Scaffold Hunter

Version 2.1

User Manual

February 4, 2013

Contents

1. Introduction

Searching chemical space for a particular compound is like looking for a needle in a haystack. To assist you with this task, the software *Scaffold Hunter* was developed. Scaffold Hunter uses the concept of *scaffolds*, i.e., underlying molecular frameworks that serve as simplified representatives for classes of similar molecules. These scaffolds can be organized in a tree-like hierarchy based on the inclusion relation, enabling the user to navigate in the associated chemical space in an intuitive way. Furthermore, Scaffold Hunter supports a variety of views allowing the user to inspect chemical compound data from different perspectives.

The first step in using Scaffold Hunter is to aggregate data in a database, and create a so-called *scaffold tree* for this data. These topics are covered in the first part of the manual. Once this is done the data can be navigated, filtered, viewed and annotated by multiple users. This is described in the second part of the manual.

History. The Scaffold Hunter project was initiated by Stefan Wetzel and Karsten Klein and a first version of Scaffold Hunter was implemented by the project group 504 [2] at the TU Dortmund in 2009. For version 2.x, many parts of it were improved, extended and rewritten by the project group 552 [1], resulting in a program with many additional features, better support of chemical workflows, and improved usability. Currently, the project is managed by Nils Kriege and Karsten Klein at TU Dortmund, and continuously improved in a cooperation of TU Dortmund and the Max-Planck-Institute for Molecular Physiology.

2. Requirements

2.1. System Requirements

CPU

A CPU with at least 2 GHz is recommended to run the program.

RAM

At least 1 GB must be available to the Java Virtual Machine running the program.
4 GB are recommended for full functionality.

Hard Drive

40 MB of free space are sufficient to store the program and the database connections.

Display

A display with a resolution of at least 1024x768 is recommended.

2.2. Database

An SQL database is required to store the imported datasets and the user's sessions. The currently supported databases systems are:

MySQL

Version 5.0 or later is required. A copy of `MySQL` can be obtained at `http://www.mysql.com`.

HSQLDB

Version 2.2.4 of `HSQLDB` is included and can be used for small personal databases. Additional space on the hard drive is required if this type of database is used. Please note that using `HSQLDB` leads to an increased memory requirement compared to `MySQL`.

2.3. Java Virtual Machine

A virtual machine capable of running `Java SE 6` code is required to execute the program. Such a virtual machine can be obtained at `http://www.java.com`.

3. Installation

3.1. Installing Scaffold Hunter

Scaffold Hunter is written completely in Java and deployed in a runnable JAR file. Therefore no installation is required to run the application. Simply download the Scaffold Hunter ZIP archive from the website and extract it somewhere on your computer (e.g., in case you have admin rights, `C:\Program Files\ScaffoldHunter` on Windows or `/opt/scaffoldhunter` on Unix/Linux/Mac, or alternatively in any local directory).

> IMPORTANT: A Java Virtual Machine must be installed on your computer. For large datasets and better overall performance, the installation of a `MySQL` Database is recommended. See *Sec. 2.3: Java Virtual Machine* for further information.

3.2. Installing Plugins

If you need to install additional plugins that are not bundled with Scaffold Hunter, you have to copy the plugin's JAR file to the `plugins` subdirectory of the Scaffold Hunter installation directory.

3.3. Starting Scaffold Hunter

To launch Scaffold Hunter, simply run the supplied start-script (`run.cmd` on Windows or `run.sh` on Unix/Linux/Mac). If you want to work with large datasets, you may want to increase the maximum memory available for your application. Be sure your computer has enough physical memory available. Increasing the usable memory can have a huge impact on Scaffold Hunter's performance, too.

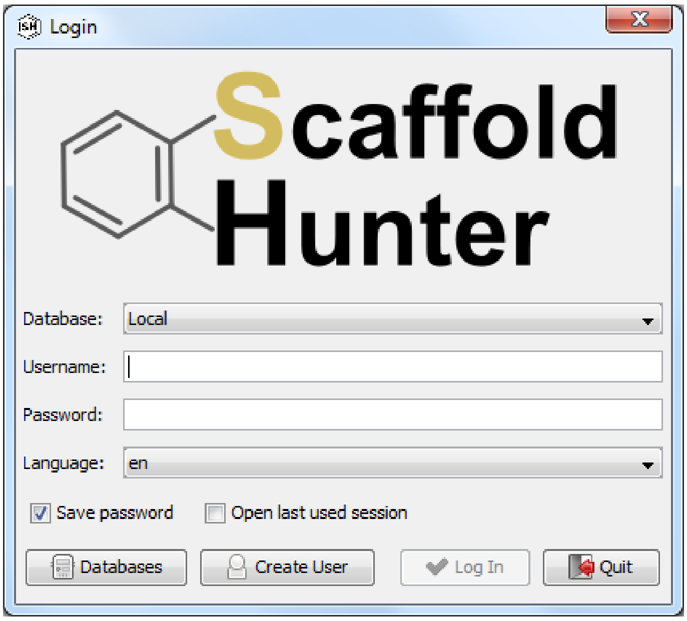
The default start-script specifies a maximum of 1 GiB of memory. To increase the available memory, edit the start-script with a text editor, find the line displayed below and change `1024m` to whatever amount of memory you would like to be available to Scaffold Hunter. If you want to specify the amount in GiB you can write, e.g., `1g`.

run.cmd (Windows): `set memory=1024m`

run.sh (Linux/Unix/Mac): `memory=1024m`

> WARNING: If you experience a *java.lang.OutOfMemoryException: Java heap space* error, try to increase the memory as described above.

4. Start Dialog



Database: Local

Username:

Password:

Language: en

☑ Save password ☐ Open last used session

Figure 4.1.: Start Dialog

The Start Dialog shown in Fig. 4.1 is the first window that appears at program startup. Here you can select the database to work with and enter your username and password. If the checkbox "Save password" is selected, the password will be saved in encrypted form. With the checkbox "Open last used session" you can choose to automatically open the session most recently used under the given username.

4.1. Setting up a Database Connection

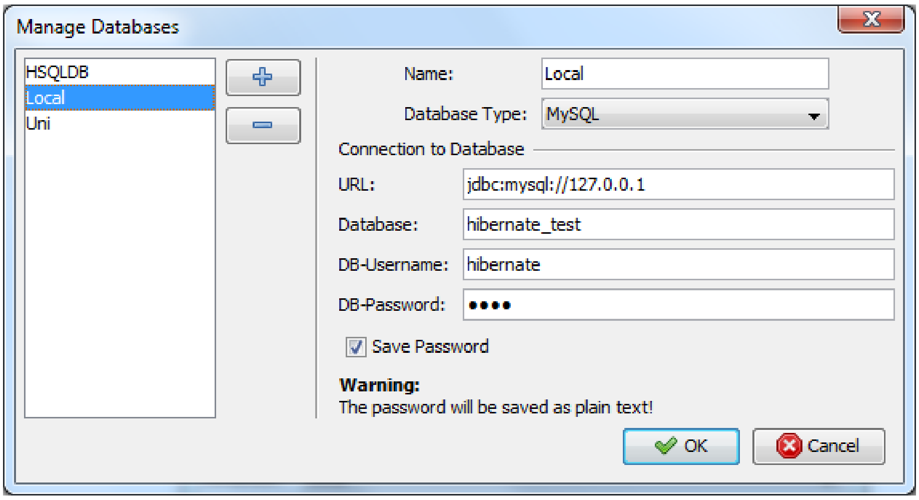
You can select the database to work with using the drop-down box in the Start Dialog. To add a new database to this list, click on the "Databases" button. This opens the Databases Dialog shown in Fig. 4.2. On the left hand side of the window, the existing database connections are shown. With the plus and minus buttons you can create a new connection or remove an existing one. On the right hand side the settings for the selected database are shown.

The name is only used for identification of the database connection – you can enter any name in this field.

"Database Type" is used to define the database system. At the moment Scaffold Hunter supports two different types of databases: `HSQLDB` and `MySQL`.

A `HSQLDB` database is an internal and local database. As it is easy to configure, this database is best if you want to test Scaffold Hunter or to have a small private database. If you choose a `HSQLDB` database, the only setting left is the database path, where you can specify a location to store the database.

`MySQL` is a server-based database. It requires a `MySQL` server that may be running on the same machine as Scaffold Hunter, or on a different machine. To configure a `MySQL` connection, there are a few settings to



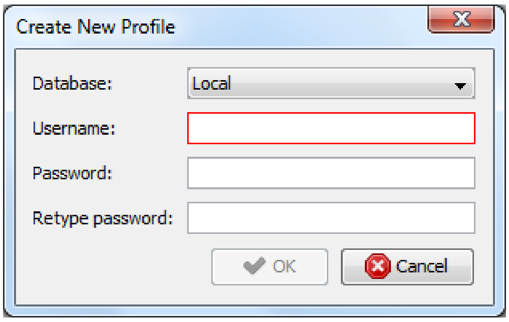
Name: Local
Database Type: MySQL
Connection to Database
URL: jdbc:mysql://127.0.0.1
Database: hibernate_test
DB-Username: hibernate
DB-Password: ••••
☑ Save Password

Warning:
The password will be saved as plain text!

Figure 4.2.: Databases Dialog

be made. The "URL" field holds the server name or IP address. For example, "localhost" or "127.0.0.1" both would point to the local machine. The protocol name "jdbc:mysql://" is not necessary and will be added automatically. In the "Database" field the database or schema name is entered. On a server, only one schema with a given name can exist. So, if an existing schema name is entered, this schema will be used (or overwritten if the schema was used by a different application or a different version of Scaffold Hunter). The "DB Username" is used to log in to the database. This user must be configured on the `MySQL` server and needs all rights to manipulate data in the database. It also needs all data definition rights if the database was not initialized before by Scaffold Hunter or has to be recreated. The connection to the database also requires a password, which can be saved in the configuration file. Please note that password is stored unencrypted. If this password is not saved, you will be asked to enter the password each time you log in to the database.

4.2. User Management



Database: Local
Username:
Password:
Retype password:

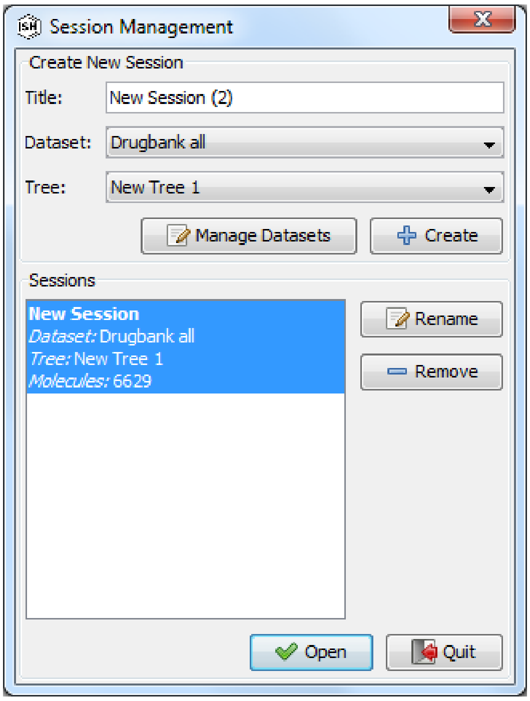
Figure 4.3.: User Creation Dialog

Scaffold Hunter includes its own user management. A profile is used to save personal settings and open sessions (see *Sec. 5.1: Session Management*). To create a new user, click on the button "Create User". Thereby the User Creation Dialog shown in Fig. 4.3 is opened, where you can choose the database on which to create the user, as well as a username and password. After clicking "Ok" the program will connect to the selected database and create a new user with the specified username. If the specified username already exists in the database user creation will fail and an error message will be displayed.

5. Getting Started

5.1. Session Management

In Scaffold Hunter, a session represents a complete working space of a user. It references a dataset consisting of molecules and a corresponding scaffold tree. The molecules of a session can be filtered to reduce the molecule count to a manageable size.



Title: New Session (2)
Dataset: Drugbank all
Tree: New Tree 1

New Session
Dataset: Drugbank all
Tree: New Tree 1
Molecules: 6629

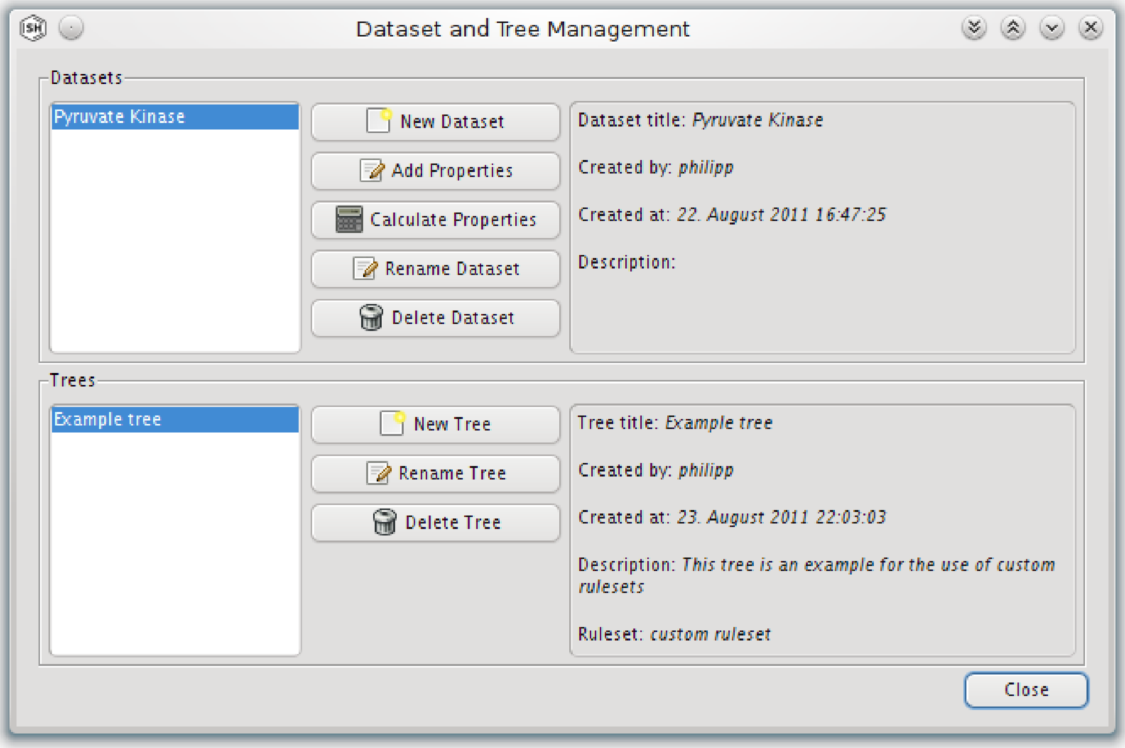
Figure 5.1.: Session Management Dialog

The Session Management Dialog shown in Fig. 5.1 appears after logging in, or by selecting “Session → Load / Create” from the menu. If “Open last used session” was selected in the Start Dialog, the session management window does not appear and the last used session is opened instead. The session management consists of two areas. The upper area is used for creating new sessions. The lower area shows a list of available sessions and options for them. At the first start the session list will be empty, and a new session has to be created in order to start Scaffold Hunter.

