MOlecule fRagmenTAtion fRamework

Tutorial

Last update: 16th September 2025

Reference version of MORTAR: v1.5.0

Contents

Introduction **1**
Installation and start-up **2**
Windows 2
MacOS 3
Debian-based Linux 3
Red Hat-based Linux 3
JAR execution via command line (all platforms) 4
Further notes 4
Molecule set import and preferences **6**
Single fragmentation **11**
Fragments tab **15**
Items tab **20**
Pipeline fragmentation **22**
Overview view **26**
Histogram view **29**
About view **32**
Application exit **34**
Integration of new fragmentation algorithms **35**
Stereochemistry **36**
Known issues **40**
Handling of polymers 40
Explicit Hydrogen atoms 40
Non-standard bond orders in MOL/SD files 41
Scaffold Generator fragmentation runtime 41
Ertl algorithm fragmentation runtime 42

Introduction

MORTAR ('MOlecule fRagmenTation fRamework') is a free and open-source graphical desktop application that supports molecular *in silico* fragmentation and substructure analysis. The Java/JavaFX rich client offers extensive graphical functions for visualizing the fragmentation results of individual compounds or entire compound sets. In addition to three currently integrated methods for fragmentation and substructure analysis - ErtlFunctionalGroupsFinder (Ertl algorithm for functional group identification in organic molecules), Sugar Removal Utility (enhanced with the Sugar Detection Utility features), and Scaffold Generator (CDK-Scaffold module) - MORTAR allows straightforward integration of additional fragmentation algorithms to support the development of new methods at an early stage. All cheminformatics functionalities are implemented based on the Chemistry Development Kit (CDK).

A scientific article describing MORTAR can be found here: MORTAR: a rich client application for in silico molecule fragmentation (Baensch et al. 2023) (please cite it if you are using MORTAR for your scientific work). The MORTAR GitHub repository, where you can download the software, browse the source code, and report issues, can be found here: https://github.com/FelixBaensch/MORTAR.

Installation and start-up

Pre-compiled and executable MORTAR distributions can be found attached as assets to the marked releases on GitHub (below the change notes).
In brief (more details in the dedicated sections below), a graphical installer executable is available for Windows, a disk image (DMG) file for installation on macOS, an RPM Package Manager (RPM) package for Red Hat-based Linux distributions like AlmaLinux, and a Debian package (DEB) is available for installation on Debian-based Linux distributions like Ubuntu.
On all three operating systems (Windows, macOS, and Linux), MORTAR can also be run from the command line using the supplied “fat” Java ARchive (JAR), which gives you full control, e.g., over how much memory should be used. A Java Development Kit or Runtime Environment (JDK/JRE) of version 21.0.1 or higher must be pre-installed on your system to run MORTAR from the command line.

Windows

A convenient Windows OS installer executable for MORTAR is available (click here to automatically download the setup.exe of the latest version). Download the installer executable, start it, and follow the instructions to install MORTAR. During installation, you are asked whether desktop or start menu icons/shortcuts should be created. The executable (batch file) denoted "MORTAR" uses up to 4 GB of RAM, while the "MORTAR 20GB" configuration allocates up to 20 GB of your RAM for running MORTAR. Note that no start menu folder will be created, so creating at least one of the shortcuts is recommended. To start MORTAR after installation, double-click one of the created shortcuts.
The MORTAR program folder is located at "C:\Program Files\MORTAR\MORTARv1.4.0.0" by default, but can be adjusted during installation. MORTAR can be uninstalled by the provided "unins000.exe" executable in the MORTAR program folder or by standard Windows functions (Settings -> Apps -> Installed Apps). Note that the installation includes a full Java Runtime Environment (JRE).
If you want to configure your own heap space settings for running MORTAR, open one of the provided batch files (in the "bin" subfolder of the MORTAR program folder, "MORTAR.bat" or "MORTAR_20GB.bat") in a text editor and adjust the line

```
set DEFAULT_JVM_OPTS="-Xms4g" "-Xmx4g"
```

with your chosen initially allocated memory (-Xms) and maximum value (-Xmx) accordingly.

Should this installation or the execution of the batch files not work for you, you can also run MORTAR directly from the command line (see below).
Please note that x32 processor architectures are not supported by MORTAR.

MacOS

On macOS, MORTAR can be installed using the disk image (.dmg) files attached to every release since v1.2, one for devices with an x86 processor architecture and one for ARM/AArch64-based systems. Download the right file for your system and double-click on it. In the window that opens, drag the MORTAR icon into the Applications folder to install it. It might be necessary to adjust your security settings to allow MORTAR to run.
The disk images are configured to allocate up to 4 gigabytes of RAM to MORTAR. Should you want to assign more memory to analyze bigger data sets, or should this installation not work for you, you can run MORTAR directly from the command line (see below).

Debian-based Linux

On Debian-based Linux distributions like Ubuntu, MORTAR can be installed using the Debian package (.deb) attached to every release since v1.2.2 (only one for devices with an x86 processor architecture). Download the package and double-click on it to install it. Alternatively, you can install it via the command line:

```
sudo dpkg -i <path to>MORTAR-1.4.0.deb
```

Execute the command in the directory where the package is situated or use its explicit path instead of <path to>.
The package is configured to allocate up to 4 gigabytes of RAM to MORTAR. Should you want to assign more memory to analyze bigger data sets, or should this installation not work for you, you can run MORTAR directly from the command line (see below).

Red Hat-based Linux

On Red Hat-based Linux distributions like AlmaLinux, MORTAR can be installed using the RPM Package Manager (.rpm) package attached to every release since v1.4.1 (only one for devices with an x86 processor architecture). Download the package and double-click on it to install it. Alternatively, you can install it via the command line:

```
sudo rpm -i <path to>MORTAR-1.4.0.rpm
```

Execute the command in the directory where the package is situated or use its explicit path instead of <path to>.
The package is configured to allocate up to 4 gigabytes of RAM to MORTAR. Should you want to assign more memory to analyze bigger data sets, or should this installation not work for you, you can run MORTAR directly from the command line (see below).

JAR execution via command line (all platforms)

Every release has the executable JAR "MORTAR-fat-1.4.0.0.jar" attached, which contains the packaged MORTAR code together with all dependencies (click here to automatically download the JAR of the latest version). To run MORTAR (with up to 4 GB of RAM available, e.g.), execute the JAR from the command line using

```
java -jar -Xms512m -Xmx4g <path to>MORTAR-fat-1.4.0.0.jar
```

A JDK or JRE of version 21.0.1 or higher must be installed on your system and linked to the "java" command. Otherwise, replace "java" with the path to the `java` command of a specific JDK or JRE.
Execute the command in the directory where the JAR is situated or use its explicit path instead of <path to>.
Adjust the initially allocated memory (-Xms) and maximum memory to be used (-Xmx) according to your preferences.

Further notes

Please note that MORTAR only supports x64 (not x32, on all four platforms) and AArch64/ARM (on macOS and Linux) architectures in general. For the latter, a special "fat JAR" named "MORTAR-fat-aarch64-1.4.0.0.jar" is available from the distributions attached to the releases and must be used (click here to automatically download the AArch64 JAR of the latest version).

Also note that using the Windows Subsystem for Linux (WSL) is not recommended, since a lot of additional configurations have to be made there to run Java GUI applications.

The X / X11 / X Window System can be used to run a graphical application like MORTAR on a remote server while displaying the graphical user interface on the local machine / personal computer. This setup can be beneficial if more computing power or memory is required to analyze large data sets. However, please note that while this is possible in principle,

unexpected behaviour from MORTAR may occur. We cannot take responsibility for or recommend this way of deployment, similar to using the WSL as mentioned above.

After successful installation and start-up of the application, the MORTAR main window (Figure 1) appears.

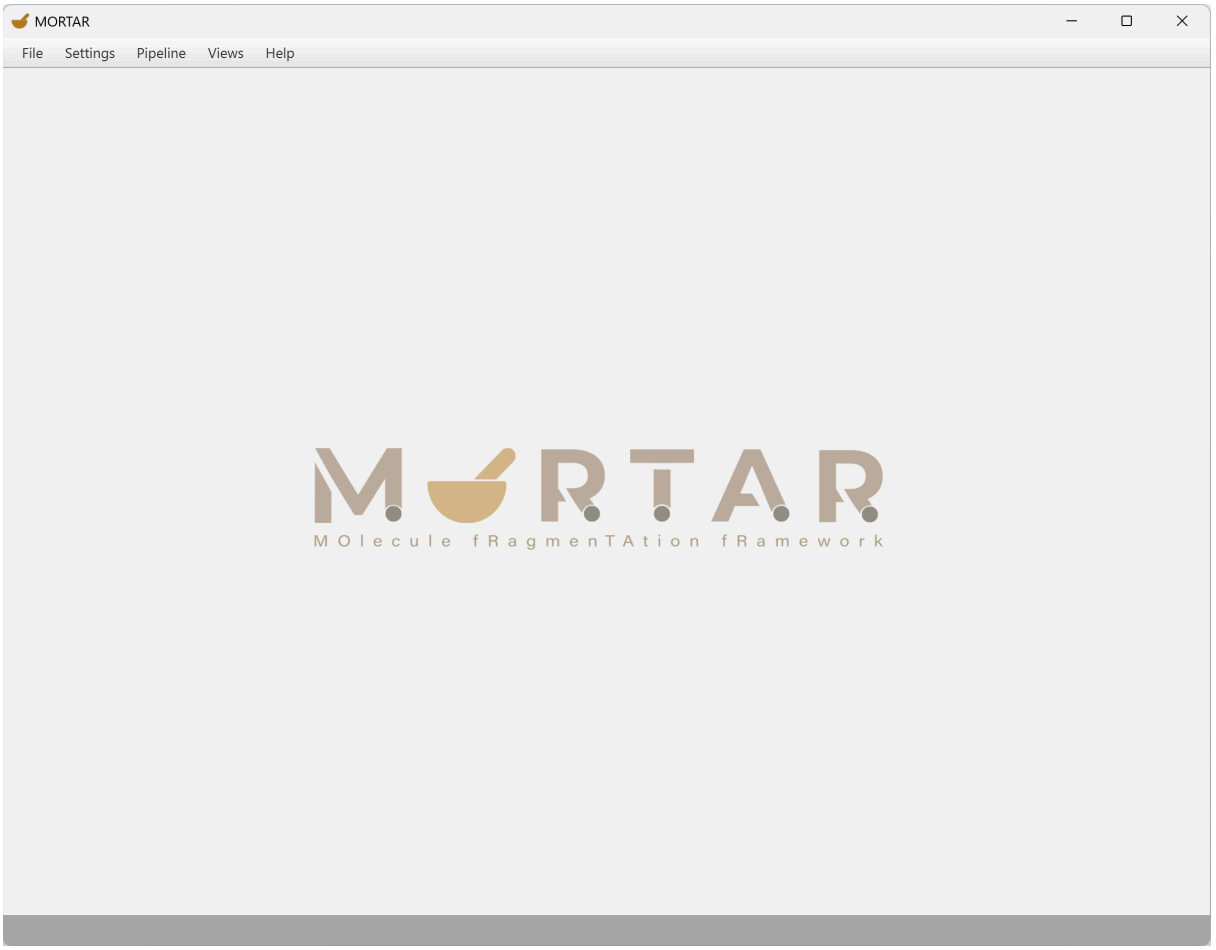

Figure 1: MORTAR main window.

Molecule set import and preferences

Before starting to work with MORTAR, inspect the global application preferences located in the upper-left menu bar (on macOS, it will be in the app menu bar at the top of the screen) at “Settings” -> “Preferences” (Figure 2). A dialog opens that allows you to adjust several settings relevant to different functionalities (Figure 3).

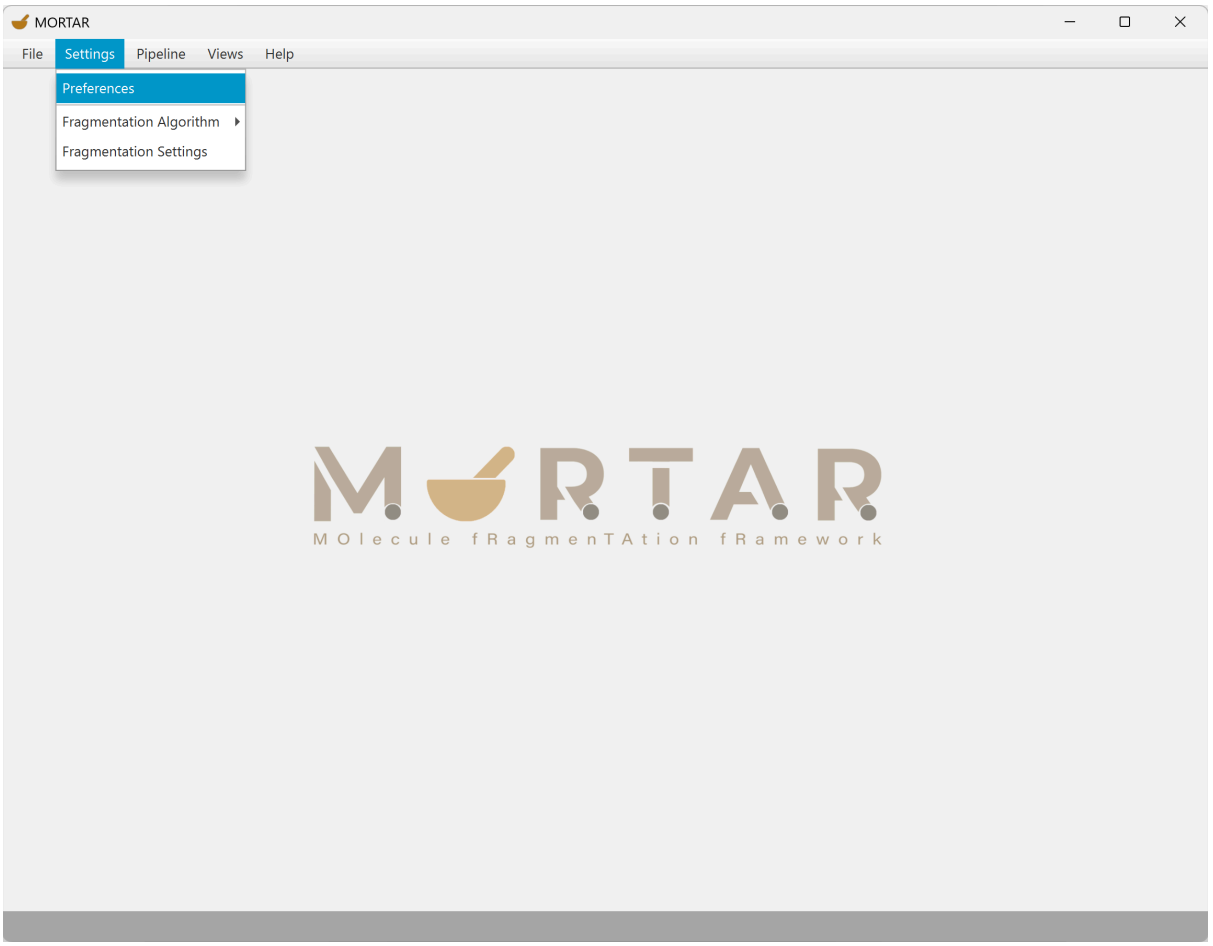

Figure 2: Settings menu.

A brief description for each setting is provided in tooltips that appear when the cursor hovers over one setting element for a few seconds. In choice boxes, every individual option also has its individual tooltip. Using the “Default” button in the bottom-left corner, all settings are reset to their default values. To save your changes to the settings and close the dialog, click the “Apply” button in the bottom-right corner. With the “Cancel” button in the bottom-right corner, the dialog is closed without saving the changes to the settings.

