



SIRIUS Documentation

Release 4.0

**Sebastian Böcker, Kai Dührkop,
Markus Fleischauer, Marcus Ludwig**

February 23, 2018

This manual is currently being updated to describe all new features of SIRIUS 4.0.

CONTENTS

1	Introduction	1
1.1	Welcome	1
1.2	Literature	2
2	Installation	4
2.1	Windows	4
2.2	Linux and Mac OS X	5
2.3	Proxy servers	5
2.4	Installing Gurobi and CPLEX	6
2.5	Getting started	6
3	Prerequisites	9
3.1	Spectral quality	9
3.2	Monoisotopic masses	9
3.3	Theoretical masses of ions	10
3.4	Mass deviations	10
3.5	Molecular formulas	10
3.6	Fragmentation trees	12
3.7	Molecular fingerprints	13
3.8	Molecular structures	14
3.9	Training data	14
4	Graphical User Interface	16
4.1	Overview	16
4.2	Data import	17
4.3	Identifying molecular formulas with SIRIUS	17
4.4	Identifying molecular structure with CSI:FingerID	19
4.5	Visualization of the results	19
4.6	Workspace	23
4.7	Settings	24
4.8	Bug Reports	25
5	SIRIUS Commandline Tool	26
5.1	Supported Input Formats	26
6	SIRIUS Java Library	34
6.1	Maven Integration	34
6.2	Main API	34
7	Changelog	39

INTRODUCTION

1.1 Welcome

SIRIUS is a *Java* software 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, and uses CSI:FingerID as a web service to search in molecular structure databases.

SIRIUS requires **high mass accuracy** data. The mass deviation of your MS and MS/MS spectra should be within 20 ppm. Mass Spectrometry instruments such as TOF, Orbitrap and FT-ICR usually provide high mass accuracy data, as well as coupled instruments like Q-TOF, IT-TOF or IT-Orbitrap. Spectra measured with a quadrupole or linear trap do not provide the high mass accuracy that is required for our method. See Sec. 3.4 on what “mass accuracy” means in detail for SIRIUS.

SIRIUS expects **MS and MS/MS** spectra as input. It is possible to omit the MS data, but it will make the analysis more time consuming and might give you worse results. In this case, you should consider limiting the candidate molecular formulas to those found in PubChem.

SIRIUS expects **processed peak lists** (centroided spectra). It does not contain routines for peak picking from profiled spectra, nor routines for merging spectra in an LC/MS run. This is a deliberate design decision: We want you to use the best peak picking software out there — or alternatively, your favorite software. There are several tools specialized for this task, such as OpenMS, MZmine or XCMS. The Trinity workflow (see <https://www.youtube.com/watch?v=zDcY7iuvyQY>) produces picked peak lists that can directly be imported into SIRIUS.

SIRIUS will identify the molecular formula of the measured precursor ion, and will also annotate the spectrum by providing a molecular formula for each fragment peak. Peaks that receive no annotation are assumed to be noise peaks. Furthermore, a **fragmentation tree** is predicted; this tree contains the predicted fragmentation reaction leading to the fragment peaks.

SIRIUS uses CSI:FingerID to identify the structure of a compound by searching in a molecular structure database. Here, “structure” refers to “conformation”:

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 such as FingerID or MAGMA to identify the 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 version 3.1, our software ships with a **Graphical User Interface** (GUI). A version without GUI is available as separate download.

1.2 Literature

The *scientific development* behind SIRIUS and CSI:FingerID required numerous man-years of PhD students, postdocs and principal investigators; an educated guess would be roughly 30 man-years. This estimate does not include building the shiny Graphical User Interface that was introduced in version 3.1. But it is not the user interface or software development that does the work here; it is our scientific research that made SIRIUS and CSI:FingerID possible. It is understood that the work of 10 years cannot be described in a single paper.

Please cite all papers that you feel relevant for your work. Please do not cite this manual or the SIRIUS or CSI:FingerID website, but rather our scientific papers.

1.2.1 Searching in molecular structure databases

- Kai Dührkop, Huibin Shen, Marvin Meusel, Juho Rousu and Sebastian Böcker. **Searching molecular structure databases with tandem mass spectra using CSI:FingerID.** *Proc Natl Acad Sci U S A*, 112(41):12580–12585, 2015.
- Huibin Shen, Kai Dührkop, Sebastian Böcker and Juho Rousu. **Metabolite Identification through Multiple Kernel Learning on Fragmentation Trees.** *Bioinformatics*, 30(12):i157–i164, 2014. *Proc. of Intelligent Systems for Molecular Biology (ISMB 2014)*.

1.2.2 Fragmentation Tree Computation

- Sebastian Böcker and Kai Dührkop. **Fragmentation trees reloaded.** *J Cheminform*, 8:5, 2016.
- W. Timothy J. White, Stephan Beyer, Kai Dührkop, Markus Chimani and Sebastian Böcker. **Speedy Colorful Subtrees.** In *Proc. of Computing and Combinatorics Conference (COCOON 2015)*, volume 9198 of *Lect Notes Comput Sci*, pages 310–322. Springer, Berlin, 2015.
- Imran Rauf, Florian Rasche, François Nicolas and Sebastian Böcker. **Finding Maximum Colorful Subtrees in practice.** *J Comput Biol*, 20(4):1–11, 2013.
- Florian Rasche, Aleš Svatoš, Ravi Kumar Maddula, Christoph Böttcher and Sebastian Böcker. **Computing fragmentation trees from tandem mass spectrometry data.** *Anal Chem* 83(4):1243–1251, 2011.
- Sebastian Böcker and Florian Rasche. **Towards de novo identification of metabolites by analyzing tandem mass spectra.** *Bioinformatics* 24(16):i49–i55, 2008.

1.2.3 Isotope pattern analysis

- Sebastian Böcker, Matthias C. Letzel, Zsuzsanna Lipták and Anton Pervukhin. **SIRIUS: decomposing isotope patterns for metabolite identification.** *Bioinformatics* 25(2): 218–224, 2009.
- Sebastian Böcker, Matthias Letzel, Zsuzsanna Lipták and Anton Pervukhin. **Decomposing metabolomic isotope patterns.** In *Proc. of Workshop on Algorithms in Bioinformatics (WABI 2006)*, volume 4175 of *Lect Notes Comput Sci*, pages 12–23. Springer, Berlin, 2006.

1.2.4 Auto-detection of elements

- Marvin Meusel, Franziska Hufsky, Fabian Panter, Daniel Krug, Rolf Müller and Sebastian Böcker. **Predicting the presence of uncommon elements in unknown biomolecules from isotope patterns.** *Anal Chem*, 88(15):7556–7566, 2016.

1.2.5 Mass decomposition

- Kai Dührkop, Marcus Ludwig, Marvin Meusel and Sebastian Böcker. **Faster mass decomposition.** In *Proc. of Workshop on Algorithms in Bioinformatics (WABI 2013)*, volume 8126 of *Lect Notes Comput Sci*, pages 45–58. Springer, Berlin, 2013.
- Sebastian Böcker and Zsuzsanna Lipták. **A fast and simple algorithm for the Money Changing Problem.** *Algorithmica* 48(4):413–432, 2007.
- Sebastian Böcker and Zsuzsanna Lipták. **Efficient Mass Decomposition.** In *Proc. of ACM Symposium on Applied Computing (ACM SAC 2005)*, pages 151-157. ACM press, New York, 2005.

INSTALLATION

To install SIRIUS, you only have to extract the archive you have downloaded to an arbitrary directory where you have write permissions. To run SIRIUS, you need to have a **Java Runtime Environment (JRE) version 7 or higher** installed. **If you run a 64-bit operating system, you also have to install a 64-bit Java runtime environment!** If you have trouble installing SIRIUS, please let us know at sirius@uni-jena.de and we will see if we can help. If you find that our installation guide is incomplete, or if you have some tricks that you want to share with your fellow scientists, please let us know so we can include them in this manual.

Warning! All advice given here on how to get SIRIUS running on your system, is given without any warranty! If you are not sure what you are doing, you might want to contact someone who does. (Remember, last Friday in July is System Administrator Appreciation Day!)

2.1 Windows

Extract the archive to an arbitrary directory where you have write permissions, such as `C:\SIRIUS`.

To check if you have a 64-bit Windows operating system, open the Windows settings, go to “System”, then “Info”. You will see a line “System type” that tells you if your operating system is 32- or 64-bit. To check if your Java version is 64-bit, open a Command Prompt (press the Windows key, then start typing “command prompt”, start the Command Prompt) and type `java -version`. If Windows complains that the java command is unknown, type `cd C:\ProgramData\Oracle\Java\javapath`, then `java -version`. In the last line of the output, you will see if you have 64-bit Java installed. If this is not the case, but your Windows is 64-bit, you have to install the correct Java version from <https://java.com/>. Press “Download”, but *do not use the big red button to download Java!* Instead, go to “See all Java Downloads”, then choose “Windows 64-bit”.

Go to the SIRIUS directory. Run `sirius-gui.exe` to start the graphical user interface. You might want to create a link on your desktop: Click and drag the file `sirius-gui.exe` to the desktop, keeping the ALT key pressed. You can rename the link on your desktop as you like. You start SIRIUS by double-clicking this link.

