidate list.
The SIRIUS commandline tool can be either called via the binary by simply running the command `sirius` in your commandline. Alternatively, you can run the sirius jar file using java with the command:

```
java -jar sirius.jar
```

You can always use the `--help` option to get a documentation about the available commands and options. Assuming you want to analyze the example data given in the CASMI\(^1\) contest, you would execute the following on the commandline:

```
sirius -1 MSpos_Challenge0.txt -2 MSMSpos_Challenge0.txt
```

### 4.1 Supported Input Formats

#### 4.1.1 Mass Spectra

The input of SIRIUS are MS and MS/MS spectra as simple peak lists. SIRIUS can read csv files which contain on each line a m/z and an intensity value separated by either a whitespace, a comma or a TAB character. For example:

```
185.041199 4034.674316
203.052597 12382.624023
245.063171 50792.085938
275.073975 124088.046875
305.084106 441539.125
335.094238 4754.061035
347.094940 13674.210938
365.105103 55487.472656
```

The intensity values can be arbitrary floating point values. SIRIUS will transform the intensities into relative intensities, so only the ratio between the intensity values is important.

SIRIUS also supports the mgf (mascot generic format). This file format was developed for peptide spectra for the mascot search engine. Each spectrum in a mgf file can contain many spectra each starting with `BEGIN IONS` and ending with `END IONS`. Peaks are again written as pairs of m/z and intensity values separated by whitespaces with one peak per line. Further meta information can be given as NAME=VALUE pairs. SIRIUS recognizes the following meta information:

- **PEPMASS**: contains the measured mass of the ion (e.g. the parent peak)
- **CHARGE**: contains the charge of the ion. As SIRIUS supports only single charged ions, this value can be either 1+ or 1-.

---

\(^1\) [http://casmi-contest.org/2014/example/MSpos_Challenge0.txt](http://casmi-contest.org/2014/example/MSpos_Challenge0.txt)
• MSLEVEL: should be 1 for MS spectra and 2 for MS/MS spectra. SIRIUS will treat higher values automatically as MS/MS spectra, although, it might be that it supports MSn spectra in future versions.

This is an example for a mgf file:

BEGIN IONS
PEPMASS=438.32382
CHARGE=1+
MSLEVEL=2
185.041199 4034.674316
203.052597 12382.624023
245.063171 50792.085938
275.073975 124088.046875
305.084106 441539.125
335.094238 4754.061035
347.09494 13674.210938
365.105103 55487.472656
END IONS

See also the GNPS database for other examples of mgf files.

A disadvantage of these data formats is that they do not contain all information necessary for SIRIUS to perform the computation. Missing meta information have to be provided via the commandline. Therefore, SIRIUS supports also an own file format very similar to the mgf format above. The file ending of this format is .ms. Each file contains one measured compound (but arbitrary many spectra). Each line may contain a peak (given as m/z and intensity separated by a whitespace), meta information (starting with the > symbol followed by the information type, a whitespace and the value) or comments (starting with the # symbol). The following fields are recognized by SIRIUS:

• >compound: The name of the measured compound (or any placeholder). This field is mandatory.
• >parentmass: the mass of the parent peak
• >formula: The molecular formula of the compound. This information is helpful if you already know the correct molecular formula and just want to compute a tree or recalibrate the spectrum
• >ion: the ionization mode. See Ion Modes for the format of ion modes.
• >charge: is redundant if you already provided the ion mode. Otherwise, it gives the charge of the ion (1 or -1).
• >ms1: All peaks after this line are interpreted as MS peaks
• >ms2: All peaks after this line are interpreted as MS/MS peaks
• >collision: The same as >ms2 with the difference that you can provide a collision energy

An example for a .ms file:

>compound Gentiobiose
>formula C12H22O11
>ionization [M+Na]+
>parentmass 365.10544

>ms1
365.10543 85.63
366.10887 11.69
367.11041 2.67

>collision 20
185.041199 4034.674316
203.052597 12382.624023
245.063171 50792.085938

2: http://gnps.ucsd.edu/
Ion Modes

Whenever SIRIUS requires the ion mode, it should be given in the following format:

- \([M+ADDUCT]+\) for positive ions
- \([M+ADDUCT]-\) for negative ions
- \([M-ADDUCT]-\) for losses
- \([M]+\) for intrinsically charged compounds

ADDUCT is the molecular formula of the adduct. The most common ionization modes are \([M+H]+\), \([M+Na]+\), \([M-H]-\), \([M+Cl]-\). Currently, SIRIUS supports only single-charged compounds, so \([M+2H]2+\) is not valid. For intrinsic charged compounds \([M]+\) and \([M]-\) should be used.