To create a new session, choose a title, the dataset to create the session on, and the scaffold tree. The title can be changed later. After clicking on the “Create” button, a filter dialog will appear, allowing to filter the selected dataset. The created session will then be displayed in the list of sessions in the lower area. To create a new dataset or to add a new tree to a dataset, click “Manage Datasets”.

In the lower area of the session management window, the available sessions are shown with their title, dataset, tree and molecule count. Here the sessions can be renamed or removed, if they are no longer needed. To open a session, select the desired session and click the “Open” button. Initially the last used session is selected in the list.

5.2. Dataset Management



Dataset title: *Pyruvate Kinase*

Created by: *philipp*

Created at: *22. August 2011 16:47:25*

Description:

Tree title: *Example tree*

Created by: *philipp*

Created at: *23. August 2011 22:03:03*

Description: *This tree is an example for the use of custom rulesets*

Ruleset: *custom ruleset*

Figure 5.2.: Dataset Management Dialog

The Dataset Management Dialog shown in Fig. 5.2 gives you the possibility to import, modify or delete the molecular data stored in Scaffold Hunter's internal database. In Scaffold Hunter you can manage your projects by the use of datasets. Each dataset consists of a set of molecules and a collection of properties for these molecules. For each dataset you can generate an arbitrary number of scaffold trees using different generation rules. You will later use the scaffold tree of your choice, to navigate through the molecules in the dataset. In the following the actions you can execute on datasets and trees are explained.

Actions on datasets

New dataset To create a new dataset by importing chemical compound data, click on "New Dataset" which opens the import wizard. Read *Sec. 5.2.1: Importing Data* to get more information about the import process.

Add properties If you have an existing dataset and want to add some additional molecular properties (e.g. bioactivity information from a bio assay), select the dataset from the list of datasets and click "Add Properties". A wizard similar to the import wizard described in *Sec. 5.2.1: Importing Data* will appear.

IMPORTANT: Please note that you can only add additional properties to existing molecules. For technical reasons it is not possible to add completely new molecules. If you try to do so, those molecules will simply be ignored.

Calculate properties Scaffold Hunter gives you the opportunity to calculate additional properties (e.g. properties derived from the structure of a compound) for the molecules of a dataset. Additionally, to enable some features like clustering molecules based on structural similarity, you may need to calculate fingerprint properties. To calculate properties for a dataset, select the dataset from the list of datasets and click “Calculate Properties”. A dialog to create calculation tasks appears. Please read *Sec. 5.2.2: Calculation of Properties* to learn more about the calculation process.

Rename dataset If you created a dataset and would like to change its name or description, select the dataset from the list of datasets and click “Rename Dataset”. A rename dialog will appear.

Delete dataset If there is an existing dataset you don’t need anymore, select it from the list of datasets and click “Delete Dataset”. A confirmation dialog will appear. If you confirm the deletion, the dataset and all trees in it will be deleted.

> WARNING: Scaffold Hunter can be used by multiple users. If you are working on a database shared with other users, you will also destroy any work done by other users on the dataset being deleted and the corresponding trees. First ensure nobody needs the dataset and the corresponding trees anymore.

Actions on trees

New tree To create a new tree, select the dataset the tree should be generated for from the list of datasets and click “New Tree”. The tree generation dialog will appear. Read *Sec. 5.2.3: Scaffold Tree Generation* to learn more about how the tree generation process works.

Rename tree If you created a tree and would like to change its name or description, select the tree from the list of trees and click “Rename Tree”. A rename dialog will appear.

Delete tree If there is an existing tree you don’t need anymore, select it from the list of trees on the left and click “Delete Tree”. A confirmation dialog will appear. If you confirm the deletion, the tree will be deleted.

> WARNING: Scaffold Hunter can be used by multiple users. If you are working on a database shared with other users, you will also destroy any work done by other users on the tree being deleted. First ensure nobody needs the tree anymore.

5.2.1. Importing Data

Using different *import plugins*, Scaffold Hunter can import data from different sources. Currently there are plugins to import data from CSV files, SDF files and MySQL databases bundled with the program for a more detailed description of the plugins see *Sec. 5.2.1.2: Import Plugins*. To perform an import you have to create one or more import jobs, one for each data source you want to import. These jobs will run sequentially and the data imported by those jobs will be merged according to rules which can be selected during the import process. In general Scaffold Hunter considers two structures equal if they have the same canonical SMILES string. If a structure is present in more than one source and properties from these sources are mapped to the same internal property the property entries for the structure will be merged. The following *merge strategies* are available to influence which data will be found in the finished dataset:

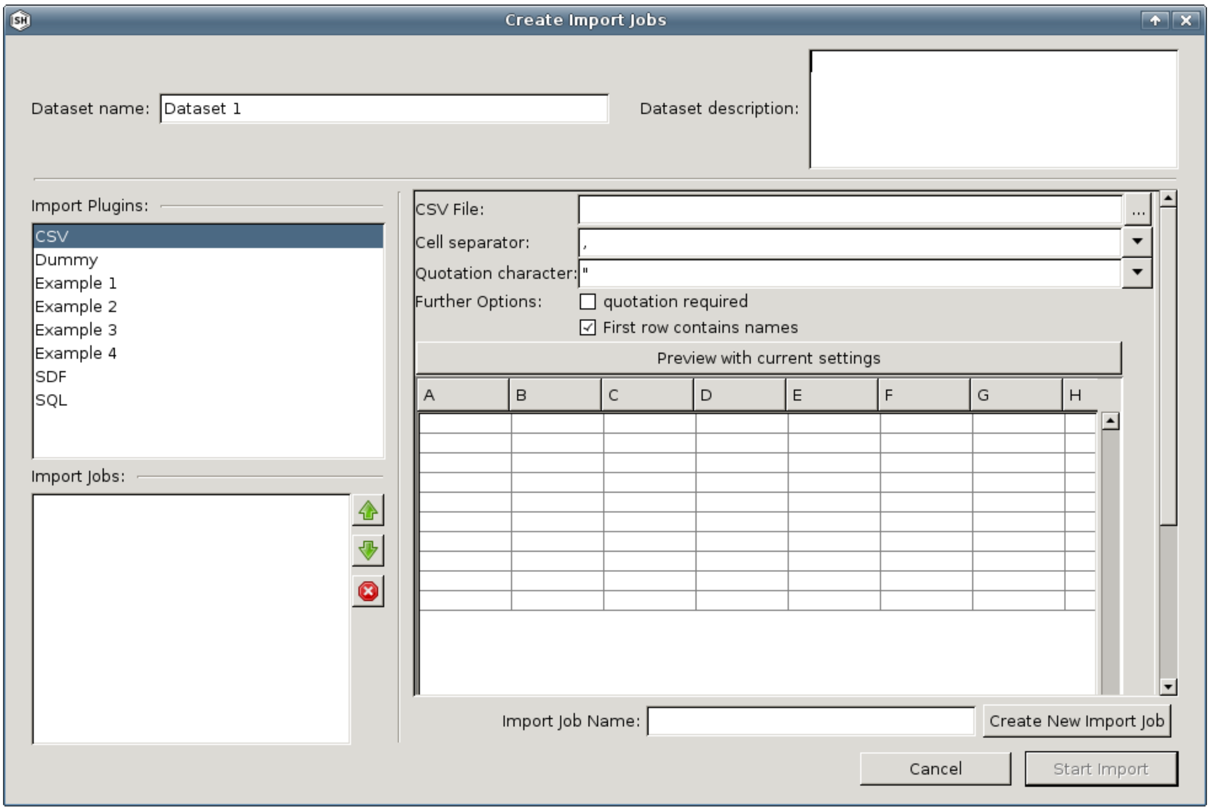
Do not overwrite: The first seen occurrence of the property is used and will not be overwritten by the current job.

Do overwrite: The property will be overwritten by the current import job.

Concatenate (only for textual properties): The property values will be appended at the end of the existing value.

Minimum / maximum (only for numeric properties): The minimum or maximum of the available property values will be retained.

5.2.1.1. Creating Import Jobs



Dataset name: Dataset 1

Dataset description:

CSV File:

Cell separator: ,

Quotation character: "

Further Options: ☐ quotation required

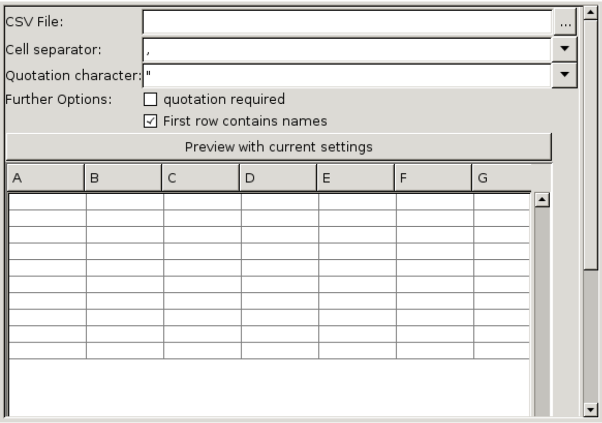
☑ First row contains names

Import Job Name:

Figure 5.3.: Create Import Jobs dialog

When you click on “New Dataset” the Create Import Jobs dialog as shown in Fig. 5.3 will open. In the top section of the dialog a name and a more detailed description for the new dataset can be entered. On the left hand side you have two lists. First the list of “Import Plugins”, where you can select a plugin appropriate for the data source you want to import. Underneath there is the list of “Import Jobs”, which contains all import jobs created so far. The buttons beside this list can be used to delete import jobs and to change their order. During import the jobs will be executed one after the other from top to bottom. On the right hand side there are the settings for the currently selected plugin. Beneath the settings you can enter an optional name for the import job. A click on the button beside the name field will create a new import job with the selected settings. Below this you can find a button to start the import process.

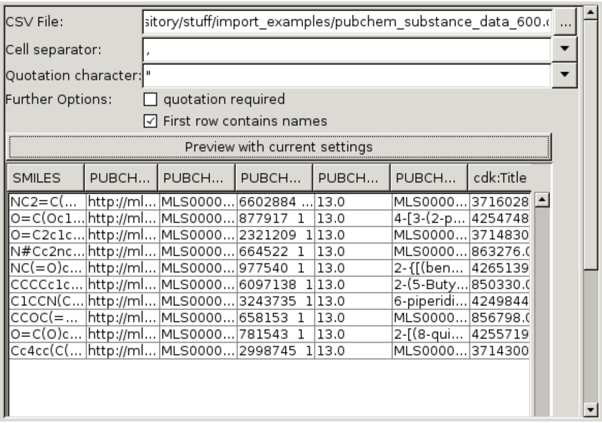
5.2.1.2. Import Plugins



CSV File:
Cell separator: ,
Quotation character: "
Further Options: ☐ quotation required
☑ First row contains names

Figure 5.4.: CSV Import Plugin

CSV Import Plugin With the CSV plugin you can import data from comma separated value (CSV) files. In Fig. 5.4 you can see the configuration panel of the CSV plugin. You can select a CSV file, different cell separators and quotation characters. Next to some default values you can type in any other character. One of these options allows you to check the "quotation required" box to configure the plugin to read only quoted content. With the "First row contains names" switch you can configure the plugin to use the first row in the CSV file for the names of the columns. If you deselect this switch the columns will be numbered. Finally there is a "Preview with current settings" button. When you click on this button the first 10 rows of the CSV file will be read according to your settings and displayed in the table below. An example can be seen in Fig. 5.5.



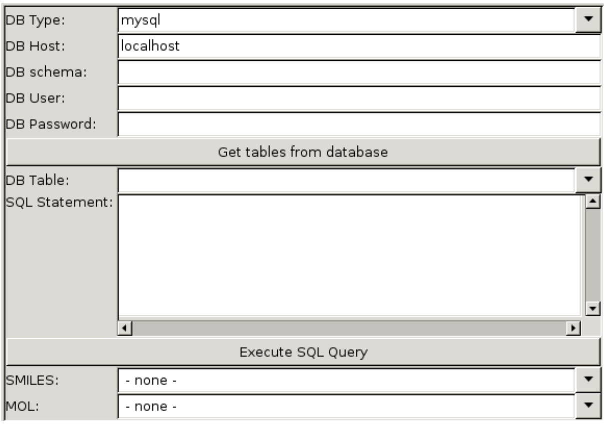
CSV File: sitory/stuff/import_examples/pubchem_substance_data_600.
Cell separator: ,
Quotation character: "
Further Options: ☐ quotation required
☑ First row contains names

SMILES	PUBCH...	PUBCH...	PUBCH...	PUBCH...	PUBCH...	cdk:Title
NC2=C(...	http://ml...	MLS0000...	6602884 ...	13.0	MLS0000...	3716028
O=C(Oc1...	http://ml...	MLS0000...	877917 1	13.0	4-[3-(2-p...	4254748
O=C2c1c...	http://ml...	MLS0000...	2321209 1	13.0	MLS0000...	3714830
N#Cc2nc...	http://ml...	MLS0000...	664522 1	13.0	MLS0000...	863276.
NC(=O)c...	http://ml...	MLS0000...	977540 1	13.0	2-{[(ben...	4265139
CCCCc1c...	http://ml...	MLS0000...	6097138 1	13.0	2-(5-Buty...	850330.
C1CCN(C...	http://ml...	MLS0000...	3243735 1	13.0	6-piperidi...	4249844
CCOC(=...	http://ml...	MLS0000...	658153 1	13.0	MLS0000...	856798.
O=C(O)c...	http://ml...	MLS0000...	781543 1	13.0	2-[(8-qui...	4255719
Cc4cc(C(...	http://ml...	MLS0000...	2998745 1	13.0	MLS0000...	3714300

Figure 5.5.: CSV Import Plugin with preview

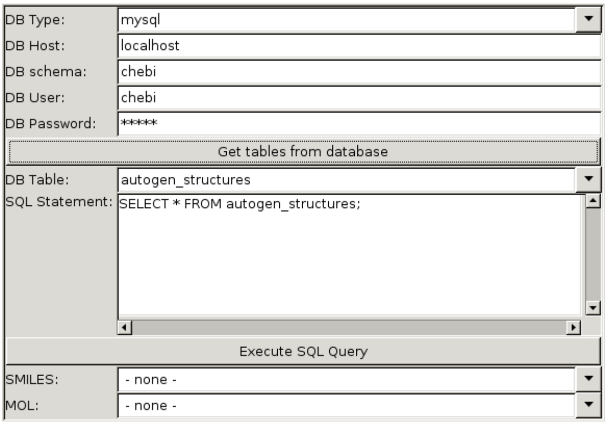
SQL Import Plugin The SQL import plugin allows to integrate data from SQL-based databases. The configuration panel, as seen in Fig. 5.6 provides various configuration options explained in the following.