For the molecule set import, it can be specified whether implicit hydrogen atoms should be added to fill possible open valences in the imported molecules. The “Regard stereochemistry

setting" is crucial for the entire MORTAR workflow, beginning with the molecule set import. If turned off, stereochemistry information will be lost in imported molecules, and extracted substructures will hence not carry any stereo configurations as well. On the other hand, some analyses will take less computation time, e.g. functional group extraction. You can read more about this in the dedicated "Stereochemistry" section of this tutorial below.

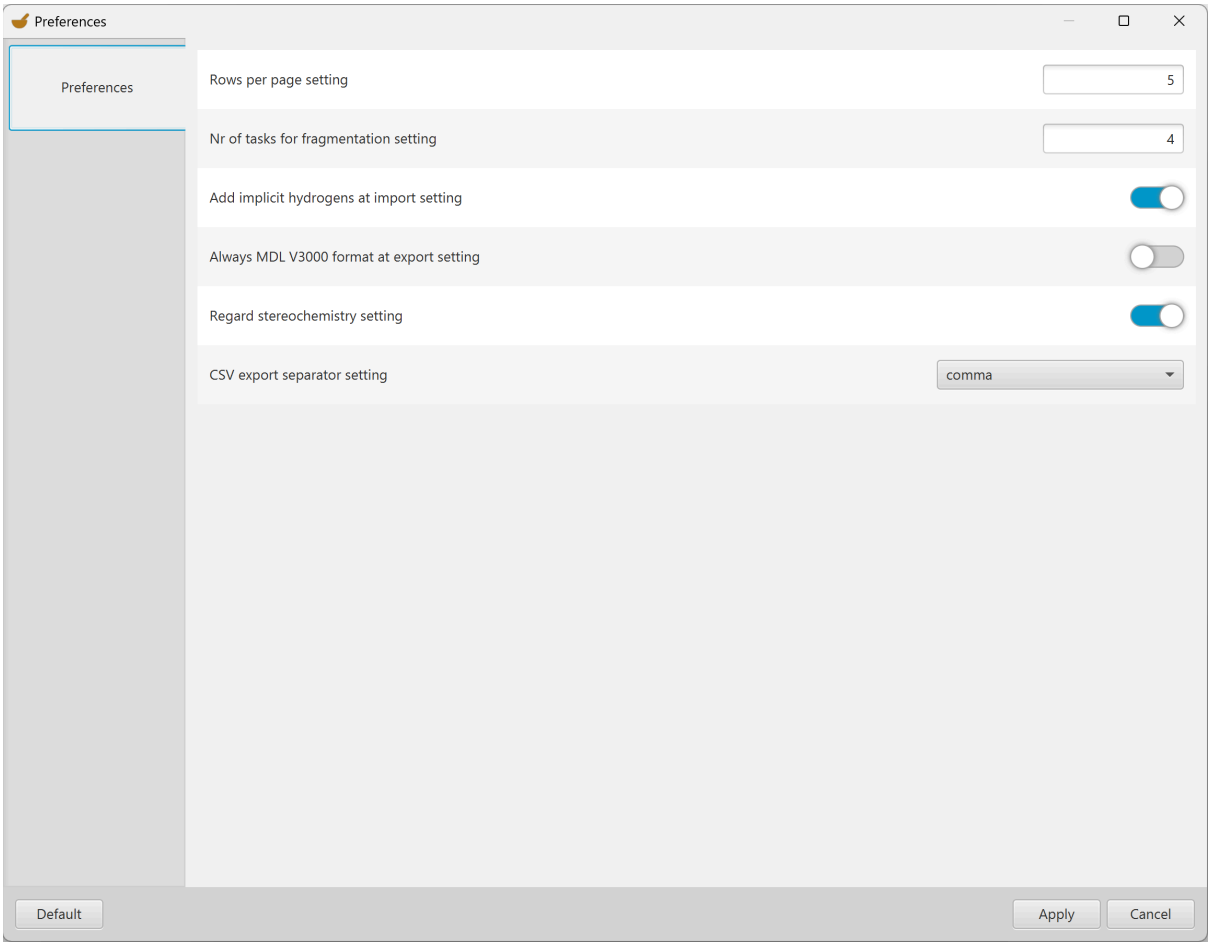

Figure 3: Global application preferences dialog.

To import a molecule set, open the "File" menu in the MORTAR main window menu bar and select the "Open" menu item (Figure 4) or simply drag and drop a MOL, SD, or SMILES file into the MORTAR main window (Figure 1).

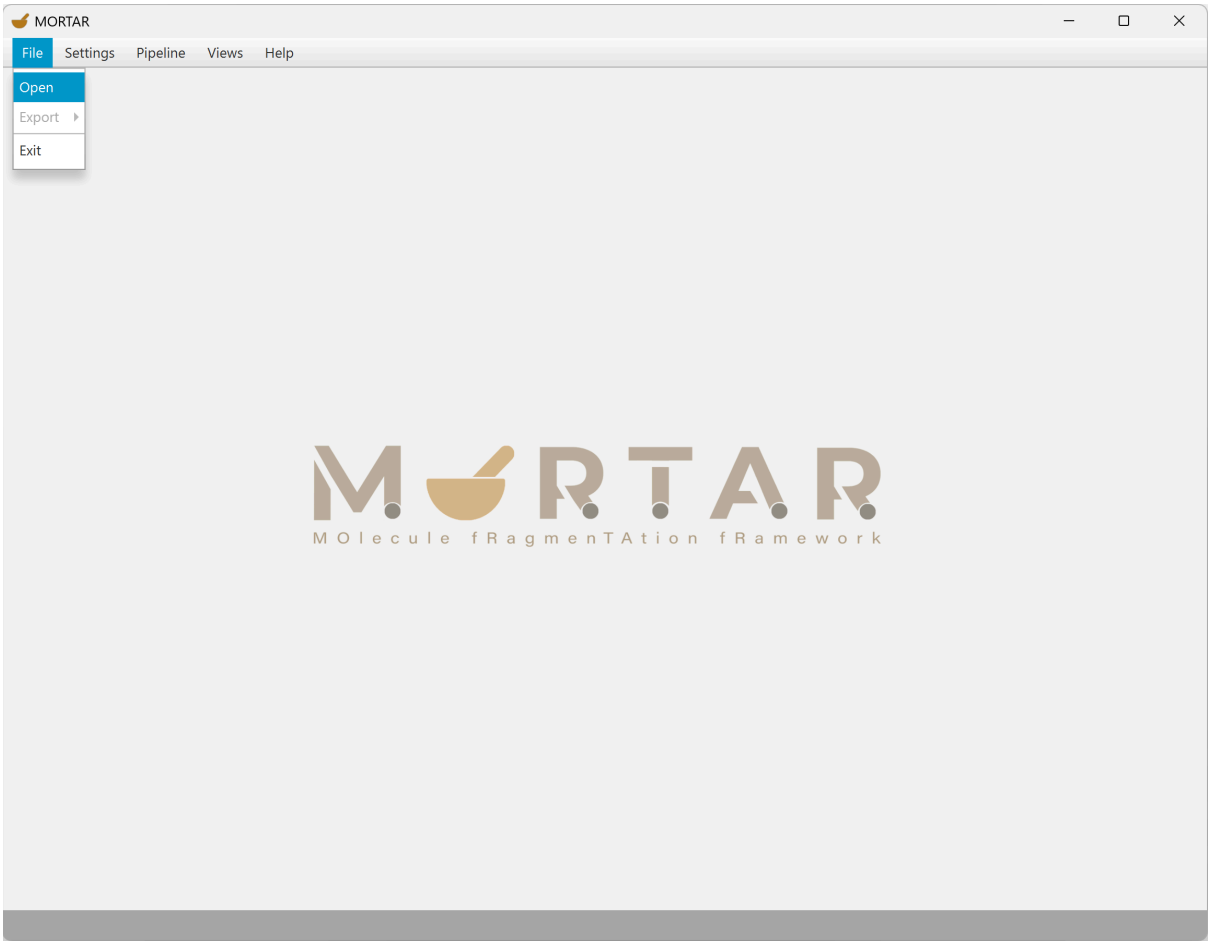

Figure 4: File menu.

An OS-dependent file chooser dialog opens. Here, you can select a MOL, SD, or SMILES file to import. An SD file of the BIOFACQUIM natural products collection (taken from the COCONUT natural products database) is shipped alongside this tutorial as an example data set. It can be found in the application's root directory in the "tutorial" folder, next to this PDF file. Alternatively, you can find it in the MORTAR GitHub repository. After importing this set or any molecular data set, the "Molecules" tab opens (Figure 5).

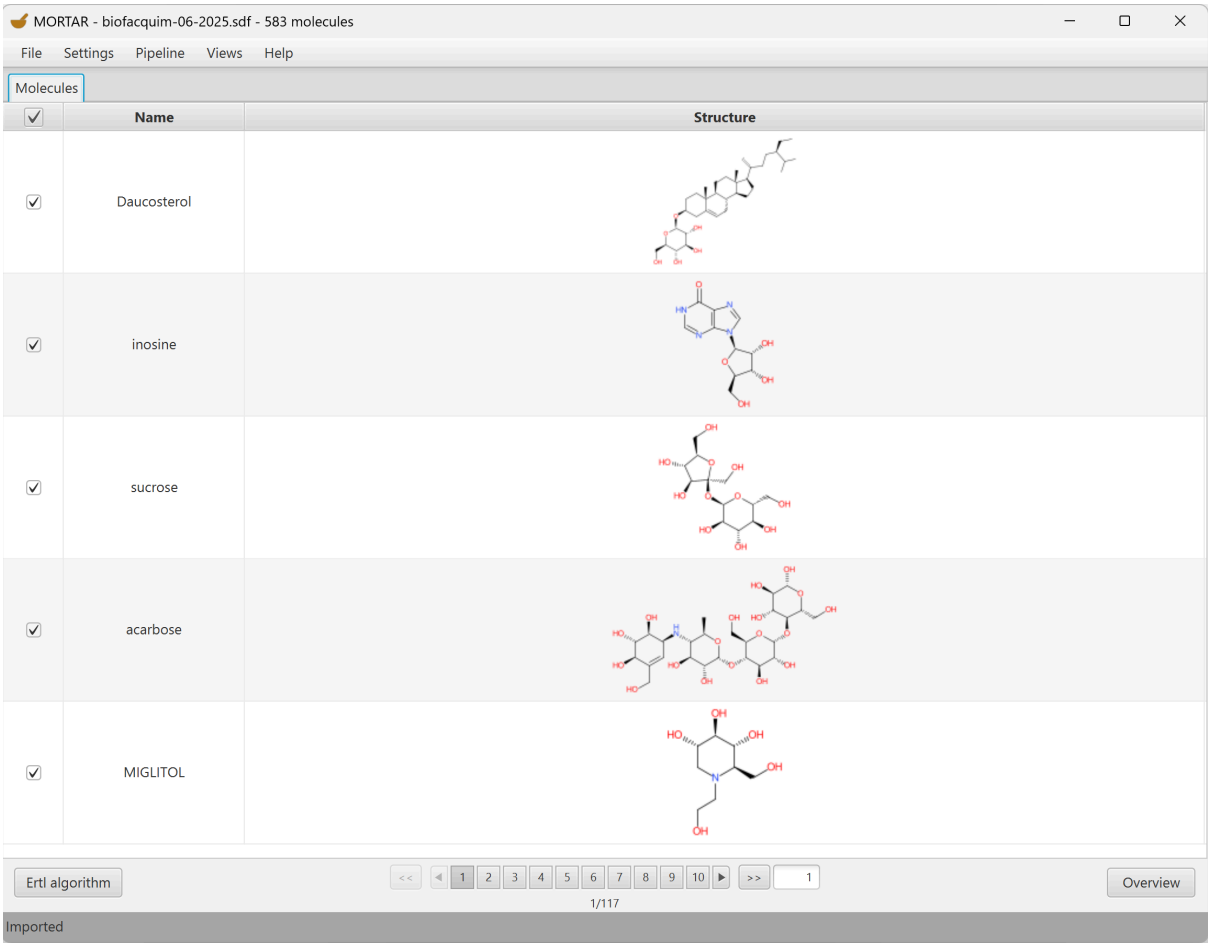

Figure 5: Molecules tab after import of the BIOFACQUIM data set.

The imported structures are displayed in a list on multiple pages and can be visually inspected. With the pagination control at the bottom, you can switch between pages, go to the first or last page, or jump directly to a specific page by entering the respective page number into the text field and pressing “Enter”. The left and right arrow keys or “page up” and “page down” keys can be used as well to switch between pages. By using the “control” key in combination with the left or right arrow key, you can jump to the first or last page, respectively. The same can be done using the “home” or “end” key.

If possible, a name for each structure is extracted from the input file. By clicking the head of the “Name” column, the structures can be sorted by their names in ascending or descending alphabetical order.

Using the first column, each molecule can be selected or deselected for fragmentation. Use the column header to select or deselect all. Single-cell values (names and structure depictions) can be copied to the clipboard using the right-click menu.

In the global preferences (Figure 3), you can adjust the number of structures/rows that are displayed per page.

For an alternative view of the imported set of structures, you can use the overview functionality, which displays all structures in a grid layout on multiple pages and also offers an enlarged structure view. In the “Molecules” tab, the overview view can be opened via the “Overview” button in the bottom-right corner or via the menu bar at the top of the window (“Views” -> “Overview”). Clicking on a structure depiction in the overview window opens the enlarged structure view, and a double-click closes the overview view and jumps to the position of the respective structure in the main window of MORTAR. For a more detailed description of the overview view, see the “Overview view” section of this tutorial below.

Single fragmentation

Using the “Settings” -> “Fragmentation Algorithm” menu, a fragmentation algorithm for a single fragmentation can be selected (Figure 6). In the current version, three algorithms are available:

- **Ertl algorithm for functional group detection**: The algorithm published by Dr Peter Ertl is the first approach to detecting functional groups in organic molecules using a rule set-based algorithm, instead of a predefined list of functional group structures. It is available in MORTAR via the open, CDK-based implementation ErtlFunctionalGroupsFinder, which is now also part of CDK as FunctionalGroupsFinder.

- **Sugar Removal Utility (SRU)**: The SRU is an algorithm and an open, CDK-based Java tool to detect sugar moieties in organic molecules, especially natural products, and remove them to produce the molecule core or aglycone. It is now also part of CDK. Since version 1.5, the Sugar Removal Utility fragmentation algorithm in MORTAR uses the Sugar Detection Utility extension of the SRU internally, to not only extract correct aglycones but also the correct sugar structures.

- **Scaffold Generator**: Scaffold Generator is an open tool that re-implements common scaffold or framework approaches like Murcko frameworks, scaffold trees, and scaffold networks based on CDK. It forms the basis of the CDK-scaffold module.

The currently selected fragmentation algorithm is also named in the bottom-left corner of the “Molecules” tab on the button that is used to start the fragmentation (Figure 6).

Figure 6: Fragmentation algorithm selection.

Each fragmentation algorithm has its own set of specific settings that can be configured via the “Fragmentation Settings” menu item in the “Settings” menu (below the “Fragmentation Algorithm” menu item, see Figure 6). When the menu item is clicked, a dialog opens that displays the settings of the currently selected algorithm (Figure 7). Using the tabs on the left, the settings for the other algorithms can be inspected and adjusted. For documentation on the individual settings, please refer to the publications linked above. A short description is given in tooltips that appear when the cursor hovers for a few seconds over one setting element. In choice boxes, every individual option also has its specific tooltip. Using the “Default” button in the bottom-left corner, all settings for the currently displayed fragmentation algorithm are reset to their default values. To save your changes to the settings and close the dialog, click the “Apply” button in the bottom-right corner. With the “Cancel” button in the bottom-right corner, the dialog is closed without saving the changes to the settings. The settings of all fragmentation algorithms will remain as they were before the dialog was opened, in this case.

