Tutorial

Last update: 8th November 2022
Reference version of MORTAR: v1.0.6.0

Contents

Introduction **1**

Installation and start-up **2**
- Windows 2
- Linux and macOS 2

Molecule set import and preferences **4**

Single fragmentation **7**

Fragments tab **10**

Items tab **13**

Pipelining fragmentation **14**

Histogram view **18**

About view **21**

Application exit **23**

Integration of new fragmentation algorithms **24**

Known issues **25**
- Handling of polymers 25
- Explicit Hydrogen atoms 25
- Non-standard bond orders in MOL/SD files 25
- Scaffold Generator fragmentation runtime 26
- Ertl algorithm fragmentation runtime 27

Introduction

MORTAR ('MOlecule fRagmenTation fRamework') is an open software project that supports workflows of molecular *in silico* fragmentation and substructure analysis. The Java/JavaFX rich-client application offers extensive graphical functions for visualising the fragmentation results of individual compounds or entire compound sets. With several views and analysis functions, MORTAR supports the interpretation of *in silico* fragmentation results. In addition to three currently integrated methods for fragmentation and substructure analysis - ErtlFunctionalGroupsFinder, Sugar Removal Utility, and Scaffold Generator - MORTAR allows straightforward integration of additional fragmentation algorithms with automatic generation of settings menus. All cheminformatics functionalities are implemented based on the Chemistry Development Kit (CDK, https://cdk.github.io/).
The preprint of a scientific article describing MORTAR can be found here: https://doi.org/10.26434/chemrxiv-2022-zgg8s

Installation and start-up

Pre-compiled and executable MORTAR distributions can be found attached to the marked releases.

Windows

A convenient Windows OS installer executable for MORTAR is available (click here to automatically download the installer .exe of the latest version). Download the installer executable, start, and follow the instructions to install MORTAR. Note that the installation includes a full Java Runtime Environment (JRE). After installation, create a shortcut to an appropriate MORTAR start batch file on your Windows desktop. E.g. for MORTAR to use up to 4 gigabytes of RAM, copy a shortcut to the batch file "MORTAR.bat" which is located in the MORTAR program folder (default “C:\Program Files\MORTAR\MORTARv1.0.6.0\bin” or the path specified at installation). To start MORTAR, double-click the created shortcut. MORTAR can be uninstalled by the provided “Uninstall.exe” executable in the MORTAR program folder or by standard Windows functions.
As an alternative to "MORTAR.bat", there is also the "MORTAR_20GB.bat" batch file available that allocates up to 20 GB of RAM for MORTAR. If you want to configure your own heap space settings, open one of the provided batch files 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, try the guidelines for Linux and macOS described below. As an alternative way, they should also work on Windows.