Run `sirius-console.exe` for the SIRIUS command line tool. To execute the SIRIUS command line tool from every location on your system, you have to add the location of the `sirius-console.exe` to your PATH environment variable: Open the Windows Setting, type “advanced” in the search window, say “yes” if Windows asks you. Press the “Environment Variables” button, select the “Path” variable in the lower panel, press “Edit...”, press “New”, enter the full directory path of SIRIUS, press RETURN. Close the Command Prompt, open a new one, type `sirius-console-64.exe`.

Frequently asked questions

- “Is running Java a security problem?” → Only if you enable execution of Java programs in your web browser. Today, there is practically no need to run Java from the browser, so you should disable this option (it is probably disabled already). Always keep your JRE up-to-date.

- “I am running Windows and I get an error message, ‘ERROR: JAVA_HOME is set to an invalid directory’.” → Probably, deleting the JAVA_HOME environment variable will solve the problem, see the instructions for the PATH variable above.
- “I want to run SIRIUS on Windows XP!” → Don’t. Running Windows XP on a computer connected to the internet is a security hazzard.

2.2 Linux and Mac OS X

For Mac OS X, make sure that you have installed JAVA version 8 or higher from <https://java.com/>; Apple only provides the deprecated version 6.

Extract SIRIUS into a folder of your choice, for example `~/sirius`.

To start the SIRIUS command line tool you then have to enter:

```
~/sirius/bin/sirius
```

To start the graphical user interface of SIRIUS:

```
~/sirius/bin/sirius-gui
```

To execute SIRIUS from every location you have to add `/path/to/sirius/bin` to your PATH variable. To do so, open `~/.bashrc` in an editor and add the following line (replacing the placeholder path):

```
export PATH=$PATH:~/sirius/bin/
```

Note that you have to reopen your “bash” shell to make the changes effective.

Frequently asked questions

- “Max OS X ships with Java, do I still have to install a Java version?” → Yes. SIRIUS does not work with Apple Java 6, the latest Java version OS X installed and updated by default. Later OS X version do not contain an Apple Java, but instead suggest installing Oracle Java.
- “I am running Mac OS X with version before 10.7.3; how can I install Java 7?” → Java 7 installation is not possible for MAC OS X versions before 10.7.3. You need to upgrade to OS X 10.7.3 or higher.

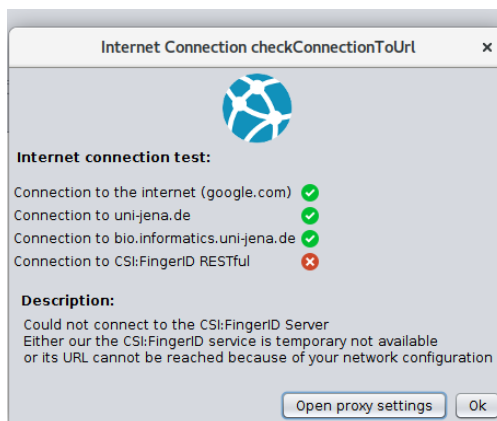
2.3 Proxy servers

To use database related functionality of Sirius, it needs an Internet connection. You have to ensure that Sirius is not blocked by any security software on your computer.

If you have to use a proxy server to connect to the Internet, Sirius automatically uses the system wide Java proxy configuration¹ if available. Alternatively you can specify the proxy configuration in the Sirius user interface setting (see Section 4.7).

If Sirius cannot connect to the Internet, it will report on which stage the error occurred.

¹ https://www.java.com/de/download/help/proxy_setup.xml



2.4 Installing Gurobi and CPLEX

SIRIUS ships with the GLPK (GNU Linear Programming Kit) ² Integer Linear Program solver which allows us to swiftly compute fragmentation trees in most cases. However, if you want to analyze large molecules and/or spectra with many peaks and/or a large number of spectra, you can greatly improve running time by using a faster solver. SIRIUS also supports Integer Linear Program solvers Gurobi ³ and CPLEX ⁴. These are commercial solvers which offer a free academic license for university members. You can find installation instruction on their websites. Using Gurobi or CPLEX will greatly improve the speed of fragmentation tree computations, which is the most time-intensive step of the computational analysis. Beside this, there will be no differences in using Gurobi, CPLEX or GLPK. To use Gurobi set the environment variable `GUROBI_HOME` to a valid Gurobi installation location e.g. `/path/to/gurobi750/linux64`. Similarly, to use CPLEX set `CPLEX_HOME` to `/path/to/CPLEX_Studio/cplex/lib`. SIRIUS will automatically use Gurobi or CPLEX as its solver if corresponding environment variables are specified. You can manually set the preferred ILP solvers in the settings dialog (GUI).

2.5 Getting started

2.5.1 Working in single mode

SIRIUS's "Single mode" corresponds to analyzing a single compound with one or more mass spectra.

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/MS* spectrum but should be an *MSI* 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 memorable name in the `<compound name>` field.
5. Press the `<OK>` button. The newly imported compound should now appear in your compound list on the left side.

² <https://www.gnu.org/software/glpk/>

³ <http://www.gurobi.com/>

⁴ <https://www.ibm.com/products/ilog-cplex-optimization-studio>

6. Choose the compound, 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 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.

2.5.2 Working in batch mode

SIRIUS's "Batch mode" corresponds to analyzing many compounds at once, each having one or more mass spectra.

1. Move the files *Bicuculline.ms* and *Kaempferol.ms* from the demo data via Drag and Drop into the application window
2. The two compounds are now displayed in the compound list
3. Just check if the ionization and parent mass is correctly annotated. You can change this values by clicking on the compound 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 compounds
6. Choose *Orbitrap* in the instrument field and press *OK*
7. A *<...>* symbol occurs on the lower right corner of each compound. This means that the compound will be computed soon. A gear symbol tells you that this compound is currently computed in background. A check mark appears in all compounds that were successfully computed, a red cross marks compounds 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 compounds with a lot of elements or very high masses). You can cancel the computation of a single compound by selecting *Cancel Computation* in the right-click context menu. You can cancel the computation of all compounds by clicking on *Cancel Computation* in the toolbar.

2.5.3 Identifying a CASMI challenge

1. Download the files http://casmi-contest.org/2014/Challenge2014/Challenge1/1_MS.txt and http://casmi-contest.org/2014/Challenge2014/Challenge1/1_MSMS.txt
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 compound
6. Choose *Q-TOF* as instrument and press the *OK* button

7. *C23H46NO7P* should be suggested as number one hit in the candidate list

PREREQUISITES

3.1 Spectral quality

SIRIUS and CSI:FingerID have been trained on a wide variety of data, including data from different instrument types. Nevertheless, certain aspects of the mass spectra are important so that our software can successfully process your data:

- Be reminded that SIRIUS requires **high mass accuracy** data: The mass deviation should be within 20 ppm. We are confident that SIRIUS can also give useful information for worse mass accuracy (say, 50 ppm), but you should know what you are doing if you are processing such data.
- It is understood that some molecules generate more fragments, whereas others have sparse fragmentation spectra. But it is also important to understand that without sufficient information, it is impossible to deduce the structure or even the molecular formula from a tandem mass spectrum that contains almost no peaks. For example, three peaks in a fragmentation spectrum measured with 1 ppm mass accuracy contain about 60 bit of information, ignoring dependencies between fragments and distribution of molecular masses. With this information, it is simply not possible to find the correct structure in a database such as PubChem, containing 100 million structures. In comparison, ten peaks measured with 20 ppm mass accuracy contain about 156 bit of information, again ignoring dependencies and distributions. To this end, we ask you to provide **rich fragmentation spectra** to SIRIUS, meaning that you **must not noise-filter** these spectra, or let the peak picking/centroiding software do that for you. At present, SIRIUS considers up to 60 peaks in the fragmentation spectrum, and decides for itself which of these peaks are considered noise.
- You will find that CSI:FingerID can sometimes identify the correct structure although the fragmentation spectrum is (almost) empty — do not get fooled, this is often nothing but lucky guessing. If you know how to structurally elucidate a compound based on an empty spectrum, please contact us and tell us how.
- You may have heard that peaks in a MS/MS spectrum with high mass carry more information than peaks with low mass: This is a misunderstanding. For example, if CSI:FingerID has to differentiate between 10000 candidates with identical molecular formula, then observing a fragment corresponding to an H₂O loss is in fact very uninformative. To this end, **do not set up your instrument to favor peaks of large masses**, sacrificing those with smaller masses.
- Some instrument types (e.g., time-of-flight) suffer from detectors that can run into saturation; saturated peaks can have mass differences much larger than those expected for other peaks. Unfortunately, most peak picking software do not mark such peaks as “misshaped”. To this end, it is possible that the most intense peak in a spectrum is not explained, as its mass deviation is extremely high.

3.2 Monoisotopic masses

The monoisotopic mass of a molecule (or ion) is formally defined as “the sum of masses of the atoms in a molecule (or ion) using the unbound, ground-state, rest mass of the most abundant isotope for each element.” Using this definition, the monoisotopic mass is usually not the most abundant isotopologue of the molecule (e.g., peptides and proteins),

it is often not resolved from other isotopologue peaks, and it may be undetectable in an MS experiment as it has intensity below noise level. In particular, given the isotope pattern of an unknown molecule, it is generally impossible to determine which of the peaks correspond to the monoisotopic peak. In total, this definition is not very practical.