DB Type This field allows to select the type of the SQL database you want to connect to. Currently only `MySQL` is tested. Every JDBC compatible database should be useable.



DB Type: mysql
DB Host: localhost
DB schema:
DB User:
DB Password:
DB Table:
SQL Statement:
SMILES: - none -
MOL: - none -

Figure 5.6.: SQL Import Plugin



DB Type: mysql
DB Host: localhost
DB schema: chebi
DB User: chebi
DB Password: *****
DB Table: autogen_structures
SQL Statement: SELECT * FROM autogen_structures;
SMILES: - none -
MOL: - none -

Figure 5.7.: SQL Import Plugin after “Get tables from database”

DB Host In “DB Host” you can select the database host, the server on which the source database is running. If you need a specific port to connect to the string is: `HOSTNAME:PORT`

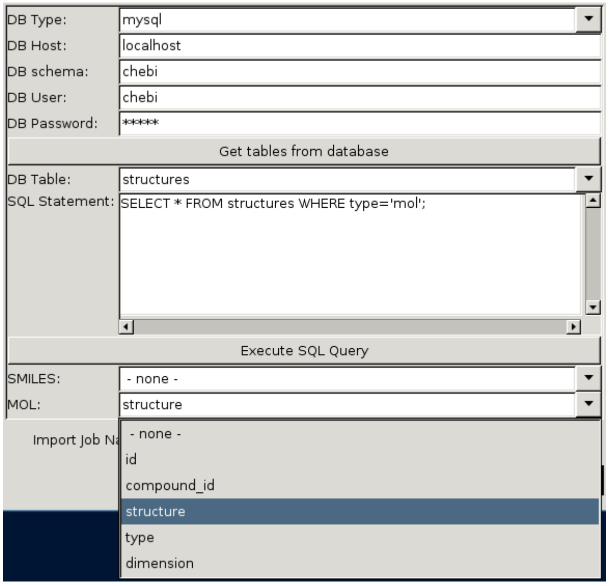
DB schema In the “DB schema” field you can enter the database (schema) name you want to use.

DB User / DB Password In the “DB User” and “DB Password” fields you can fill in your database connection credentials. Only read access is needed.

Get tables from database Once you have entered the connection data you can click the “Get tables from database” button. The plugin will connect to the database and put all table names found in the schema in the “DB Table” list. A connection done with the ChEBI database is shown in Fig. 5.7.

DB Table The “DB Table” list contains the tables found after clicking on “Get tables from database”. When selecting a table a simple `SELECT` statement will be generated.

SQL Statement In the “SQL Statement” field you can type in an SQL SELECT statement that will be run on the selected database. The resulting rows from this statement will be used as source for the import.



DB Type: mysql
DB Host: localhost
DB schema: chebi
DB User: chebi
DB Password: *****
DB Table: structures
SQL Statement: SELECT * FROM structures WHERE type='mol';
SMILES: - none -
MOL: structure

Figure 5.8.: SQL Import Plugin after “Execute SQL Query”

Execute SQL Query By clicking on “Execute SQL Query” your SQL Statement will be executed and the “SMILES” and “MOL” lists will be filled with the resulting column names. This is shown in Fig. 5.8.

SMILES / MOL In the “SMILES” and “MOL” lists you can select the column names which contain structure information in SMILES and/or MOL format.

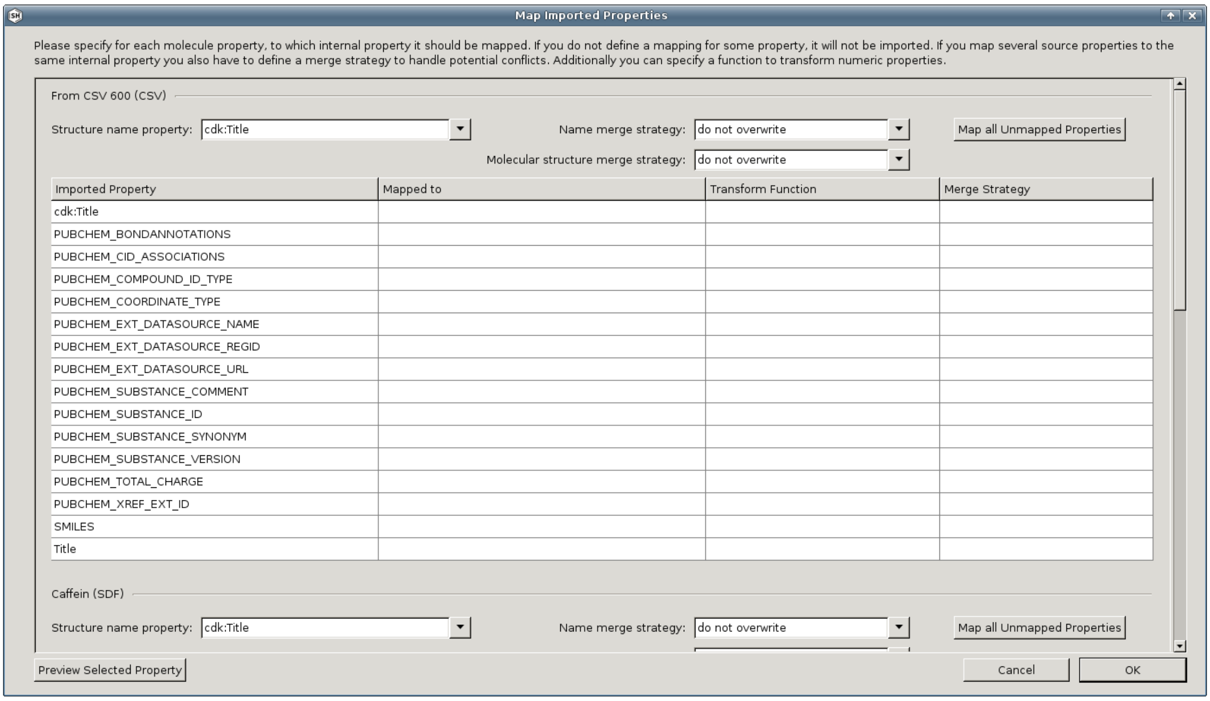
SDF Import Plugin The SDF Import plugin is very easy to use. It just has a field “SDF file name” where you enter the path to the SDF file which you want to import.

5.2.1.3. Map Imported Properties Dialog

Once you have created at least one import job, you can start the import process by clicking on “Start Import”. A new Dialog named “Map Imported Properties” will appear. In Fig. 5.9 you can see an example of this dialog with two import jobs. For every import Job there are the same GUI elements. At the top you have a line with the name of the following import job.

Structure name property In the “Structure name property” list you select one of the properties for the name field of the structures. The name field will be used at several places in the program to present a name together with a structure, however it is not used internally so there are no special requirements for this property. If the name property is missing for some structure, the SMILES string will be used instead.

Name/Molecular structure merge strategy In the “Name merge strategy” and “Molecular structure merge strategy” you can select the merge strategies for the molecule name or 3D/2D structure according to *Par. 5.2.1: Importing Data*.



Map Imported Properties
Map all Unmapped Properties
Map all Unmapped Properties
Preview Selected Property
Cancel
OK

Please specify for each molecule property, to which internal property it should be mapped. If you do not define a mapping for some property, it will not be imported. If you map several source properties to the same internal property you also have to define a merge strategy to handle potential conflicts. Additionally you can specify a function to transform numeric properties.

From CSV 600 (CSV)

Structure name property: cdk:Title

Name merge strategy: do not overwrite

Molecular structure merge strategy: do not overwrite

Imported Property	Mapped to	Transform Function	Merge Strategy
cdk:Title			
PUBCHEM_BONDANNOTATIONS			
PUBCHEM_CID_ASSOCIATIONS			
PUBCHEM_COMPOUND_ID_TYPE			
PUBCHEM_COORDINATE_TYPE			
PUBCHEM_EXT_DATASOURCE_NAME			
PUBCHEM_EXT_DATASOURCE_REGID			
PUBCHEM_EXT_DATASOURCE_URL			
PUBCHEM_SUBSTANCE_COMMENT			
PUBCHEM_SUBSTANCE_ID			
PUBCHEM_SUBSTANCE_SYNONYM			
PUBCHEM_SUBSTANCE_VERSION			
PUBCHEM_TOTAL_CHARGE			
PUBCHEM_XREF_EXT_ID			
SMILES			
Title			

Caffein (SDF)

Structure name property: cdk:Title

Name merge strategy: do not overwrite

Figure 5.9.: Map Imported Properties dialog

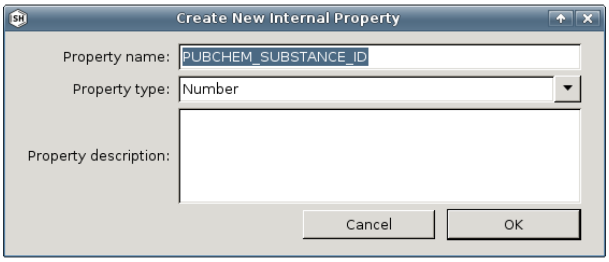
Map all Unmapped Properties With the "Map all Unmapped Properties" button all properties for which you did not give mapping information will be mapped automatically. The property name will be the name obtained from the import plugin, the type (numeric or text) is set also according to the plugin information as well. If multiple sources have properties with the same name, these properties will be mapped to the same internal property.

> IMPORTANT: Please note that plugins typically recognize properties as numeric only if all values of the property provided by the data source can be interpreted as floating point number. As a result properties as IC_{50} are not recognized as numeric if some values contain characters like $<$, $>$ or $\sim$. Since this may be undesirable, you may want to inspect the default mapping and explicitly mark some properties as numeric. Values that can not be interpreted as floating point number will then be discarded during import.

The mapping table In the table you have four columns. The first column ("Imported Property") shows the property name obtained from the plugin. In the next column ("Mapped to") you can select an existing internal property for this value, alternatively you can choose to create a "new internal property" which will open the "Create New Internal Property" Dialog (see *Sec. 5.2.1.4: Create New Internal Property dialog*) or you can choose "do not map" which will clear the table cell. In the "Transform Function" column you can input a transformation function for numerical properties, for example to convert logarithmic to linear values. Symbols you can use are numbers, $+, -, *, /,$ ^, (,), `log`, `log10`, `exp`, `ceil` and `floor`. For the input value use the variable x. So in case you want to convert aforementioned logarithmic values you would enter `exp`(x). In the last column ("Merge Strategy") you can select the merge strategy for this property.

Preview Selected Property With "Preview Selected Property" you can open a list where you can see the first 100 occurrences of the property, which is currently selected in the table.

5.2.1.4. Create New Internal Property dialog



Create New Internal Property

Property name: PUBCHEM_SUBSTANCE_ID

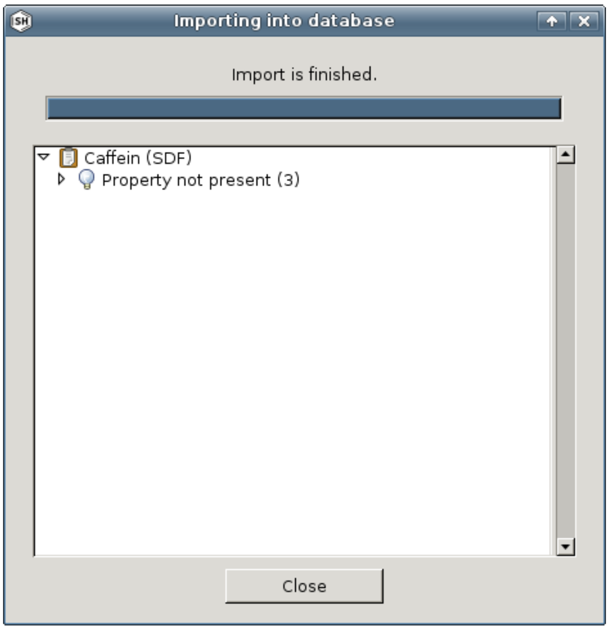
Property type: Number

Property description:

Figure 5.10.: Create New Internal Property dialog

In the “Create New Internal Property” dialog, see Fig. 5.10 you can create new internal properties. First you give the property a name using the “Property name” field. In the “Property type” list you can select one of the property types (Number, Text, fingerprint types that can be used for clustering in the dendogram view (see *Sec. 7.3: Dendrogram*)). In “Property description” you can enter a detailed description for this property.

5.2.1.5. Import into database dialog



Importing into database

Import is finished.

Caffein (SDF)
Property not present (3)

Figure 5.11.: Importing into database dialog

When you start the import in the “Map Imported Properties” dialog the import runs and the “Import into database” dialog will show up. At the top you see the current process, in the list underneath you can see some information during the import process. When the import is finished click on “Close” which concludes creation of a new dataset.

5.2.1.6. Add Properties

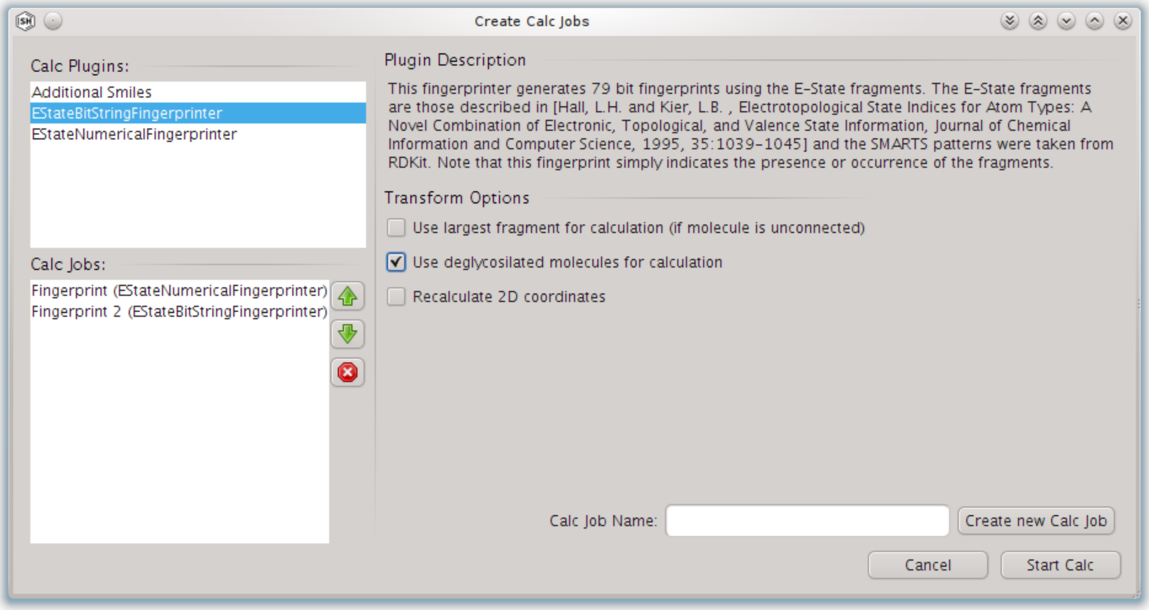
It is possible to add new properties to an existing dataset as well, by using the “Add Properties” button in the “Dataset and Tree Management” dialog. The process is very similar to the normal import process.