Linux and macOS

Every release has the executable Java ARchive (JAR) "MORTAR-fat-1.0.6.0.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-1.0.6.0.jar
```

A JDK or JRE of version 17.0.4 or higher needs to 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.

Please note that MORTAR only supports x64 (on all three platforms) and AArch64 (on macOS and Linux) architectures in general. For the latter, a special "fat JAR" named "MORTAR-fat-aarch64-1.0.6.0.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.

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

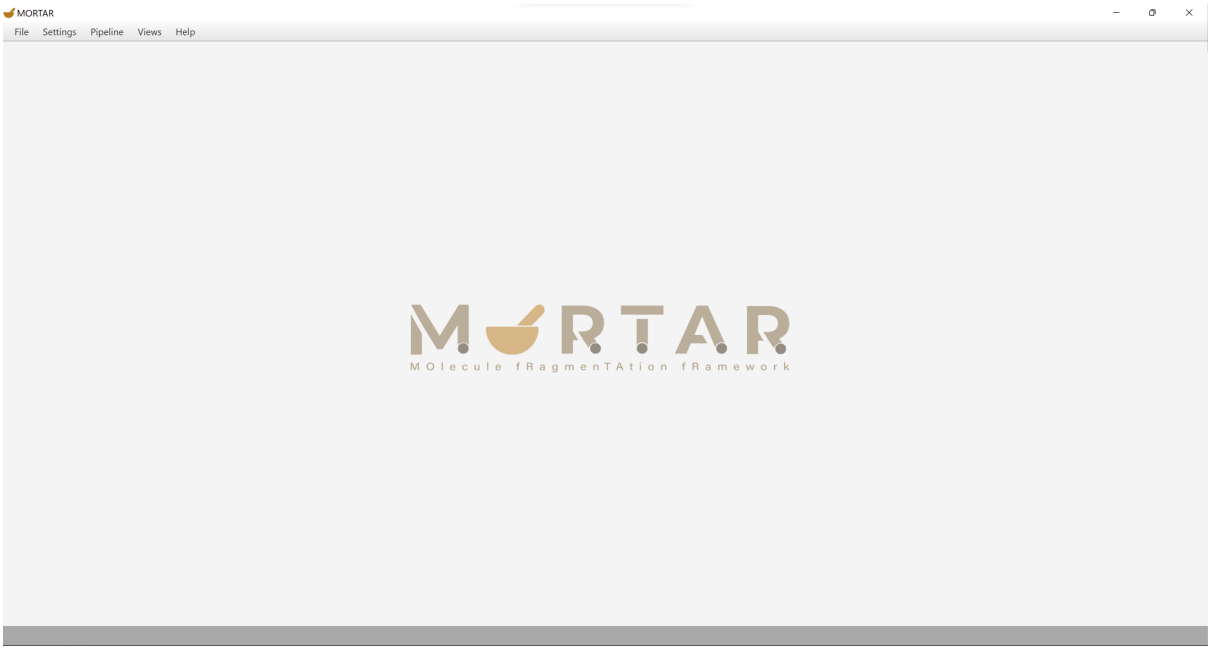


Figure 1: MORTAR main window.

Molecule set import and preferences

Before starting to work with MORTAR, inspect the global application preferences that can be found in the upper-left menu bar at “Settings” -> “Preferences” (Figure 2). A dialog opens that allows to adjust multiple settings relevant to different functionalities (Figure 3).

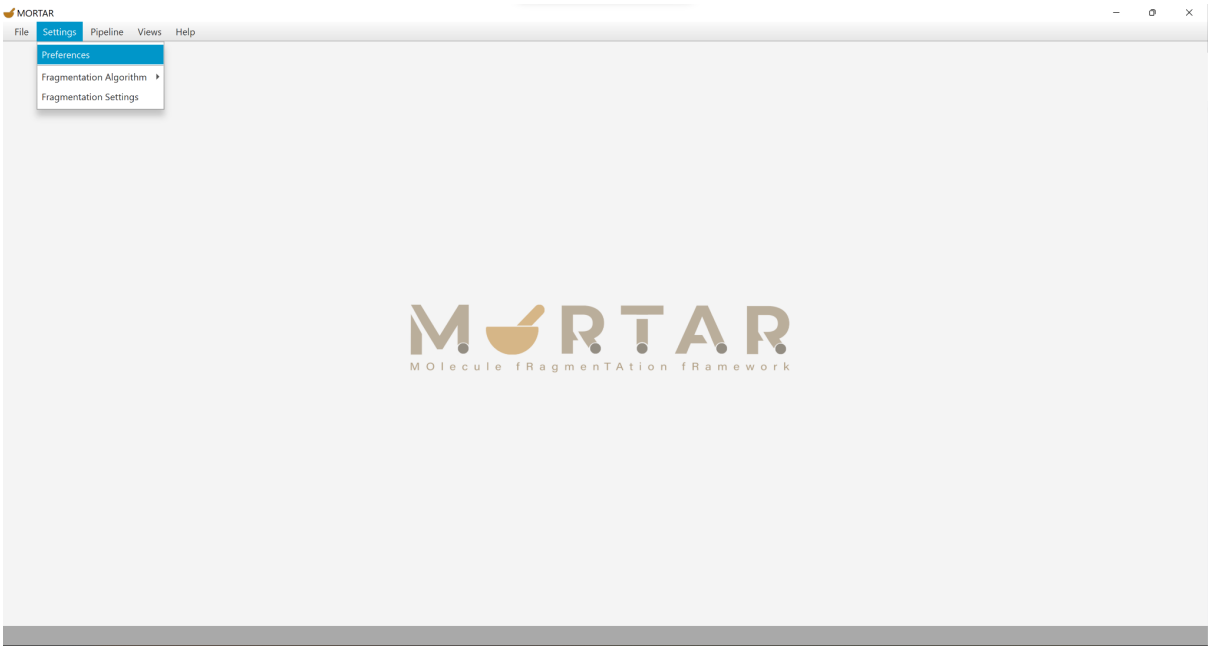


Figure 2: Settings menu.

For the molecule set import, it can be specified whether implicit hydrogen atoms should be added to fill possible open valances in the imported molecules. A short description for every setting is given in tooltips that appear when the cursor hovers for a few seconds over one setting element. Using the “Default” button in the bottom-left corner, all settings are reset to their default values. For saving your changes to the settings and closing the dialogue, click the “Apply” button in the bottom-right corner. With the “Cancel” button in the bottom-right corner, the dialogue is closed without saving the changes to the settings.

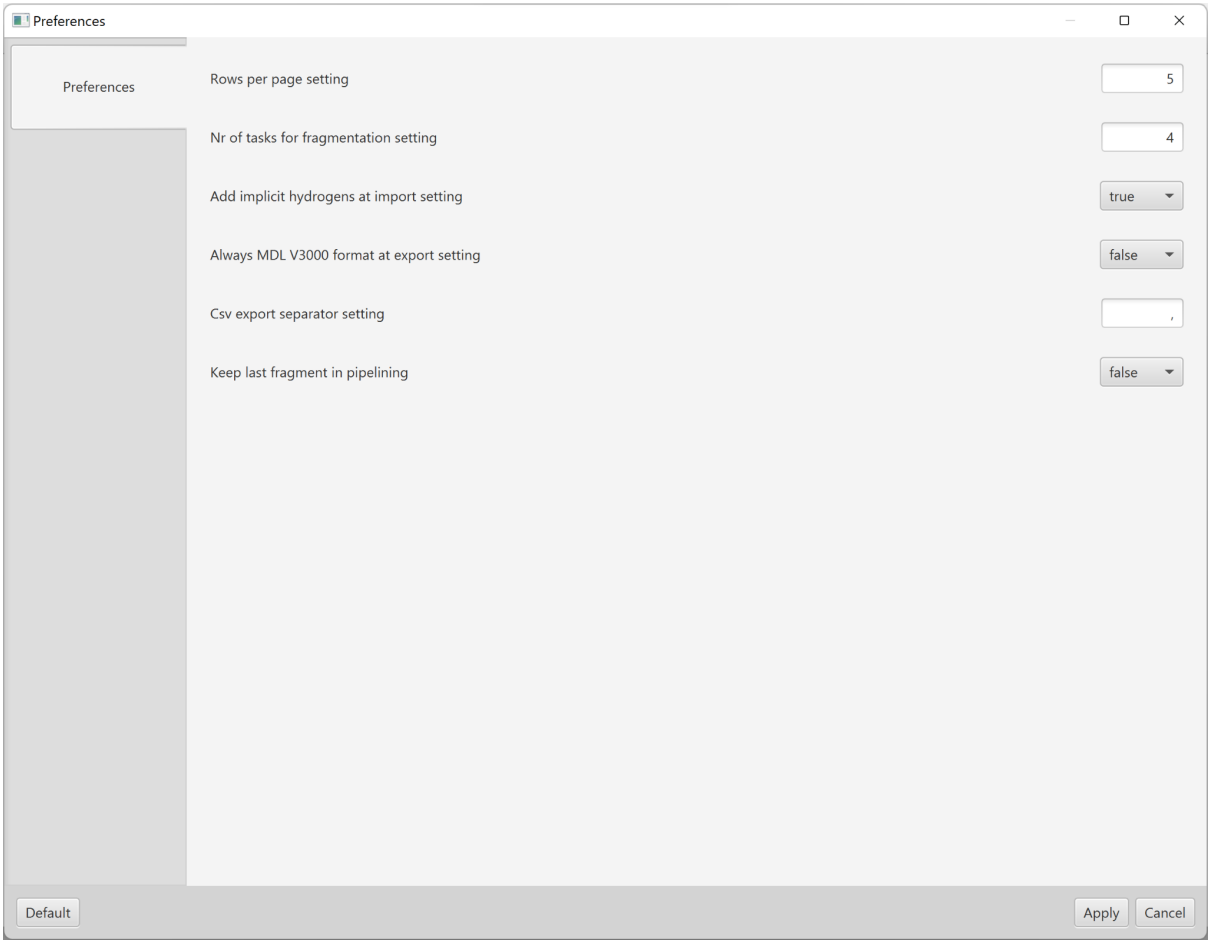


Figure 3: Global application preferences dialog.

To import a molecule set, open the “File” menu in the MORTAR main window menu bar and select the “Open” menu item (Figure 4).

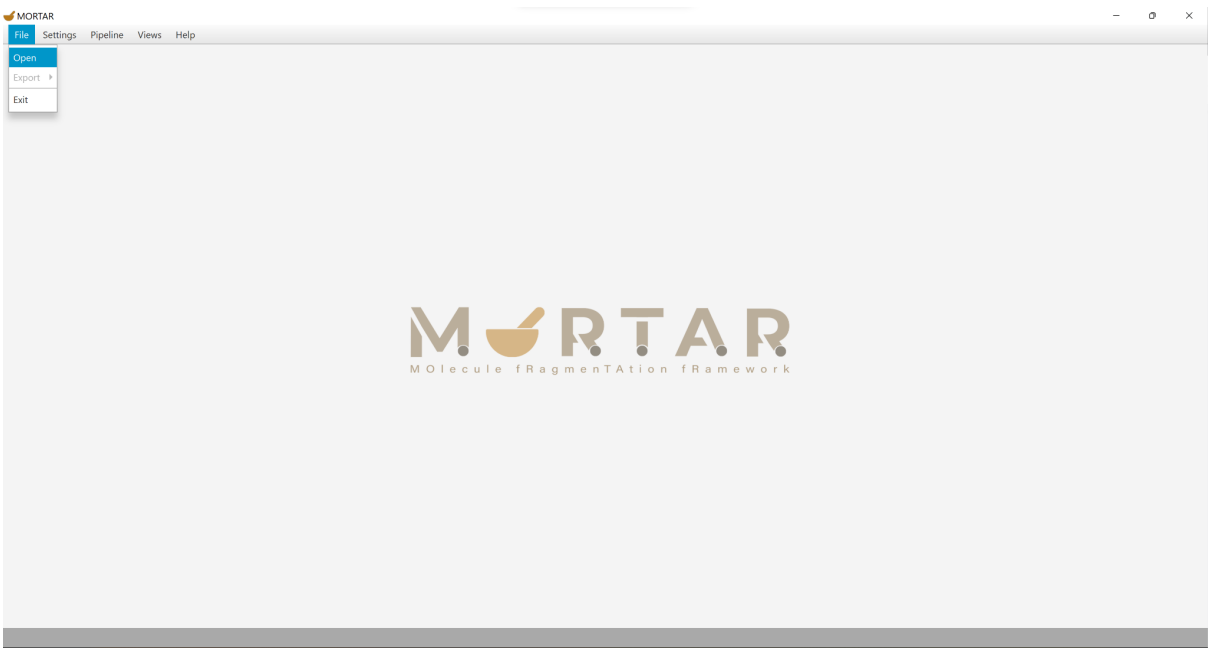


Figure 4: File menu.