Many researchers that work on the simulation and interpretation of isotope patterns have therefor introduced a slightly different and more practical definition of the monoisotopic mass of a molecule, see for example Dittwald *et al.* [1] and Meusel *et al.* [2]: Here, the isotopologue of a molecule where each atom is the isotope with the lowest nominal mass (according to the natural isotope distribution of elements) is referred to as *monoisotopic*. This definition has the advantages that the monoisotopic mass of a molecule is always the sum of monoisotopic masses of the atoms, which can be defined analogously; the monoisotopic peak is in all cases the first peak of the ideal isotope pattern; and, the monoisotopic (isotopologue) peak is always resolved from all other isotopologue peaks, even at unit mass accuracy. Clearly, the monoisotopic peak of a molecule may again be undetectable in an MS experiments.

SIRIUS uses the second, more practical definition of “monoisotopic”. This results in notable differences only for molecules that contain “uncommon elements” such as boron or selenium.

3.3 Theoretical masses of ions

There are different ways of computing the mass of an ionized molecule such as $C_6H_7O^+$ or $C_6H_6ONa^+$ that will result in slightly different results: in particular, adding the mass of a proton vs. subtracting the mass of an electron. Following suggestions by Ferrer & Thurman [3], SIRIUS computes this mass by *subtracting the rest mass of an electron*. To this end, the monoisotopic mass of $C_6H_7O^+$ is the monoisotopic mass of the molecule C_6H_7O (95.049690 Da) minus the rest mass of an electron (0.000549 Da), which totals as 95.049141 Da. Similarly, the monoisotopic mass of $C_6H_6ONa^+$ equals 117.031634 Da - 0.000549 Da = 117.031085 Da.

Above, masses have been rounded to six decimals; internally, SIRIUS uses double precision¹ for representing masses. Masses of isotopes are taken from the AME2016 atomic mass evaluation [4]. See Table 3.1 for the isotope masses and abundances used by SIRIUS, again rounded to six decimals for presentation.

We suggest to calibrate your instrument with ion masses as calculated above. In any case, you should *be aware of this tiny mass difference*, as this can result in unexpected behavior when decomposing masses; see for example Pluskal *et al.*, *Anal Chem* 84:4396–4403.

3.4 Mass deviations

SIRIUS assumes that mass deviations (the difference between the measured mass and the theoretical mass of the ion) are normally distributed [5–7]. The user-defined parameter “mass accuracy” is given in parts-per-million (ppm). SIRIUS interpretes this parameter as a “guarantee” and, hence, assumes that **this is the maximum allowed mass deviation**; it will **discard** all explanations that require a larger mass deviation. This implies that **if in doubt, you should use a larger mass accuracy** to ensure that SIRIUS can successfully annotate peaks in the spectrum. For masses below 200 Da, we use the absolute mass deviation at 200 Da, as we found that small masses vary according to an absolute rather than a relative error.

3.5 Molecular formulas

Unless instructed otherwise, SIRIUS will consider all molecular formulas that are chemically feasible and explain the precursor mass of the molecule/ion: For example, if your query compound is pinensin A ($C_{96}H_{139}N_{27}O_{30}S_2$, monoisotopic mass 2213.962 Da) then SIRIUS will consider all 19 746 670 candidate molecular formulas that explain this monoisotopic mass (assuming set of elements CHNOPS, see below, and 10 ppm mass accuracy). SIRIUS penalizes

¹ https://en.wikipedia.org/wiki/Double-precision_floating-point_format

Table 3.1: Isotopes with masses and abundances as used by SIRIUS. In this table, *masses have been rounded to six decimals* for the purpose of presentation; internally, SIRIUS uses masses with higher precision. ‘AN’ is atomic number. *Isotope abundances of boron can vary strongly, so isotope pattern analysis is of little use for identifying the correct molecular formula in case boron is present.

element (symbol)	AN	isotope	abundance	mass (Da)
hydrogen (H)	1	¹ H	99.988 %	1.007825
		² H	0.012 %	2.014102
boron (B)	5	¹⁰ B	19.9* %	10.012937
		¹¹ B	80.1* %	11.009305
carbon (C)	6	¹² C	98.93 %	12.0
		¹³ C	1.07 %	13.003355
nitrogen (N)	7	¹⁴ N	99.636 %	14.003074
		¹⁵ N	0.364 %	15.001090
oxygen (O)	8	¹⁶ O	99.757 %	15.994915
		¹⁷ O	0.038 %	16.999131
		¹⁸ O	0.205 %	17.999160
fluorine (F)	9	¹⁸ F	100 %	18.000938
silicon (Si)	14	²⁸ Si	92.223 %	27.976927
		²⁹ Si	4.685 %	28.976495
		³⁰ Si	3.092 %	29.973770
phosphor (P)	15	³¹ P	100 %	30.973762
sulfur (S)	16	³² S	94.99 %	31.972071
		³³ S	0.75 %	32.971459
		³⁴ S	4.25 %	33.967867
		³⁶ S	0.01 %	35.967081
chlorine (Cl)	17	³⁵ Cl	75.76 %	34.968853
		³⁷ Cl	24.24 %	36.965903
selenium (Se)	34	⁷⁴ Se	0.89 %	73.922476
		⁷⁶ Se	9.37 %	75.919214
		⁷⁷ Se	7.63 %	76.919914
		⁷⁸ Se	23.77 %	77.917309
		⁸⁰ Se	49.61 %	79.916521
bromine (Br)	35	⁸² Se	8.73 %	81.916699
		⁷⁹ Br	50.69 %	78.918337
		⁸¹ Br	49.31 %	80.916291
iodine (I)	53	¹²⁷ I	100 %	126.904473

candidate molecular formulas that deviate too strongly of what we assume a molecular formula of a biomolecule to look like (for example, $C_2H_2N_{12}O_{12}$ will receive a penalty), but this penalty is used cautiously: Only 2.6 % of the molecular formulas of all PubChem compounds — and, hence, only a tiny fraction of molecular formulas from compounds not marked as biomolecules — are penalized. Molecular formulas are never rewarded by SIRIUS.

SIRIUS uses a short list of outlier molecular formulas which would be penalized by the above method, as they are not “biomolecule-like”; these molecular formulas are not penalized, as they have been observed in metabolomics experiments (for example, as solvents), but are also not rewarded. These outlier molecular formulas will be considered as candidates by SIRIUS even if they violate elemental constraints such as “at most 2 fluorine”.

Considering all molecular formulas implies that a set of elements has to be provided from which these molecular formulas are generated. SIRIUS includes methods for the auto-detection of elements from the isotope and fragmentation pattern of the query compound [2].

In case your compound is large (above 600 Da) or you have incomplete information (no isotope pattern), you can restrict SIRIUS to only consider molecular formulas found in PubChem. Doing so, it is not possible to ever detect molecules with a novel molecular formula, though.

Recent evaluations (for example, as part of the CASMI contest²) indicated that one can determine molecular formulas by searching in a structure database using tool such as MAGMa, CFM-ID or CSI:FingerID. (Somewhat consequently, the CASMI 2016 contest did no longer have a category for molecular formula identification.) **We strongly advise against doing so, as this is apparently a hidden prior problem.** Molecular formulas in a structure database are far from being uniformly distributed: For a certain precursor mass plus mass inaccuracy, you may find 90% molecular structures with only one molecular formula. Assuming that the molecular structures in our evaluation set are uniformly chosen at random, even a method that uniformly draws a molecular structure, then reports its molecular formula will get 81% correct identifications for the molecular formula identification task. A method that still ignores the mass spectrometry data beyond the monoisotopic mass, but reports the majority vote in the structure database would even reach 90% correct identifications. (This is comparable to an app for bird identification that always outputs “It is a house sparrow!” for Britain, because the house sparrow is the most common bird in Britain. This app will get a lot of correct identifications.) Clearly, computational methods such as MAGMa, CFM-ID or CSI:FingerID are not random, but they will fall for this pit, too. To this end, we advise for the “classical chemical identification pipeline” where **molecular formulas are identified first, then molecular structure.**

3.6 Fragmentation trees

Fragmentation trees annotate the fragmentation spectrum with molecular formulas, and identify likely losses between the ions in the fragmentation spectrum. Fragmentation trees can be used both to identify the molecular formula of a query compound, and to derive information about its fragmentation: For example, this is used in CSI:FingerID to predict the molecular fingerprint of the query compound. Fragmentation trees are computed directly from the fragmentation spectrum, and do not use or require any spectral libraries or molecular structure databases (for the subtle “exemptions” from this rule, see Böcker & Dührkop [7]). Fragmentation trees are computed by combinatorial optimization; the underlying optimization problem constitutes a Maximum A Posterior Estimator. The optimization problem (finding a maximum colorful subtree) is NP-hard but nevertheless solved optimally, explaining why computations sometimes require significant running time for large molecules with rich fragmentation spectra.

If you think that the computation process should be further speeded up, we ask you to cite our papers on swiftly computing fragmentation trees [8, 9], as this would give us an incentive to continue our work on this topic: We stress that the current version of SIRIUS is **many million times faster** than the initial version [10]. In fact, this initial version could not process more than 15 peaks in the fragmentation spectrum, due to exploding running times *and* memory requirements.