Molecular Formulas

Molecular Formulas in SIRIUS must not contain brackets. So \(2(C2H2)\) is not a valid molecular formula. Write \(C4H4\) instead. Furthermore, all molecular formulas in SIRIUS are always neutral, there is no possibility to add a charge on a molecular formula (instead, charges are given separately). So \(CH3+\) is not a valid molecular formula. Write \(CH3\) instead and provide the charge separately via commandline option.

Chemical Alphabets

Whenever SIRIUS requires the chemical alphabet, you have to provide which elements should be considered and what is the maximum amount for each element. Chemical alphabets are written like molecular formulas. The maximum amount of an element is written in square brackets behind the element. If no square brackets are given, the element might occur arbitrary often. The standard alphabet is CHNOP[5]S, allowing the elements C, H, N O and S as well as up to five times the element P.

Identifying Molecular Formulas

The main purpose of SIRIUS is identifying the molecular formula of the measured ion. The syntax for this command is:

sirius [OPTIONS] -z <PARENTMASS> -i <IONIZATION> -l <MS FILE> -2 <MS/MS FILE>

Where MS FILE and MS/MS FILE are either csv or mgf files. If mgf files are used, you might omit the PARENTMASS option. If you omit the IONIZATION option, \([M+H]+\) is used as default. It is also possible to give a list of MS/MS files if you have several measurements of the same compound with different collision energies. SIRIUS will merge these MS/MS spectra into one spectrum.

If your input files are in .ms or .mgf format (containing MSLEVEL and PEPMASS meta information), you can omit the -1 and -2 flag. For example:

sirius [OPTIONS] demo-data/ms

SIRIUS will pick the meta information (parentmass, ionization etc.) from the .ms files in the given directory. This allows SIRIUS to run in batch mode (analyzing multiple compounds without starting a new jvm process every time).
SIRIUS will output a candidate list containing the rank, overall score, fragmentation pattern score, isotope pattern score, the number of explained peaks and the relative amount of explained intensity. See the following example output:

```
sirius -z 354.1347 -p orbitrap -1 demo-data/txt/chelidonine_ms.txt
-2 demo-data/txt/chelidonine_msms1.txt demo-data/txt/chelidonine_msms2.txt
```

1.) C20H19NO5 score: 33.17 tree: +27.48 iso: 5.69 peaks: 13 95.44 %
2.) C16H22N2O5P score: 32.35 tree: +26.77 iso: 5.58 peaks: 13 95.44 %
3.) C12H23N3O7S score: 24.62 tree: +24.62 iso: 0.00 peaks: 13 95.44 %
4.) C18H17N4O4 score: 23.28 tree: +23.28 iso: 0.00 peaks: 14 95.79 %
5.) C14H20N5O4P score: 21.61 tree: +21.61 iso: 0.00 peaks: 14 95.79 %

The overall score is the sum of the fragmentation pattern score and the isotope pattern score. If the isotope pattern score is negative, it is set to zero. If at least one isotope pattern score is greater than 10, the isotope pattern is considered to have good quality and only the candidates with best isotope pattern scores are selected for further fragmentation pattern analysis.

If you want to analyze spectra measured with Orbitrap or FTICR, you should specify the appropriated analysis profile. A profile is a set of configuration options and scoring functions SIRIUS will use for its analysis. For example, the Orbitrap and FTICR profiles having tighter constraints for the allowed mass deviation but do not rely so much on the intensity of isotope peaks. You can set the profile with the `-p <name>` option. By default, qtof is used as profile.

SIRIUS recognizes the following options:

- `-p <name>, --profile <name>`
  Specify the used analysis profile. Choose either qtof, orbitrap or fticr. By default, qtof is selected.

- `-o <dirname>, --output <dirname>`
  Specify the output directory. If given, SIRIUS will write the computed trees into this directory.

- `-O <format>, --format <format>`
  Specify the format of the output of the fragmentation trees. This can be either json (machine readable), dot (visualizable) or sirius (can be viewed with the Sirius User Interface).