An OS-dependent file chooser dialogue opens. Here, you can select a MOL, SD, or SMILES file to import. With this tutorial, an SD file containing the first 250 hits of an anthraquinone substructure search in the COCONUT natural product database (https://coconut.naturalproducts.net) is supplied. It can be found in the application's root directory in the "tutorial" folder, next to the PDF file of this tutorial. Alternatively, you can find it in the MORTAR GitHub repository: https://github.com/FelixBaensch/MORTAR/tree/master/Tutorial. After importing this set or any molecular data set, the "Molecules" tab opens (Figure 5).

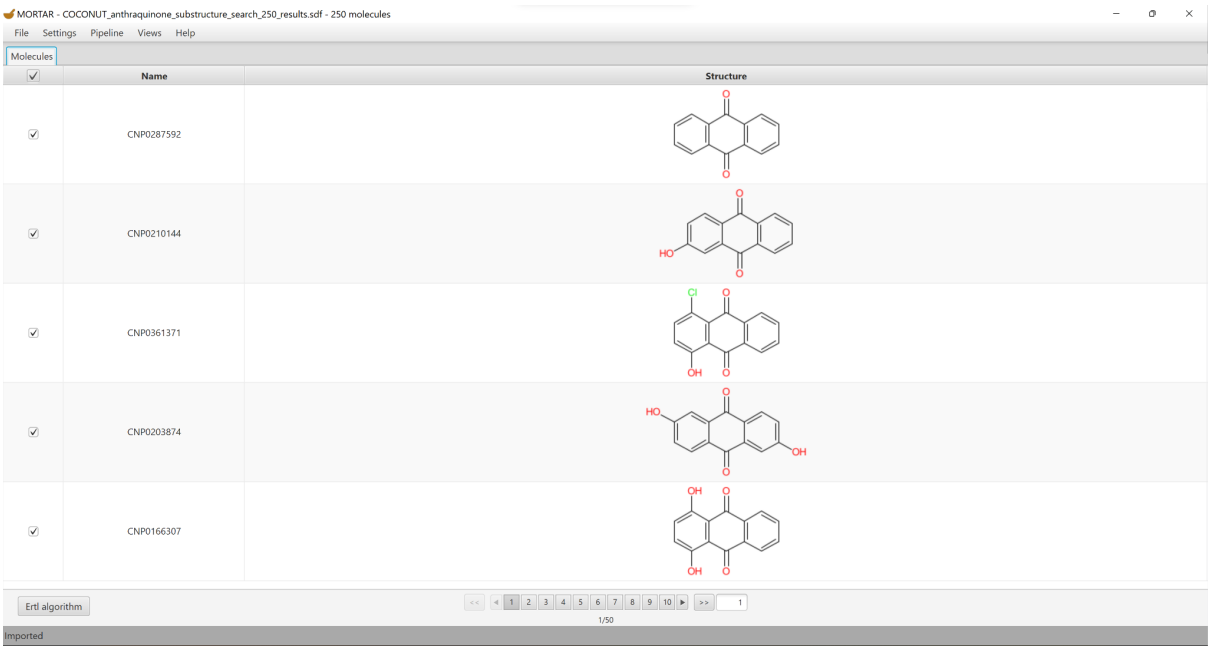


Figure 5: Molecules tab after import of the COCONUT anthraquinone substructure search set.

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 box 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 individual 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 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 how many structures/rows should be displayed per page.

Single fragmentation

Using the “Settings” menu, a fragmentation algorithm for a single fragmentation can be selected (Figure 6). In the current version, 3 algorithms are available:

- **Ertl algorithm for functional group detection:** The algorithm published by Dr. Peter Ertl (https://doi.org/10.1186/s13321-017-0225-z) is a 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 (https://doi.org/10.1186/s13321-019-0361-8).
- **Sugar Removal Utility (SRU):** The SRU is an algorithm and 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 (https://doi.org/10.1186/s13321-020-00467-y and https://doi.org/10.3390/biom11040486).
- **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 is described in this preprint: https://doi.org/10.26434/chemrxiv-2022-7tf0h-v2

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).

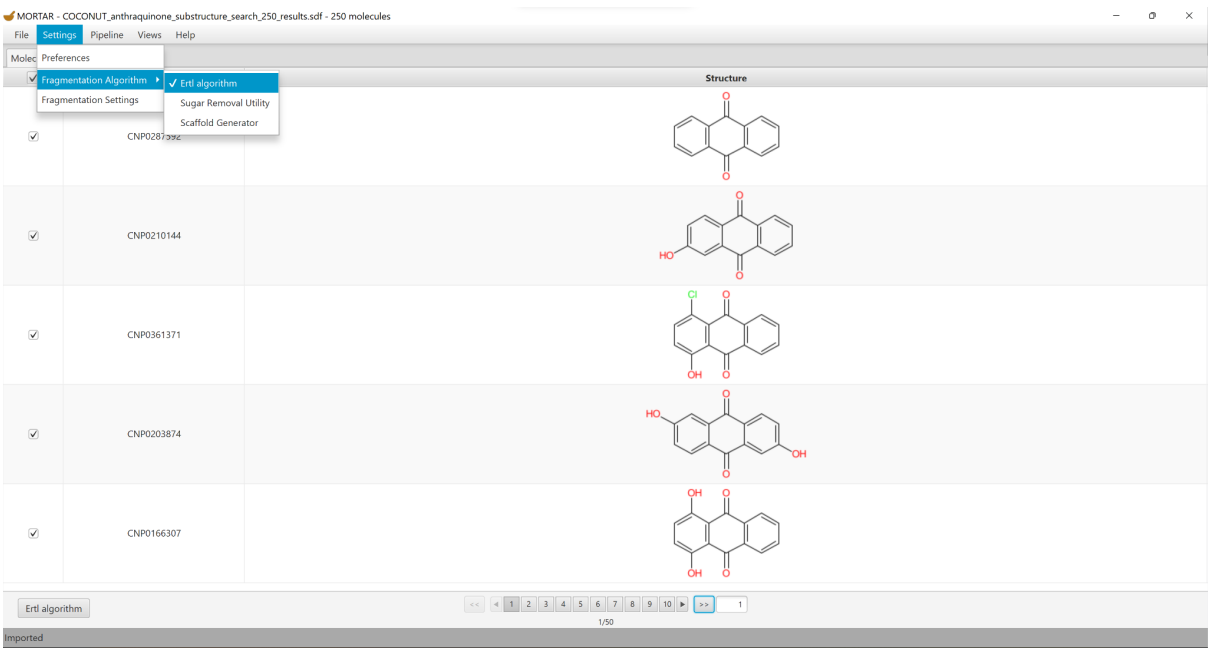


Figure 6: Fragmentation algorithm selection.