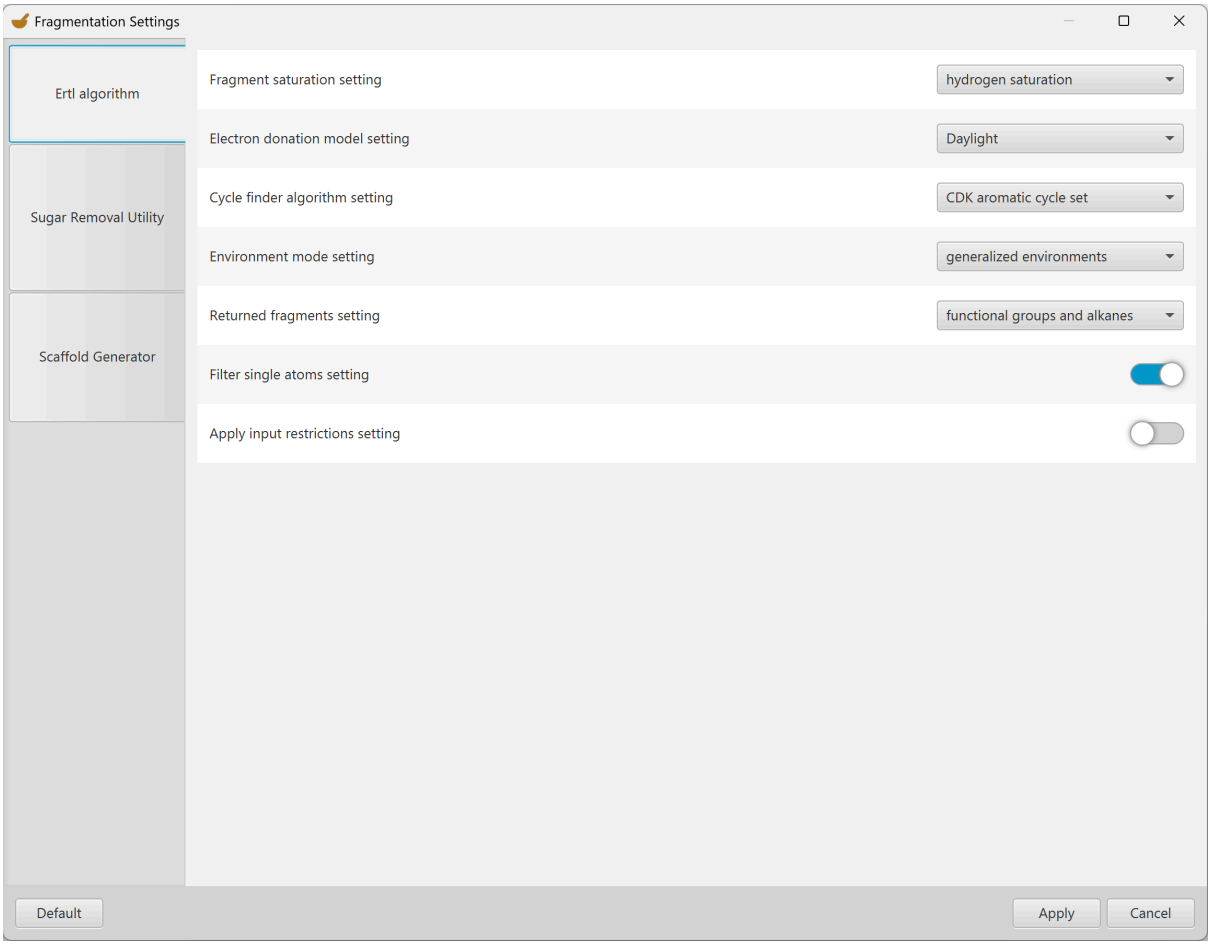

Figure 7: Fragmentation settings.

After adjusting the settings and closing the dialog, the fragmentation can be started using the “Ertl algorithm” button in the bottom-left corner of the “Molecules” tab (Figure 5). If a different fragmentation algorithm is selected, its name is displayed on the button.

MORTAR uses parallel computing to speed up the fragmentation process. In the global preferences, it can be specified how many parallel calculation threads should be employed (Figure 3). The default value is set to 4 if your system has 4 or more available threads. For some fragmentation algorithms, employing more than 4 parallel fragmentation threads did not yield an additional performance boost in testing. But all users are invited to test this by themselves on their own machines. The maximum value of the setting is the maximum number of logical processors the specific machine has to offer. This number is determined by MORTAR and named in the tooltip of the setting in the global preferences dialog (Figure 3). Additionally, an error will be raised if a number higher than the maximum value is entered in the text field. In general, it is also not recommended to set the setting to this maximum value exactly because MORTAR cannot block every available thread of your machine at the same time anyway.

Note that the fragmentation settings and the global preferences are persisted for the next session when the application is shut down.

Fragments tab

When the fragmentation has finished, two new tabs open to display the fragmentation results. One is the “Fragments” tab, which focuses on the different fragments generated in the process (Figure 8). For the Ertl algorithm, these are functional groups and alkane remnants resulting from the extraction of the former.

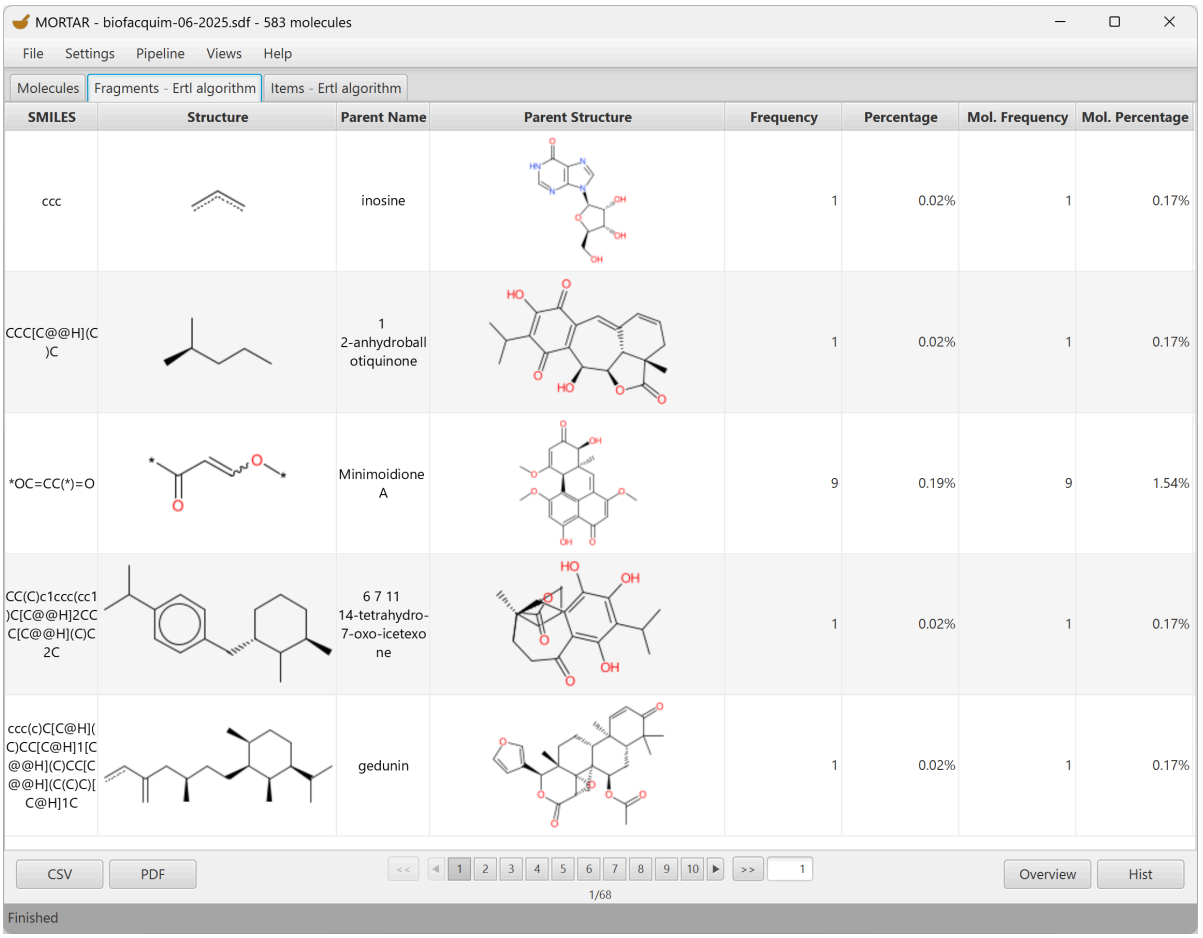

Figure 8: Fragments tab.

The first two columns display the SMILES codes and structures of the fragments. Note that aromaticity is displayed in the depictions if possible and denoted in the SMILES representations by the usage of lower-case letters for the respective aromatic atoms. Asterisks (“*”) in the structure depictions and SMILES codes denote pseudo (“R”) atoms. The third and fourth columns contain the name and structure of the first molecule from which the respective fragment was extracted. The last four columns display the frequency of each fragment in the given data set. The basic “Frequency” indicates how often a fragment was identified, i.e. if a molecule contained the same type of fragment multiple times, it is counted multiple times here. The “Molecule Frequency”, on the other hand, expresses how many

molecules the respective fragment was identified in, i.e. the same fragment appearing multiple times in one molecule is counted only once here. Single-cell values can be copied to the clipboard using the right-click menu. The table can be sorted in ascending or descending order according to most of the columns' values by clicking the column heads. E.g. by double-clicking the “Frequency” column label, the fragments are sorted according to their frequency in descending order and the most frequent fragments are displayed first (Figure 9). The column arrangement can also be adjusted by dragging and dropping the columns.

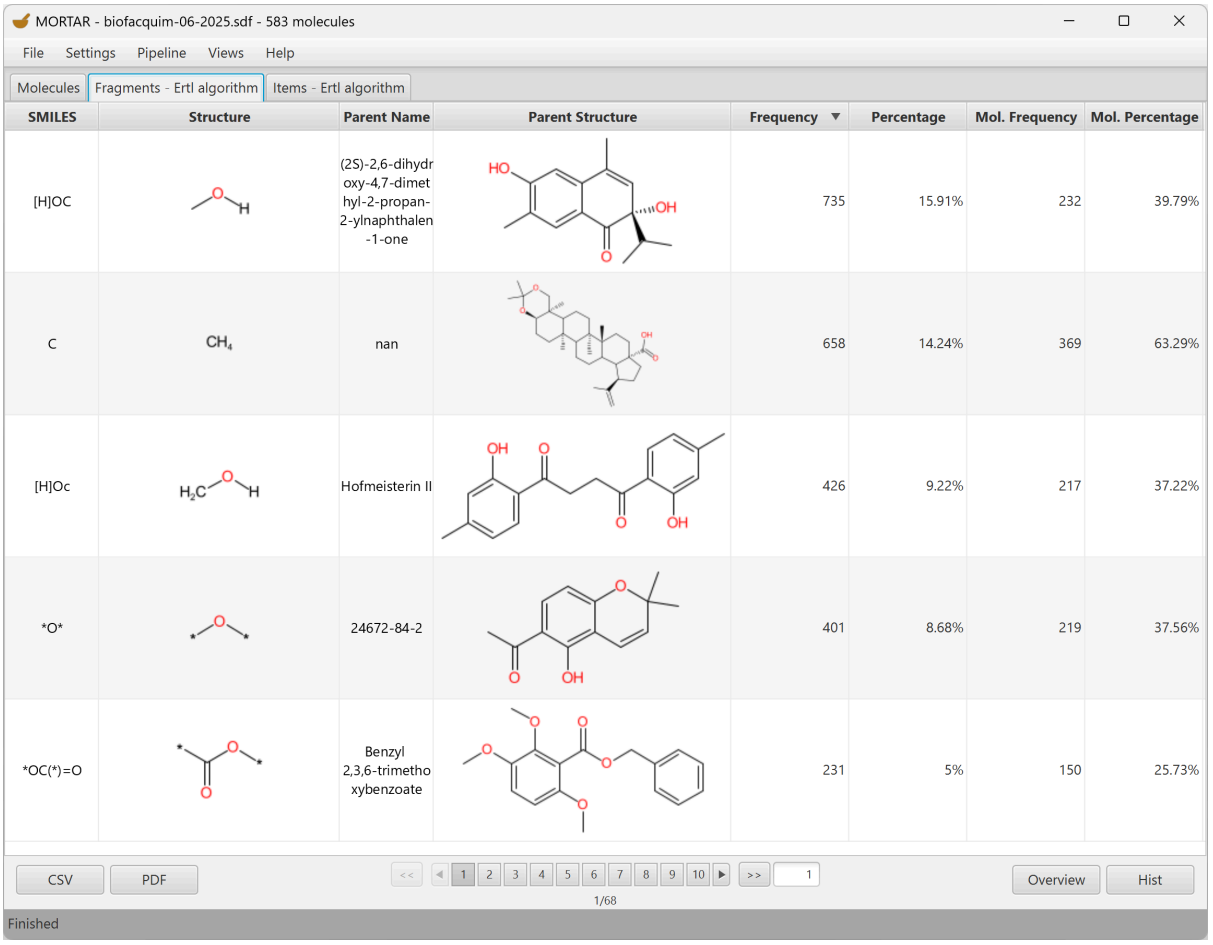

Figure 9: Fragments sorted by decreasing frequency.

For an alternative view of the set of fragments, the overview view functionality can be used. As for the “Molecules” tab, it is accessible via a button in the bottom-right corner of the window or via the menu bar at the top (“Views” -> “Overview”). The overview view can also be used to display the whole set of parent structures of an individual fragment. This option is accessible via the context menu in the respective row of the “Fragments” tab (“Parent Structures Overview” menu item). In the “Parent Structures Overview”, the fragment is placed in the first position on the first page of the view and highlighted. For more details, see the “Overview view” section of this tutorial below.

The fragmentation results can also be visualized using the histogram view functionality. In this window, the most frequent fragments are displayed in a configurable histogram in decreasing order of their frequencies. By hovering over a bar in the histogram, a 2D depiction of the respective fragment is displayed in the bottom-right corner of the window; via the right-click menu of a specific bar, the SMILES code or structure depiction of a fragment can be copied to the clipboard. As for the overview view functionality, the histogram view functionality is available via a button in the bottom-right corner of the fragments tab or via the menu bar at the top (“Views” -> “Histogram”). It is described in more detail in the “Histogram view” section of this tutorial below.

With the two buttons in the bottom-left corner of the fragments tab, the displayed data can be exported to a Comma-Separated Value (CSV) or Portable Document Format (PDF) file. A file chooser dialog allows you to specify where the created files should be written. The CSV file contains the SMILES code and all four frequency notations for every fragment. The employed separation character can be adjusted in the global preferences dialog (Figure 3). The PDF file additionally includes depictions of the fragment structures. The PDF export functionality is implemented based on the OpenPDF library by LibrePDF. Note that the PDF export may take some time, especially for a greater number of fragments. When the export is still running, it is indicated in the status bar at the bottom of the MORTAR window.

Additional export functionalities can be found in the “File” menu of the main menu bar in the upper-left corner (Figure 10).