- `-f [list of formulas], --formula [list of formulas]`
  Specify a list of candidate formulas (separated by whitespaces) that should be considered during analysis. This option is helpful if you performed a database search beforehand and only want to consider molecular formulas found in the database. It is recommendet to first consider all molecular formulas (and omit this option) and filter the candidate list afterwards. However, specifying a subset of molecular formulas with this option might greatly improve the speed of the analysis especially for large molecules.

- `-a, --annotate`
  If set, SIRIUS will write the annotated spectrum containing the explanations (molecular formulas) for all identified peaks in a csv file within the specified output directory.

- `-c <num>, --candidates <num>`
  The number of candidates in the output. By default, SIRIUS will only write the five best candidates.

- `-s <val>, --isotope <val>`
  This option specifies the way SIRIUS will handle the isotope patterns. If it is set to omit, SIRIUS will omit the isotope pattern analysis. If it is set to filter, SIRIUS will use the isotope pattern to select a subset of candidates before starting the fragmentation pattern analysis (this will improve the speed of the analysis). Only if it is set to score, SIRIUS will use it for filtering and scoring the candidates. The default setting is score.

- `-e <alphabet>, --elements <alphabet>`
  Specify the used chemical alphabet. See Chemical Alphabets. By default, CHNOP[5]S is used.

- `-i <ion>, --ion <ion>`
  Specify the used ionization. See Ion Modes. By default, [M+H]+ is used.
-z <mz>, --parentmass <mz>
Specify the parentmass of the input spectra. You have to give the exact measured value, not the selected ion mass.

-1 <file>, --ms1 <file>
Specify the file path to the MS spectrum of the measured compound.

-2 <file>, --ms2 <file>
Specify one or multiple file paths to the MS/MS spectra of the measured compound

--ppm-max <value>
Specify the allowed mass deviation of the fragment peaks in ppm. By default, Q-TOF instruments use 10 ppm and Orbitrap instruments use 5 ppm.

--auto-charge
If this option is set, SIRIUS will annotate the fragment peaks with ion formulas instead of neutral molecular formulas. Use this option if you do not know the correct ionization.

--no-recalibrate
If this option is set, SIRIUS will not recalibrate the spectrum during the analysis.

-h, --help
display help

See the following examples for running SIRIUS commandline tool:

```bash
sirius -p orbitrap -z 239.0315 -i [M+Na]+ -1 bergapten_ms.csv
-2 bergapten_msms1.csv bergapten_msms2.csv
-1 unknown_ms1.csv -2 unknown_ms2.csv
sirius -p qtof -z 215.035 -i 1- --auto-charge -2 unknown_ms2.csv
sirius -c 10 -o trees -O json msdir
sirius -f C6H12O6 C5H6N7O C7H16OS2 -i [M+H]+ -1 ms.csv -2 msms.csv
```

**Computing Fragmentation Trees**

If you already know the correct molecular formula and just want to compute a tree, you can specify a single molecular formula with the -f option. SIRIUS will then only compute a tree for this molecular formula. If your input data is in .ms format, the molecular formula might be already specified within the file. If a molecular formula is specified, the parentmass can be omitted. However, you still have to specify the ionization (except for default value [M+H]+):

```bash
sirius -f C20H19NO5 -2 demo-data/txt/chelidonine_msms2.txt demo-data/txt/chelidonine_msms2.txt
```

**Visualizing Fragmentation Trees**

SIRIUS supports three output formats for fragmentation trees: dot (graphviz format), json (machine readable format), and sirius (can be viewed with the Sirius User Interface). The commandline tool Graphviz ³ can transform dot files into image formats (pdf, svg, png etc.). After installing Graphviz you can display tree files as follows:

```bash
sirius -p orbitrap -f C20H17NO6 -o trees demo-data/ms/Bicuculline.ms
dot -Tpdf -O trees/Bicuculline.dot
```

This creates a file Bicuculline.dot.pdf (Fig.1). Remark that SIRIUS uses automatically the file name of the input spectrum to name the output file. You can specify another filename with the -o option (as long as only one tree is computed).