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 dialogue opens, displaying 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 indicated above. A short description is given in tooltips that appear when the cursor hovers for a few seconds over one setting element. 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 dialogue, click the “Apply” button in the bottom-right corner. With the “Cancel” button in the bottom-right corner, the dialogue 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.

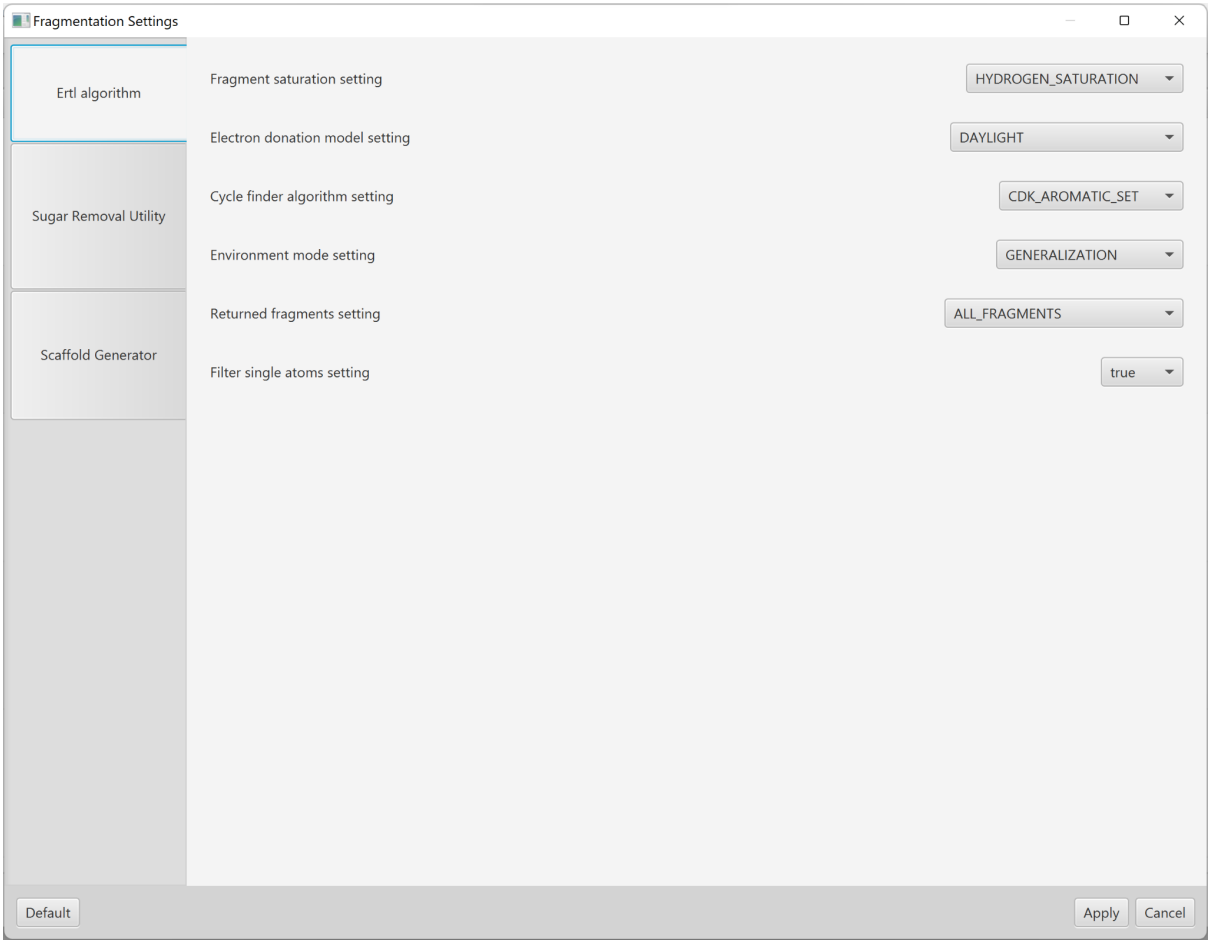


Figure 7: Fragmentation settings.

After adjusting the settings and closing the dialogue, the single fragmentation can be started using the “Ertl algorithm” button in the bottom-left corner of the “Molecules” tab (Figure 5/6). 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 box. In general, it is also not recommended to set the setting to this maximum value exactly because MORTAR anyway cannot block every available thread of your machine at a time.
Note that the fragmentation settings — as well as the global preferences — are persisted for the next session when the application is shut down.

Fragments tab

When the fragmentation is finished, two new tabs open to display the fragmentation results. One of them 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.

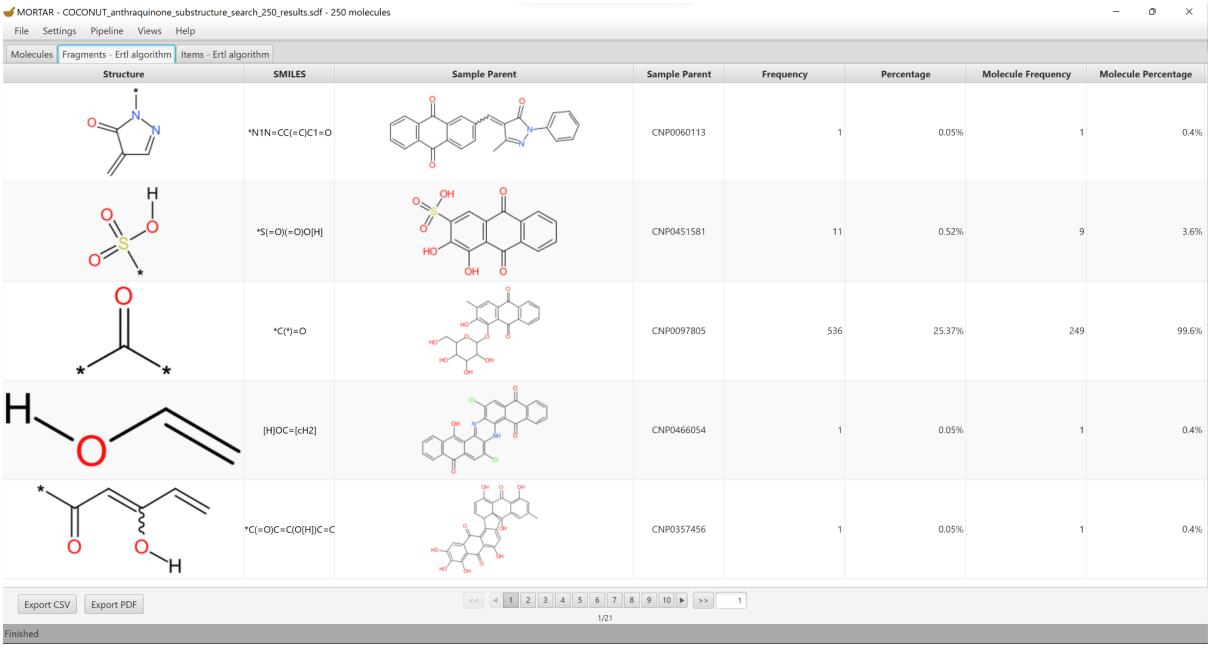


Figure 8: Fragments tab.

The first two columns display the fragment structures and the respective SMILES codes 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. The third and fourth columns contain the structure and name of the first molecule the respective fragment was identified in. The last four columns display the frequency of each fragment in the given data set. The simple “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 in how many molecules the respective fragment was identified, i.e. the same fragment appearing multiple times in the same 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 column values by clicking the column heads. E.g. by double-clicking the “Frequency” column label, the fragments are sorted according to their frequency descending and the most frequent fragments are displayed first (Figure 9).