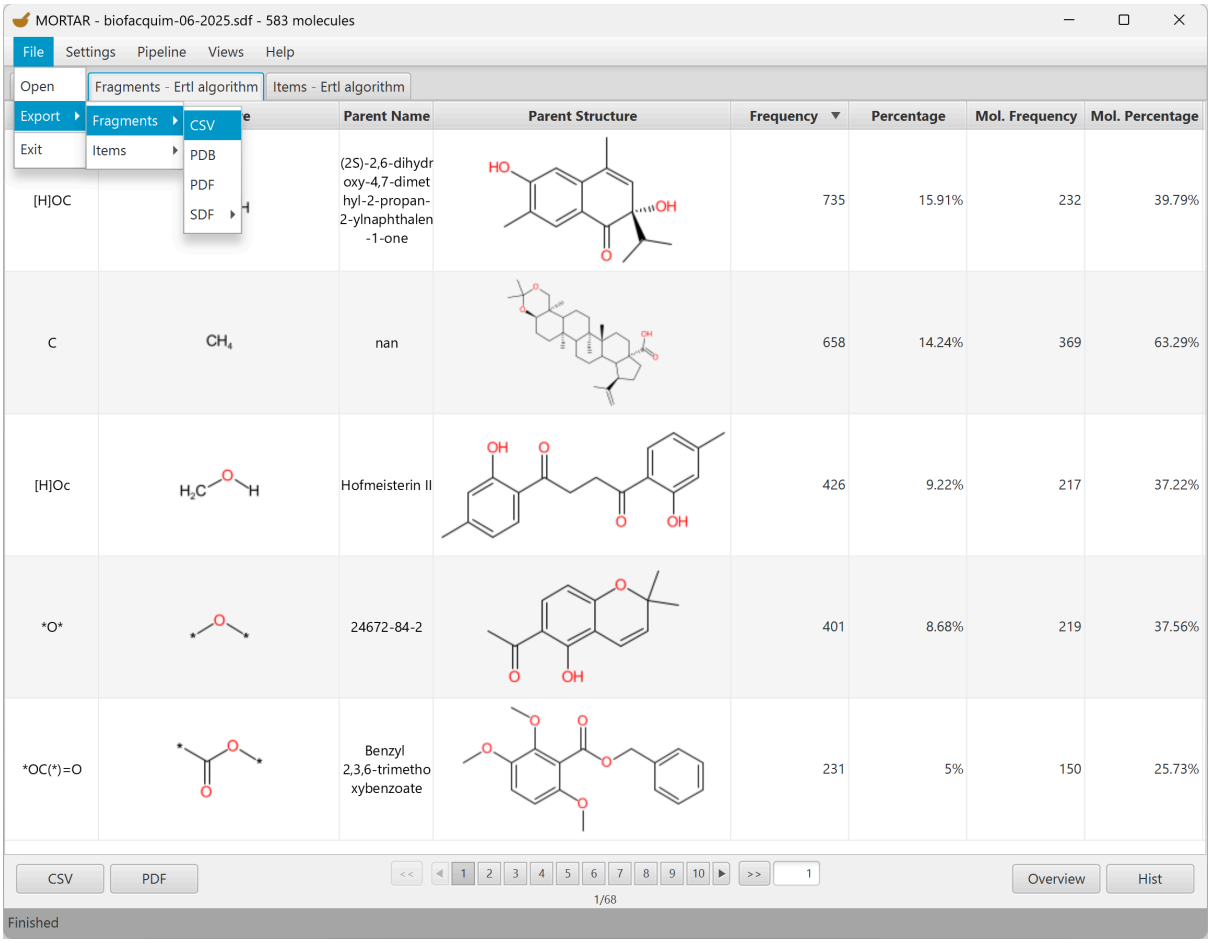

Figure 10: Fragments export menu.

In addition to the CSV and PDF export options, the fragment structures (without their frequencies) can also be exported as Structure Data (SD) files using this menu. Optionally, they can all be exported to one SD file or to separate files. In the global preferences dialog (Figure 3), it can be specified whether SD/MOL file exports should always be done in the MOL version 3000 format. If this is set to false, the export will be done in the MOL version 2000 format by default, except for molecules with more than 999 atoms that are too big for the older MOL file version. The fragment molecules can also be exported as Protein Data Bank (PDB) format files. In the generated PDB files, the “ATOM” and “CONECT” blocks are used to store the molecular connection information. All bonds are represented as single bonds, and only explicit hydrogen atoms are included. If multiple individual files are exported (SD or PDB), they are given the fragments’ molecular formulae as file names. The PDB export requires atom coordinates for the fragments to be set, but MORTAR does not retain them if they were given in the input file. Therefore, during PDB export, 2D atom coordinates that were originally intended for graphical layout are generated for the fragments. Note that these do not represent genuine conformers of the fragments, but they can be used as start geometries for conformer sampling or structure optimization in external tools. For the SDF

export, it is optional to generate these atom coordinates and include them in the exported file(s).

Items tab

The other tab that displays the fragmentation results is the “Items” tab (Figure 11). Its focus is on the individual molecules that were fragmented and the individual itemisation of their resulting fragments.

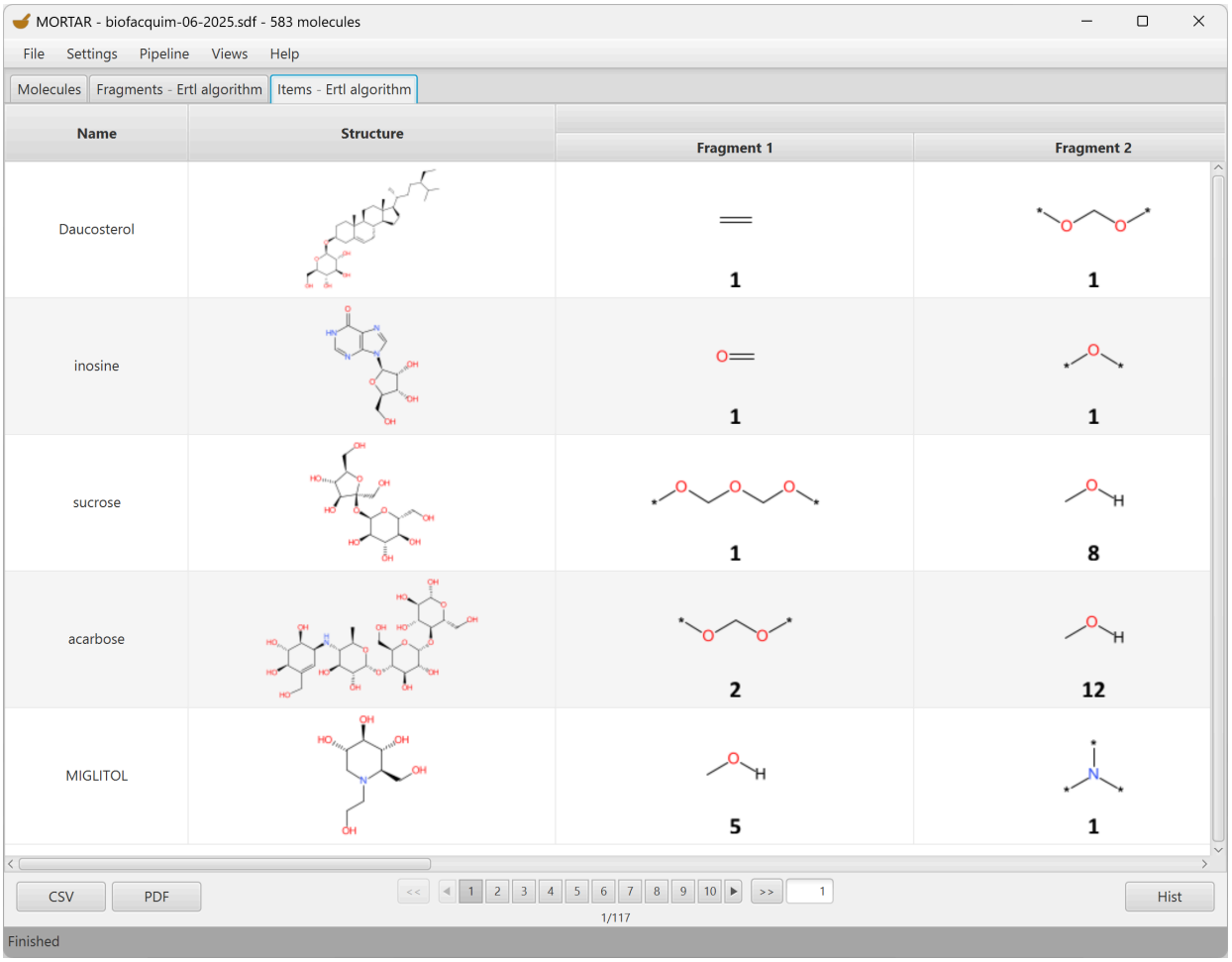

Figure 11: Items tab.

In the first two columns, the fragmented molecules are listed with their names and structures. The rows can be sorted according to the molecule names by clicking the column header. Single-cell values can be copied to the clipboard using the right-click menu. The following columns contain the fragments extracted from each molecule and how often they appeared in the molecule. For the Ertl algorithm, functional group fragments are listed first and alkane fragments second. For the Sugar Removal Utility, it is aglycones first and sugar moieties second. But in general, there is not necessarily an order in which the fragments are listed. Fragments that are not displayed initially can be accessed by scrolling horizontally. Also, an overview view of all fragments of a respective molecule can be opened via the

right-click menu in a specific row. The histogram view functionality, which displays the fragment frequencies in decreasing order, is available via the button in the bottom-right corner or via the menu bar at the top of the window (“Views” -> “Histogram”). Note that it displays the same results when opened from the “Fragments” or “Items” tab of a specific fragmentation. For a more detailed description of both functionalities, see the “Overview view” and “Histogram view” sections of this tutorial below.
Analogously to the fragments tab, the items tab data can be exported to CSV and PDF files using the buttons in the bottom-left corner (Figure 11). The same two functionalities are available as well in the main menu bar at “File” -> “Export” -> “Items” (Figure 10).

The fragments and items tabs of every individual fragmentation are assigned unique names based on the used fragmentation algorithm. When a new fragmentation is started from the molecules tab (e.g. using different settings or a different fragmentation algorithm), the already existing result tabs remain. They are only cleared away when a new molecule set is imported. Please note that no tab persists when the application is shut down. You therefore have to export all the data you want to keep.

Pipeline fragmentation

In the menu bar at the top of the MORTAR main window, the menu item “Pipeline” -> “Create Pipeline” opens a dialog where a fragmentation pipeline with different steps can be set up (Figure 12).

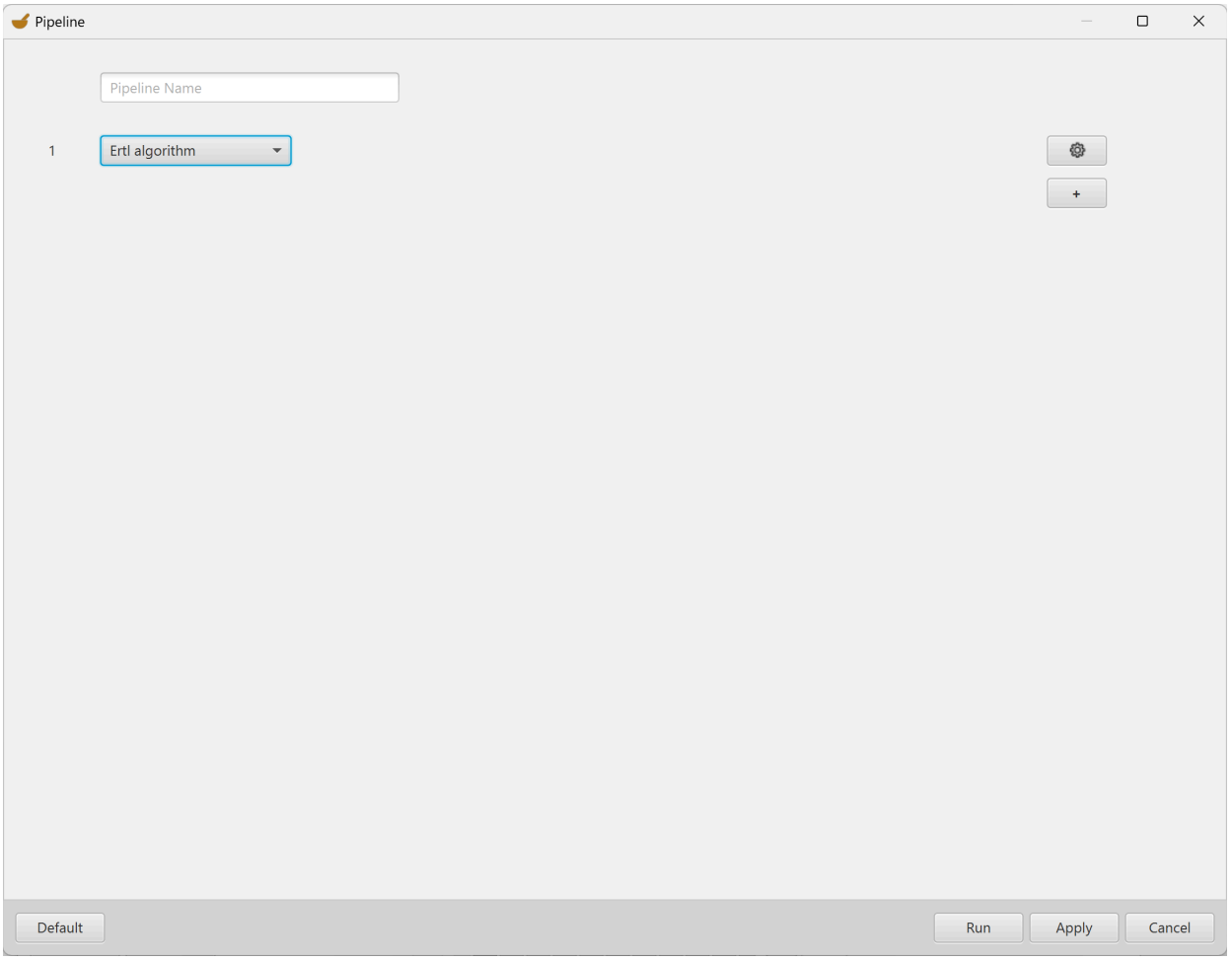

Figure 12: Pipeline dialog.

The text field at the top of the dialog allows you to define an individual name for the configured pipeline that will be used to label the result tabs. Below, the algorithm to use in the first pipeline step can be chosen from a drop-down menu. Its settings can be configured via the gear button on the right-hand side that opens a fragmentation settings dialog for the selected fragmenter (compare Figure 7). An additional fragmentation step can be added via the “+” button below the gear. Note that the fragmentation settings here are specifically set for the individual pipeline step, i.e. it is possible to employ the same fragmentation algorithm multiple times with different settings. The fragmentation settings configured in the pipeline dialog do not affect the algorithm settings for the single fragmentations in the main view, but

note that when a new pipeline fragmenter is added, it is initially configured analogously to its settings in the main view and not according to its default settings.

A very important setting in this context is the “returned fragments” setting that all three currently available algorithms have (in Scaffold Generator, it is called “side chain setting”). For the Ertl algorithm, for example, it allows you to define whether only functional groups, only alkanes, or both types of fragments should be returned. This is important here because all fragments generated in one pipeline step are processed with the next defined fragmenter.