However you cannot add new molecules to the dataset, due to technical reasons. Should your source contain molecules which are not contained in the current dataset, these molecules will simply be ignored during import.

Furthermore the “Map Imported Properties” Dialog contains a new field named “merge by” where you can select either to merge on base of the canonical SMILES strings of the molecules or another property, which can already be found in the dataset, such as a database id from some public database.

5.2.2. Calculation of Properties

A plugin system was integrated into Scaffold Hunter to support calculation of properties for the molecules of a dataset. Scaffold Hunter provides a basic set of built-in plugins, and the plugin system allows users to write calculation plugins suited to their particular needs. See *Chap. B: How to Write Plugins* for a detailed description how to write a plugin.



Fingerprint (EStateNumericalFingerprinter)
Fingerprint 2 (EStateBitStringFingerprinter)

This fingerprinter generates 79 bit fingerprints using the E-State fragments. The E-State fragments are those described in [Hall, L.H. and Kier, L.B. , Electrotopological State Indices for Atom Types: A Novel Combination of Electronic, Topological, and Valence State Information, Journal of Chemical Information and Computer Science, 1995, 35:1039-1045] and the SMARTS patterns were taken from RDKit. Note that this fingerprint simply indicates the presence or occurrence of the fragments.

☐ Use largest fragment for calculation (if molecule is unconnected)
☑ Use deglycosilated molecules for calculation
☐ Recalculate 2D coordinates

Figure 5.12.: Create Calc Job Dialog

In the Create Calc Job Dialog shown on Fig. 5.12 you can create a batch of calc jobs, that should be executed one after each other. Each calc job is an instance of a calc plugin, which is configured based on your needs. Each plugin instance obtains a list of all molecules with their structural data and their existing properties, and can use this data to calculate new properties for all (or some) molecules. To create a batch of calc jobs, perform the following steps:

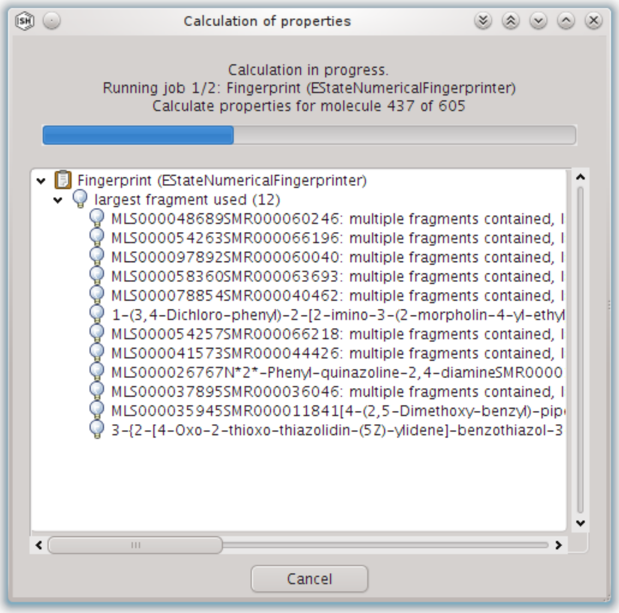
1. Select a calc plugin In the top-left corner of the dialog you will find the list of calc plugins. Select one by clicking on it and proceed with the next step.

2. Configure the calc plugin As you selected a calc plugin, the configuration panel on the right hand side of the dialog will show the plugin description and, depending on the plugin, a bunch of options to adjust the plugin. Choose the options you would like to use, and proceed with the next step.

3. Add the configured plugin to the list of calc jobs After you have finished configuring the plugin, you can make the configured plugin instance a calc job. To do this, click on “Create New Calc Job” in the bottom-right corner of the dialog. If you want to give the calc job a name, enter it in the text field labeled “Calc Job Name” first. Otherwise leave the field blank. Now the calc job of the chosen name (or a default name) appears in the list of calc jobs in the bottom-left corner of the dialog. Proceed with step 1 if you want to add another calc job.

4. (optional) Edit the list of calc jobs The list of calc jobs defines the order in which the jobs are executed. To change the order, select a calc job and move it up or down using the arrow buttons. To delete a calc job, select it and delete it by clicking on the “X” button. If you want to change the configuration of a calc job, select it. The configuration panel on the right hand side of the dialog then lets you modify the job.

If you are ready to start the calculation, click “Start Calc”. If you want to return to the previous dialog without any changes on your dataset, click “Cancel”. After you have started the calculation, a progress window as shown in Fig. 5.13 will appear and inform you about the calculation status and possible issues during calculation.



Calculation in progress.
Running job 1/2: Fingerprint (EStateNumericalFingerprinter)
Calculate properties for molecule 437 of 605

- Fingerprint (EStateNumericalFingerprinter)
  - largest fragment used (12)
    - MLS000048689SMR000060246: multiple fragments contained, l
    - MLS000054263SMR000066196: multiple fragments contained, l
    - MLS000097892SMR000060040: multiple fragments contained, l
    - MLS000058360SMR000063693: multiple fragments contained, l
    - MLS000078854SMR000040462: multiple fragments contained, l
    - 1-(3,4-Dichloro-phenyl)-2-[2-imino-3-(2-morpholin-4-yl-ethyl
    - MLS000054257SMR000066218: multiple fragments contained, l
    - MLS000041573SMR000044426: multiple fragments contained, l
    - MLS000026767N*2*-Phenyl-quinazoline-2,4-diamineSMR0000
    - MLS000037895SMR000036046: multiple fragments contained, l
    - MLS000035945SMR000011841[4-(2,5-Dimethoxy-benzyl)-pip
    - 3-{2-[4-Oxo-2-thioxo-thiazolidin-(5Z)-ylidene]-benzothiazol-3

Figure 5.13.: Calculation Progress

5.2.2.1. Transform Options

In addition to the specific options provided by each plugin, there are several plugins that provide transform options. Transform options are used to transform the structure of all molecules, before the plugin starts the calculation. The following transform options are available:

Use largest fragment Sometimes multiple molecule fragments are stored (unconnected) in one single structure. If, for example, the plugin calculates a fingerprint which takes the molecular structure into account, you may want just the largest fragment of the compound to be used for the calculation.

Use deglycosylated molecules Perhaps deglycosylation (Read *Sec. 5.2.3.2: Deglycosylate Molecules First* for an explanation of deglycosylation) has an effect on the accuracy of a calculated fingerprint. In this case you may want to deglycosylate before the calculation.

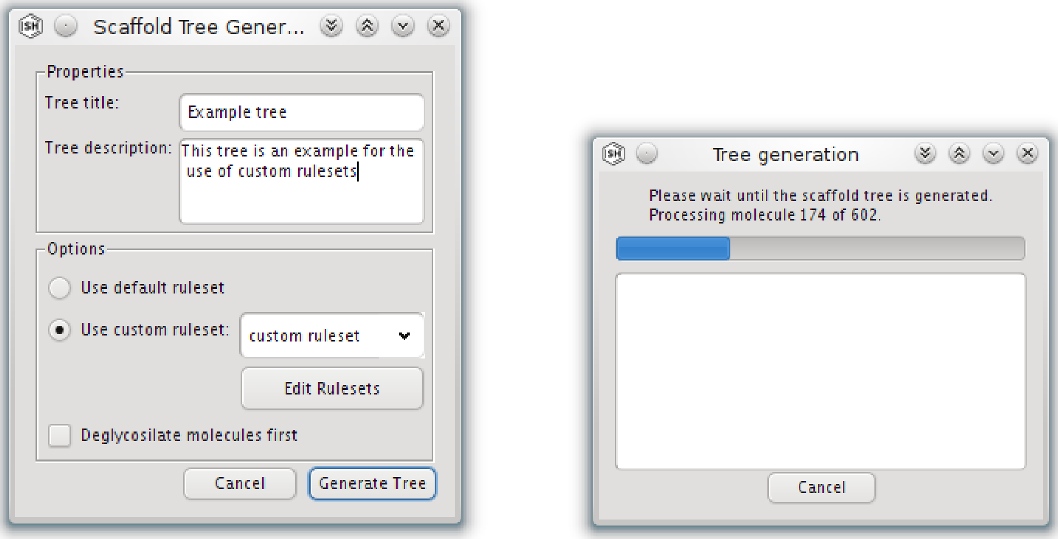
Recalculate 2D-coordinates Some calc plugins (e.g. fingerprints) may depend on the 2D-coordinates of the molecules. If you are not sure whether all imported molecules have appropriate 2D-coordinates, you propably want to recalculate them for all molecules before calculating the fingerprint.

5.2.2.2. Calc plugins

Currently there are just three calc plugins available; one plugin to calculate additional SMILES strings, and two plugins to calculate fingerprints which can be used for clustering of molecules. Each plugin supplies its own description, so just select a plugin from the list of calc plugins as shown in Fig. 5.12 and consult the plugin description for an explanation of what the plugin does.

5.2.3. Scaffold Tree Generation

Scaffold Hunter reduces each molecule of a dataset to its scaffold and arrange the resulting scaffolds hierarchically in a tree structure called *Scaffold Tree*. Scaffold Trees were first described by [4] and allow navigation through the molecules in a meaningful manner. In this chapter you will learn how to generate a scaffold tree.



Tree title: Example tree

Tree description: This tree is an example for the use of custom rulesets

● Use custom ruleset: custom ruleset

Please wait until the scaffold tree is generated.
Processing molecule 174 of 602.

(a) Scaffold Tree Generation Dialog

(b) Tree Generation Progress

Figure 5.14.: Dialog (a) allows you to configure tree generation; (b) shows the dialog indicating the process during tree generation

Fig. 5.14a shows the Scaffold Tree Generation Dialog, where you have to fill in a tree title, can fill in an explaining description of the tree and have to choose some generation options. See *Sec. 5.2.3.1: Using Custom Rules* and *Sec. 5.2.3.2: Deglycosylate Molecules First* to learn more about the options you have. By clicking "Cancel", you will be returned to the *Sec. 5.2: Dataset Management* dialog. By clicking "Generate Tree" Scaffold Hunter will generate and store the tree with the given options, while showing the progress in a window shown in Fig. 5.14b.

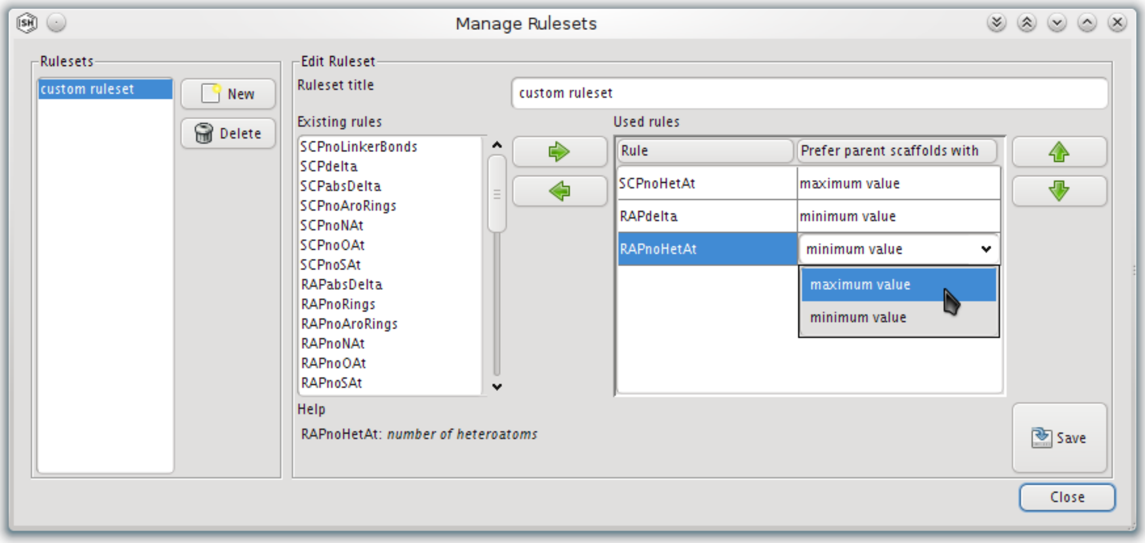
5.2.3.1. Using Custom Rules

The ring pruning rules define an order according to which rings are pruned from the scaffold for the generation of the scaffold tree. In each step of the tree generation, starting from a child scaffold, all possible parent scaffolds are generated that could be obtained by the removal of one peripheral ring. The hierarchical set of rules is used to select one pruning among the candidates, i.e. one ring to be pruned. Thereby each rule might decrease the number of candidate prunings until only a single solutions remains. In the default mode the rules are applied exactly as described in the Scaffold Tree publication [4]. To provide more flexibility with ring pruning, the custom rules option allows to define a customized subset of rules in a desired order selected from a wider range of implemented rules. For this purpose a generic framework for rules was defined. Each rule consists of two parts:

1. a numerical descriptor that describes either each of the potentially pruned rings or each of the different remaining parent scaffolds that could be generated from the child scaffold, and
2. a selection criterion that defines whether the candidate prunings with the minimum or the maximum descriptor value should be retained (i.e. each pruning candidate except the minimum / maximum are removed from the set of remaining pruning operations) and passed on to the next rule until only one pruning remains.

If multiple solutions have the minimum / maximum rule descriptor value, all of these are retained. If all pruning candidates of a scaffold result in the same descriptor value, all pruning candidates are passed on to the next rule. Like with in the original ruleset, if the custom rules are not successful to identify a single pruning, a tie-break rule is evoked that selects the first scaffolds after lexicographically sorting the parent scaffolds according to their canonical SMILES strings.

To choose a set of rules for tree generation, select a ruleset from the drop-down box in the **Scaffold Tree Generation Dialog** shown on Fig. 5.14a. If no appropriate ruleset exists, you first have to click on "Edit Rulesets" to open the **Ruleset Management Dialog** and create a custom ruleset.



Ruleset title: custom ruleset

Rule	Prefer parent scaffolds with
SCPnoHetAt	maximum value
RAPdelta	minimum value
RAPnoHetAt	minimum value

Help
RAPnoHetAt: *number of heteroatoms*

Figure 5.15.: Ruleset Management Dialog

On the left side of Fig. 5.15 you see a list of currently available rulesets. If no ruleset is available you can create a new one by clicking "New Ruleset".