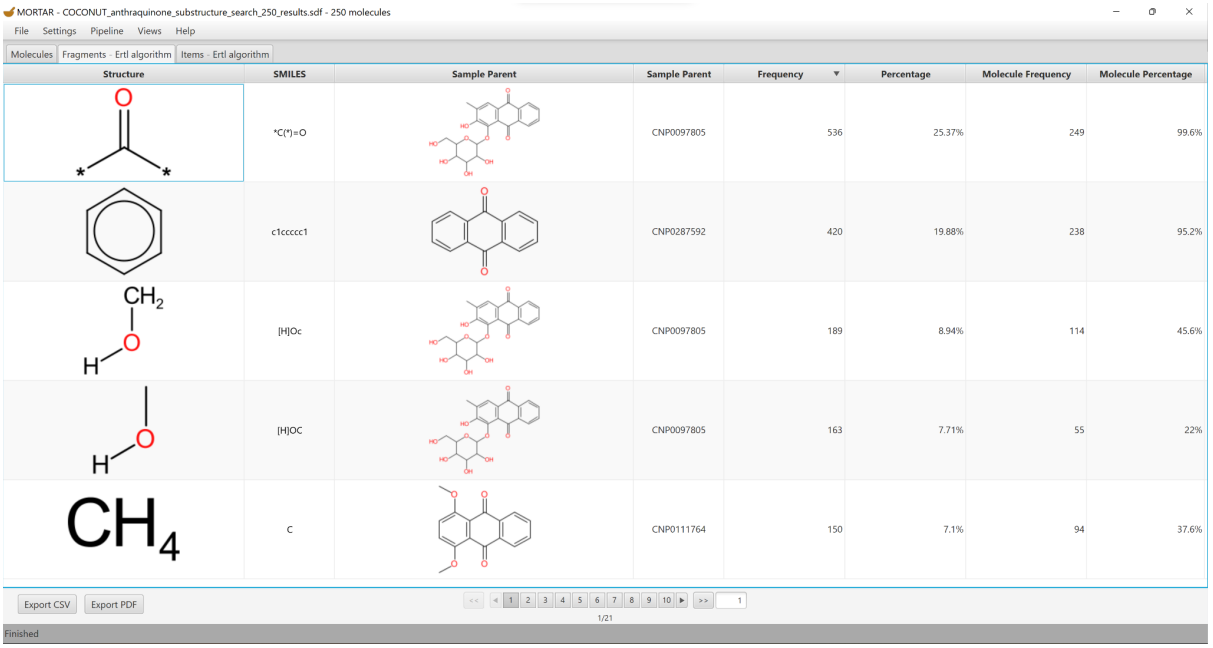


Figure 9: Fragments sorted with decreasing frequency.

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 dialogue allows to define where the created files should be written to. 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 (https://github.com/LibrePDF/OpenPDF). 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).

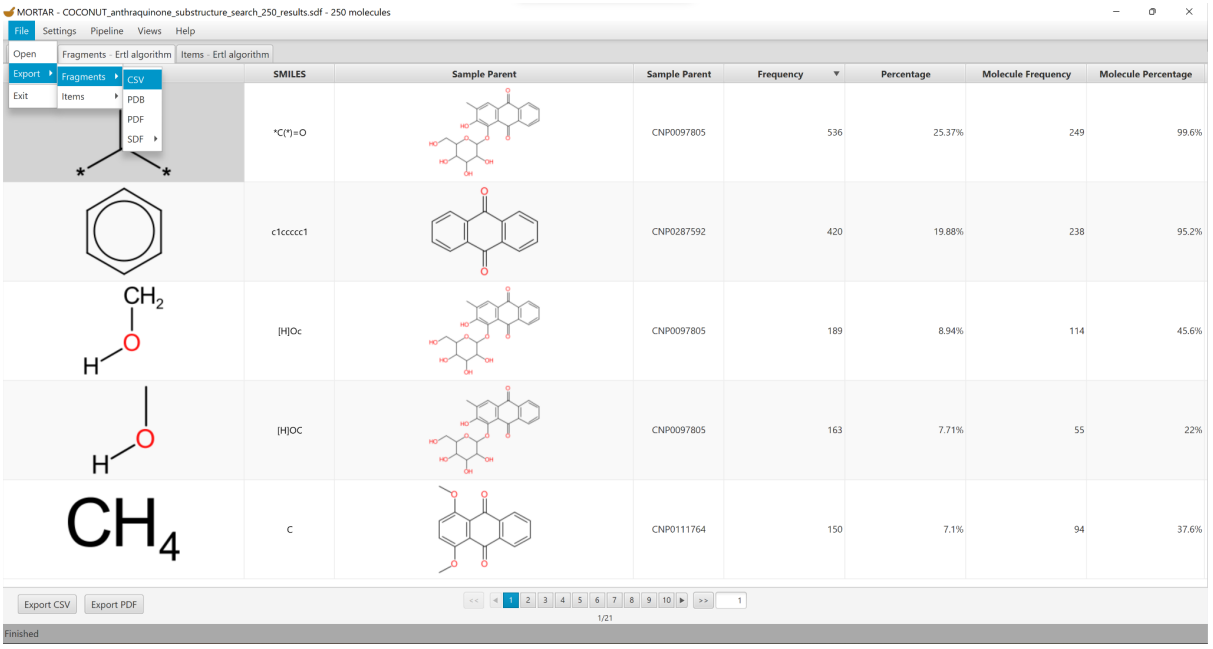


Figure 10: Fragments export menu.

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, respectively. 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 are generated for the fragments that are originally intended for graphical layout. 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 an individual itemisation of their resulting fragments.

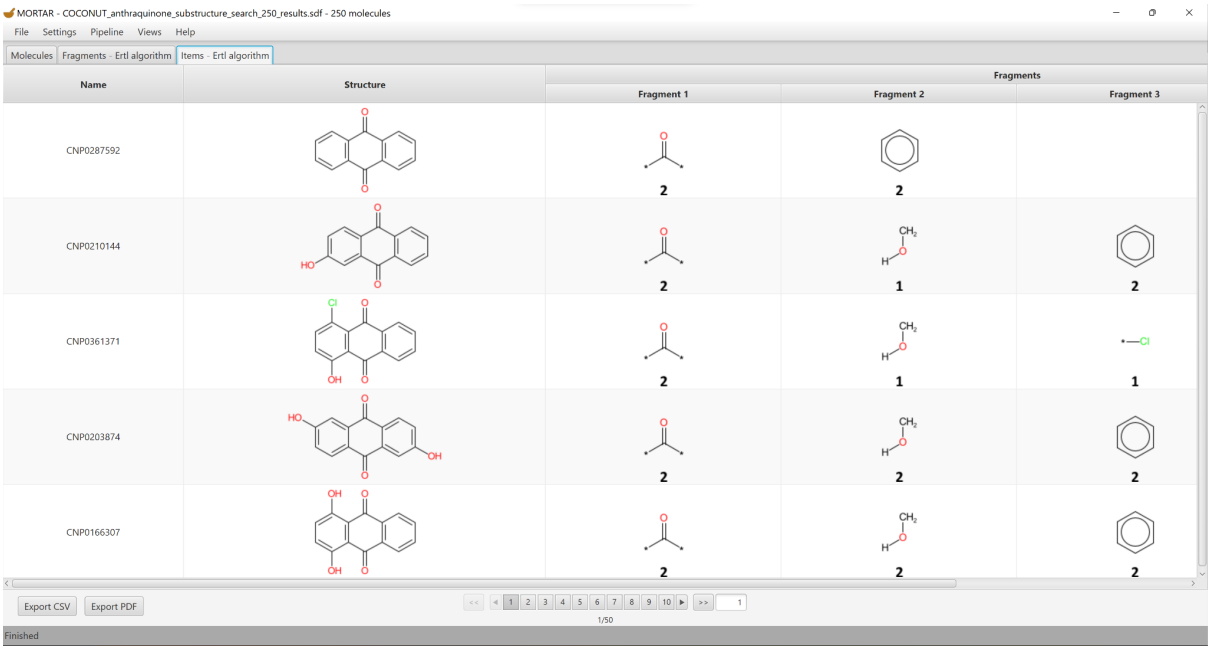


Figure 11: Items tab.

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

The fragments and items tab 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 results tabs remain. They are only cleared away when a new molecule set is imported. Please note that no tab is persisted when the application is shut down. Please export all data you want to keep.

Pipelining fragmentation

In the main 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).

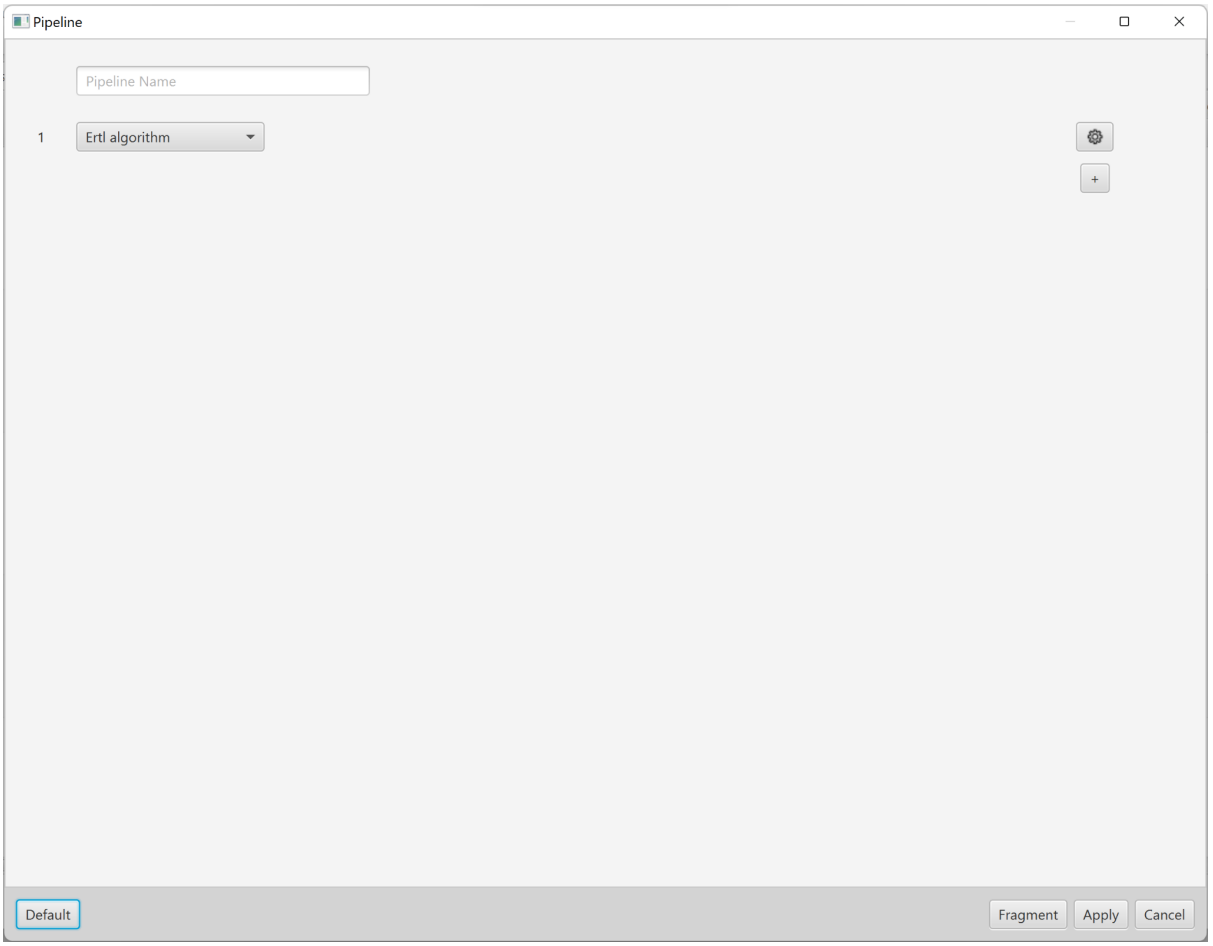


Figure 12: Pipeline dialog.

The text field at the top of the dialog allows the definition of 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. But they do not affect the algorithm settings for the single fragmentations described above.
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 to define whether only functional group, only alkane, 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.