Modeling the fragmentation process as a tree comes with two restrictions: Namely, “pull-ups” and “parallelograms”. A **pull-up** is a fragment which is inserted too deep into the trees. Due to our combinatorial optimization, SIRIUS will try to generate deep trees, assuming that there are many small fragmentation steps instead of few larger ones. SIRIUS

² <http://casmi-contest.org/>

will, for example, prefer three consecutive C_2H_2 losses to a single C_6H_6 loss. This does not affect the quality of the molecular formula identification; but when interpreting fragmentation trees, you should keep in mind this side effect of the combinatorial optimization. **Parallelograms** are consecutive fragmentation processes that happen in more than one order: For example, the precursor ion loses H_2O then CO_2 , *but also* CO_2 then H_2O . SIRIUS will always decide for one order of such fragmentation reactions, as this is the only valid way to model the fragmentation as a tree.

At present, SIRIUS does not make use of isotope patterns in the fragmentation spectra: Usually, SIRIUS will simply flag these peaks and ignore them in the optimization process. This will likely change in an upcoming version of SIRIUS.

3.7 Molecular fingerprints

Molecular fingerprints can be used to encode the structure of a molecule: Most commonly, these are binary vector of fixed length where each bit describes the presence or absence of a particular, fixed *molecular property*, usually the existence of a certain substructure. As an example, consider PubChem CACTVS fingerprints with length 881 bits: Molecular property 121 encodes the presence of at least one “unsaturated non-aromatic heteroatom-containing ring size 3”. Most bits are just explained via their SMARTS (SMiles ARbitrary Target Specification) string³: For example, molecular property 357 of PubChem CACTVS encodes SMARTS string “[#6] (~[#6]) (:c) (:n)”, corresponding to a central carbon atom connected to a second carbon atom via any bond, to a third aromatic carbon atom via an aromatic bond, and to an aromatic nitrogen atom via an aromatic bond. See ftp://ftp.ncbi.nlm.nih.gov/pubchem/specifications/pubchem_fingerprints.pdf for the full description of the CACTVS fingerprint. We ignore all molecular properties that can be derived from the molecular formula of the query compound (for example, bits 0 to 114 of PubChem CACTVS).

Given the molecular structure of a compound, we can deterministically compute its molecular fingerprint: We use the Chemical Development Kit CDK [11–13] for this purpose. Heinonen *et al.* [14] pioneered the idea of predicting a complete molecular fingerprint from the fragmentation spectrum of a query compound: Before this, only few, usually hand-selected properties (presence or absence of certain substructures) were predicted from fragmentation spectra, in particular for GC-MS with Electron Ionization; see Curry & Rumelhart [15] for an excellent example.

Given the fragmentation spectrum and fragmentation tree of a query compound, CSI:FingerID predicts its molecular fingerprint using Machine Learning (linear Support Vector Machines), see Shen *et al.* [16] and Dührkop *et al.* [17] for the technical details. CSI:FingerID does not predict a single fingerprint type but instead, five of them: Namely, CDK Substructure fingerprints, PubChem CACTVS fingerprints, Klekota-Roth fingerprints [18], FP3 fingerprints, and MACCS fingerprints. In addition, CSI:FingerID predicts ECFP2 and ECFP4 fingerprints [19] that appear sufficiently often in the training data. Different from other fingerprints, ECFP are not encoded via SMARTS matching; instead, a hash function encodes the neighborhood of each atom in the molecule. In principle, these fingerprints can encode $2^{32} \approx 4.2 \cdot 10^9$ different substructures (molecular properties); in practice, it is possible but very unlikely that two substructures share the same value, due to a hash collision.

CSI:FingerID predicts only those molecular properties that showed reasonable prediction quality in cross validation (F_1 at least 0.25, see below). In total, 3 215 molecular properties are predicted by CSI:FingerID 1.1.

CSI:FingerID does not only predict if some molecular property is zero (absent) or one (present); it also provides an **estimate how sure it is about this prediction**. Mathematically speaking, we estimate the posterior probability that the molecular property is present: Estimates close to one indicate that CSI:FingerID is rather sure that the molecular property is present; similarly, estimates close to zero for an absent molecular property; whereas estimates between 0.1 and 0.9 hint towards an unsure situation. Posterior probabilities are estimated using a method by Platt [20], so we also refer to these estimates as “Platt probabilities”. **But even if CSI:FingerID is 99 % sure that a molecular property is present, this does not mean that it is indeed present!** CSI:FingerID predicts thousands of molecular properties, and 10 out of 1000 predictions should be incorrect at this level of accuracy. Furthermore, estimation parameters were derived from the training data, and if your query molecule structures are very different from those in the training data, it is rather likely that some estimates are imprecise. In addition to Platt probabilities, we also report the performance

³ https://en.wikipedia.org/wiki/Smiles_arbitrary_target_specification

of each molecular property classifier in cross validation: The F_1 score⁴ is the harmonic mean⁵ of precision (fraction of retrieved instances that are relevant) and recall (fraction of relevant instances that are retrieved).⁶ Molecular properties that have a classifier with F_1 score close to one, are more trustworthy than those with F_1 score close to zero; again, this has to be treated with some care, as these measures were estimated from the training data using cross validation.

It is important to understand that the predicted molecular fingerprint which is returned by the CSI:FingerID web service, has *per se* no connections to any structures in any molecular structure database. That means that **even if the correct molecular structure is not contained in any structure database, the predicted fingerprint is still valid** within the prediction power of the method. For example, you can use it to hypothesize about the structure of an “unknown unknown” not present in any structure database. We have added a tab in the Graphical User Interface that allows you to examine the predicted molecular fingerprint.

3.8 Molecular structures

By default, SIRIUS searches in either PubChem or a biomolecule structure database; in addition, SIRIUS now offers to search in your own “suspect database”.

- When searching *PubChem*, we use a local copy of the database where we have precomputed all molecular fingerprints, as computing the fingerprints of the candidates “on the fly” is too time-consuming. We are sporadically updating our local copy of PubChem. You can lookup the date of the latest database update in the database dialog.
- The *biomolecule structure database* is an amalgamation of several structure databases that contain biomolecules (metabolites and other compounds of biological relevance; molecules that are products of nature, or synthetic products with potential bioactivity). Currently, this biomolecule structure database consists of KNApSAcK [21], HMDB [22], ChEBI [23], KEGG [24], HSDB [25], MaConDa [26], BioCyc [27], UNPD [28], a subset of biomolecules from ZINC [29], all structures from GNPS [30] and MassBank [31], and MeSH-annotated compounds from PubChem [32, 33].

3.9 Training data

The fragmentation tree computation of SIRIUS **is not trained on any data**, since no machine learning is used for this step. The parameters for fragmentation tree computation were estimated from two MS/MS spectra datasets, with 2005 compounds from GNPS [30] and 2046 compounds from Agilent (“MassHunter Forensics/Toxicology PCDL” version B.04.01 from Agilent Technologies Inc., Santa Clara, CA, USA). Parameters of this step were not optimized to maximize, say, the molecular formula identification rate, and estimates should be very robust. All spectra were recorded in positive ion mode. Fragmentation tree computation and molecular formula estimation appear to work very well for negative ion mode data, too; but there is no guarantee for that.

The Machine Learning part of CSI:FingerID, namely the ensemble of linear Support Vector Machines, is currently (CSI:FingerID version 1.1) trained on 12 108 compounds from [30] and 2 073 compounds from MassBank [31]. In total, CSI:FingerID is trained on 14 181 compounds with 8 210 unique structures. Again, all spectra were recorded in positive ion mode. Different from fragmentation tree computation, we assume that the **prediction of molecular fingerprints is challenging for negative ion mode data**, given that we have no training data. Hopefully, enough negative ion mode data will become available in the near future, so that we can retrain CSI:FingerID on this data and solve this issue for good.

We would like to explicitly and emphatically thank everyone who made their spectra publically available. With that, you have done a huge favor not only to us, but to everyone in the metabolomics community; which, unfortunately, is not recognized by the community at the moment. We sincerely hope that the metabolomics community will become

⁴https://en.wikipedia.org/wiki/F1_score

⁵https://en.wikipedia.org/wiki/Harmonic_mean

⁶https://en.wikipedia.org/wiki/Precision_and_recall

aware of the urgent need for open data and data sharing in the near future (just like the genomics community did 25 years ago, or the proteomics community 10 year ago); and that you will receive your well-deserved accolades then.

We are constantly adding new training data that becomes publically available. If you have data from reference compounds, we ask you to upload these to a public database such as GNPS or MassBank; if this is not possible for some reason, **you can contact us so that we can add your data to the CSI:FingerID training data without making it publically available.** Please help us improve the performance of CSI:FingerID by providing additional training data!