³ http://www.graphviz.org/

### 4.1. Supported Input Formats

16
sirius -p orbitrap -f C20H17NO6 -o compound.dot demo-data/ms/Bicuculline.ms dot -Tpdf -O compound.dot

Demo Data

You can download some sample spectra from the SIRIUS website at http://bio.informatik.uni-jena.de/sirius2/wp-content/uploads/2015/05/demo.zip

The demo-data contain examples for three different data formats readable by SIRIUS. The mgf folder contains an example for a mgf file containing a single compound with several MS/MS spectra measured on an Orbitrap instrument. SIRIUS recognizes that these MS/MS spectra belong to the same compound because they have the same parent mass. To analyze this compound, run:
sirius -p orbitrap demo-data/mgf/laudanosine.mgf

The output is:

1.) C21H27NO4 score: 25.41 tree: +17.55 iso: 7.86 peaks: 12 97.94 %
2.) C17H30N2O4P score: 21.46 tree: +13.97 iso: 7.49 peaks: 12 97.94 %
3.) C15H28N5O3P score: 15.00 tree: +15.00 iso: 0.00 peaks: 11 87.04 %
4.) C19H25N4O3 score: 14.66 tree: +14.66 iso: 0.00 peaks: 11 87.16 %
5.) C14H27N7O2S score: 13.69 tree: +13.69 iso: 0.00 peaks: 11 97.38 %

This is a ranking list of the top molecular formula candidates. The best candidate is C21H27NO4 with an overall score of 25.41. This score is the sum of the fragmentation pattern scoring (17.55) and the isotope pattern scoring (7.86). For the last three candidates, the isotope pattern scoring is 0. In fact, this score can never fall below zero. If all isotope pattern scores are zero, you can assume that the isotope pattern has very low quality and cannot be used to determine the molecular formula. If the isotope pattern score of the top candidate is over 10, it is assumed to be a high quality isotope pattern. In this case, the isotope pattern is also used to filter out unlikely candidates and speed up the analysis.

The last two columns contain the number of explained peaks in MS/MS spectrum as well as the relative amount of explained intensity. The last value should usually be over 80 % or even 90 %. If this value is very low, you either have strange high intensive noise in your spectrum or the allowed mass deviation might be too low to explain all the peaks.

If you want to look at the trees, you have to add the output option:
sirius -p orbitrap -o outputdir demo-data/mgf/laudanosine.mgf

Now, SIRIUS will write the computed trees into the outputdir directory. You can visualize these trees in pdf format using Graphviz:
dot -Tpdf -O outputdir/laudanosine_1_C21H27NO4.dot

This creates a pdf file outputdir/laudanosine_1_C21H27NO4.dot.pdf.

The directory ms contains two examples of the ms format. Each file contains a single compound measured with an Orbitrap instrument. To analyze this compound run:
sirius -p orbitrap -o outputdir demo-data/ms/Bicuculline.ms

As the ms file already contains the correct molecular formula, SIRIUS will directly compute the tree. For such cases (as well as when you specify exactly one molecular formula via -f option) you can also specify the concrete filename of the output file:
sirius -p orbitrap -o mycompound.dot demo-data/ms/Bicuculline.ms

If you want to enforce a molecular formula analysis and ranking (although the correct molecular formula is given within the file) you can specify the number of candidates with the -c option:

4.1. Supported Input Formats
Figure 4.1: The output of the dot program to visualize the computed fragmentation tree
sirius -p orbitrap -c 5 demo-data/ms/Bicuculline.ms

SIRIUS will now ignore the correct molecular formula in the file and output the 5 best candidates.

The txt folder contains simple peaklist files. Such file formats can be easily extracted from Excel spreadsheets. However, they do not contain meta information like the MS level and the parent mass. So you have to specify this information via commandline options:

sirius -p orbitrap -z 354.134704589844 -1 demo-data/txt/chelidonine_ms.txt
-2 demo-data/txt/chelidonine_msms1.txt demo-data/txt/chelidonine_msms2.txt

The demo data contain a clean MS spectrum (e.g. there is only one isotope pattern contained in the MS spectrum). In such cases, SIRIUS can infer the correct parent mass from the MS data (by simply using the monoisotopic mass of the isotope pattern as parent mass). So you can omit the -z option in this cases.
You can integrate the SIRIUS library in your Java project, either by using Maven \(^1\) or by including the jar file directly. The latter is not recommendet, as the SIRIUS jar contains also dependencies to other external libraries.