Another important setting for the pipelining functionality can be found in the global preferences dialog (Figure 3), the “keep last fragment in pipelining” setting. If it is set to “false”, a molecule producing no fragments, e.g. an alkane not having any functional groups, will be lost in the pipeline at the respective step and not processed further. If the setting is set to “true”, the fragments resulting from the previous pipeline step or the input molecule will be kept as fragments in the respective step and will be able to be processed by following pipeline steps.

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 processing to extract side chains from the resulting aglycones, discarding rings. And 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 “scaffold only” 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” (Figure 13). 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. Another recommended setting here is to set the global preference “keep last fragment in pipelining” to “false”. A molecule having no side chains or an alkane side chain having no functional groups should not appear in the final result.

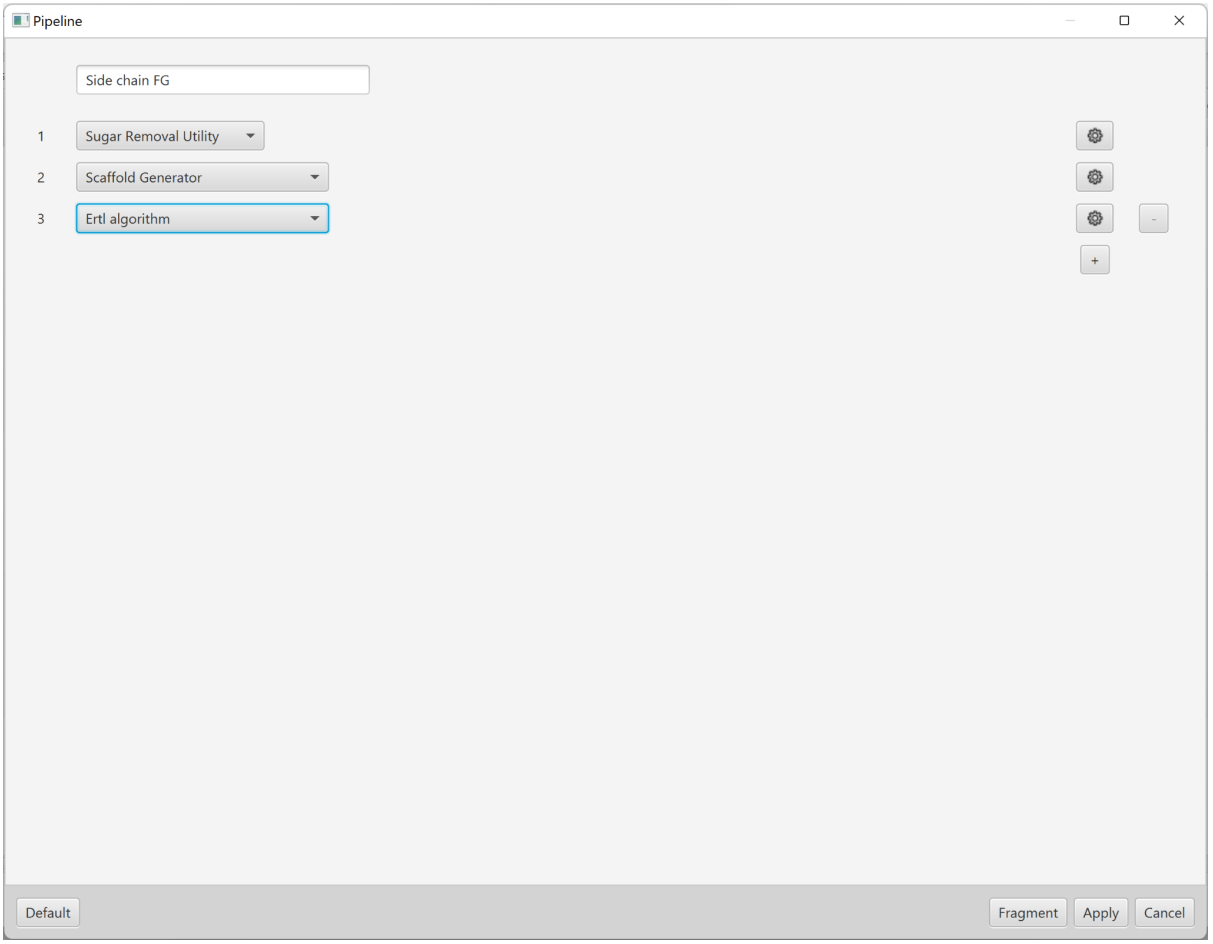
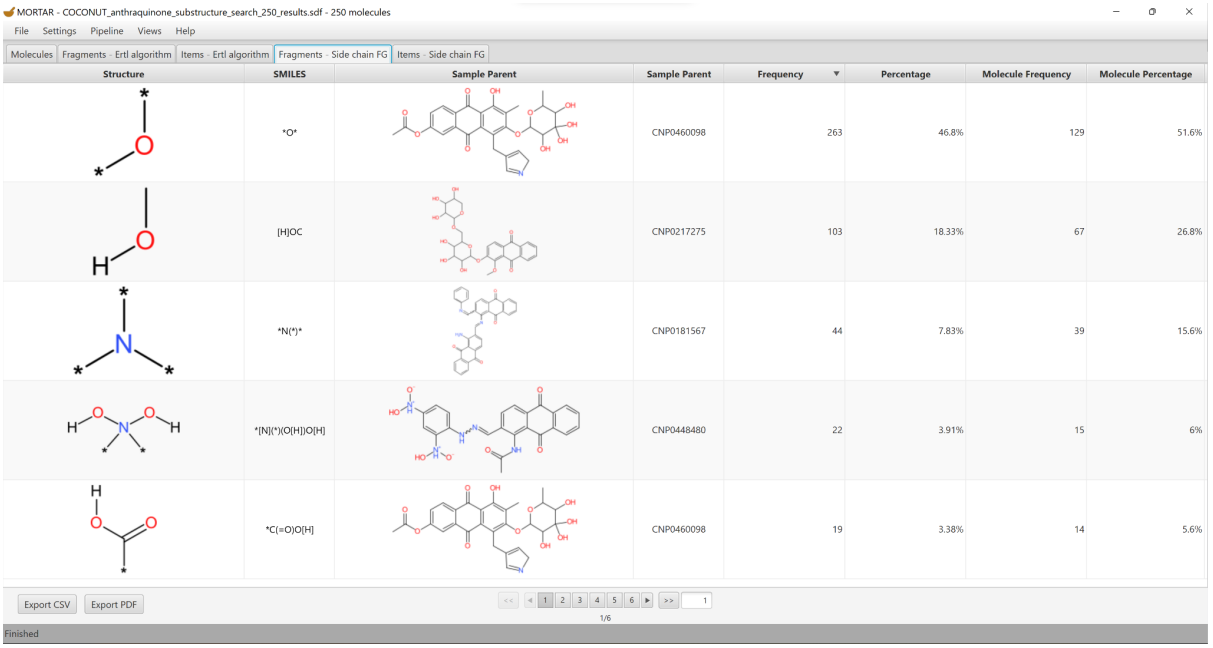


Figure 13: Pipeline configuration to extract side chain functional groups.

When the fragmentation has been set up, 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 “Fragment” button in the bottom-right corner of the pipeline settings dialog. Using the COCONUT anthraquinone subset, 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 an ether or hydroxy group, followed by a hydroxy group connected to an aliphatic carbon atom that in most cases results from a methyl-ether side chain. The third-frequent functional group is a primary amine, followed by a nitro group whose charge was neutralised by the Ertl algorithm fragmenter in preprocessing. The fifth-frequent functional group is an ester or carboxy acid functionality. For more details, inspect the items tab.
Note that the configured pipeline is persisted at application exit and re-imported at the next session.