One example use case of the pipeline functionality would be this: First, a deglycosylation step with the Sugar Removal Utility (SRU) to remove sugar moieties. Then, a Scaffold Generator step to extract side chains from the resulting aglycones, discarding rings and linkers. Finally, a functional group extraction using the Ertl algorithm to assess the functional groups found in ring side chains in the given data set. To set up this pipeline, the first fragmenter needs to be set to the SRU via the drop-down menu, and the returned fragments setting needs to be set to “only aglycone” via the gear button. The second fragmentation step (added via the “+” button) needs to be set to Scaffold Generator, and its side chain setting set to “only side chains”. Also, the fragmentation type setting should be set to “main scaffold” because parent scaffolds are not needed to generate the side chains. Finally, the third fragmentation step needs to be set to the Ertl algorithm, and its returned fragments setting set to “only functional groups”. Here, the “Filter single atoms setting” should also be set to “false” for the given use case. By default, the Ertl algorithm fragmenter does not process input molecules that consist of only one heavy atom. But in the given example, we would like to keep ring side chains like hydroxy groups or chlorine substituents and be able to detect them as functional groups in the final pipeline result.

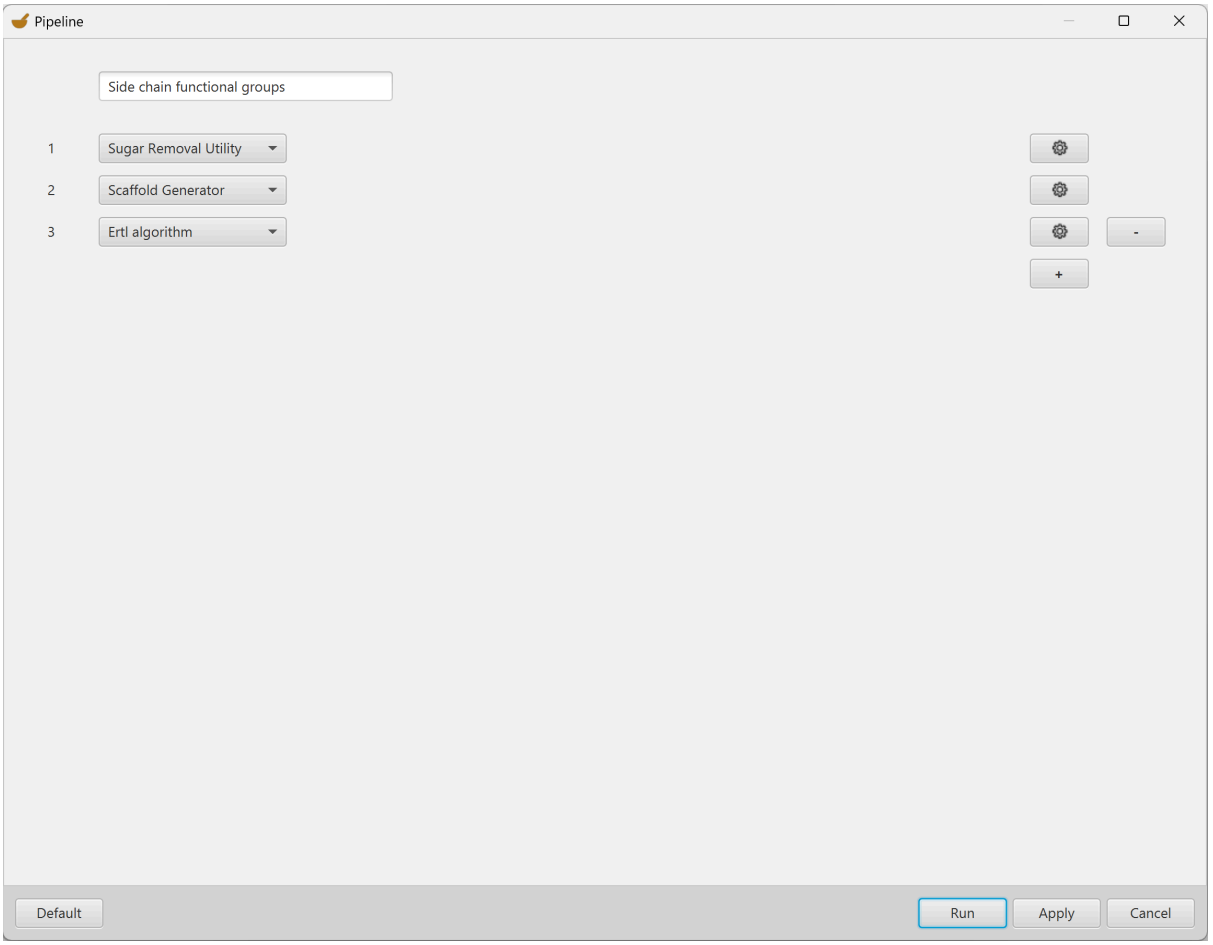

Figure 13: Pipeline configuration to extract side chain functional groups.

When the fragmentation has been set up (Figure 13), the settings can be saved (and the dialog closed) using the “Apply” button in the bottom-right corner of the dialog. “Cancel” discards all changes made to the pipeline definitions, and “Default” (in the bottom-left corner) resets everything to default values (compare Figure 12). To start the pipeline fragmentation as defined (and save the settings), use the “Run” button in the bottom-right corner of the pipeline settings dialog. Using the BIOFACQUIM natural products collection, the results should look like Figure 14 when sorted for their frequency decreasing, i.e. displaying the five most frequent functional groups identified in the side chains. The most frequent group is a hydroxy group that is connected directly to the ring system, followed by a hydroxy group connected to an aliphatic carbon atom in the side chain. The third most frequent functional group is an ester (directly connected to a ring) or carboxylic acid, followed by an alkene side chain. The fifth most frequent functional group is an ester functionality that is more or less in the middle of a side chain (just not directly attached to any rings). For more details, inspect the parent structures overview or the items tab.

Note that the configured pipeline, like all other settings, is persisted at application exit and re-imported at the next session.

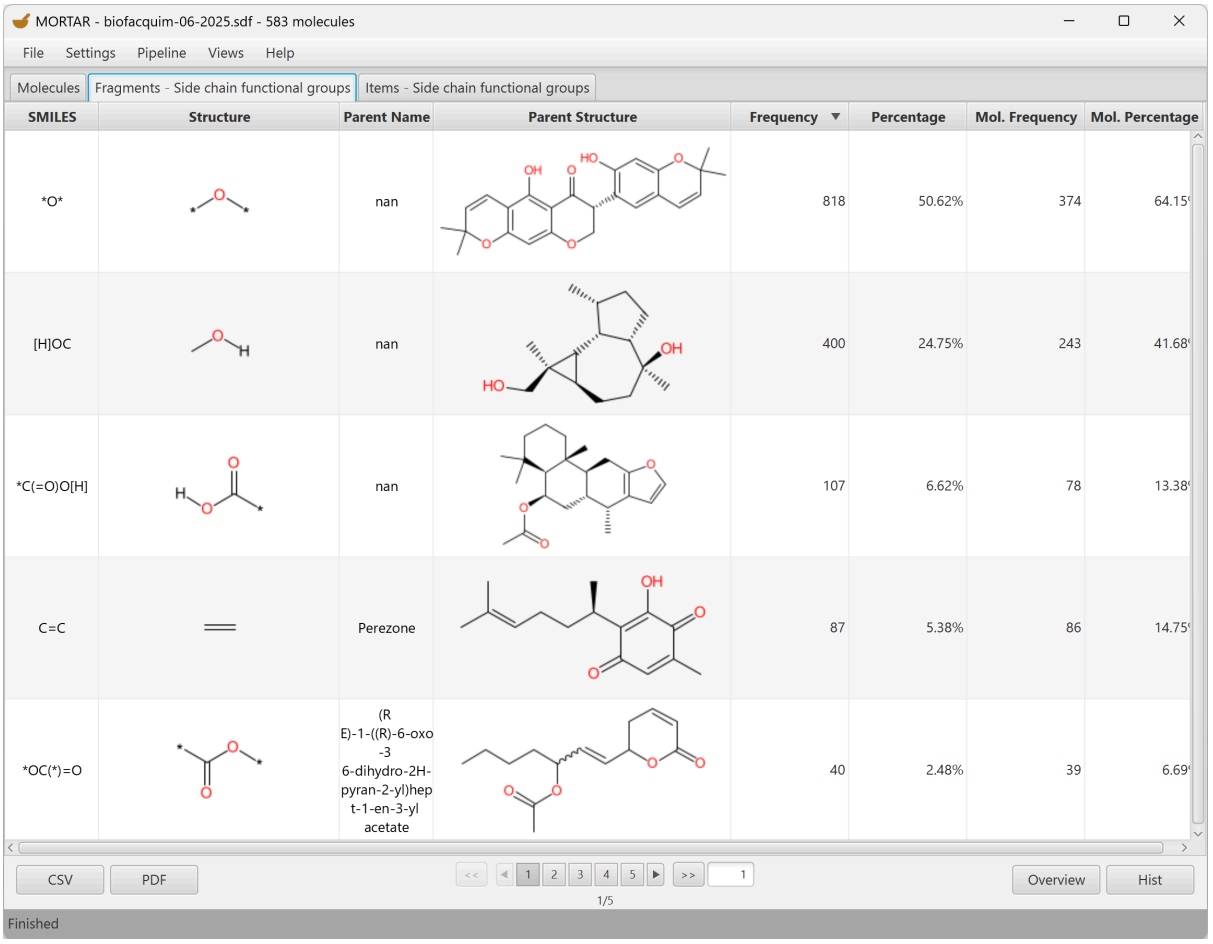

Figure 14: Side chain functional groups of the BIOFACQUIM natural products collection.

Overview view

The overview functionality mentioned in multiple places above serves as an alternative, grid-based visualization of molecular structures that allows you to inspect them more quickly than in the table-based MORTAR main window. The imported molecule set, the generated fragment sets, all fragments of an individual molecule, or all parent molecules of a specific fragment can be visualized this way (see Figure 15). It can be opened via the menu bar ("Views" -> "Overview"), via action buttons in the lower-right corners of the respective tabs, or via the context menu.

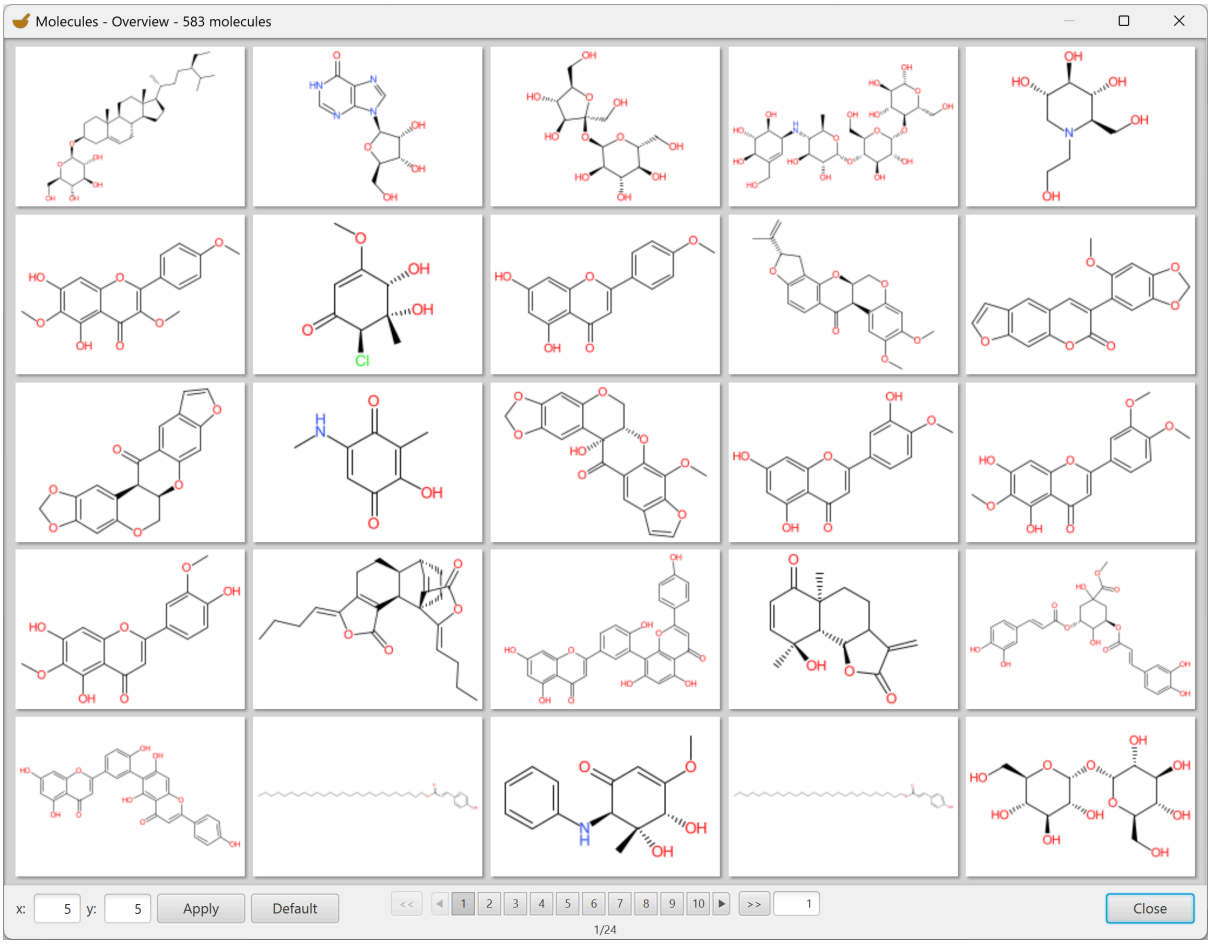

Figure 15: Overview view showing the first 25 structures of the BIOFACQUIM natural products collection.

The pagination control at the bottom can be used to switch between the individual pages of the view. Its handling here is analogous to the pagination controls in the MORTAR main window. Via the text fields and buttons in the bottom-left corner of the view, the number of rows and columns of structures that are displayed per page can be adjusted. Changes are

applied via the “Apply” button or by pressing the “Enter” key in one of the text fields. High input values that would cause the structure depictions to be smaller than a defined minimum size are ruled out by a window size-dependent maximum value for the x and y values. The “Default” button can be used to return to the default grid configuration.

The individual structures can be enlarged in an additional view, the so-called “Enlarged Structure View” (see Figure 16), with a single click on their depiction. In the overview view of the “Molecules” and the “Fragments” tab, there is also the option of double-clicking on a structure depiction to jump to the specific structure in the respective tab of the MORTAR main window. A context menu, which is also available in the enlarged structure view, offers the additional option to copy the respective SMILES code, Name/ID string, or image of each structure to the clipboard. In the “Parent Structures Overview”, which visualizes all parent molecules of an individual fragment, the fragment is placed in the first position on the first page of the view and highlighted, as can be seen in Figure 17. The same applies to the “Items Overview”, where the fragmented molecule is highlighted against its fragments. The overview window can be closed by clicking on the “Close” button in the bottom-right corner.