If you select a ruleset from the ruleset list, then you can edit the ruleset with the controls located on the right side of the dialog. You have to enter a name for the ruleset and you can choose rules from the existing ruleset list on the left side of the dialog to be used in your custom ruleset. To use a rule, select it and move it to the used rule list by clicking on the right arrow button. You can even select and move multiple rules by using the `Shift` or `Ctrl` modifier keys during selection.

Each row in the used rule list represents one rule, with the descriptor name in the first column, and the selection criterion in the second column. To change the selection criterion of a rule, select ”maximum value” or ”minimum value” from the corresponding drop-down box.

The order of the rules in the used rules list determines the order of execution of the rules. You can change the execution order of a rule by selecting it first and then clicking on the arrow buttons to move it up or down.

If you finished editing your custom ruleset, click the “Save” button to save the ruleset.

An adaption of the default rules within this framework would look like (for detailed explanation of the rule name see below):

```
RRPsize11p "minimum value"
SCPnoLinkerBonds "minimum value"
SCPabsDelta "maximum value"
SCPdelta "maximum value"
RRPsize6 "maximum value"
RRPsize5 "maximum value"
RRPsize3 "maximum value"
RRPnoHetAt "minimum value"
RRPnoNAt "minimum value"
RRPnoOAt "minimum value"
RRPnoSAt "minimum value"
RRPringSize "minimum value"
SCPnoAroRings "minimum value"
RRPhetAtLinked "maximum value"
```

Implemented Rule Descriptors Three classes of descriptors are provided:

`SCP` (SCaffold Properties)	properties of the resulting parent scaffold after pruning
`RAP` (Ring Assembly Properties)	properties of the ring assembly containing the ring to be pruned
`RRP` (Removed Ring Properties)	properties of the ring to be pruned

Detailed Description of the Properties The names on the left are the rules that can be used in a ruleset.

Rule	Description
`SCPnoLinkerBonds`	number of acyclic linker bonds
`SCPdelta`	delta value indicating nonlinear ring fusions, spiro systems bridged systems as defined in [4]
`SCPabsDelta`	absolute delta value indicating nonlinear ring fusions, spiro systems bridged systems as defined in [4]
`SCPnoAroRings`	number of aromatic rings
`SCPnoHetAt`	number of heteroatoms

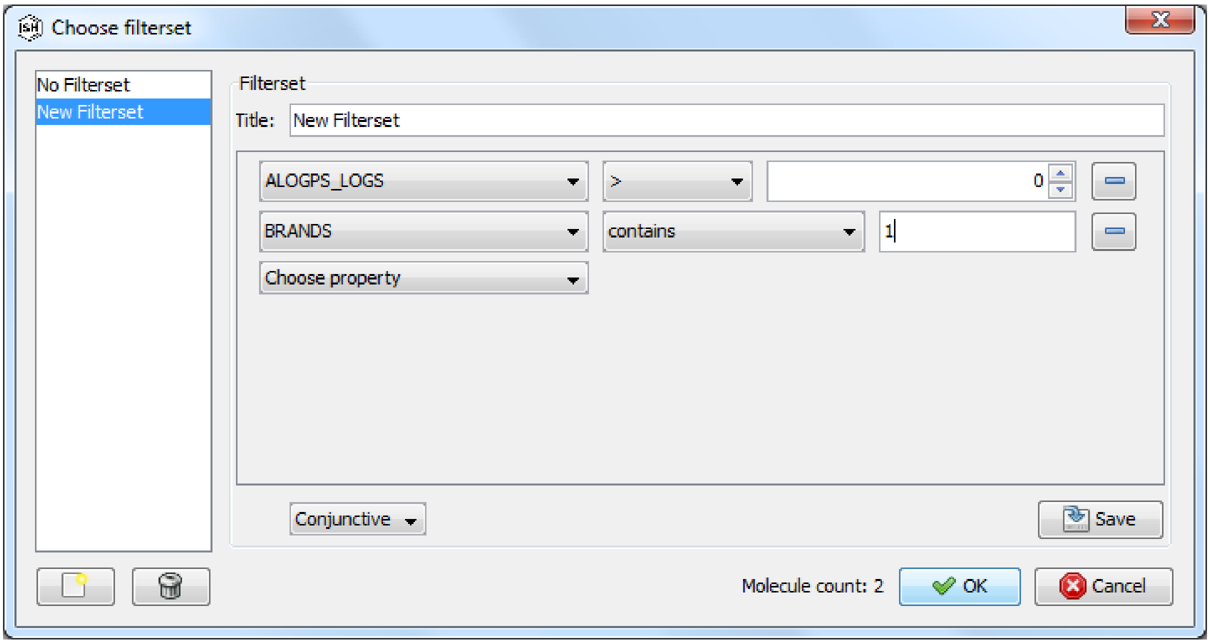
`SCPnoNAt`	number of nitrogen atoms
`SCPnoOAt`	number of oxygen atoms
`SCPnoSAt`	number of sulfur atoms
`RAPdelta`	delta value
`RAPabsDelta`	absolute delta value
`RAPnoRings`	number of rings
`RAPnoAroRings`	number of aromatic rings
`RAPnoHetAt`	number of heteroatoms
`RAPnoNAt`	number of nitrogen atoms
`RAPnoOAt`	number of oxygen atoms
`RAPnoSAt`	number of sulfur atoms
`RRPringSize`	size of removed ring
`RRPnoHetAt`	number of heteroatoms
`RRPnoNAt`	number of nitrogen atoms
`RRPnoOAt`	number of oxygen atoms
`RRPnoSAt`	number of sulfur atoms
`RRPhetAtLinked`	binary descriptor ("maximum value" = `True`, "minimum value" = `False`) indicating whether removed ring was linked via a linker to a heteroatom in a ring
`RRPsize3`	binary descriptor: removed ring of size 3
`RRPsize4`	binary descriptor: removed ring of size 4
`RRPsize5`	binary descriptor: removed ring of size 5
`RRPsize6`	binary descriptor: removed ring of size 6
`RRPsize7`	binary descriptor: removed ring of size 7
`RRPsize8`	binary descriptor: removed ring of size 8
`RRPsize9`	binary descriptor: removed ring of size 9
`RRPsize10`	binary descriptor: removed ring of size 10
`RRPsize11`	binary descriptor: removed ring of size 11
`RRPsize11p`	binary descriptor: removed ring of size more than 11
`RRPlinkerLen1`	binary descriptor: removed ring connected via linker of length 1
`RRPlinkerLen2`	binary descriptor: removed ring connected via linker of length 2
`RRPlinkerLen3`	binary descriptor: removed ring connected via linker of length 3
`RRPlinkerLen4`	binary descriptor: removed ring connected via linker of length 4
`RRPlinkerLen5`	binary descriptor: removed ring connected via linker of length 5
`RRPlinkerLen6`	binary descriptor: removed ring connected via linker of length 6
`RRPlinkerLen7`	binary descriptor: removed ring connected via linker of length 7
`RRPlinkerLen7p`	binary descriptor: removed ring connected via linker of length more than 7

Table 5.1.: Rule descriptors

5.2.3.2. Deglycosylate Molecules First

This check box activates the deglycosylation of molecules that was described in [3]. The procedure iteratively removes terminal pentose and hexose sugars based on a substructure matching before generation of the scaffold tree. It is especially useful when processing natural product structures, i.e. from the Dictionary of Natural Products (DNP), that contain molecules with multiple glycosylation patterns.

5.3. Filtering



No Filterset
New Filterset

Title: New Filterset

ALOGPS_LOGS > 0
BRANDS contains 1
Choose property

Conjunctive

Molecule count: 2

Figure 5.16.: Filter Dialog

The Filter Dialog shown in Fig. 5.16 is used to filter the molecules of a dataset, so that you can work with molecules relevant to your problem. On the left hand side of the dialog a list shows the previously used filtersets. A filterset is a set of filters. One filter defines a constraint for one property of the molecules or scaffolds of the dataset. A combination of filters can be used to filter the dataset by more than one property. In the lower left corner of the window there are buttons to add a new filterset, or to delete an existing one. After selecting one of these filtersets, or after creating a new one, the right hand side of the window shows the details of the selected filterset. Here the title can be changed and filters can be added or modified. To add a new filter, simply select a property from the "Choose property" box. For this property, the user can then select a constraint in the appearing drop-down box. For numerical properties, this box holds comparison operators like "greater" (>) or "equal" (=). For text properties, the user can select comparison operators like "contains" or "ends with". Both property types can also be filtered by whether or not they are defined. With the "minus" button at the end of each row, the corresponding filter can be removed. With the "conjunctive / disjunctive" box you can define the operator used to combine the filters. With the conjunctive method, all filters have to match to allow a molecule to be added to the resulting dataset. With the disjunctive method, only one filter needs to match.

The "Save" button simply saves the changes made to the selected filterset. To use a filterset, select it and click the "Ok" button. If the filterset was not saved before clicking "Ok" and you continue without saving, the current filterset is used for filtering, but the changes are discarded afterwards.

6. Main Window

Scaffold Hunter includes several *views*, that allow the user to visualize and navigate through the imported chemical data in various ways. Currently the following types of views are supported:

Scaffold tree view The scaffold tree view shows chemical structures arranged in a scaffold tree. This is the only type of visualization that was supported by Scaffold Hunter version 1.x, and is still a pivotal part of the application.

Dendrogram view The dendrogram view shows the result of a hierarchical clustering of chemical structures, supporting various linkage methods and distance measures.

Plot view The plot view shows molecules in a two- or three-dimensional scatter plot based on the molecules' properties.

Table view The table view shows molecules in tabular form, including their structure, description and properties.

See *Sec. 6.3: Managing and Arranging Views* for an explanation of how to open and manage views within Scaffold Hunter. The individual types of views are described in greater detail in *Chap. 7: Views*. Another concept crucial to Scaffold Hunter is that of subsets, as explained in *Sec. 6.2: Subset Management*.

6.1. Structure of the Main Window

The basic structure of the Scaffold Hunter main window stays the same regardless of the type of view that's currently in use. However, most of the individual parts of the user interface will change when switching between different views, to accommodate the functionality of each view type.

The main window can roughly be divided into five parts:

Menu bar This is the main menu bar at the top of the Scaffold Hunter window, through which most of Scaffold Hunter's functionality is accessible. Most of the menu items are available regardless of the current view. One notable exception is the view-specific sub-menu, which changes to reflect the actions available in each given view type.

Toolbar The toolbar is located below the menu bar. Most of the toolbar items are view-specific, and will change when switching to a different type of view.

View tabs This tab bar is the central part of the Scaffold Hunter window, and contains tabs for each view. Managing views is described in *Sec. 6.3: Managing and Arranging Views*.

Sidebar The sidebar is shown on the left hand side of the main window, and can optionally be hidden using the “Show Side Bar” button in the “Window” menu or in the toolbar. It consists of several panels that can be folded or expanded independently by clicking the corresponding caption bar. Each panel displays additional information about the data that is visible in the currently active view.

Subset bar The subset bar on the right hand side of the window shows a tree of all subsets in the current session. Like the sidebar, this part of the user interface can be hidden, using the “Show Subset Bar” button in the “Window” menu or in the toolbar. In addition to the subset tree, the subset bar also contains buttons for creating new subsets from the selection. Subset management is explained in detail in *Sec. 6.2: Subset Management*.

6.2. Subset Management

A central part of the Scaffold Hunter user interface is dedicated to the concept of *subsets*. Each subset identifies a set of molecules, all of which are part of the main dataset (hence a subset thereof). Subsets are arranged in a tree structure, where each subset can have any number of child subsets. Every subset is guaranteed to contain only molecules that are also in its parent subset.

At the top of the subset tree is the *root subset*, which is automatically created when starting a new Scaffold Hunter session. The root subset may be identical to the dataset that has been imported, but it can also be smaller in case a filter was used when creating the session.

6.2.1. The Subset Tree

The subset tree is the main part of the subset bar at the right hand side of the Scaffold Hunter main window. In a newly created session, the tree contains only the root subset. All subsets, including the root subset, can be assigned a custom name and comment. The current subset, i.e. the subset that is shown in the currently active view, is always shown in bold. The subset tree itself only contains the name of each subset. Additional information about subsets can be obtained by hovering the mouse cursor above the subset name, which will pop up a tooltip window.

Most functionality regarding subsets is available using the context menu of the subset tree. For convenience the main menu also contains the “Subset” menu, which always refers to the current subset.

6.2.2. Creating Subsets from the Selection

The most straightforward way to create a new subset is by first selecting all molecules to be included in the subset, and then making a subset from the selection. Since there is only one, global selection in Scaffold Hunter, which may include molecules that are not part of the current view, there are two slightly different ways to decide which molecules are to be included in the new subset.

Subset from the total selection In order to create a subset including all selected molecules, regardless of the current view, choose “Selection → Make Subset” in the main menu, or use the first “Make Subset” button at the bottom of the subset bar. Since the new subset does not necessarily have any relation to any of the existing subsets, it will be added to the subset tree as a child of the root subset.

Subset from the selection in the current view To create a subset including only the selected molecules that are also in the current view, choose “Selection → Make Subset from Current View”, or use the second “Make Subset” button in the subset bar. the new subset is guaranteed to include no molecules that are not part of the current view’s subset, so that subset will be used as its parent.

6.2.3. Deleting Subsets

The subset of the current view can be deleted by selecting “Subset → Delete” from the main menu. In the subset tree, subsets are deleted by selecting them and choosing “Delete” from the context menu. Deleting a subset always closes any open views currently associated with that subset.
If the subset being deleted has any children, those will not automatically be deleted as well. Instead, they will be re-attached to the parent of the subset being deleted. It is not possible to delete the root subset.

6.2.4. Set Operations

The basic set operations of *union*, *intersection* and *difference* can be used to create new subsets from existing subsets. Since these operations require a selection of two or more subsets, they are not available in the “Subset” menu, but only in the context menu of the subset tree.
As in many other user interfaces, the selection of multiple subsets in the tree is possible using `Ctrl + Left click` to add or remove a single subset to/from the selection. To select multiple adjacent subsets, `Shift + Left click` can be used. Once all the desired subsets have been selected, open the context menu by right-clicking on one of them, and choose “Make Union”, “Make Intersection” or “Make Difference”. You will then be asked to enter a name for the new subset.

Since the subset tree requires each subset to be an actual subset of its parent, the set operations differ slightly in where the newly created subset will be added to the tree. For the difference operation, the order of subsets is also relevant.

Union Generally speaking, none of the original subsets is a superset of the new subset. As a result the new subset cannot be added below any of them in the tree. In order to ensure that the new subset’s position is both deterministic and as meaningful as possible, the parent will always be the lowest common ancestor of all the selected subsets.

Intersection All selected subsets would be a valid parent for the new subset. For deterministic results, the subset that was clicked on when opening the context menu will be the new subset’s parent.