Structure	SMILES	Sample Parent	Sample Parent	Frequency	Percentage	Molecule Frequency	Molecule Percentage
	O		CNP0460098	263	46.8%	129	51.6%
	[H]OC		CNP0217275	103	18.33%	67	26.8%
	N(*)		CNP0181567	44	7.83%	39	15.6%
	*[N](*)(O[H])O[H]		CNP0448480	22	3.91%	15	6%
	*C(=O)O[H]		CNP0460098	19	3.38%	14	5.6%

Figure 14: Side chain functional groups of the COCONUT anthraquinone subset.

Histogram view

Another visualisation option for fragmentation results in MORTAR is the histogram view. When a fragmentation process has been finished, the histogram view can be opened via “Views” -> “Histogram” in the menu bar at the top of the MORTAR main window (Figure 15).

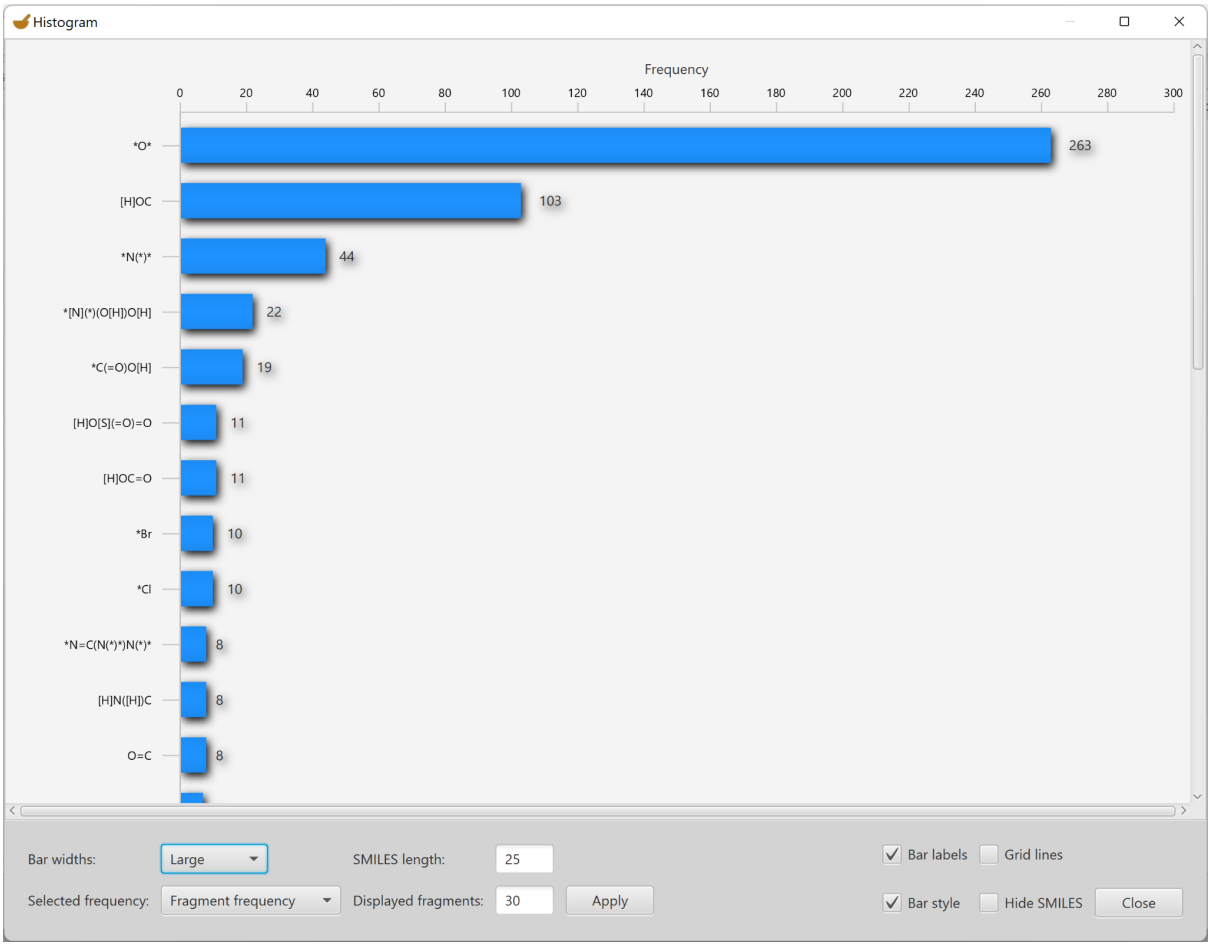


Figure 15: Histogram view.

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 16).

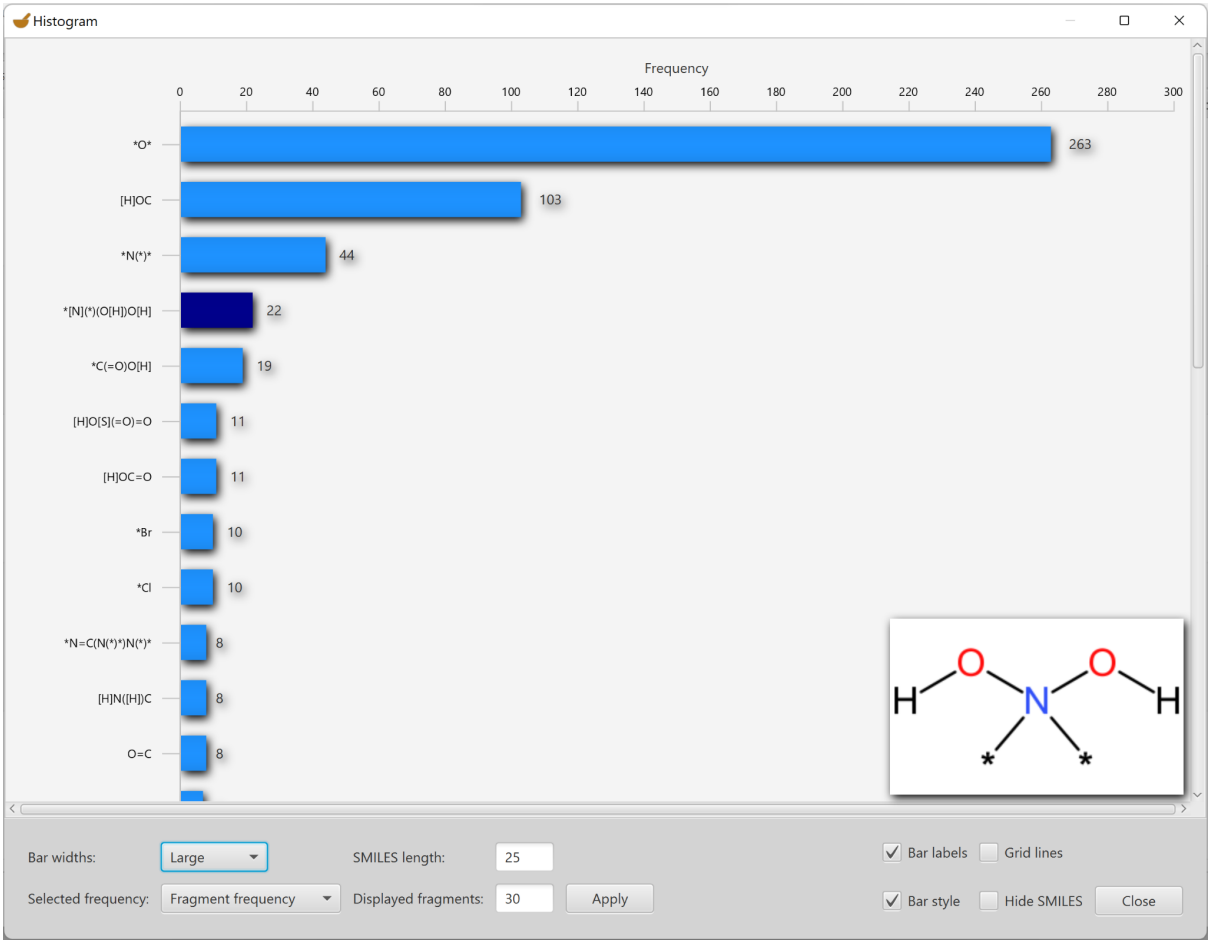


Figure 16: Structure display in histogram view.

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 20 most frequent fragments, enter “20” into the respective text box. 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 in 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 if the “Apply” button is clicked.

The settings in the bottom-right of the histogram view can be used to show or hide the frequency labels next to the bars or grid lines in the histogram. Also, the shadows of the bars can be deactivated. The “Hide SMILES” option can be used to hide the fragment SMILES representations on the y-axis of the histogram. Figure 17 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 on the “Close” button in the bottom-right corner.

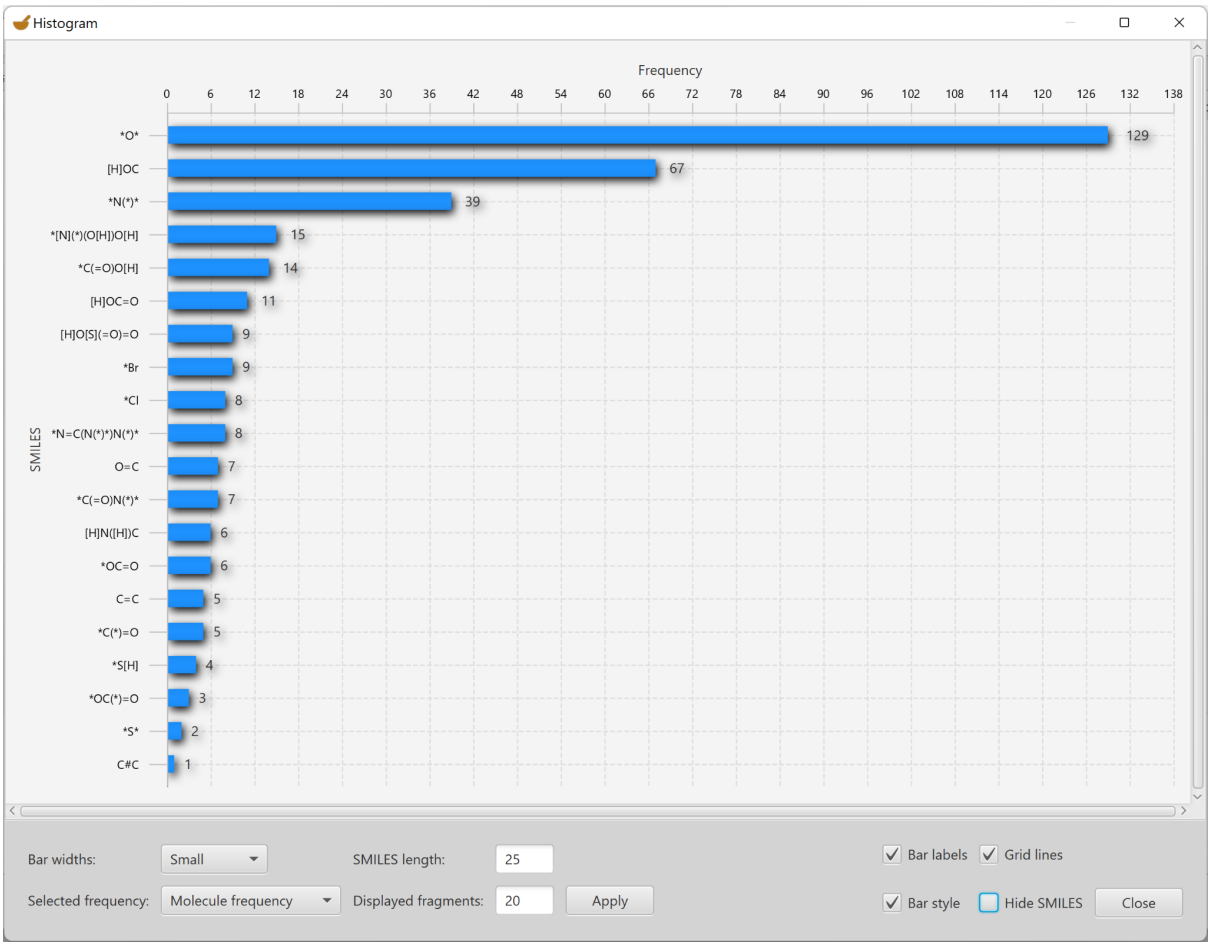


Figure 17: Alternative styling for histogram view.

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 18).

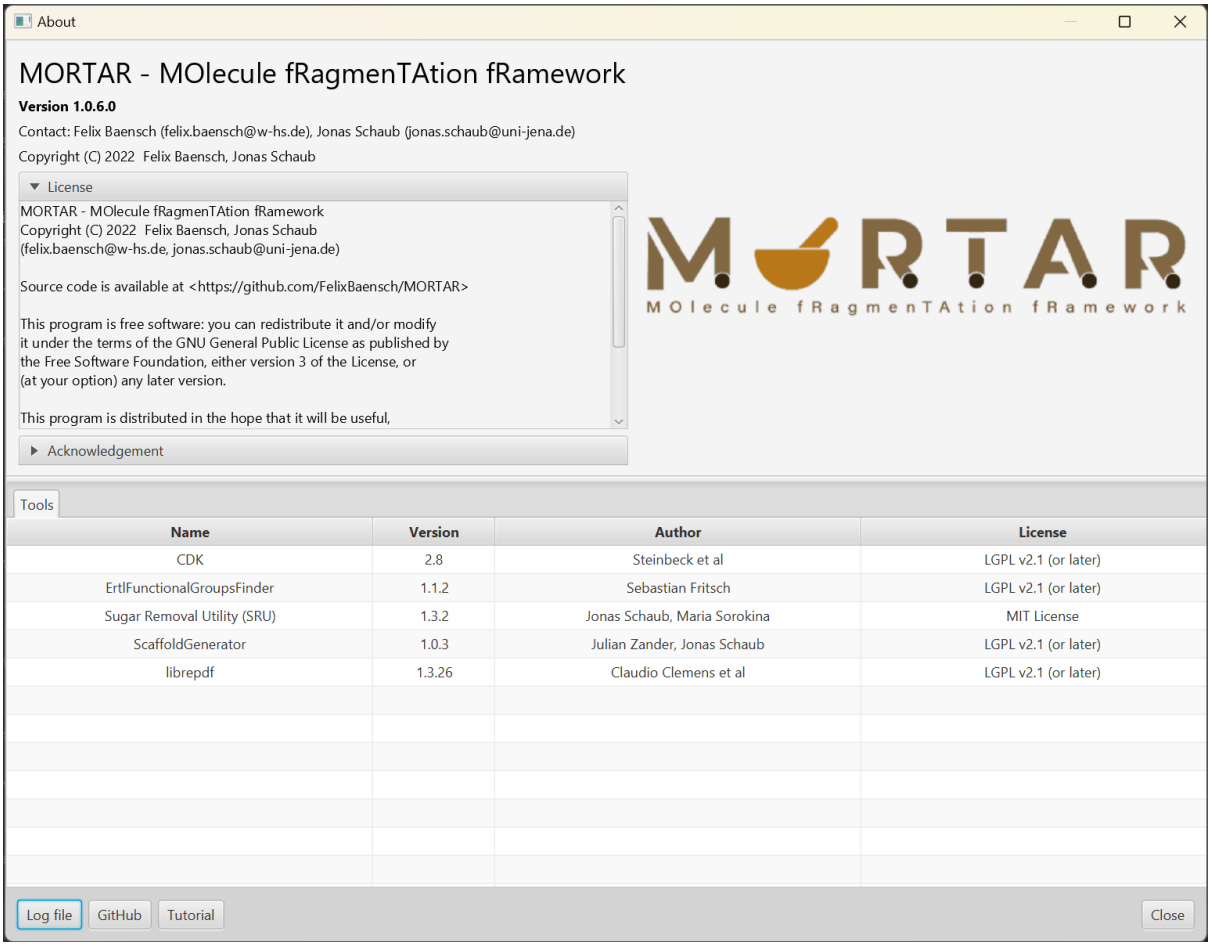


Figure 18: About view dialog.