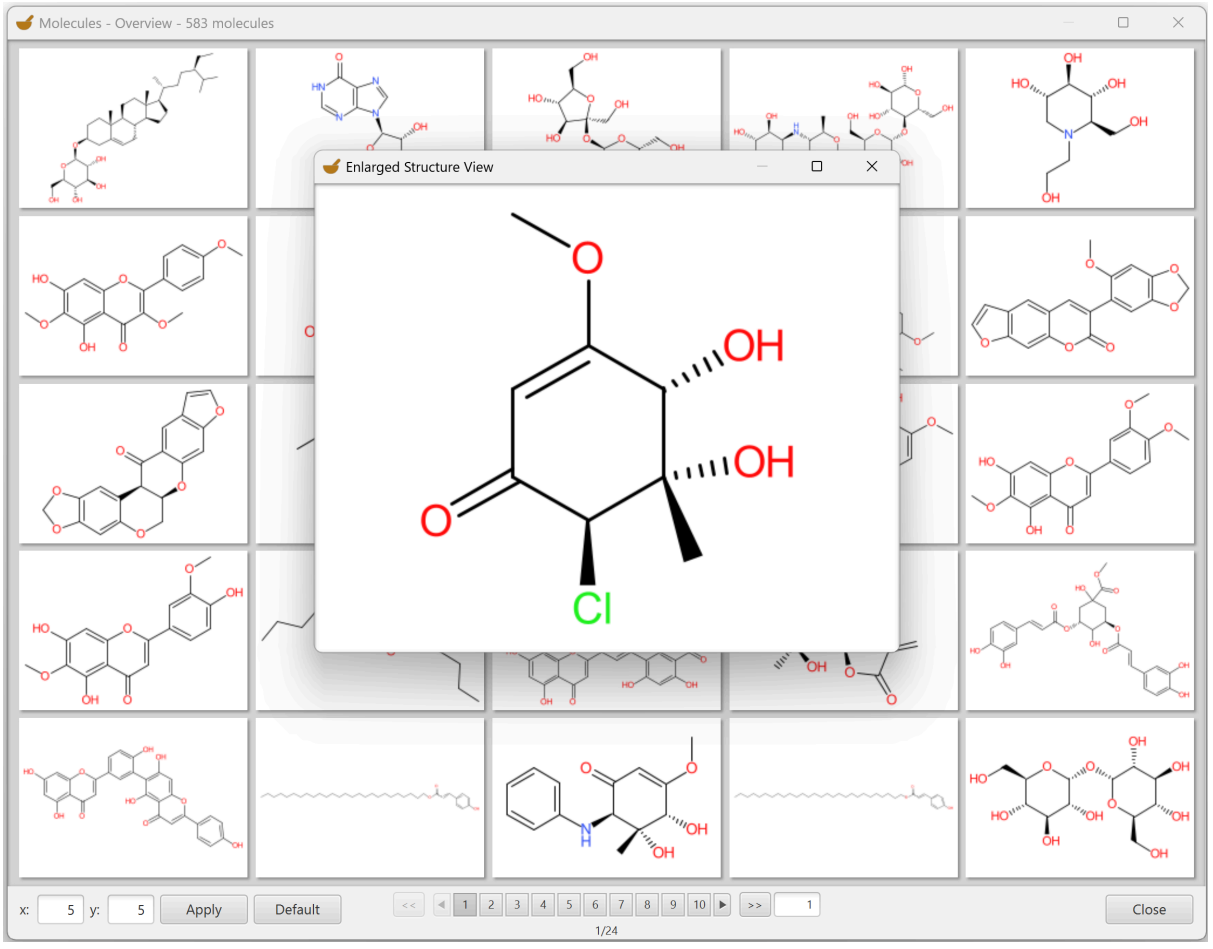

Figure 16: Depiction of a structure in the enlarged structure view.

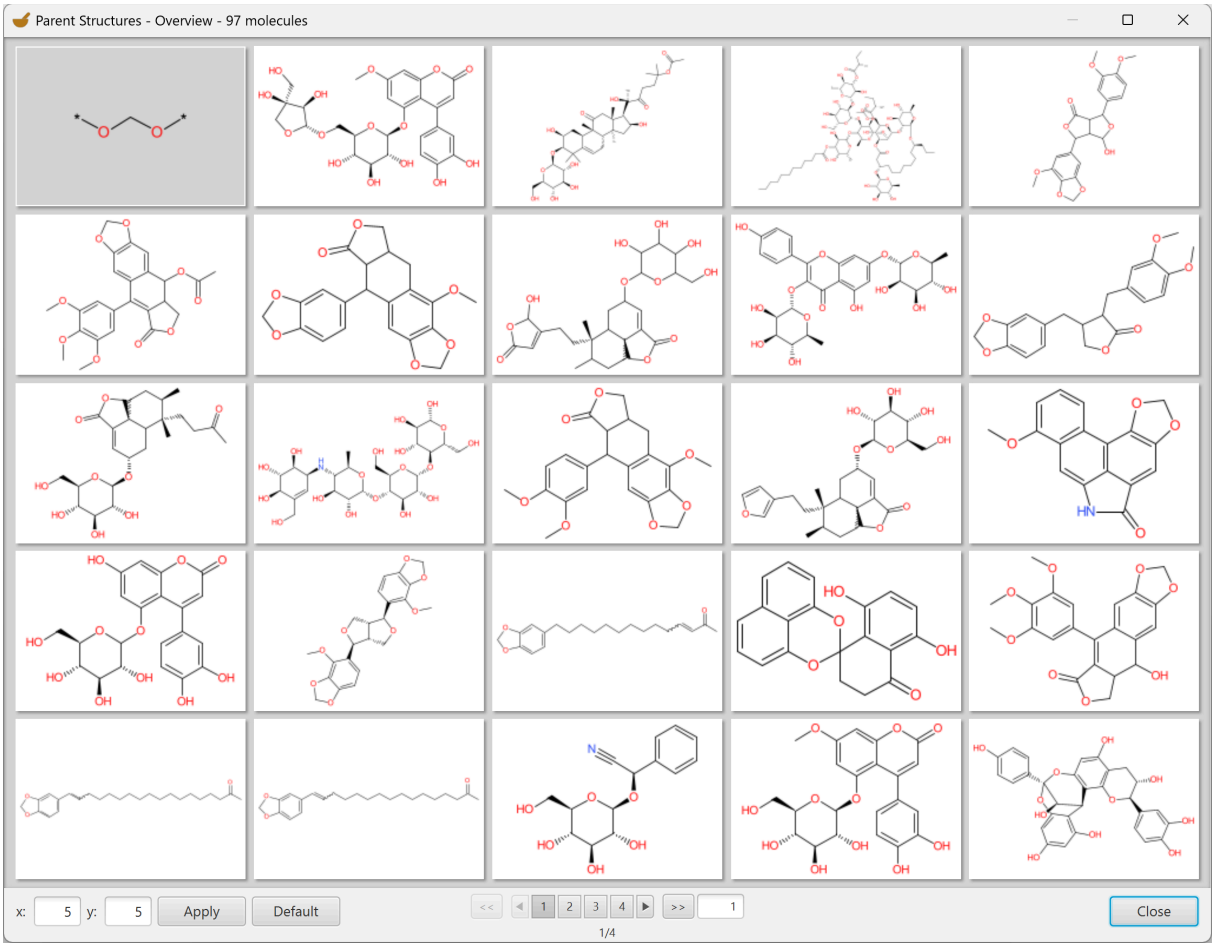

Figure 17: Overview view of all parent structures of the highlighted fragment.

Histogram view

Another visualization option for fragmentation results in MORTAR is the histogram view (Figure 18). When a fragmentation process is finished, the histogram view can be opened via “Views” -> “Histogram” in the menu bar at the top of the MORTAR main window or via the “Hist” button in the lower-right corner of the “Fragments” or “Items” tab.

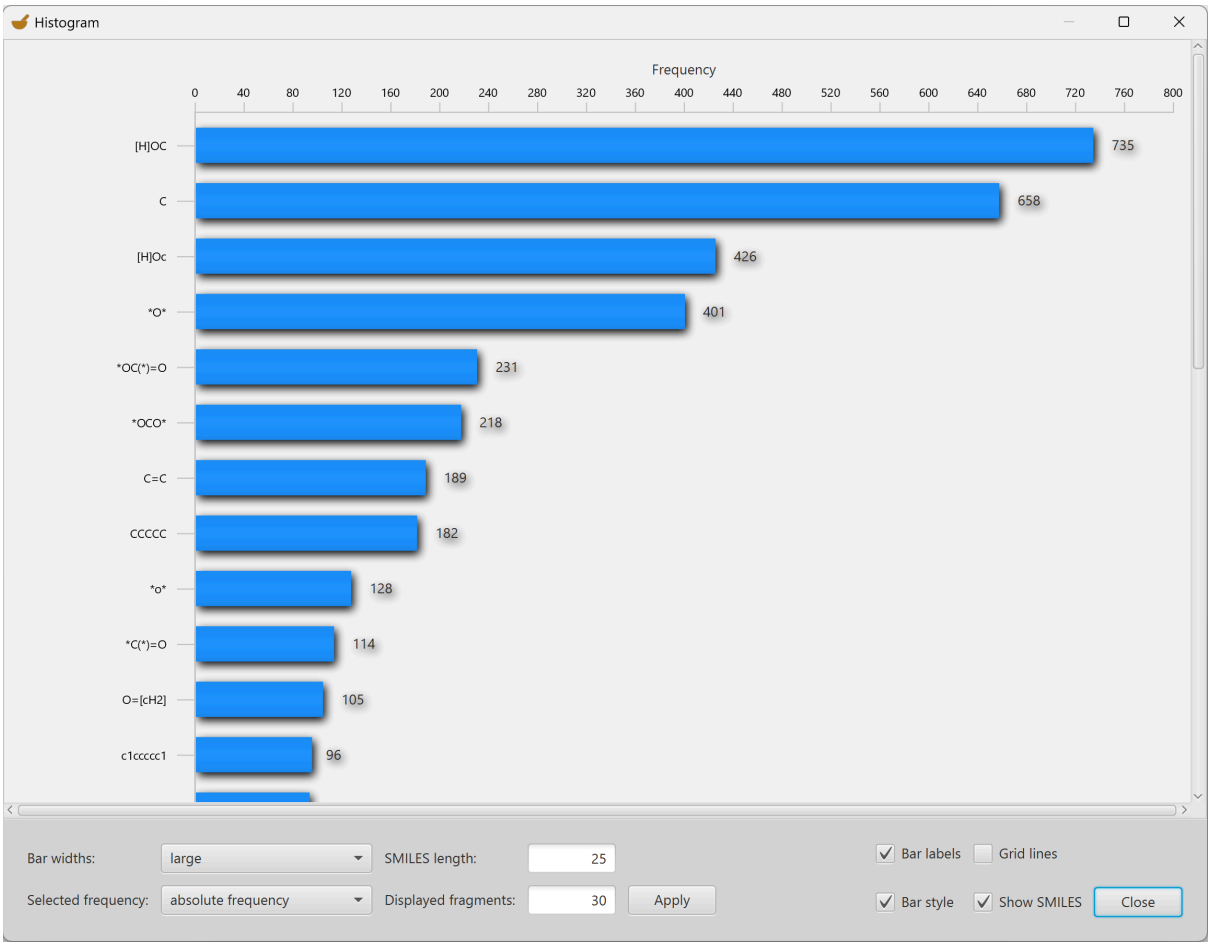

Figure 18: Histogram view.

The histogram displays the fragment frequencies in decreasing order, i.e. the most frequent ones are at the top. On the left axis, the fragment structures are listed as SMILES strings. By hovering over a bar, the respective fragment structure can also be displayed as a 2D structure diagram in the bottom-right corner of the window (Figure 19).

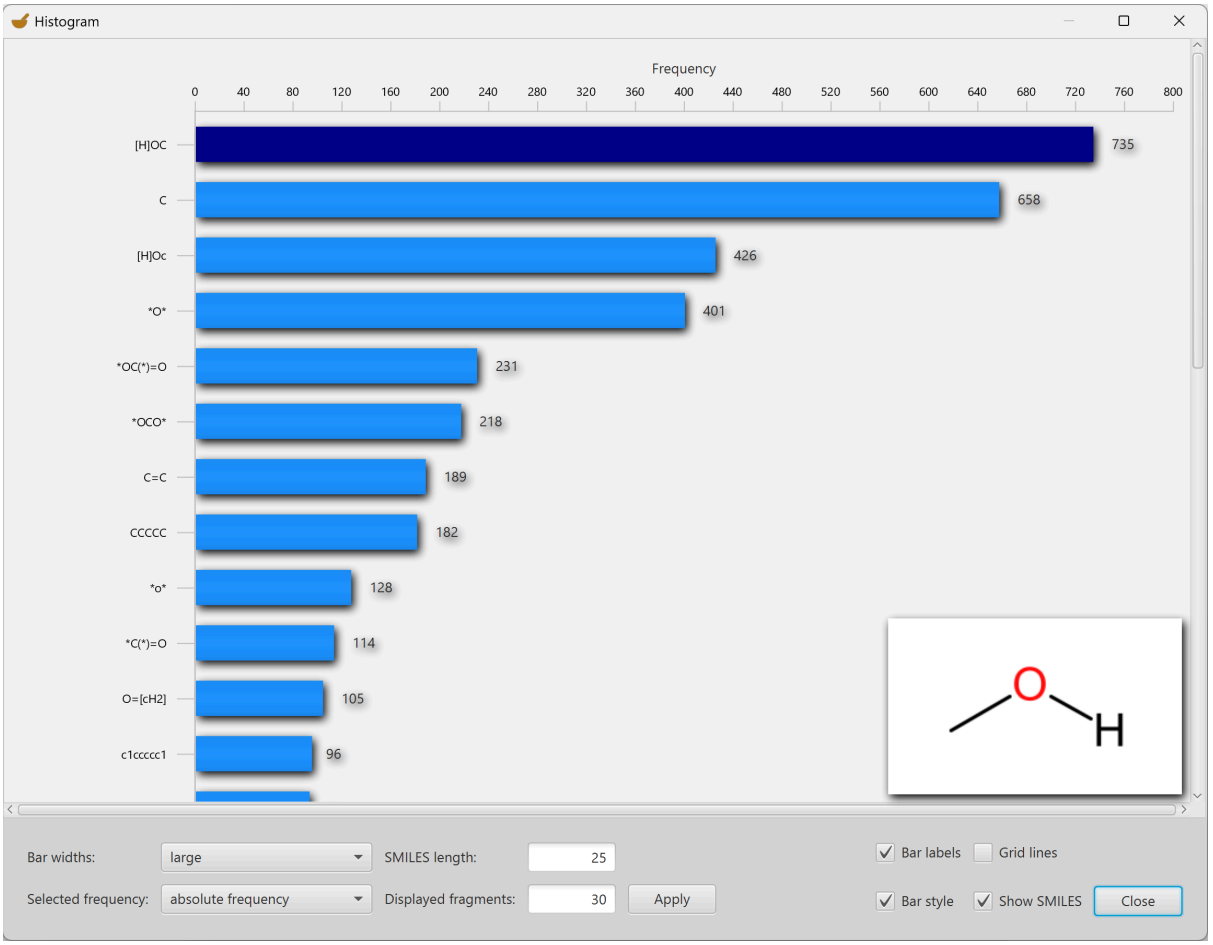

Figure 19: Structure display in histogram view.

At the bottom, multiple settings can be made to the visual appearance of the histogram. The “Bar widths” setting can be used to adjust the bar widths and hence, how many fragment frequencies are visible at once. Use the scroll bar at the right to see the other fragment frequencies below. The overall number of fragments included in the histogram can be set via “Displayed fragments”. To, e.g., display only the 10 most frequent fragments, enter “10” into the respective text field. Via the “Selected frequency” drop-down menu, it can be set whether the fragment frequency (absolute occurrence of one fragment, see above) or the molecule frequency (number of molecules the respective fragment appears in, see above) should be displayed on the bars. The “SMILES length” setting can be used to set the maximum SMILES string length to display on the left axis. If a SMILES representation is longer than the defined maximum, the label on the axis will display “SMILES too long” for the respective fragment. Note that changes to these four described settings in the bottom-left corner only come into effect after the “Apply” button is clicked.

The settings at the bottom right of the histogram view can be used to show or hide the frequency labels next to the bars and the grid lines in the histogram. Also, the shadows of the bars can be deactivated. The “Show SMILES” option can be used to show/hide the

fragment SMILES representations on the y-axis of the histogram. Figure 20 shows the histogram view with alternative styling using some of the described options.