GRAPHICAL USER INTERFACE

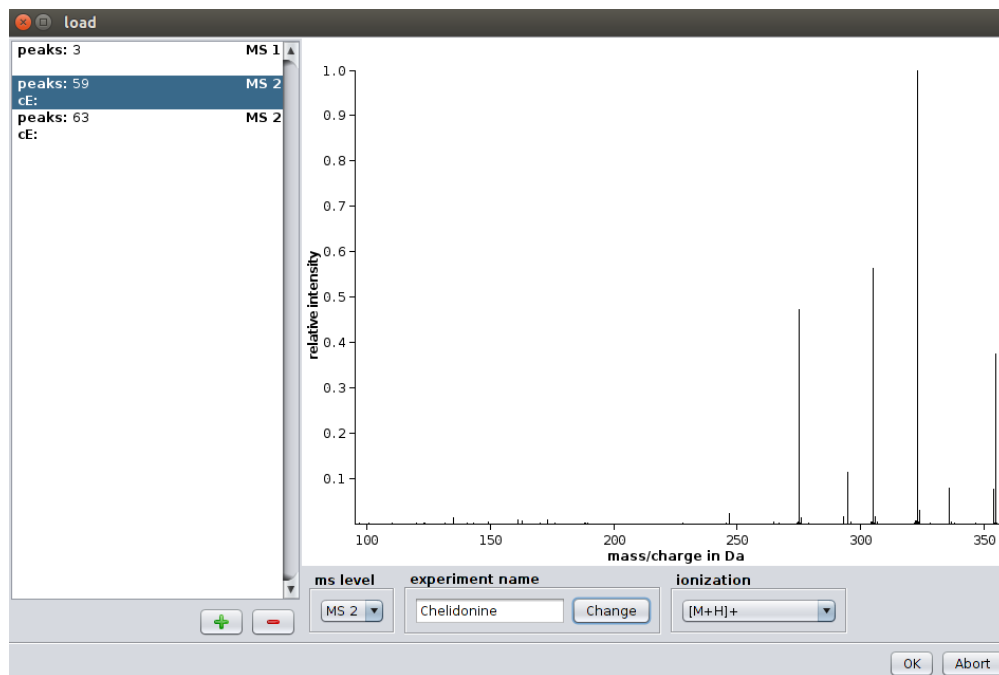


4.1 Overview

Starting with version 3.1, our software ships with a Graphical User Interface. On top of the screen you find the toolbar (1). On the left side is the compound list (2) displaying all imported compounds. Each <compound> lists MS and MS/MS spectra corresponding to a single measured compound. If a compound has been processed successfully, 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 a compound an overview is displayed. It shows a list of all molecular formula candidates (5), sorted by score, the corresponding spectrum (6) and the fragmentation tree of the selected candidate molecular formula (7). Explained peaks are highlighted in the spectrum. Nodes in the fragmentation tree are colored according to their score. In the upper right corner are settings and bug report dialogs (8).

4.2 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** compound. (We will use the term “compound” 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, such as 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 MS1 or MS/MS). If you have MS_n spectra you can just import them as MS/MS spectra. You can select a name for the compound 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, parent mass) directly from the file. You can, however, change these attributes afterward by selecting the imported compound and clicking on the *Edit* button.

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

4.2.1 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*.

4.3 Identifying molecular formulas with SIRIUS

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 compound. You can trigger it by right-clicking on a compound and choosing *Compute* in the context menu. The Batch Computation will compute all compounds in the workspace. Besides, you can select multiple compounds and choose *Compute* to only compute a subset of your imported compounds.

4.3.1 Parent mass

The exact m/z of the parent peak. If MS1 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/MS spectra.

4.3.2 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 within the *«Select elements»* dialog. Adding additional elements will increase running time. Using (too many) elements that do not occur in the correct molecular formula of the compound might worsen the results.

The automated detection of a set of “uncommon elements” is available if the isotope pattern is provided. These elements are sulfur (S), chlorine (Cl), bromine (Br), boron (B), and selenium (Se). Using *«Auto detect»* will clear your element selection and will set new values based on the detection. Autodetection is usually quite sensitive and rather overpredicts the actual quantity of an element.

4.3.3 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 below the chosen ppm; for masses below 200 Da, the allowed mass deviation is $200 \cdot \frac{ppm_{max}}{10^6}$.

Finally, you can select the number of molecular formula candidates that should be reported in the output, and what molecular formulas are considered as candidates: If you select option *«all possible molecular formulas»* then SIRIUS will enumerate over all molecular formulas that match the ion mass, filtering out only molecular formulas with negative ring double bond equivalent. If you choose *«all PubChem formulas»* then SIRIUS will select all molecular formulas from PubChem. Option *«organic PubChem formulas»* ignores molecular formulas containing elements untypical for organic compounds such as Si or Mg; molecular formulas pass this filter if they are composed solely of CHNOPSBBrcIIF. When choosing *«formulas from biomolecule databases»*, SIRIUS will use all formulas contained in databases with biological compounds or compounds that could be expected in biological experiments (e.g. KEGG, BioCyc, HMDB, but also MaConDa).

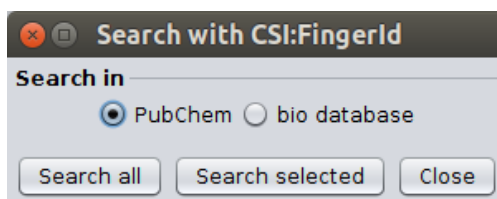
Please consider the following:

- We never search in these databases directly, but rather in our local database copies. Although we regularly update our database, it may happen that some new compound in, say, ChEBI is not already contained in our local copy.
- When choosing a *«molecular formulas from a database»* option, SIRIUS will ignore your element restrictions and instead allow all elements.

- We do not recommend to restrict molecular formula searching to biomolecule databases, but doing so significantly speeds up computations, as SIRIUS has to consider significantly less molecular formulas and download significantly smaller candidate structure lists.

4.4 Identifying molecular structure with CSI:FingerID

After computing the fragmentation trees you can search these in a structure database. Again we provide a *single mode* and a *batch mode*. The single mode is available by clicking on the molecular formula of interest, then switching to the *CSI:FingerID* tab and pressing on the *Search online with CSI:FingerID* button. The batch mode can be triggered by pressing on the *CSI:FingerID* in the toolbar.



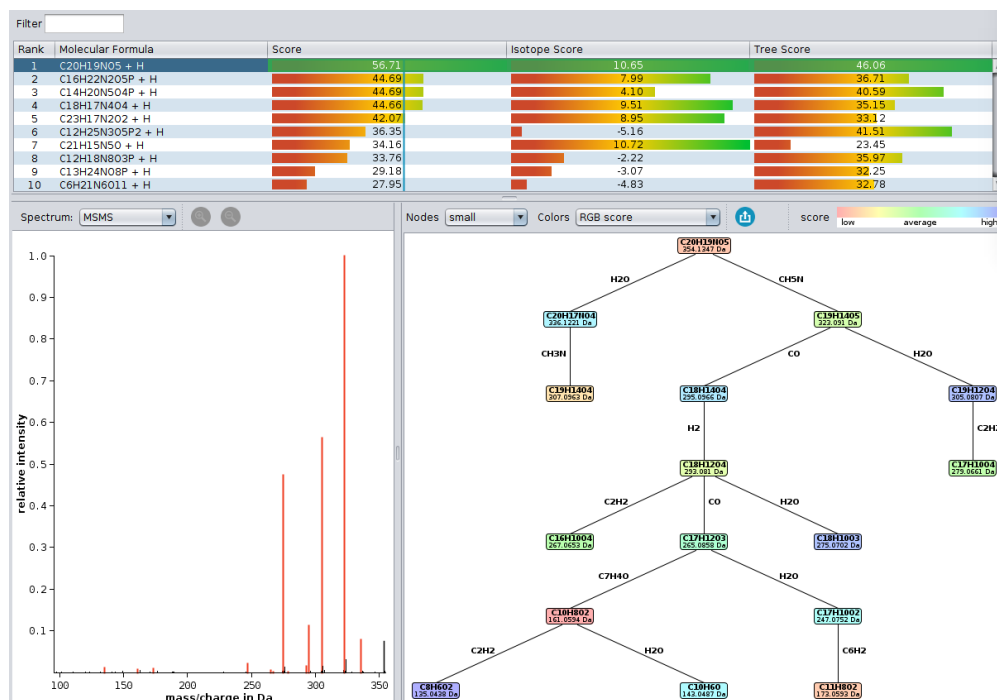
When starting the CSI:FingerID search you are again asked to choose between PubChem or biomolecule databases. This is mainly a performance issue because you can filter your result lists afterwards by any database you want to. Our biomolecule database is several magnitudes smaller than PubChem and downloading and searching structure lists from biomolecule databases is significantly faster. However, when searching in biomolecule databases you might never see if there are structures with possibly much better score from PubChem. Therefore, we recommend to search in PubChem and filter the result list if you expect the result to be contained in biomolecule databases.

4.5 Visualization of the results

Each compound has an *Overview* panel to display the most important information. The candidate list contains the best candidate molecular formulas ordered by score. Molecular formulas are always written in neutral form, except for compounds with unknown ionization mode. For the selected molecular formula candidates the *Spectra view* visualizes which peak is assigned to a fragment. The corresponding fragmentation tree is visualized in the *Tree view*. Both views can be displayed in a separate panel to have a more detailed look. The *CSI:FingerID* panel displays candidates from structure prediction.