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 occur in 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 our MORTAR GitHub repository (https://github.com/FelixBaensch/MORTAR) 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 into 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 the 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 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.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 for 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 (https://github.com/FelixBaensch/MORTAR#readme). If you want to contribute to MORTAR publicly and openly, be invited to start a pull request to the MORTAR repository on GitHub (https://github.com/FelixBaensch/MORTAR) after successfully adding your new functionality. And if you need any assistance, feel free to contact us via GitHub, e.g. in an issue on the MORTAR repository.

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 file 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 19).

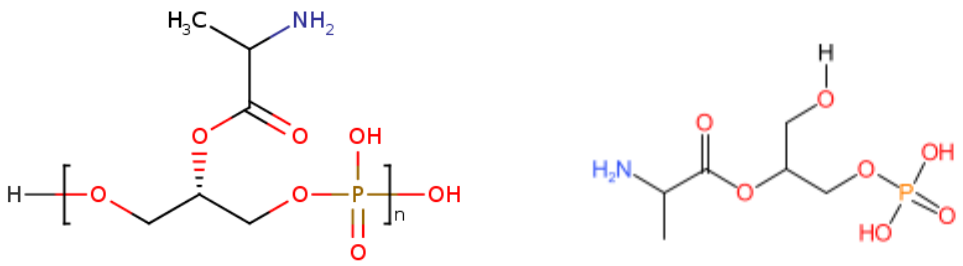


Figure 19: Representation of ChEBI compound 15338 in ChEBI as a polymer versus representation in MORTAR as a monomer.

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). For example, the metal complexation bonds in ChEBI compound 48572 (pseudocoenzyme B_{12}) (Figure 20) raise an error at import of the structure into MORTAR from an SD or MOL file because they are defined as bond order 8 (any bond order). It also

has to be noted that the SMILES representation (that is internally used by MORTAR in general) does not support such bond types either.

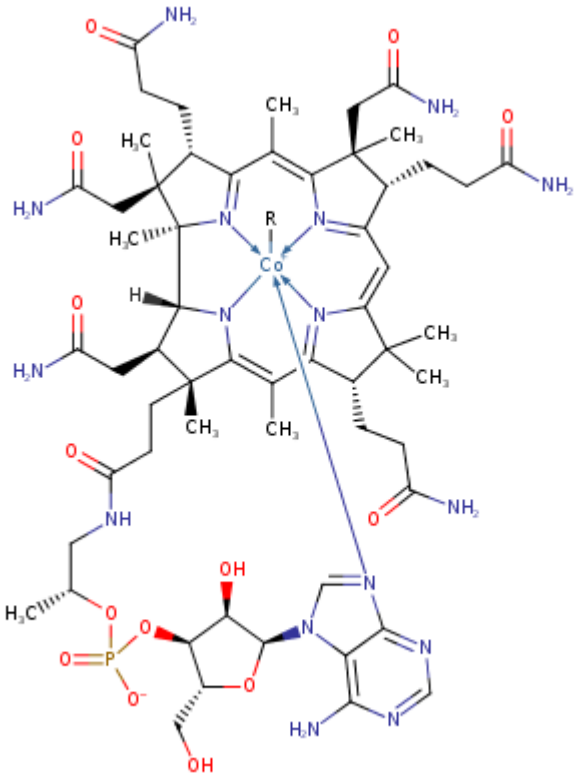


Figure 20: Representation of pseudocoenzyme B12 in ChEBI with metal complexation bonds.

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. And 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 in 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 of computation time is currently inexplicable to us. We therefore recommend using at least 2 parallel computation threads. For the other fragmentation algorithms available, this effect could not be observed.