By right-clicking on a bar, the SMILES code or structure depiction of a fragment can be copied to the clipboard.

The histogram view can be closed by clicking the “Close” button in the bottom right corner.

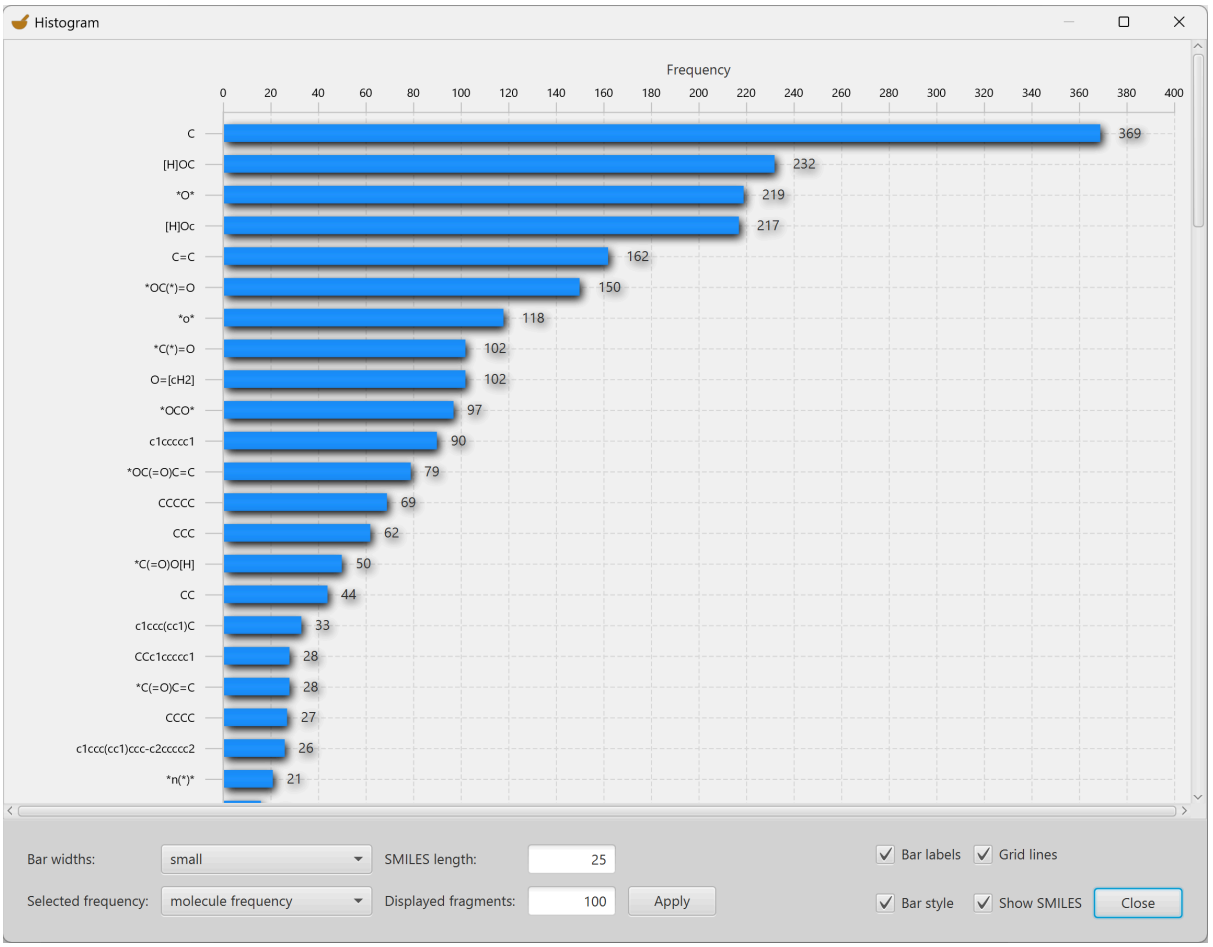

Figure 20: Alternative styling for histogram view.

About view

In the main menu bar, select “Help” -> “About” to open the “About” dialog of MORTAR. Here, the software version, licence, acknowledgements, contact information of the developers, and information on the external open software projects MORTAR uses can be found (Figure 21).

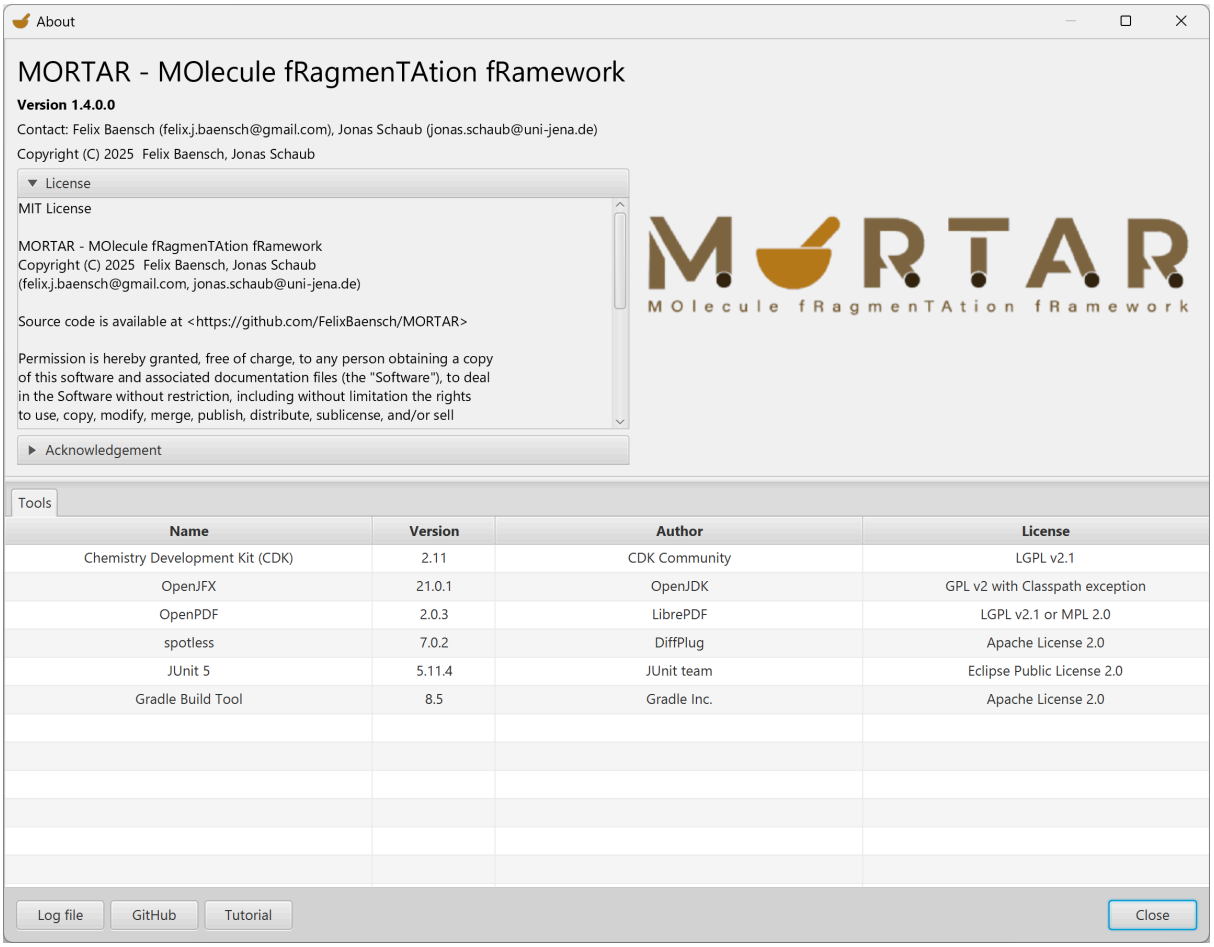

Figure 21: About view dialog.

The first button in the bottom-left corner opens the log file directory in an OS-dependent file explorer. Internally, MORTAR writes information, like how many molecules were imported and how many fragments were generated, to a text-based log file in this directory. Most importantly, exceptions/problems that might occur during an application session are logged there. For every session, a new log file is created, and older ones are deleted after some time.

The second button opens the MORTAR GitHub repository in your standard browser. Here, all important information about MORTAR can be found, like the most recent version of the software and known issues. To update your MORTAR distribution, download the most recent

version from GitHub. You can also have a look at the source code there. If you have a problem with MORTAR, have any questions, or want to suggest a new feature, please post this in the “Issues” section of the GitHub repository. If it is a problem, please describe it as well as you can and attach the respective log file to your issue. This helps us to understand and solve the issue faster.

The “Tutorial” button opens this tutorial document in your standard PDF file viewer (only available on Windows OS; macOS and Linux users will be redirected to the online version of the tutorial document in the GitHub repository).

Application exit

The MORTAR session can be ended via the OS-dependent controls at the top of the main window or via “File” -> “Exit” in the main menu bar in the top-left corner.
A warning will be displayed that all generated fragmentation results will be lost when the application is shut down. If you want to save them, export them in a format of your choosing (see above). Note that no export format can later be used to fully restore the fragmentation result as it is displayed in MORTAR (with fragments and items tab).

MORTAR checks at start-up that there is only one instance of the application running at a time. Running multiple instances can cause problems in internal file access processes, like logging and settings persistence. If MORTAR crashes, it may display a warning at restart that there could be a second instance already running. In that case, this warning can be ignored, but otherwise, it is not recommended to run multiple MORTAR instances at once.

Integration of new fragmentation algorithms

MORTAR is intended to support the development of new *in silico* fragmentation algorithms in a way that the integration of additional fragmentation algorithms should be achievable in a straightforward manner. In principle, all substructure analysis logic that accepts one molecule as an input and returns one or multiple substructures of it as output can be integrated if it is implemented in Java based on the Chemistry Development Kit. The new functionality has to be wrapped in a Java class implementing the `IMoleculeFragmenter` interface in the package `de.unijena.cheminf.mortar.model.fragmentation.algorithm`. When this wrapper class is implemented, it has to be linked in the `FragmentationService` class in the package `de.unijena.cheminf.mortar.model.fragmentation`. All details on how to achieve the implementation and the linking are described in the JavaDoc documentation of the `IMoleculeFragmenter` interface, and examples of how to do it can be found in the same package in the wrapper classes for the fragmentation algorithms already available in MORTAR. To get started inspecting and augmenting the MORTAR source code, follow the installation instructions for MORTAR as a software project in the GitHub readme document. If you want to contribute to MORTAR publicly and openly, feel free to start a pull request to the MORTAR repository on GitHub after successfully adding your new functionality. If you need any assistance, feel free to contact us via GitHub, e.g. in an issue on the MORTAR repository.

Stereochemistry

Before version 1.4.0, MORTAR did not support stereochemistry, i.e. the internal chemical structure representation based on canonical SMILES did not encode this information, and hence, it was lost in the imported molecules and not defined in any resulting fragments. Since MORTAR v.1.4.0, there is the “regard stereochemistry” setting in the global application preferences (see above). It is turned on by default, which means that imported molecules retain their stereo configurations, and they are transferred to specific types of fragments during the fragmentation process (details below). This setting can be turned off to establish backwards compatibility with earlier MORTAR versions. If it is turned off, processes like molecule set imports and fragmentations are faster because generating the internal SMILES representations is less computationally demanding if stereochemistry is disregarded. In our tests, importing the COCONUT natural products database was about three times faster without considering stereochemistry.

For the fragmentation algorithms, the inherent treatment of stereochemistry is different in every individual case (if the setting is turned on):

Ertl algorithm for functional group detection:
The generated functional group fragments do not retain the stereo information of their parent molecules because even though Ertl’s functional groups can be quite large, stereochemistry is usually not of interest when assessing the functional group diversity in a given data set. The alkyl remnants, on the other hand, still carry their original stereo information because they can be considered scaffold-like building blocks where the stereo configuration is important (as an example, see the fragmentation results of 3-epiursolic acid in Figure 22).

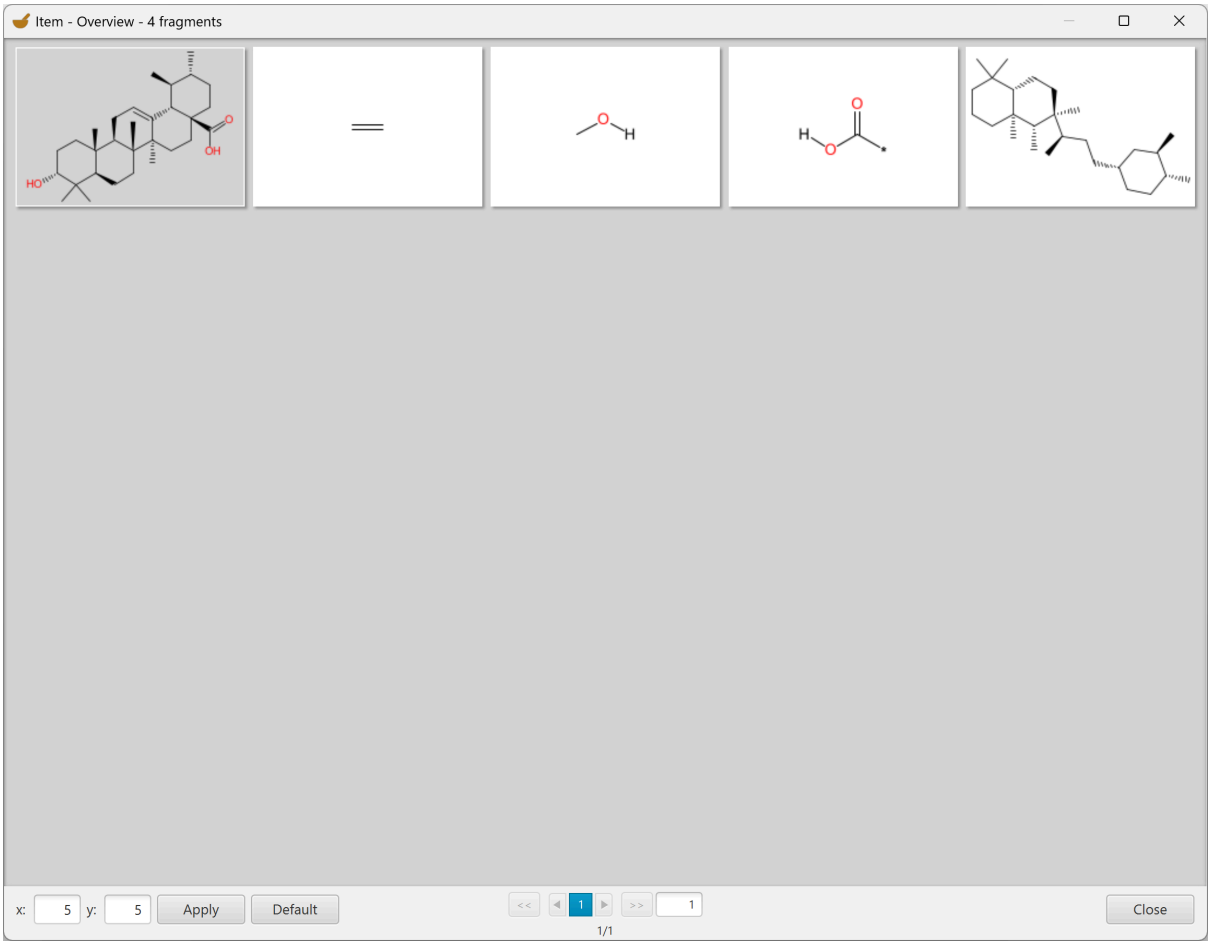

Figure 22: Functional group and alkane fragments of 3-epiursolic acid (CNP0219063.1, part of the BIOFACQUIM dataset); the functional groups have lost their stereo configuration, while the alkyl remnant has retained it.