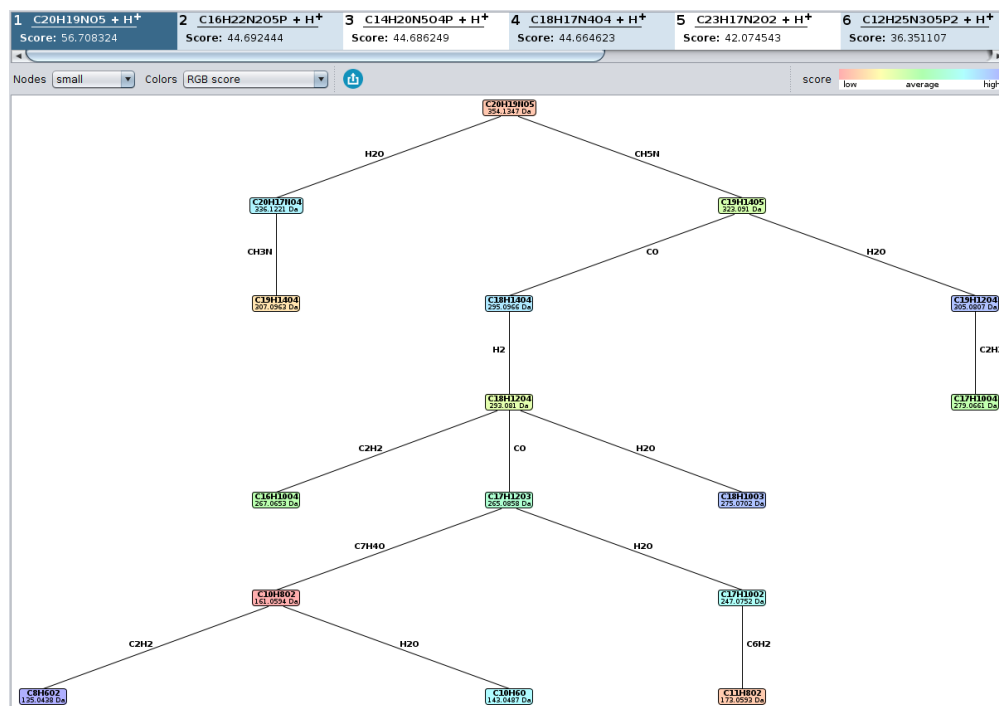
4.5.1 Overview tab

The *Overview* tab displays the candidate list, spectrum and fragmentation tree of the selected candidate. Candidates are ordered by total score, but can be sorted by any other column. Moreover, the list can be filtered using the corresponding text field. A green row highlights the molecular formula of the best candidate structure found by CSI:FingerID.



The length of the bars for the different score columns (isotope pattern, fragmentation pattern, total) as well as the displayed numbers for columns *Isotope Score* and *Tree Score*, correspond to *logarithms* of maximum likelihoods (probability that this hypothesis, i.e. molecular formula, will generate the observed data). In contrast, the number in the *Score* column is the posterior probability of the hypothesis (molecular formula), and these probabilities sum to one. A higher posterior probability of the top hit may indicate that this molecular formula has a higher chance of being correct; but we stress that a **posterior probability of 90 %, must not be misunderstood as a 90 % probability that this molecular formula identification is correct!** The displayed probabilities are neither q-values nor Posterior Error Probabilities.

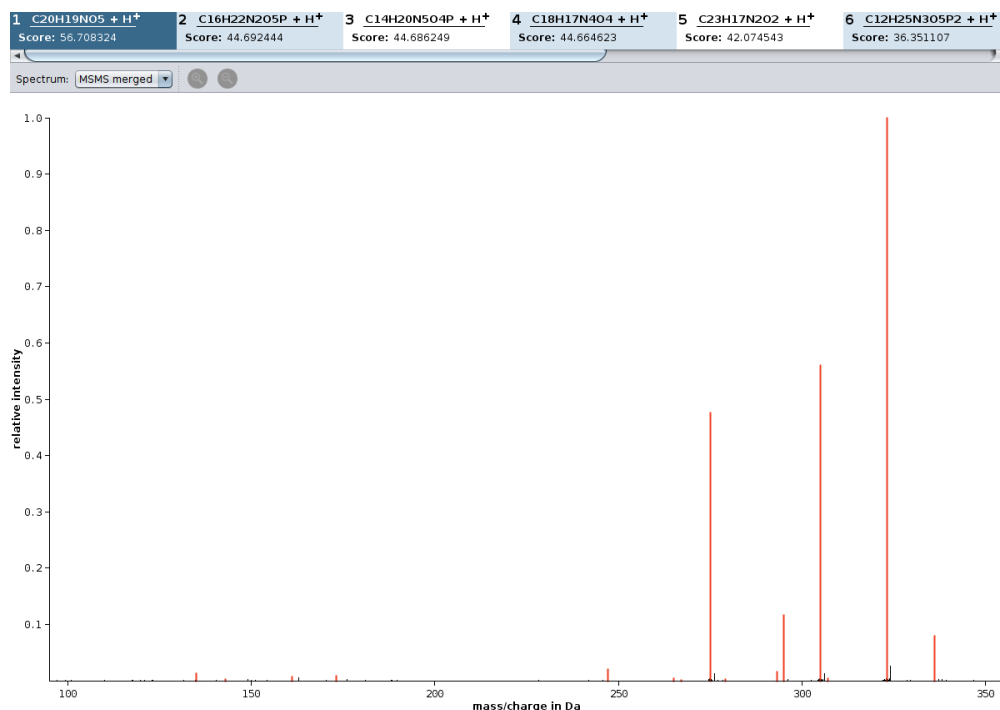
4.5.2 Tree view tab



The *<Tree view>* tab displays the estimated fragmentation tree. Each node in this tree assigns a molecular formula to a peak in the (merged) MS/MS spectrum. Each edge is a hypothetical fragmentation reaction. The user has the choice between different node styles and color schemes.

The displayed fragmentation tree can be exported as JPEG, GIF, and PNG. Alternatively, the Dot file format contains a text 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.

4.5.3 Spectrum view tab



In the *<Spectrum view>* tab, all peaks that are annotated by the fragmentation tree are colored in orange. Peaks that are annotated as noise are colored black. Hovering with the mouse over a peak shows its detailed annotation.

4.5.4 CSI:FingerID view

This tab shows you the candidate structures for the selected molecular formula ordered by the CSI:FingerID search score. If you want to filter the candidate list by a certain database (e.g. only compounds from KEGG and BioCyc) you can press the filter button. A menu will open displaying all available databases. Only candidates will be displayed that are enabled in this filter menu. Note that PubChem is enabled by default and, therefore, always the complete list is shown. If you want to see only compounds from KEGG and BioCyc you have to disable PubChem and enable KEGG and BioCyc.

Another way of filtering is the XLogP slider. If you have information about retention times and expected logP values of your measured compound you can use this slider to filter the candidate list by certain XLogP values. The slider

allows you to define min and max values. XLogP is calculated using the Chemical Development Kit CDK [11–13].

The blue and red squares are some visualization of the CSI:FingerID predictions and scoring. All blue squares represent molecular structures that are found in the candidate structure and are predicted by CSI:FingerID to be present in the measured compound. The more intense the color of the square the higher is the predicted probability for the presence of this substructure. The larger the square the more reliable is the predictor. The red squares, however, represent structures that are predicted to be absent but are, nevertheless, found in the candidate structure. Again, as more intense the square as higher the predicted probability that this structure should be absent. Therefore, a lot of large intense blue squares and as few as possible large intense red squares are a good indication for a correct prediction.

When hovering with the mouse over these squares the corresponding description of the molecular structure (usually a SMART expression) is displayed. When clicking on one of these squares, the corresponding atoms in the molecule that belong to this substructure are highlighted. If the substructure matches several times in the molecule, it is once highlighted in dark blue while all other matches are highlighted in a translucent blue.

Even if the correct structure is not found by CSI:FingerID — in particular if the correct structure is not contained in any database — you can get information about the structure by looking at the predicted structures: When clicking on the large light green squares you see which molecular substructures are expected in the measured compound.

You can open a context menu by right click on the compound. It offers you to open the compound in PubChem or copy the InChI or InChI key in your clipboard.

If the compound is contained in any biomolecule database, a blue label with the name of this database is displayed below the compound. You can click on most of these labels to open the database entry in your browser window.

You can export a single candidate list by clicking on the `<export list>` button.

4.6 Workspace

All imported compounds 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 Graphical User Interface to get a visualization of the results.

Currently, CSI:FingerID predictions are not stored in the `.sirius` file. This will hopefully change in future releases.

4.6.1 Export results

You can export your results as a simple `csv` file for all compounds by clicking on the `<Export Results>` button and selecting single file export. Such a `.csv` file can then be imported into *Excel*, *Matlab* or *R*.

The CSV file contains the following fields:

- name of the compound
- 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 compounds, the number of fields per row might differ, too. However, most software programs do not have a problem with such *malformed* CSV files.