Difference Mathematically, the difference operation is only defined for two sets. The difference operation used here deviates from this definition by extending it to mean “one subset, minus the union of an arbitrary number of other subsets”. The base subset, from which the others will be subtracted, is the one that was clicked on when opening the context menu. This is also the subset that will then become the new subset’s parent.

6.2.5. Filtering

The same filtering that is available while creating a session can also be used on individual subsets. Select “Subset → Filter” to start filtering the subset of the current view, or choose “Filter” in the context menu of any subset in the subset tree. The Filter Dialog itself works just like the one that is used during session creation, as described in *Sec. 5.3: Filtering*.
Once the filter settings have been applied, you will be asked to enter a name for the new subset, and the subset will be added to the subset tree as a child of the original subset.

6.3. Managing and Arranging Views

A newly created Scaffold Hunter session will contain four views, one of each type (scaffold tree, dendrogram, plot and table), with each view showing the root subset. This setup is meant to serve only as a starting point. The number of simultaneous views is only limited by the system's available memory, and each view can be associated with a different subset and different settings.

6.3.1. Opening and Closing Views

There are several ways of creating a new view. The "Window → Add View" submenu can be used to create a new view of a given type, showing the root subset. A more flexible way of opening new views is using either the "Subset" menu or the subset tree's context menu:

Show in Current View (only in subset tree context menu)
Replaces the subset shown in the currently active view, while otherwise retaining the view's settings.

Show in New View
Creates a new tab in the current window, showing the selected subset in the chosen type of view.

Show in New Window
Creates a new window and adds a new tab to it, showing the selected subset in the chosen type of view.

Note that the "Subset" menu always refers to the current view's subset, while the context menu refers to the subset being clicked on.

Usually the fastest way to close a view is by using the 'X' button in the view's tab, but there are also menu items ("Window → Close View") and a keyboard shortcut ("Ctrl+W") available.

6.3.2. Split Windows

It is possible to simultaneously show two views in the same window, either side by side or one above the other. To split the window, use "Window → Split Horizontally" and "Window → Split Vertically". This will create a new, initially empty tab bar, that can then be filled by moving views to it (see *Sec. 6.3.4: Moving Views*). The same menu items can also be used to change the orientation of an existing split.

To get back to a window with no split, use "Window → Single Tab Bar". This will join all views from both sides of the split in a single tab bar.

6.3.3. Multiple Main Windows

Sometimes it can be convenient to open views in separate windows, for example to place them on different monitors, or to avoid a tab bar getting too crowded. Scaffold Hunter supports this by allowing multiple main windows within the same instance of the application. Windows can be opened using "Window → Open New Window", and closed using "Window → Close Window". Note that while each window can have its own views and layout, all windows still work on the same dataset and subset tree. Thus, opening multiple main windows is not the same as opening multiple instances of Scaffold Hunter.

6.3.4. Moving Views

Within a single tab bar, views can easily be moved by simply dragging their tab to a new position.
It's currently not possible to move views to a different tab bar (in case of split views) or to a different

window via drag & drop. To move windows from one window or tab bar to another, use “Window → Move View to Window” and “Window → Move View to Tab Bar”.

6.3.5. Renaming Views

Views are initially named after their type. For example, the tab of a newly created scaffold tree view will be called “Scaffold Tree”. Views can be renamed to give them a more descriptive label. To do so, select “Window → Rename View” in the main menu, or “Rename” in the tab’s context menu.

6.4. Interaction between Views

6.4.1. Selection

In order to create subsets containing molecules of interest it is possible to select several molecules in the views, which are then highlighted in red in all views.

Single molecules or the molecules belonging to a scaffold can be selected or deselected by left clicking a molecule or a scaffold or in some views by dragging a selection box around them or by using the scaffold’s context menu. Another possibility to change the selection is the subset menu. If the menu has been opened via the main menu, it offers options to add the current subset of the active view to the selection, to remove it, or to replace the selection. The same applies to the subset which has been used to open the context menu in the subset bar.

The user can also select all molecules of the dataset, clear the selection, invert the current selection or confine the current selection to the molecules that are visible in the active view by using the selection menu in the main menu.

The total number of molecules selected and the number of molecules selected that are visible in the active view are displayed at the bottom of the subset bar. Both have an associated “Make Subset” button that can be used to create a new subset from the respective set.

Selection colors Table 6.1 shows the possible colors of molecules and scaffold, depending on their selection state:

Display color	Selection state
black	Unselected molecule or scaffold
red	Selected molecule
red	Completely selected scaffold (all molecules belonging to the scaffold are selected)
orange	Partially selected scaffold (some molecules belonging to the scaffold are selected)
gray	Virtual scaffold (scaffold with no child molecules; cannot be selected)

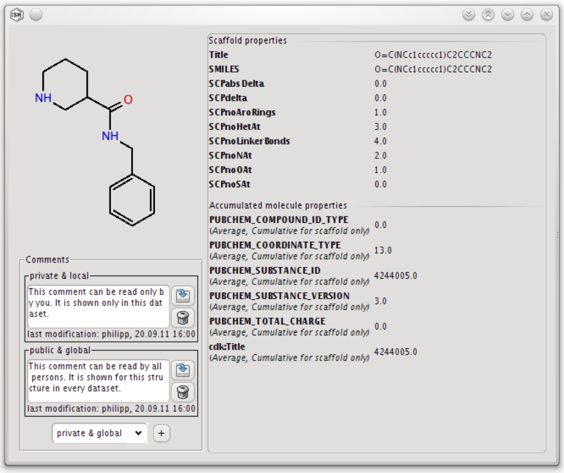
Table 6.1.: Selection colors

6.4.2. Flags

Besides the selection, there is another kind of marking that is shared by all views: flags. Flags are intended as a fast way of marking molecules and scaffolds, for example as *already seen* or *investigate later*, and are available in a public and in a private variant. Public flags are visible to everyone working on the dataset, while private flags are only visible to the user who created them.

The user can mark molecules and scaffolds with flags by using the context menu or the main menu entry "Selection". If the context menu is used, the private and public flags of the molecule or scaffold under the cursor can be toggled. The selection menu allows the user to add or remove a public or private flag for all molecules of the current selection.

6.5. Tooltip Window and Comments



Scaffold properties

Title	O=C(NCc1ccccc1)C2CCCNC2
SMILES	O=C(NCc1ccccc1)C2CCCNC2
SCPabs Delta	0.0
SCPdelta	0.0
SCPnoAroRings	1.0
SCPnoHetAt	3.0
SCPnoLinkerBonds	4.0
SCPnoNAt	2.0
SCPnoOAt	1.0
SCPnoSAt	0.0

Accumulated molecule properties

PUBCHEM_COMPOUND_ID_TYPE *(Average, Cumulative for scaffold only)*	0.0
PUBCHEM_COORDINATE_TYPE *(Average, Cumulative for scaffold only)*	13.0
PUBCHEM_SUBSTANCE_ID *(Average, Cumulative for scaffold only)*	4244005.0
PUBCHEM_SUBSTANCE_VERSION *(Average, Cumulative for scaffold only)*	3.0
PUBCHEM_TOTAL_CHARGE *(Average, Cumulative for scaffold only)*	0.0
cdk:Title *(Average, Cumulative for scaffold only)*	4244005.0

This comment can be read only by you. It is shown only in this dataset.

last modification: philipp, 20.09.11 16:00

This comment can be read by all persons. It is shown for this structure in every dataset.

last modification: philipp, 20.09.11 16:00

Figure 6.1.: Tooltip

An interactive tooltip window (see Fig. 6.1) is available for the scaffold tree view and the dendrogram view when you move your mouse over a molecule or scaffold. It combines the following components in one window.

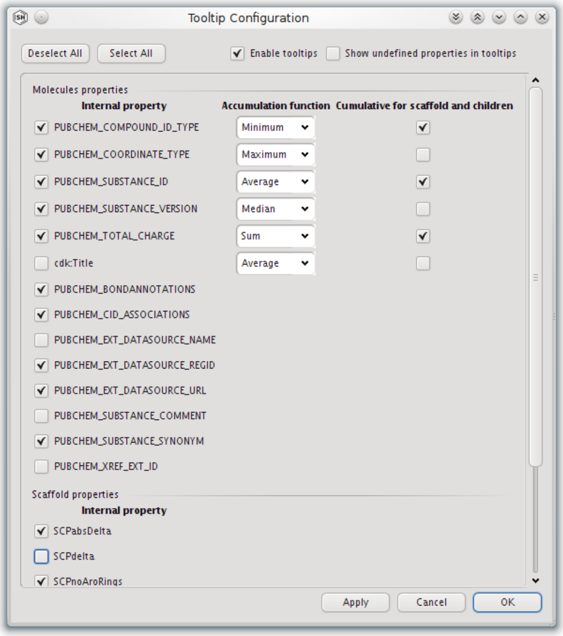
- a large **image** of the structure
- configurable information on the **properties**
- a component to view, add, delete or modify **comments**

The window behaves different from a normal tooltip in some circumstances. Like a normal tooltip it disappears as soon as the mouse leaves molecule, scaffold or tooltip window. But if you click inside the window the window will stick and stay until the focus of the window is lost. This enables you to move around the mouse freely and (more important) lets you edit the comments more safely.

6.5.1. Configuring the Tooltip

You can find some general options at `Session` → `Preferences` → `General Configuration` in the menubar.

- **enable/disable** tooltip
- the **size** of the image shown in the tooltip
- whether it shows **undefined properties** or not
- the **delay** after which the tooltip is shown



☑ Enable tooltips ☐ Show undefined properties in tooltips

Molecules properties

Internal property	Accumulation function	Cumulative for scaffold and children
☑ PUBCHEM_COMPOUND_ID_TYPE	Minimum	☑
☑ PUBCHEM_COORDINATE_TYPE	Maximum	☐
☑ PUBCHEM_SUBSTANCE_ID	Average	☑
☑ PUBCHEM_SUBSTANCE_VERSION	Median	☐
☑ PUBCHEM_TOTAL_CHARGE	Sum	☑
☐ cdk:Title	Average	☐
☑ PUBCHEM_BONDANNOTATIONS		
☑ PUBCHEM_CID_ASSOCIATIONS		
☐ PUBCHEM_EXT_DATASOURCE_NAME		
☑ PUBCHEM_EXT_DATASOURCE_REGID		
☑ PUBCHEM_EXT_DATASOURCE_URL		
☐ PUBCHEM_SUBSTANCE_COMMENT		
☑ PUBCHEM_SUBSTANCE_SYNONYM		
☐ PUBCHEM_XREF_EXT_ID		

Scaffold properties

Internal property
☑ SCPabsDelta
☐ SCPdelta
☑ SCPnoAroRings

Figure 6.2.: Tooltip Configuration

6.5.2. Configuring the Properties

You can find a dialog to configure the properties at `Session` → `Tooltip Configuration` in the menubar. As you can see in Fig. 6.2 you have a check box in front of every property to enable or disable the property in the tooltip. As the tooltip can be shown for molecules and scaffolds there are two types of properties. If the tooltip is displayed over a molecule only the molecule properties will be shown.

The properties for scaffolds are somewhat more complex. You can derive accumulated properties for scaffolds from the molecule properties that belong to the displayed scaffold. Therefore you can choose an accumulation function for each numerical molecule property. For example you can display the average, minimum or maximum of the molecule property when using the tooltip with scaffolds. If you enable the checkbox `Cumulative for scaffold and children` this accumulation function is applied to the molecules of the scaffold and all molecules that belong to any child scaffold.

6.5.3. Comments

You can add, modify and delete comments for a molecule or scaffold directly in the tooltip. You can find the controls in the bottom left corner of the tooltip (see Fig. 6.1).

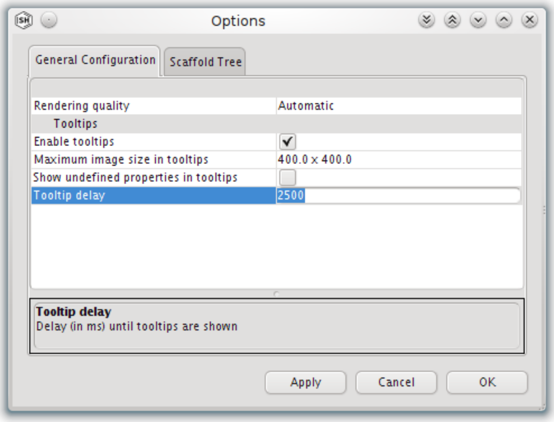
Private, public, local and global comments Comments can be private or public and local or global. Therefore you have 4 different types of comments (private & local, private & global, public & local, public & global). If a comment is private it is only accessible by the user who created it, while a public comment can be seen by any other user working on the same database. If a comment is local it is limited to the current tree. If you choose global instead it will be visible for all datasets and all trees on the current database for molecules or scaffolds with the same structure.

Add, modify or delete a comment To add a new comment select the type in the drop-down box and click the plus button. Note that only one comment of each type can exists for each molecule or scaffold.

Therefore not all or no types may be available in the drop-down box. After adding a new comment a text field with a save button, delete button and some information about the modification will appear. You can now modify the (empty or previously existent) comments by editing the text in the text field and pressing the save button afterwards. The save button will be only accessible if you already modified the text. Otherwise it will be greyed out. To delete a comment simply press the delete button. The text field with the according controls will disappear.

6.6. Settings

The Scaffold Hunter settings are divided into two types. There are global preferences and settings for a specific view. All settings are stored in the database. Therefore your settings travel with your profile.



Rendering quality	Automatic
Tooltips	
Enable tooltips	☑
Maximum image size in tooltips	400.0 x 400.0
Show undefined properties in tooltips	☐
Tooltip delay	2500

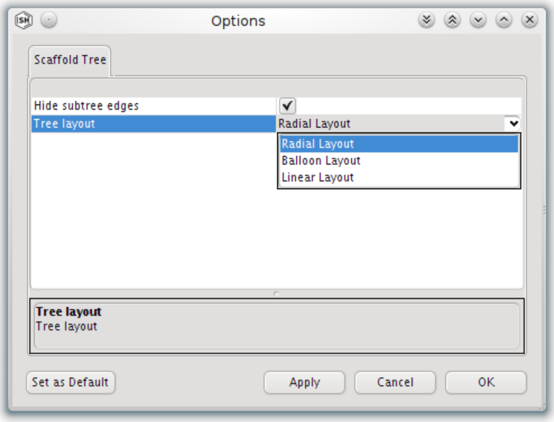
Tooltip delay
Delay (in ms) until tooltips are shown

Figure 6.3.: Preferences Dialog

Preferences The global preferences are accessible via `Session` → `Preferences` in the menu bar. The Dialog Fig. 6.3 has a `General Configuration` tab with settings that will apply to all views and the main window. The options are self explaining. If you select an option by clicking on it you get a more detailed description in the label below.