Sugar Removal Utility (SRU):

The generated aglycones still carry their original stereo information because deglycosylation just removes the sugar part of a glycoside.

The focus during the development of the Sugar Removal Utility was on producing a sensible aglycone structure and not on detecting the glycosidic substructures correctly. Because of that, extracted circular sugars were only represented by their furanose or pyranose rings, did not carry any substituents, and hence, had no stereocenters. This situation was largely improved with the Sugar Detection Utility extension that is used in MORTAR internally since version 1.5. Now, the sugar structures are extracted correctly (see Figure 23). The new feature also added new settings to the Sugar Removal Utility which are documented in the description linked above.

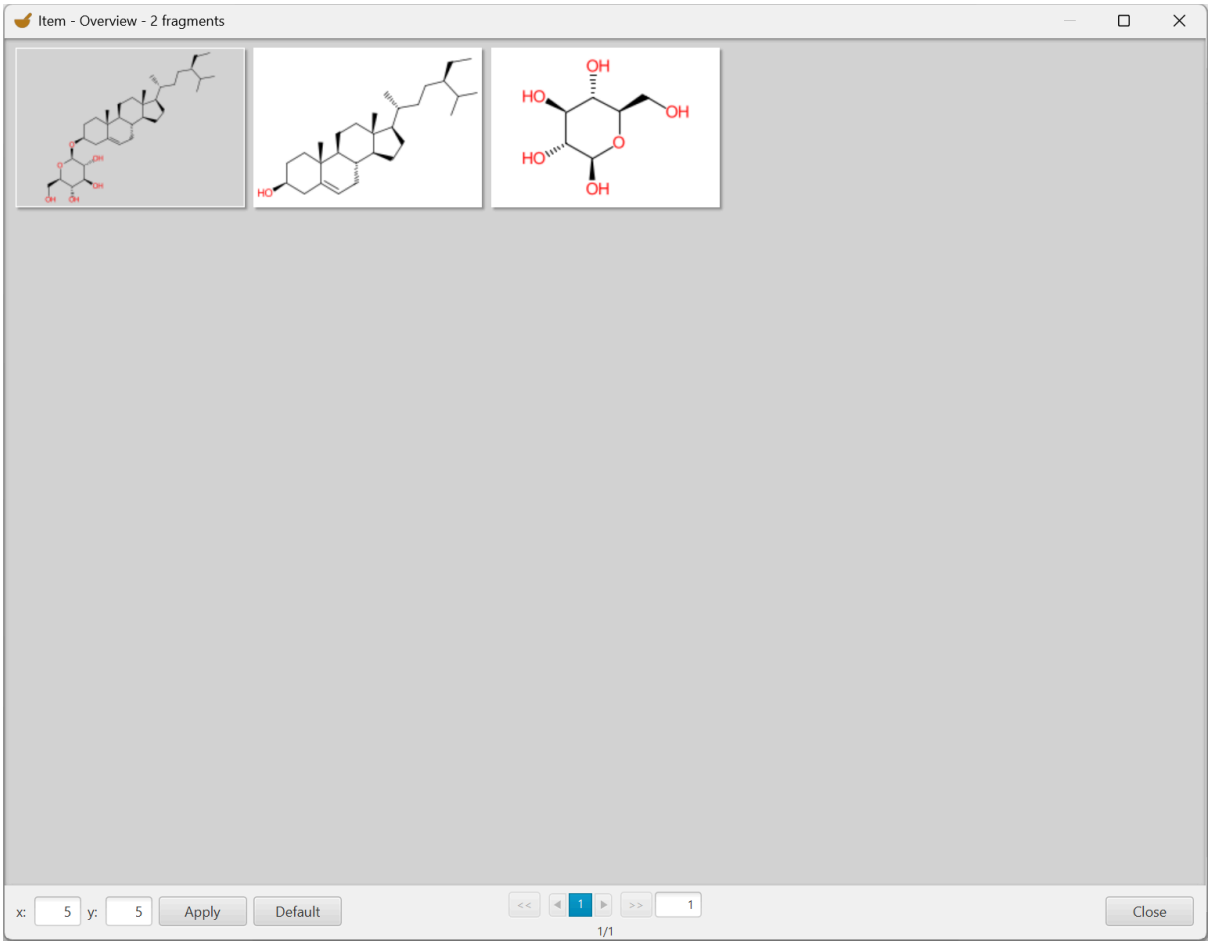

Figure 23: Sugar Removal/Detection Utility fragmentation result of Daucosterol (CNP0324146.1, part of the BIOFACQUIM dataset); aglycone and sugar moiety retain their stereo configurations.

Scaffold Generator:

Scaffold and side chain fragments generated by Scaffold Generator retain their original stereo information if possible, but many scaffold atoms lose their stereo configuration when their side chains are clipped off, of course (Figure 24). In the fragmentation settings, you can also define whether stereochemistry should be considered when the parent scaffolds generated for one molecule are deduplicated in the scaffold tree or scaffold network fragmentation schemes.

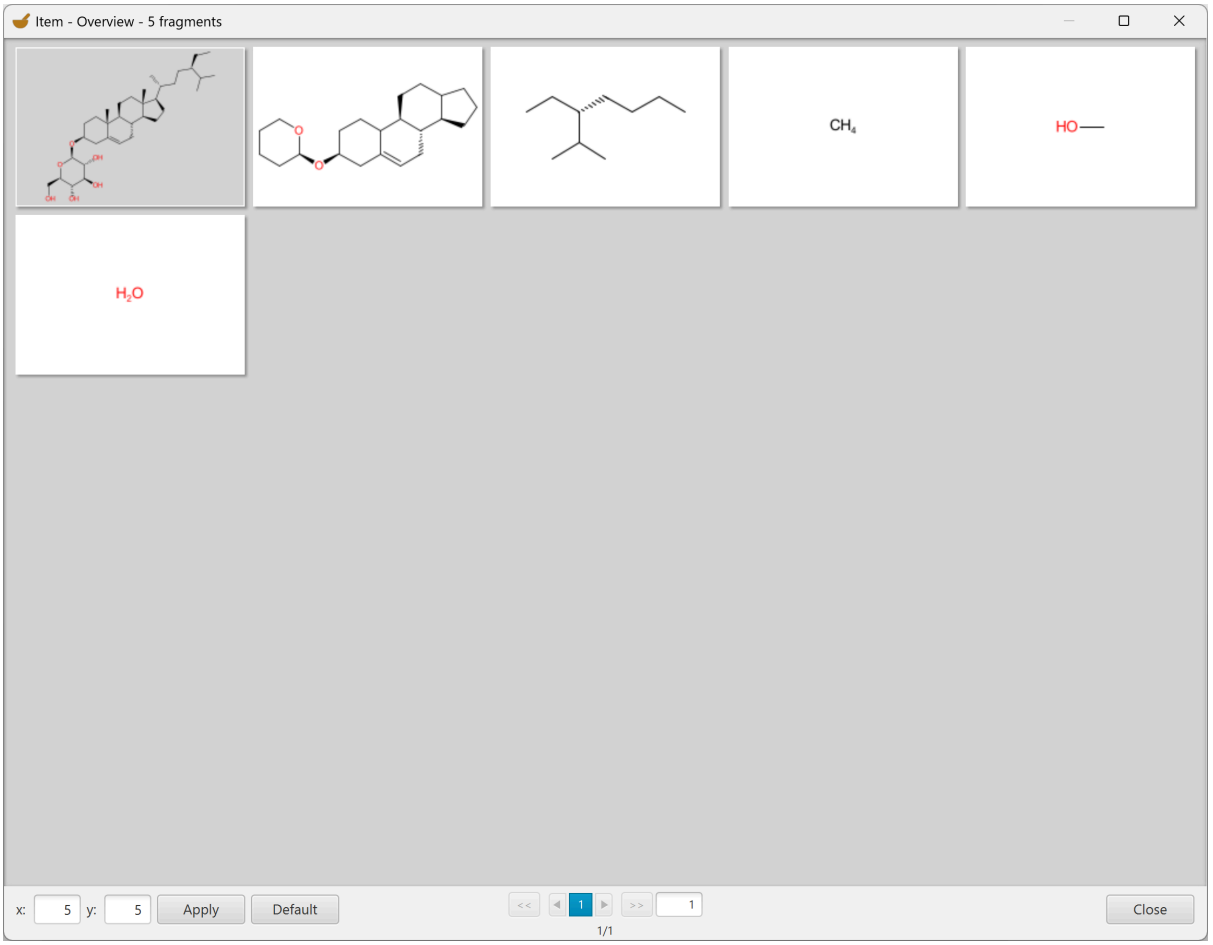

Figure 24: Scaffold Generator fragmentation result of Daucosterol (CNP0324146.1, part of the BIOFACQUIM dataset); The scaffold and side chain fragments retain their stereo configuration if possible.

For those types of fragments that carry stereo configurations, note the following general caveats: If ligands / side chains - e.g. hydroxy groups - are extracted as functional groups, the formerly connected carbon atom in the alkyl remnant might not be a stereocenter anymore as a result of the fragmentation (see Figure 22, the two former functional group attachment points in the alkyl fragment, or Figure 24, the pyranose ring in the extracted scaffold). Another possible effect is that a stereo center in the alkyl remnant might change its R/S configuration because a functional group ligand is replaced by a hydrogen atom. But, in this case, the overall 3D stereo configuration of the carbon atom still remains the same, just the priorities and hence the R/S label might change. Also note that the depiction might change (or definitely does in most cases, see, e.g., the alkyl remnant in Figure 22).

Known issues

Handling of polymers

For example, the ChEBI compound number 15338 represents “a poly(glycerol phosphate) having an alanyl group attached to the hydroxy function of the repeating unit”. The information about the polymer (which atoms and bonds are included in the repeating unit and where the brackets should be placed in a depiction) is included in the MOL and SD files, but gets lost when imported into MORTAR because of the internal conversion to the SMILES format. Therefore, MORTAR handles the polymer as only one entity of its monomer (Figure 22).

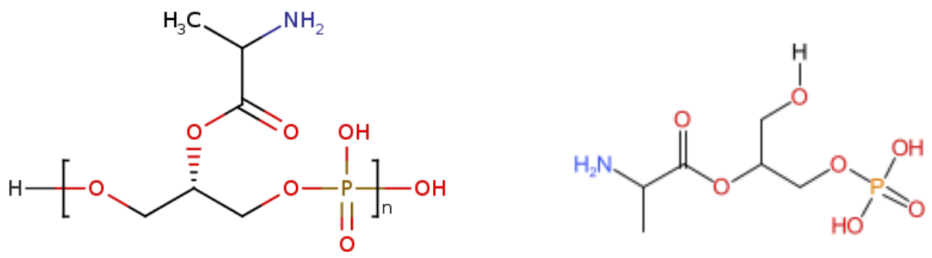

Figure 26: Representation of ChEBI compound 15338 in ChEBI as a polymer versus representation in MORTAR as a monomer.

Explicit Hydrogen atoms

MORTAR generally turns explicit hydrogen atoms of imported molecules into implicit hydrogen atom counts of the connected heavy atoms. But in some cases, e.g. when the hydrogen atoms are needed to define stereochemical configurations, they are not turned into implicit hydrogen counts. This has the effect that, e.g. among the non-functional group fragments of the Ertl algorithm, an H_2 fragment (SMILES code “[H][H]”) can appear. In 51,000 ChEBI lite 3-star compounds, for example, this fragment appears 114 times.

Non-standard bond orders in MOL/SD files

The CDK MOL/SD file reader functionality cannot parse bond orders higher than 4 (aromatic bond; higher bond orders are interpreted as query bonds). For example, the intended metal complexation bonds in ChEBI compound 48572 (pseudocoenzyme B_{12}) (Figure 23) raise an error at the import of the structure into MORTAR from an SD or MOL file because they are defined there as bond order 8. The SMILES representation (that is internally used by MORTAR in general) does not support such bond types.

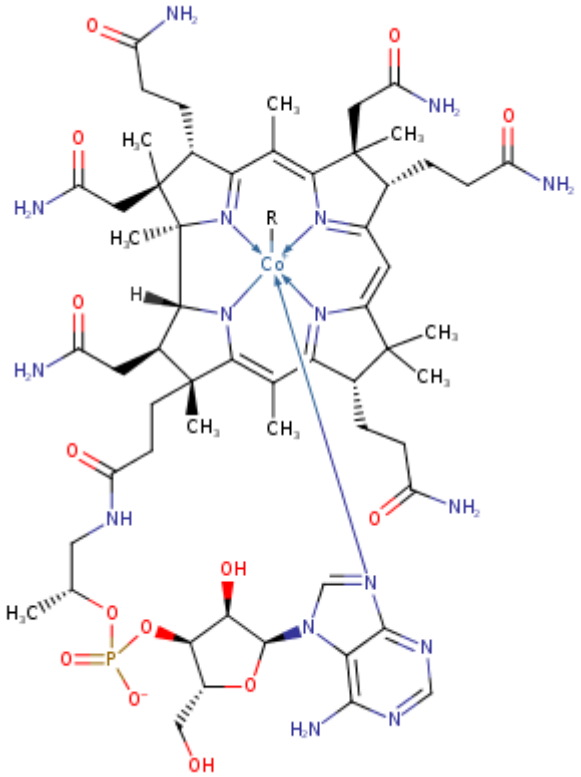

Figure 27: Representation of pseudocoenzyme B12 in ChEBI with metal complexation bonds.

Scaffold Generator fragmentation runtime

The “enumerative fragmentation” routine of Scaffold Generator generates every possible parent scaffold of a given molecule, i.e. every possible ring system that can result from the removal of one ring in the first iteration, two rings in the next iteration, etc., until all one-ring scaffolds are enumerated in the last step. Therefore, the number of fragments MORTAR generates with this routine and the time it needs for the process can scale close to exponentially with the number of rings in a molecule (since only terminal rings are

considered at every step and duplicate parent scaffolds are eliminated, both numbers usually scale below exponentially). While this is hardly noticeable when dealing with small molecules, it is important to know when analysing larger molecules, e.g. natural products. On a standard notebook, it took up to half a second to generate all possible parent scaffolds for a molecule with 10 rings in MORTAR. For COCONUT natural product CNP0075232, for example, a molecule with 20 rings, it took about 17 seconds and generated almost 2,500 fragments. For a molecule with 24 rings, like CNP0050552, it took more than a minute to generate 7,300 parent scaffolds. As said before, with more rings, the time consumption and fragment number can scale close to exponentially.

This is also important for the cancellation of a fragmentation process: A fragmentation thread can only be cancelled between the processing of two molecules, not while a molecule is processed. So if the processing of one molecule takes more than a minute, like in the last example above, the release of computing resources after the cancellation of a fragmentation process can take equally long (and longer for molecules with more rings).

Ertl algorithm fragmentation runtime

When processing a molecular data set with the Ertl algorithm for functional group detection in MORTAR with 1 vs. 2 parallel computation threads, the overall runtime scales in an unexpected way: processing the complete COCONUT database (406,747 molecules) with ErtlFunctionalGroupsFinder in default settings on a standard notebook took about 30 minutes employing a single processing thread (see global settings -> “Nr of tasks for fragmentation setting” above). With two parallel threads, it only took about 100 seconds to generate 35,791 fragments. This 18-fold speed-up instead of the expected 2-fold decrease in computation time is currently inexplicable to us. Therefore, we recommend using at least 2 parallel computation threads. For the other fragmentation algorithms available, this effect could not be observed.