Selecting the *multiple files* option will create one file per compound. Exporting the *Tree results* will give you a `.csv` file for each compound containing the molecular formula candidates with the following fields:

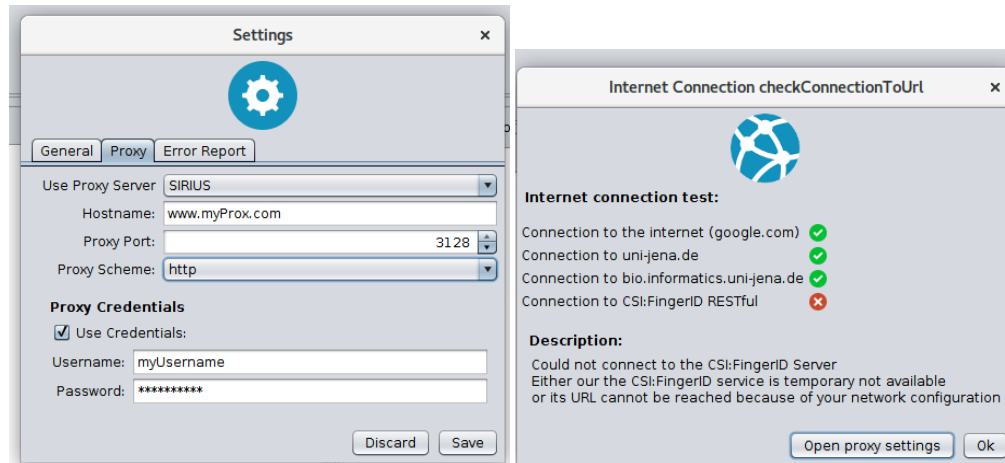
- molecular formula
- rank
- overall score
- fragmentation tree score
- isotope pattern score
- number of peaks which were annotated with a fragment molecular formula
- ratio of explained intensity. It is the sum of intensities of all annotated peaks divided by the total intensity.

Exporting *CSI:FingerID results* will create one `.csv` file for each compound with the following fields:

- InChI key's first 14 positions (no stereochemistry information)
- InChI
- molecular formula
- rank
- score
- name of the candidate compound (if available)
- SMILES
- XlogP
- PubChem identifier
- occurrence in additional databases (if available)

4.7 Settings

- *General settings*
 - *Allowed solvers:* chose the ILP solver for SIRIUS to use for fragmentation tree computation. GLPK is free, Gurobi is commercial but offers free academic license.
 - *Database cache:* location of cache directory. CSI:FingerID download candidate structures from our server and caches them for faster retrieval.
- *Proxy settings*
 - Sirius support three different kinds of proxy configuration SYSTEM, SIRIUS and NONE. If SYSTEM (default) is selected Sirius uses the system wide Java proxy settings. If SIRIUS is selected it uses the configuration you have specified in the Settings -> Proxy panel. If NONE is selected Sirius ignores all proxy settings.
 - Edit the information in the Settings -> Proxy panel if you want to address CSI:FingerID via a proxy server. Your specified configuration will be tested if you hit the save button (see Figure below).



- *«Error report settings»*
 - Add an email address which will be sent with a bug report. This makes it possible for us to contact you, in case we need additional information to solve your problem.
 - Decide whether specific hardware and operating system information is send with your bug report.

4.8 Bug Reports

We do our best so that you will not be confronted with errors while using SIRIUS. But we cannot test every possible scenario. We encourage you to send us a bug report in case you encounter an error. It is very helpful if you specify your email address. Often, errors are very specific and can only be reproduced and understood with help of the input file and knowledge of the used parameter settings. Therefore, we might reach out to you. With your help, we will continue to improve SIRIUS. You can also contact us at sirius@uni-jena.de.

SIRIUS COMMANDLINE TOOL

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
```

5.1 Supported Input Formats

5.1.1 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
347.09494 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

- **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 MS_n 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 GNPS² 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 fragmentation 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
```

² <http://gnps.ucsd.edu/>

```
245.063171 50792.085938
275.073975 124088.046875
305.084106 441539.125
335.094238 4754.061035
347.09494 13674.210938
365.105103 55487.472656
```

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. Hence, $2(C_2H_2)$ is not a valid molecular formula; write C_4H_4 instead. Furthermore, all molecular formulas in SIRIUS are always neutral, and there is no possibility to add a charge on a molecular formula (instead, charges are given separately). Hence, CH_3^+ is not a valid molecular formula. Write CH_3 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> -1 <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 FT-ICR, you should specify the appropriate analysis profile. A profile is a set of configuration options and scoring functions SIRIUS will use for its analysis. For example, the Orbitrap and FT-ICR 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 fragmentation 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` (for vizualization) or `sirius` (can be viewed with the SIRIUS Graphical 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 recommended 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.
- fingerprintid**
If this option is set, SIRIUS will search for molecular structure using CSI:FingerID after determining the molecular formula
- database** , **-d**
If this option is set, SIRIUS will only consider molecular formulas from the given database. Either PubChem or bio can be chosen, as well as other concrete databases like hmdb, kegg, knapsack or biocyc. When used with the option `--fingerprintid`, the chosen database will also be used to search for candidate structures. When setting this value to `all`, SIRIUS will consider all possible molecular formulas (without database search) and CSI:FingerID will use PubChem.
- fingerprintid-db**
With this option you can specify the structure database to search in with CSI:FingerID. You only have to set this option if you want to have different settings for searching molecular formulas and structures. For example, it might be common to search in the space of all possible molecular formulas (using `-d all`) but restrict the structures to a user specified database.
- h** , **--help**
display help

See the following examples for running SIRIUS commandline tool:

```
sirius -p orbitrap -z 239.0315 -i [M+Na]+ -1 bergapten_ms.csv
                -2 bergapten_msms1.csv bergapten_msms2.csv
sirius -p fticr -z 215.0350 -i [M-H]- -e CHNOPSCl[2] -c 10 -s omit
                -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 fragmentation 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 parent mass can be omitted. However, you still have to specify the ionization (except for default value `[M+H]+`):

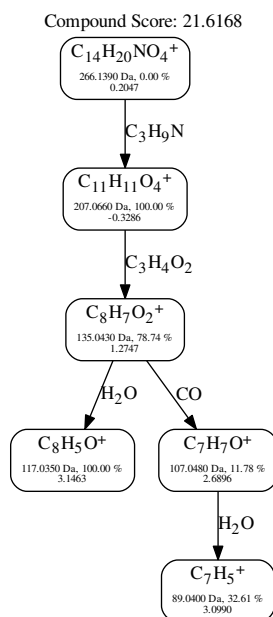
```
sirius -f C20H19NO5 -2 demo-data/txt/chelidonine_msms2.txt demo-data/txt/chelidonine_
↪msms2.txt
```

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 Graphical User Interface). The commandline tool Graphviz³ can transform dot files into image formats (PDF, SVG, PNG etc.). After installing Graphviz you can display tree files as follows:

```
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`:



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 fragmentation tree is computed.

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

³ <http://www.graphviz.org/>

Identifying Molecular Structures

You can also search for molecular structures with CSI:FingerID. Just add the option `--fingerid` to trigger a CSI:FingerID search after fragmentation tree computation. With `--database` can now also specify the database SIRIUS should search in. Available are `pubchem` and `bio`. You can also specify certain databases like `code` and `code`; however, we strongly recommended to instead filter the output list afterwards.

SIRIUS will generate CSV files for each input spectrum containing a ordered candidate list of structures with the CSI:FingerID score. Furthermore, another result CSV file is generated only the top candidates from all input spectra ordered by their confidence.

```
sirius -c 10 --database=pubchem --fingerid demo-data/ms/Bicuculline.ms
```

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 contain 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.) C ₂₁ H ₂₇ NO ₄	score: 25.41	tree: +17.55	iso: 7.86	peaks: 12	↳
↳ 97.94 %					
2.) C ₁₇ H ₃₀ N ₂ O ₄ P	score: 21.46	tree: +13.97	iso: 7.49	peaks: 12	↳
↳ 97.94 %					
3.) C ₁₅ H ₂₈ N ₅ O ₃ P	score: 15.00	tree: +15.00	iso: 0.00	peaks: 11	↳
↳ 87.04 %					
4.) C ₁₉ H ₂₅ N ₄ O ₃	score: 14.66	tree: +14.66	iso: 0.00	peaks: 11	↳
↳ 87.16 %					
5.) C ₁₄ H ₂₇ N ₇ O ₂ S	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 a 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 fragmentation trees, you have to add the output option:

```
sirius -p orbitrap -o outputdir demo-data/mgf/laudanosine.mgf
```

Now, SIRIUS will write the computed fragmentation trees into the `outputdir` directory. You can visualize this 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 fragmentation 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:

```
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 -l demo-data/txt/chelidone_ms.txt  
-2 demo-data/txt/chelidone_msms1.txt demo-data/txt/chelidone_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.

SIRIUS JAVA LIBRARY

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 recommended, as the SIRIUS jar contains also dependencies to other external libraries.

6.1 Maven Integration

Add the following repository to your pom file:

```
<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:

```
<dependency>
  <groupId>de.unijena.bioinf</groupId>
  <artifactId>sirius_api</artifactId>
  <version>3.5.0</version>
</dependency>
```

6.2 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.

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 compound, or computing a fragmentation 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.

6.2.1 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 an 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 a 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 has to be given.

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 on 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 upper bounds for each of this elements. A formula constraint can be given like a molecular formula. Upper bounds 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 compounds contained in this file. a compound 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.