### 5.1 Maven Integration

Add the following repository to your pom file:

```xml
<distributionManagement>
  <repository>
    <id>bioinf-jena</id>
    <name>bioinf-jena-releases</name>
    <url>http://bio.informatik.uni-jena.de/artifactory/libs-releases-local</url>
  </repository>
</distributionManagement>
```

Now you can integrate SIRIUS in your project by adding the following dependency:

```xml
<dependency>
  <groupId>de.unijena.bioinf</groupId>
  <artifactId>SiriusCLI</artifactId>
  <version>3.0.0</version>
</dependency>
```

### 5.2 Main API

The main class in SIRIUS is `de.unijena.bioinf.sirius.Sirius`. It is basically a wrapper around the important functionalities of the library. Although there are special classes for all parts of the analysis pipeline it is recommended to only use the Sirius class as the API of all other classes might change in future releases. The Sirius class also provides factory methods for the most important data structures. Although, for many of this data structures you could also use their constructors directly, it is recommended to use the methods in the Sirius class.

```java
public class Sirius
```

The main class in SIRIUS. Provides the basic functionality of the method.

**Parameters**

- **profile** – the profile name. Can be one of ‘qtof’, ‘orbitrap’ or ‘fticr’. If omitted, the default profile (‘qtof’) is used.

\(^1\) https://maven.apache.org/
The main functions of SIRIUS are either identifying the molecular formula of a given MS/MS experiment or com-
puting a tree for a given molecular formula and MS/MS spectrum. The Sirius class provides two methods for this
purpose: identify and compute. The basic input type is an Ms2Experiment. It can be seen as a set of MS/MS spectra
derived from the same precursor as well as a MS spectrum containing this precursor peak. The output of Sirius is
an instance of IdentificationResult, containing the score and the corresponding fragmentation tree for the candidate
molecular formula.

5.2.1 Create Datastructures

Sirius provides the following functions to create the basic data structures:

public Spectrum<Peak> wrapSpectrum (double[] mz, double[] intensities)
Wraps an array of m/z values and and array of intensity values into a spectrum object that can be used by the
SIRIUS library. The resulting spectrum is a lightweight view on the array, so changes in the array are reflected
in the spectrum. The spectrum object itself is immutable.

Parameters
• mz – mass to charge ratios
• intensities – intensity values. Can be normalized or absolute values - SIRIUS will normalize
  them itself if necessary

Returns view on the arrays implementing the Spectrum interface

public Element getElement (String symbol)
Lookup the symbol in the periodic table and returns the corresponding Element object or null if no element with
this symbol exists.

Parameters
• symbol – symbol of the element, e.g. H for hydrogen or Cl for chlorine

Returns instance of Element class

public Ionization getIonization (String name)
Lookup the ionization name and returns the corresponding ionization object or null if no ionization with this
name is registered. The name of an ionization has the syntax [M+ADDUCT]CHARGE, for example [M+H]+
or [M-H]-.

Parameters
• name – name of the ionization

Returns Adduct instance

public Charge getCharge (int charge)
Charges are subclasses of Ionization. So they can be used everywhere as replacement for ionizations. A charge
is very similar to the [M]+ and [M]- ionizations. However, the difference is that [M]+ describes an intrinsically
charged compound where the Charge +1 describes an compound with unknown adduct.

Parameters
• charge – either 1 for positive or -1 for negative charges.

Returns a Charge instance which is also a subclass of Ionization

public Deviation getMassDeviation (int ppm, double abs)
 Creates a Deviation object that describes a mass deviation as maximum of a relative term (in ppm) and an
absolute term. Usually, mass accuracy is given as relative term in ppm, as measurement errors increase with
higher masses. However, for very small compounds (and fragments!) these relative values might overestimate
the mass accuracy. Therefore, an absolute value have to be given.