There are also tabs for types of views. For example you can find general options for the scaffold tree view. These settings will apply to all views of one type that are open (e.g. all scaffold tree views). Currently only the scaffold tree view uses this global settings, but in future there may be also other views available here.

Configure View The preferences for one specific view are accessible over `Session` → `Configure View`. The dialog Fig. 6.4 has the same layout as the global preferences. It shows one tab for each open view. If you have two open scaffold tree views you will find two tabs here. The name of the tabs matches the name of the view tabs in the main window. Renaming the views can thereby significantly improve the assignment between preferences tabs and the corresponding view (see *Sec. 6.3.5: Renaming Views*). Some options may also be available directly over the menu bar or tool bar (for further details see *Sec. 7: Views*).



Options

Scaffold Tree

Hide subtree edges ☑

Tree layout Radial Layout

Tree layout
Tree layout

Figure 6.4.: Configure View Dialog

7. Views

This chapter will give you a deeper understanding of each available view. If you want to know how to handle, manage and arrange views in general terms you may want to read *Sec. 6.3: Managing and Arranging Views* instead.

7.1. Scaffold Tree

The scaffold tree view visualizes a scaffold tree, which was generated as described in *Sec. 5.2.3: Scaffold Tree Generation*.

7.1.1. Navigation

A new scaffold tree view will show an overview of the whole graph. You may zoom in to get more detailed information on a part of the graph. Depending on the scale, scaffolds will be represented simplified by a rectangle or by their structural formula. There are several ways to zoom in/out: You may use the mouse wheel to magnify/demagnify the view on the area of the mouse pointer or by using the buttons in the toolbar or “Tree” menu. You may also select the area of interest by a rectangle on the GraphMap, see *Sec. 7.1.1.1: GraphMap*. The other navigation mechanism is panning: By keeping the left mouse button pressed on an empty area in the graph pane and moving the mouse the viewpoint is panned. You may also use the scrollbars for panning.

Usually not all scaffolds are shown when a new scaffold tree view is opened. Subtrees beginning at a scaffold can be shown or hidden respectively by clicking on the $+/-$ symbol at the lower left corner of the scaffold. In addition, the context menu (*Sec. 7.1.2: Context Menu*) contains options to open complete subtrees or to open all children on the next hierarchy level. Inverse operations exist to hide scaffolds you are not interested in. The command to expand the whole tree or to reset it to the default hierarchy level can be invoked via the toolbar or “Tree” menu (*Sec. 7.1.3: Tree Menu & Toolbar*).

A scaffold will be shown in different colors based on the global selection as described in *Sec. 6.4.1: Selection*. A left-click on an unselected or partially selected scaffold will add all molecules associated with the scaffold to the selection, a left-click on a fully selected scaffold will deselect all associated molecules.

You may also navigate using keyboard shortcuts, listed in *App. A: Keyboard Shortcuts*. There is a cursor set on one scaffold marked by a blue rectangle. The cursor can be moved using the arrow keys. The camera automatically retains focus on the cursor.

7.1.1.1. GraphMap

The GraphMap is found in the sidebar on the left hand side and provides an overview on the whole graph. The current viewport of the graph pane is marked by a transparent red rectangle. Dragging this rectangle with the mouse will change the viewport of the graph pane accordingly. Click the left mouse button at an area that is not currently visible to center the viewport on the selected point. You can define a new viewport by marking a rectangle on the graph map: Click the left mouse button on an empty area and keep it pressed while moving the mouse. When releasing the mouse button the view will be panned and zoomed to completely contain the selected area. Using your mouse wheel on the GraphMap will magnify or demagnify the graph pane at the specified point.

7.1.1.2. Magnifying Glass

The Magnifying Glass in the sidebar below the GraphMap is opened by a mouse click on its title bar. It shows the area below the mouse pointer magnified. You may move the mouse pointer over the graph pane and navigate in the graph as usual while the Magnifying Glass is active.

7.1.2. Context Menu

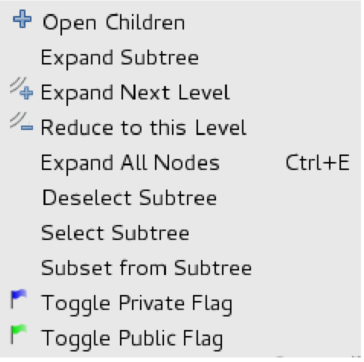


Figure 7.1.: The context menu invoked by a right-click on a scaffold.

Click the right mouse button on a scaffold to access the context menu. The menu mainly contains options that are directly related to the scaffold: You can open/close the children of the scaffold, open the whole subtree rooted at the scaffold or open all children on the next hierarchy level by the option "Expand next level". In addition you can choose to expand the whole tree, by clicking on "Expand all Nodes". You can hide the hierarchy levels under the scaffold by choosing "Reduce to this level".

The commands "Select Subtree" and "Deselect Subtree" add or remove all molecules from the selection, which are associated with scaffolds located in the subtree that is rooted at the current scaffold. The command "Subset from Subtree" will create a new subset which contains these molecules.

There are two additional entries to set or remove public and private flags, see *Sec. 6.4.2: Flags*.

7.1.3. Tree Menu & Toolbar

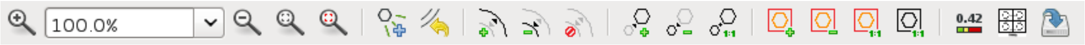


Figure 7.2.: Meaning of the icons (from left to right): Zoom In, Zoom Out, Fit Graph, Fit Selection, Expand all Nodes, Number of Rings to Default Level, Increase/Decrease Radius, Lock Radii while Zooming, Scale cusor node up/down, Normalize cursor node, Scale Selected Scaffolds up/down, Normalize Selected Scaffolds, Normalize all Scaffolds, Configure Property Mappings, Show Scaffold Molecules, Export Image

The "Tree" menu provides access to many different functions related to the scaffold tree view. The toolbar shown in *Fig. 7.2* allows quick access to most of these functions. When you hover the cursor over an item of the menu or toolbar a tooltip with a description will appear. It follows a list of the menu items, each with a short description.

Layout Can be used to choose one of the three available layout algorithms. The graph in the current tab will be laid out accordingly. See *Sec. 7.1.4: Layout* for a brief description of the layouts.

Show Scaffold Molecules Toggles the molecule view for each node in the scaffold tree, which displays the molecules associated with a scaffold.

Configure Property Mappings Allows to map different imported properties to several visual features of the nodes in the scaffold tree. See *Sec. 7.1.6: Property Mappings* for details.

Zoom In/Out Changes the zoom level of the graph pane.

Fit Graph Changes the zoom level and pans the graph pane to show the whole graph.

Fit Selection Changes the zoom level to show all selected and partially selected scaffolds.

Expand all Nodes Expands the whole Graph.

Number of Rings to Default Level Resets the graph, such that only the hierarchy levels which are shown by default are displayed.

Increase/Decrease Radius Allows adjustment of the distance between two adjacent rings in the radial layout.

Lock Radii While Zooming Disables/enables the automatic adjustment of the distance between two adjacent rings in the radial layout.

Node Scale Resizes the scaffolds of the graph. There are buttons for scaling up/down all currently selected scaffolds or just the scaffold the cursor is on. In addition "Normalize" buttons are provided to reset scaffolds to their original size.

Export Image Allows exporting the current viewport or the whole graph as an image.

7.1.4. Layout

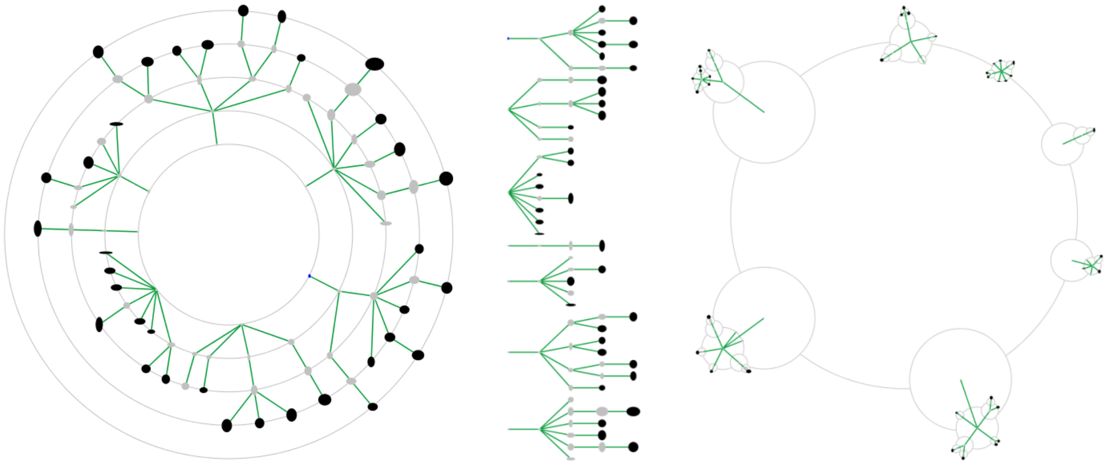

(a) (b) (c)

Figure 7.3.: Different layouts of the same graph: (a) Radial Layout, (b) Linear Layout, (c) Balloon Layout

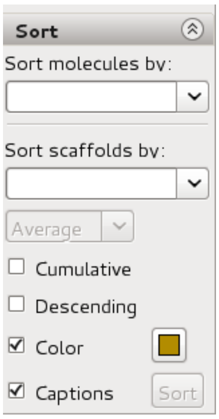
Scaffold Hunter offers three different layouts, which can be selected by using the "Layout" option in the "Tree" menu *Sec. 7.1.3: Tree Menu & Toolbar*. The different layouts are shown in *Fig. 7.3*. The Radial Layout (*Fig. 7.3a*) is the default layout method. With this layout the scaffolds are located according to their depth on circles with a common center. The Linear Layout (*Fig. 7.3b*) orders the scaffolds from left to right. Both layouts emphasize the depth of scaffolds. With the Balloon Layout (*Fig. 7.3c*) each scaffold is the center of a circle on which its children are distributed. This layout separates subtrees more clearly from each other.

For the Radial Layout the distance between the circles may be customized. By default the distance is automatically adjusted depending on the zoom. The “Lock Radii While Zooming” button disables this mechanism and allows the user to set the distance using the buttons “Increase Radius”/“Decrease Radius”.

7.1.5. Sorting

Initially the ordering of scaffolds in the tree is chosen in a deterministic way but without any meaning. Alternatively scaffolds can be sorted according to some property value using the “Sort” sidebar panel show in *Fig. 7.4*.

The same panel can be used to define the ordering of the molecules which are shown when the option “Show Scaffold Molecules” is toggled. To do so select the property by which these molecules should be sorted in the uppermost box.



☐ Cumulative
☐ Descending
☑ Color
☑ Captions

Figure 7.4.: The Sidebar sort panel

The property used for sorting can be chosen using the box below “Sort Scaffolds by”. An accumulation method to determine how scaffold properties will be derived from the properties of associated molecules can be selected in the box below. The checkboxes underneath can be used to add a colored background to the graph, where each segment which has the same value of the selected property is colored in the same shade of the selected color. A label showing this value can be displayed in each segment.

After clicking on “Sort” the scaffolds on the first layer will be reordered according to their property values. Sorting by property values is applied only to the scaffolds on the first layer. You can however create a subset out of a subtree and sort the scaffold tree displaying the subset.

7.1.6. Property Mappings

Properties associated with the scaffolds can be mapped to different *visual properties* of the Scaffold Tree. Scaffold properties are either directly associated with a scaffold, such as *Number of Oxygen Atoms* or can be derived from the properties of the structures which are associated with a scaffold.

The following *visual properties* are available for mapping:

Node Background Color This changes the background color of the scaffold nodes in the scaffold tree. Either two colors can be selected and the value range of the property is mapped to the gradient between those two colors or value intervals can be specified and a color can be assigned to each interval.

Edge Thickness This maps the absolute difference between the values associated with two adjacent scaffolds onto their connecting edge. The maximum and minimum displayed thickness are predefined and cannot be adjusted. A gradient mapping maps the greatest difference to the maximum thickness and the smallest difference to the minimum thickness, intermediate differences are linearly mapped to intermediate thickness values. Alternatively intervals of differences can be defined, which will then be mapped to different thickness values.

Edge Color Similarly to the way values can be mapped to the background color of a node they can also be mapped to the adjacent edges. On the edge a gradient between the colors will be displayed, which is determined by the adjacent scaffolds.

Label The value associated with a scaffold can be displayed by a label below the node in the scaffold tree.

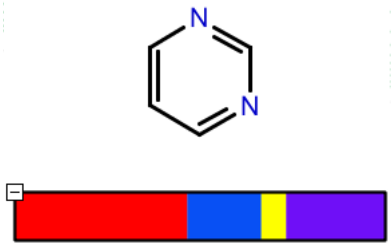


Figure 7.5.: An example of an Info Bar.

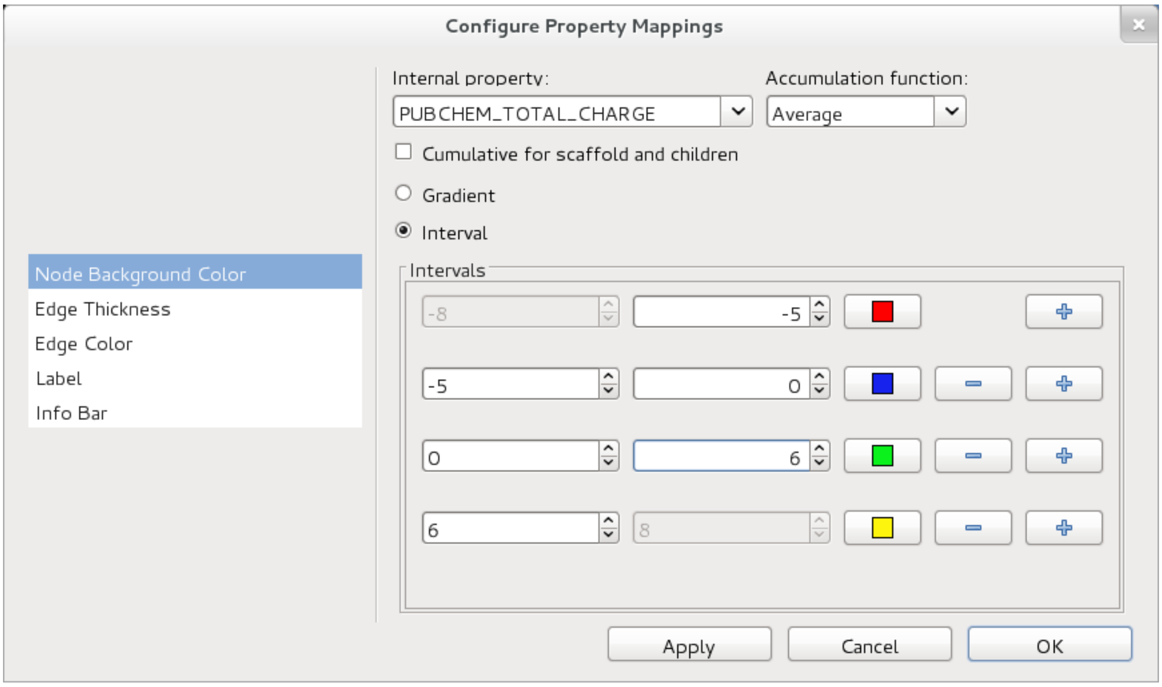
Info Bar This visualizes distributions of a property for the molecules associated with a scaffold. Intervals can be specified and a color is associated with each interval. The distribution of these intervals can then be visualized as shown in *Fig. 7.5*. Textual properties with ten different values or less can also be mapped to colors to display such a distribution.