6.2.2 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 fragmentation 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 fragmentation trees

```
public IdentificationResult compute ( Ms2Experiment experiment, MolecularFormula formula, boolean re-
                                     calibrating)
```

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 fragmentation tree computation

Returns A single instance of IdentificationResult containing the computed fragmentation tree

```
public List<MolecularFormula> decompose ( double mass, Ionization ion, FormulaConstraints constr, Devi-
                                     ation dev)
```

```
public List<MolecularFormula> decompose ( double mass, Ionization ion, FormulaConstraints constr)
```

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

6.2.3 Output type

public class **IdentificationResult**

The compute and identify methods return instances of IdentificationResult. This class wraps a fragmentation tree and its scores. You can write the tree to a file using the writeTreeToFile method.

public void **writeTreeToFile** (File *target*)

Writes the fragmentation 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

CHANGELOG

4.0

- **Fragmentation tree heuristics**
- **Negative ion mode data is now supported**
- **CPLEX** ILP solver support
- Consider a specific list of **ionizations** for **Sirius**
- Consider a specific list of **adducts** for **CSI:FingerID**
- Custom ionizations/adducts can be specified (CLI and GUI)
- **Full-featured** standalone **command line version** (headless version)
- Improved **parallelization** and task management
- Improved stability of the **CSI:FingerID** webservice
- Time limit for fragmentation tree computations
- Specify fields to import name and ID from .sdf into a custom database (GUI).
- **CSI:FingerID** results can be **filtered by Custom databases** (GUI).
- Better filtering performance (GUI)
- Bug fix in Database filtering view (GUI)
- Error Reporter bug fixed (GUI)
- Logging bugs fixed
- Many minor bug fixes

3.5

- **Custom databases** can be imported by hand or via csv file. You can manage multiple databases within **Sirius**.
- New **Bayesian Network scoring** for **CSI:FingerID** which takes dependencies between molecular properties into account.
- **CSI:FingerID Overview** which lists results for all molecular formulas.
- **Visualization of the predicted fingerprints.**
- **ECFP fingerprints** are now also in the **CSI:FingerID** database and do no longer have to be computed on the users side.

- Connection error detection and refresh feature. No restart required to apply Sirius internal proxy settings anymore.
- **System wide proxy** settings are now supported.
- Many minor bug fixes and small improvements of the GUI

3.4

- **element prediction** using isotope pattern
- CSI:FingerID now predicts **more molecular properties** which improves structure identification
- improved structure of the result output generated by the command line tool **to its final version**

3.3

- fix missing MS2 data error
- MacOSX compatible start script
- add proxy settings, bug reporter, feature request
- new GUI look

3.2

- integration of CSI:FingerID and structure identification into SIRIUS
- it is now possible to search formulas or structures in molecular databases
- isotope pattern analysis is now rewritten and hopefully more stable than before

3.1.3

- fix bug with penalizing molecular formulas on intrinsically charged mode
- fix critical bug in CSV reader

3.1.0

- 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

BIBLIOGRAPHY

1. Dittwald, P., Valkenborg, D., Claesen, J., Rockwood, A. L. & Gambin, A. On the Fine Isotopic Distribution and Limits to Resolution in Mass Spectrometry. *J Am Soc Mass Spectrom* **26**, 1732–1745 (2015).
2. Meusel, M. *et al.* Predicting the presence of uncommon elements in unknown biomolecules from isotope patterns. *Anal Chem* **88**, 7556–7566 (2016).
3. Ferrer, I. & Thurman, E. M. Importance of the electron mass in the calculations of exact mass by time-of-flight mass spectrometry. *Rapid Commun Mass Spectrom* **21**, 2538–2539 (2007).
4. Wang, M. *et al.* The AME2016 atomic mass evaluation (II). Tables, graphs and references. *Chinese Physics C* **41**, 030003 (2017).
5. Jaitly, N. *et al.* Robust algorithm for alignment of liquid chromatography-mass spectrometry analyses in an accurate mass and time tag data analysis pipeline. *Anal Chem* **78**, 7397–7409 (2006).
6. Zubarev, R. & Mann, M. On the proper use of mass accuracy in proteomics. *Mol Cell Proteomics* **6**, 377–381 (2007).
7. Böcker, S. & Dührkop, K. Fragmentation trees reloaded. *J Cheminform* **8**, 5 (2016).
8. White, W. T. J., Beyer, S., Dührkop, K., Chimani, M. & Böcker, S. *Speedy Colorful Subtrees* in *Proc. of Computing and Combinatorics Conference (COCOON 2015)* **9198** (Springer, Berlin, 2015), 310–322.
9. Rauf, I., Rasche, F., Nicolas, F. & Böcker, S. Finding Maximum Colorful Subtrees in practice. *J Comput Biol* **20**, 1–11 (2013).
10. Böcker, S. & Rasche, F. Towards de novo identification of metabolites by analyzing tandem mass spectra. *Bioinformatics* **24**. *Proc. of European Conference on Computational Biology (ECCB 2008)*, 149–155 (2008).
11. Steinbeck, C. *et al.* The Chemistry Development Kit (CDK): An Open-Source Java Library for Chemo- and Bioinformatics. *J Chem Inf Comput Sci* **43**, 493–500 (2003).
12. Steinbeck, C. *et al.* Recent developments of the Chemistry Development Kit (CDK) - an open-source Java library for chemo- and bioinformatics. *Curr Pharm Des* **12**, 2111–2120 (2006).
13. Willighagen, E. L. *et al.* The Chemistry Development Kit (CDK) v2.0: atom typing, depiction, molecular formulas, and substructure searching. *J Cheminf* **9**, 33 (2017).
14. Heinonen, M., Shen, H., Zamboni, N. & Rousu, J. Metabolite identification and molecular fingerprint prediction via machine learning. *Bioinformatics* **28**, 2333–2341 (2012).
15. Curry, B. & Rumelhart, D. E. MSnet: A Neural Network That Classifies Mass Spectra. *Tetrahedron Com Methodol* **3**, 213–237 (1990).
16. Shen, H., Dührkop, K., Böcker, S. & Rousu, J. Metabolite Identification through Multiple Kernel Learning on Fragmentation Trees. *Bioinformatics* **30**. *Proc. of Intelligent Systems for Molecular Biology (ISMB 2014)*, i157–i164 (2014).
17. Dührkop, K., Shen, H., Meusel, M., Rousu, J. & Böcker, S. Searching molecular structure databases with tandem mass spectra using CSI:FingerID. *Proc Natl Acad Sci U S A* **112**, 12580–12585 (2015).
18. Klekota, J. & Roth, F. P. Chemical substructures that enrich for biological activity. *Bioinformatics* **24**, 2518–2525 (2008).
19. Rogers, D. & Hahn, M. Extended-connectivity fingerprints. *J Chem Inf Model* **50**, 742–754 (2010).
20. Platt, J. C. in *Advances in large margin classifiers* chap. 5 (MIT Press, Cambridge, Massachusetts, 2000).
21. Shinbo, Y. *et al.* in *Plant Metabolomics* (eds Saito, K., Dixon, R. A. & Willmitzer, L.) 165–181 (Springer-Verlag, 2006).

22. Wishart, D. S. *et al.* HMDB 3.0: The Human Metabolome Database in 2013. *Nucleic Acids Res* **41**, D801–D807 (2013).
23. Hastings, J. *et al.* The ChEBI reference database and ontology for biologically relevant chemistry: enhancements for 2013. *Nucleic Acids Res* **41**, D456–D463 (2013).
24. Kanehisa, M., Sato, Y., Kawashima, M., Furumichi, M. & Tanabe, M. KEGG as a reference resource for gene and protein annotation. *Nucleic Acids Res* **44**, D457–D462 (2016).
25. Fonger, G. C., Hakkinen, P., Jordan, S. & Publicker, S. The National Library of Medicine’s (NLM) Hazardous Substances Data Bank (HSDB): background, recent enhancements and future plans. *eng. Toxicology* **325**, 209–216 (2014).
26. Weber, R. J. M., Li, E., Bruty, J., He, S. & Viant, M. R. MaConDa: A publicly accessible mass spectrometry contaminants database. *Bioinformatics* **28**, 2856–2857 (2012).
27. Caspi, R. *et al.* The MetaCyc database of metabolic pathways and enzymes and the BioCyc collection of Pathway/Genome Databases. *Nucleic Acids Research* **42**, D459–D471 (2014).
28. Gu, J. *et al.* Use of natural products as chemical library for drug discovery and network pharmacology. *PLoS One* **8**, e62839 (2013).
29. Irwin, J. J., Sterling, T., Mysinger, M. M., Bolstad, E. S. & Coleman, R. G. ZINC: a free tool to discover chemistry for biology. *J Chem Inf Model* **52**, 1757–1768 (2012).
30. Wang, M. *et al.* Sharing and community curation of mass spectrometry data with Global Natural Products Social Molecular Networking. *Nat Biotechnol* **34**, 828–837 (2016).
31. Horai, H. *et al.* MassBank: A public repository for sharing mass spectral data for life sciences. *J Mass Spectrom* **45**, 703–714 (2010).
32. Nelson, S. J., Johnston, W. D. & Humphreys, B. L. in *Relationships in the organization of knowledge* (eds Bean, C. A. & Green, R.) 171–184 (Kluwer Academic Publishers, 2001).
33. Kim, S. *et al.* PubChem Substance and Compound databases. *Nucleic Acids Res* **44**, D1202–D1213 (2016).