5.2. Main API
Parameters

- **ppm** – mass deviation as relative value (in ppm)
- **abs** – mass deviation as absolute value (m/z)

Returns Deviation object

MolecularFormula `parseFormula(String f)`

Parses a molecular formula from the given string

Parameters
- **f** – molecular formula (e.g. in Hill notation)

Returns immutable molecular formula object

public Ms2Experiment `getMs2Experiment(MolecularFormula formula, Ionization ion, Spectrum<Peak> ms1, Spectrum... ms2)`

public Ms2Experiment `getMs2Experiment(double parentmass, Ionization ion, Spectrum<Peak> ms1, Spectrum... ms2)`

Creates a Ms2Experiment object from the given MS and MS/MS spectra. A Ms2Experiment is NOT a single run or measurement, but a measurement of a concrete compound. So a MS spectrum might contain several Ms2Experiments. However, each MS/MS spectrum should have one precursor or parent mass. All MS/MS spectra with the same precursor together with the MS spectrum containing this precursor peak can be seen as one Ms2Experiment.

Parameters
- **formula** – neutral molecular formula of the compound
- **parentmass** – if neutral molecular formula is unknown, you have to provide the ion mass
- **ion** – ionization mode (can be an instance of Charge if the exact adduct is unknown)
- **ms1** – the MS spectrum containing the isotope pattern of the measured compound. Might be null
- **ms2** – a list of MS/MS spectra containing the fragmentation pattern of the measured compound

Returns a MS2Experiment instance, ready to be analyzed by SIRIUS

public FormulaConstraints `getFormulaConstraints(String constraints)`

Formula Constraints consist of a chemical alphabet (a subset of the periodic table, determining which elements might occur in the measured compounds) and upperbounds for each of this elements. A formula constraint can be given like a molecular formula. Upperbounds are written in square brackets or omitted, if any number of this element should be allowed.

Parameters
- **constraints** – string representation of the constraint, e.g. “CHNOP[5]S[20]”

Returns formula constraint object

public CloseableIterator<Ms2Experiment> `parseExperiment(File file)`

parses a file and return an iterator over all MS/MS experiments contained in this file. An experiment consists of all MS and MS/MS spectra belonging to one feature (=compound). Supported file formats are .ms and .mgf. The returned iterator supports the close method to close the input stream. The stream is closed automatically, after iterating the last element. However, it is recommended to use the try-close syntax from Java 7 to ensure a proper closing of the stream.

5.2. Main API 22
5.2.2 Provided Algorithms

List<IdentificationResult> **identify** (Ms2Experiment *uexperiment*, int *numberOfCandidates*, boolean *recalibrating*, IsotopePatternHandling *deisotope*, Set<MolecularFormula> *whiteList*)

Identify the molecular formula of the measured compound by combining an isotope pattern analysis on MS data with a fragmentation pattern analysis on MS/MS data.

**Parameters**

- *uexperiment* – input data
- *numberOfCandidates* – number of candidates to output
- *recalibrating* – true if spectra should be recalibrated during tree computation
- *deisotope* – set this to ‘omit’ to ignore isotope pattern, ‘filter’ to use it for selecting molecular formula candidates or ‘score’ to rerank the candidates according to their isotope pattern
- *whiteList* – restrict the analysis to this subset of molecular formulas. If this set is empty, consider all possible molecular formulas

**Returns** a list of identified molecular formulas together with their tree

public IdentificationResult **compute** (Ms2Experiment *experiment*, MolecularFormula *formula*, boolean *recalibrating*)

Compute a fragmentation tree for the given MS/MS data using the given neutral molecular formula as explanation for the measured compound.

**Parameters**

- *experiment* – input data
- *formula* – neutral molecular formula of the measured compound
- *recalibrating* – true if spectra should be recalibrated during tree computation

**Returns** A single instance of IdentificationResult containing the computed fragmentation tree

public List<MolecularFormula> **decompose** (double *mass*, Ionization *ion*, FormulaConstraints *constr*, Deviation *dev*)

Decomposes a mass and return a list of all molecular formulas which ionized mass is near the measured mass. The maximal distance between the neutral mass of the measured ion and the theoretical mass of the decomposed formula depends on the chosen profile. For qtof it is 10 ppm, for Orbitrap and FTICR it is 5 ppm.

**Parameters**

- *mass* – mass of the measured ion
- *ion* – ionization mode (might be a Charge, in which case the decomposer will enumerate the ion formulas instead of the neutral formulas)
- *constr* – the formula constraints, defining the allowed elements and their upperbounds
- *dev* – the allowed mass deviation of the measured ion from the theoretical ion masses

**Returns** list of molecular formulas which theoretical ion mass is near the given mass

public Spectrum<Peak> **simulateIsotopePattern** (MolecularFormula *compound*, Ionization *ion*)

Simulates an isotope pattern for the given molecular formula and the chosen ionization.

**Parameters**

- *compound* – neutral molecular formula
- *ion* – ionization mode (might be a Charge)
Returns spectrum containing the theoretical isotope pattern of this compound

### 5.2.3 Output Type

```java
public class IdentificationResult {
    // The compute and identify methods return instances of IdentificationResult. This class wraps a tree and its scores.
    // You can write the tree to a file using the writeTreeToFile method.
    public void writeTreeToFile(File target) {
        // Writes the tree into a file. The file format is determined by the file ending (either '.dot' or '.json')
        // Parameters
        // • target – file name
    }

    public void writeAnnotatedSpectrumToFile(File target) {
        // Writes the annotated spectrum into a csv file.
        // Parameters
        // • target – file name
    }
}
```
3.1

• Sirius User Interface
• new output type `-O sirius`. The .sirius format can be imported into the User Interface.
• Experimental support for in-source fragmentations and adducts

3.0.3

• fix crash when using GLPK solver

3.0.2

• fix bug: SIRIUS uses the old scoring system by default when `-p` parameter is not given
• fix some minor bugs

3.0.1

• if MS1 data is available, SIRIUS will now always use the parent peak from MS1 to decompose the parent ion, instead of using the peak from an MS/MS spectrum
• fix bugs in isotope pattern selection
• SIRIUS ships now with the correct version of the GLPK binary

3.0.0

• release version
Symbols

--auto-charge
  command line option, 16
--no-recalibrate
  command line option, 16
--ppm-max <value>
  command line option, 16
-1 <file>, --ms1 <file>
  command line option, 16
-2 <file>, --ms2 <file>
  command line option, 16
-O <format>, --format <format>
  command line option, 15
-a, --annotate
  command line option, 15
-c <num>, --candidates <num>
  command line option, 15
-e <alphabet>, --elements <alphabet>
  command line option, 15
-f [list of formulas], --formula [list of formulas]
  command line option, 15
-h, --help
  command line option, 16
-i <ion>, --ion <ion>
  command line option, 15
-o <dirname>, --output <dirname>
  command line option, 15
-p <name>, --profile <name>
  command line option, 15
-s <val>, --isotope <val>
  command line option, 15
-z <mz>, --parentmass <mz>
  command line option, 15

C
  command line option
    --auto-charge, 16
    --no-recalibrate, 16
    --ppm-max <value>, 16
    -1 <file>, --ms1 <file>, 16
    -2 <file>, --ms2 <file>, 16
    -O <format>, --format <format>, 15
-a, --annotate, 15
-c <num>, --candidates <num>, 15
-e <alphabet>, --elements <alphabet>, 15
-f [list of formulas], --formula [list of formulas], 15
-h, --help, 16
-i <ion>, --ion <ion>, 15
-o <dirname>, --output <dirname>, 15
-p <name>, --profile <name>, 15
-s <val>, --isotope <val>, 15
-z <mz>, --parentmass <mz>, 15

compute(Ms2Experiment, MolecularFormula, boolean)
  (Java method), 23

D
  de.unijena.bioinf.sirius (package), 20, 24
  decompose(double, Ionization, FormulaConstraints)
    (Java method), 23
  decompose(double, Ionization, FormulaConstraints, Deviation)
    (Java method), 23

G
  getCharge(int) (Java method), 21
  getElement(String) (Java method), 21
  getFormulaConstraints(String) (Java method), 22
  getIonization(String) (Java method), 21
  getMassDeviation(int, double) (Java method), 21
  getMs2Experiment(double, Ionization, Spectrum, Spectrum)
    (Java method), 22
  getMs2Experiment(MolecularFormula, Ionization, Spectrum, Spectrum)
    (Java method), 22

I
  IdentificationResult (Java class), 24
  identify(Ms2Experiment, int, boolean, IsotopePattern-Handling, Set)
    (Java method), 23

P
  parseExperiment(File) (Java method), 22
  parseFormula(String) (Java method), 22

S
  simulateIsotopePattern(MolecularFormula, Ionization)
    (Java method), 23
Sirius (Java class), 20

W

wrapSpectrum(double[], double[]) (Java method), 21
writeAnnotatedSpectrumToFile(File) (Java method), 24
writeTreeToFile(File) (Java method), 24