The dialog to configure these mappings is shown in *Fig. 7.6*. On the left the visual property can be selected, on the right settings for the selected visual property can be configured. These settings may vary depending on the visual property. In general the scaffold property to be mapped can be selected using the box at the top, in case of *derived properties* a second box will be displayed to select the accumulation function. If the box “Cumulative for scaffolds and children” below is selected the accumulation function will not only be applied to values of the structures associated with the scaffold, but also include the values of all structures, which belong to scaffolds located in the subtree below the scaffold.

7.1.7. Exporting Images

Clicking the “Export Image” button in the toolbar or the “Tree” menu will open the export dialog shown in *Fig. 7.7*, which allows you to save the current graph as an image. You can either export the current view on the graph, which will save only the part of the graph which is currently visible in the graph pane, or the whole graph. There are three file formats available (SVG, PNG and TIFF) and options to control the image size. By default the aspect ratio of the exported image is fixed, this can be changed by clicking on the lock.

Since SVG is a file format for vector graphics the exported image can be scaled without quality loss. Consequently the specified image size will not have much effect on the quality or file size for SVGs but is very important for raster formats such as PNG and TIFF. Even if the current view shows rectangles for scaffolds the exported image will always show their structural formulas.



Internal property: PUBCHEM_TOTAL_CHARGE

Accumulation function: Average

☐ Cumulative for scaffold and children

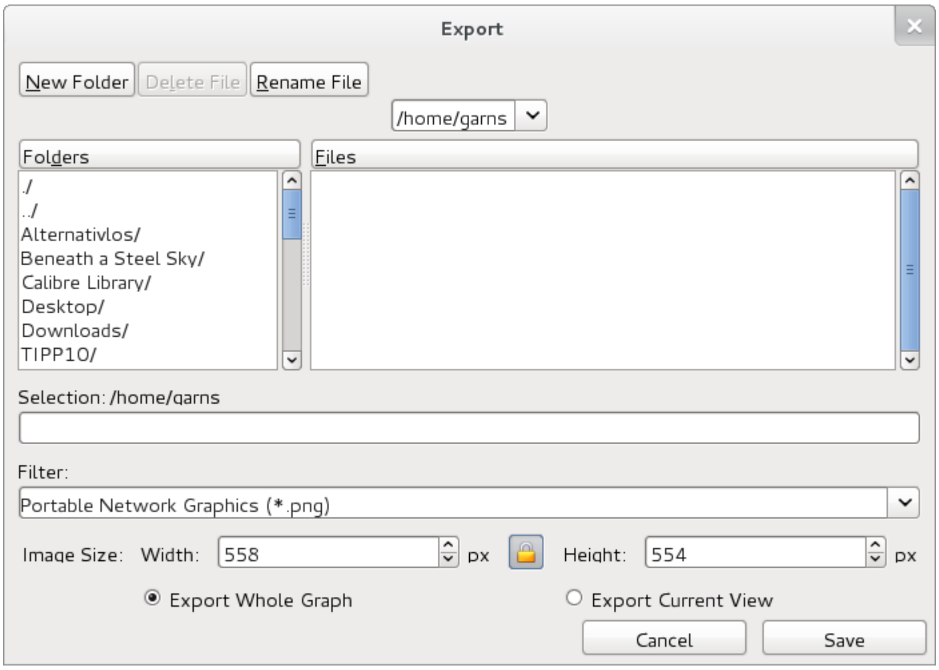
○ Gradient

● Interval

Intervals

-8	-5	🟥
-5	0	🟦
0	6	🟩
6	8	🟨

Figure 7.6.: The property mapping dialog showing an interval mapping to the Node Background Color.



/home/garns

./
../
Alternativlos/
Beneath a Steel Sky/
Calibre Library/
Desktop/
Downloads/
TIPP10/

Selection: /home/garns

Filter: Portable Network Graphics (*.png)

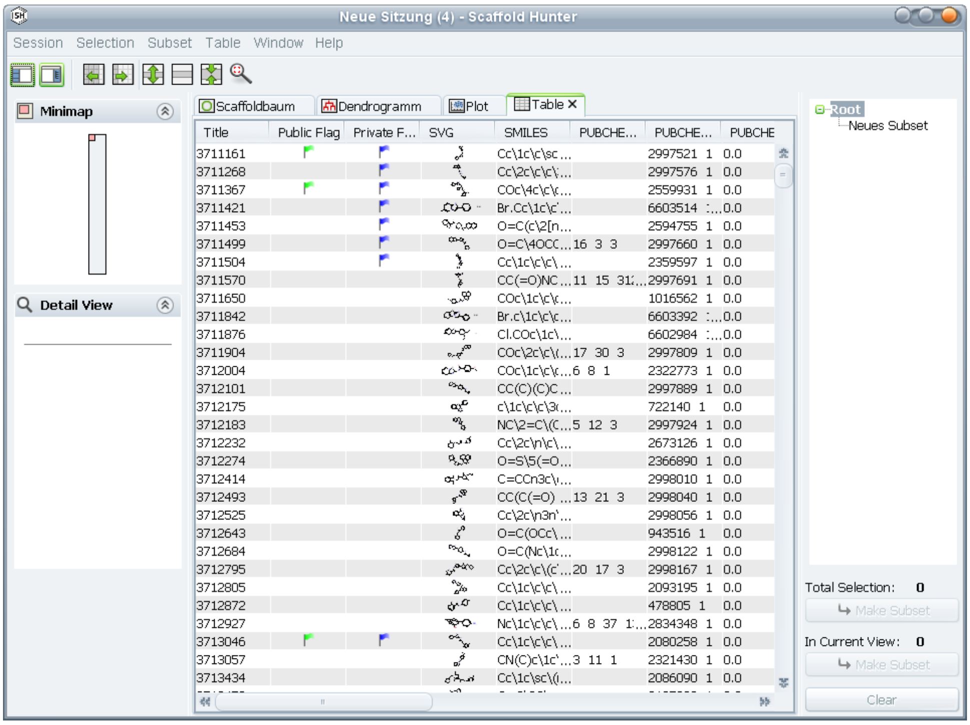
Image Size: Width: 558 px Height: 554 px

● Export Whole Graph ○ Export Current View

Figure 7.7.: The export dialog.

7.2. Table

An overview of the aggregated molecule information can be seen in the table view. All the molecule properties, as well as their titles, SVG-images and the flags set by users, are shown in form of a table.



Title	Public Flag	Private F...	SVG	SMILES	PUBCHE...	PUBCHE...	PUBCHE
3711161	⚑	⚑		Cc\1c\c\sc ...		2997521 1	0.0
3711268		⚑		Cc\2c\c\c\...		2997576 1	0.0
3711367	⚑	⚑		COc\4c\c\c...		2559931 1	0.0
3711421		⚑		Br.Cc\1c\c'...		6603514 :...	0.0
3711453		⚑		O=C(c\2[n...		2594755 1	0.0
3711499		⚑		O=C\4OCC...	16 3 3	2997660 1	0.0
3711504		⚑		Cc\1c\c\c\ ...		2359597 1	0.0
3711570				CC(=O)NC ...	11 15 31:...	2997691 1	0.0
3711650				COc\1c\c\c...		1016562 1	0.0
3711842				Br.c\1c\c\c...		6603392 :...	0.0
3711876				Cl.COc\1c\...		6602984 :...	0.0
3711904				COc\2c\c\(...	17 30 3	2997809 1	0.0
3712004				COc\1c\c\c...	6 8 1	2322773 1	0.0
3712101				CC(C)(C)C ...		2997889 1	0.0
3712175				c\1c\c\c\3c...		722140 1	0.0
3712183				NC\2=C\(C...	5 12 3	2997924 1	0.0
3712232				Cc\2c\n\c\...		2673126 1	0.0
3712274				O=S\5(=O...		2366890 1	0.0
3712414				C=CCn3c\...		2998010 1	0.0
3712493				CC(C(=O) ...	13 21 3	2998040 1	0.0
3712525				Cc\2c\n3n'...		2998056 1	0.0
3712643				O=C(OCc\ ...		943516 1	0.0
3712684				O=C(Nc\1c...		2998122 1	0.0
3712795				Cc\2c\c\(c'...	20 17 3	2998167 1	0.0
3712805				Cc\1c\c\c\ ...		2093195 1	0.0
3712872				Cc\1c\c\c\ ...		478805 1	0.0
3712927				Nc\1c\c\c\ ...	6 8 37 1:...	2834348 1	0.0
3713046	⚑	⚑		Cc\1c\c\c\ ...		2080258 1	0.0
3713057				CN(C)c\1c'...	3 11 1	2321430 1	0.0
3713434				Cc\1c\sc\(...		2086090 1	0.0

Figure 7.8.: The tableview

7.2.1. Sorting and Ordering the Table

When you open a table view the data is initially sorted by the molecule title, but the sort order can be changed anytime by clicking on a column header. The table view supports three sort criteria, which are applied by consecutive clicks on column headers. The column selected last gets the highest sort priority.

Apart from sorting the rows of the table you can reorder the columns at any time by dragging them to their new position.

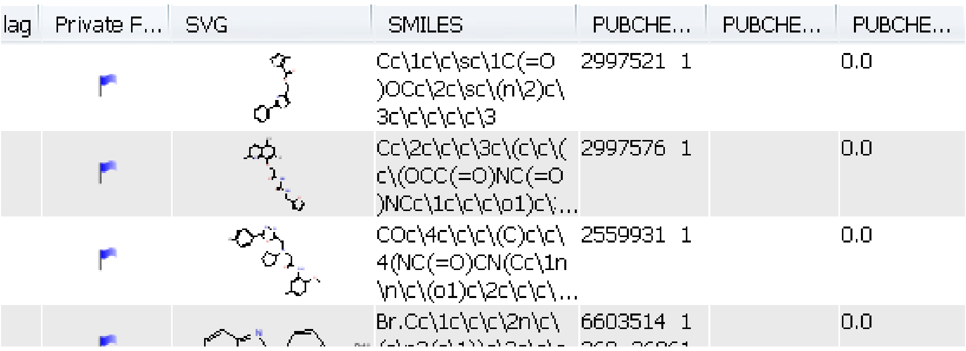
7.2.2. Resizing Table Cells

A table cell is often too small to show the information that it holds. In this case three little dots appear at the end of the cell, to indicate that the content is not completely shown. To address this problem there are two ways to resize table cells: To change the width of a column you can simply click on the right boundary line of the column header and move it to the left or right, to make the whole column smaller or wider. The height of the rows can be changed via three buttons in the toolbar:

- ⇕ the *increase* button allows you to increase the number of text lines that are shown in a table cell. Each time it is clicked one more line is added.

- the *decrease* button decreases the number of text lines per cell.
- the *normalize* button sets the number of text lines back to the default value of one.

The content of a table cell is also shown in the *Detail View* panel in the sidebar, as soon as you move the mouse pointer to a cell.



lag	Private F...	SVG	SMILES	PUBCHE...	PUBCHE...	PUBCHE...
	⚑		Cc\1c\c\sc\1C(=O)OCc\2c\sc\(n\2)c\ 3c\c\c\c\c\3	2997521 1		0.0
	⚑		Cc\2c\c\c\3c\(c\c\(c\(OCC(=O)NC(=O)NCc\1c\c\c\o1)c\...	2997576 1		0.0
	⚑		COc\4c\c\c\(C)c\c\ 4(NC(=O)CN(Cc\1n \n\c\(o1)c\2c\c\c\...	2559931 1		0.0
	⚑		Br.Cc\1c\c\c\2n\c\	6603514 1		0.0

Figure 7.9.: Example of a table that shows three text lines in a table cell

7.2.3. The Minimap

The Minimap shows an abstract view of the complete table, where the section that is currently shown in the main view is highlighted in light red. Apart from just informative reasons the minimap can also be used to scroll the table. Just click on the highlighted box and drag it to another position.

7.2.4. Sticky Columns

When the table holds a lot of columns – which is a very common case – then it is eligible that some columns are not scrolled horizontally but stay visible the whole time. The molecule title column is an example; it may be helpful if this column is always visible. This can be accomplished by putting table columns into the *sticky mode*: A sticky column does not scroll horizontally.

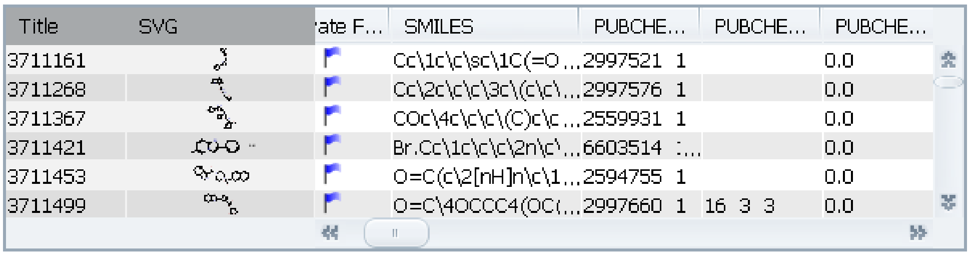
This feature can be accessed by two buttons in the toolbar:

- A click on this button turns the leftmost floating column (a column which is not yet sticky) into the sticky mode. It will not scroll horizontally anymore.
- This button turns the rightmost sticky column back into floating mode, which means that the column will scroll as usual.

To indicate that a column is sticky it is shown a bit darker than a normal, floating column.

7.2.5. Setting Flags

Flags (public and private) can be set by using the context menu that appears when you right-click anywhere in the table. The menu item “Toggle public flag (for *molecule title*)” lets you toggle the public flag for the specified molecule. The item “Toggle private flag (for *molecule title*)” does the same for private flags. The flags are shown in the table in separate columns.



Title	SVG	ate F...	SMILES	PUBCHE...	PUBCHE...	PUBCHE...
3711161		⚑	Cc\1c\c\sc\1C(=O ...	2997521 1		0.0
3711268		⚑	Cc\2c\c\c\3c\(c\c\...	2997576 1		0.0
3711367		⚑	COc\4c\c\c\(C)c\c ...	2559931 1		0.0
3711421		⚑	Br.Cc\1c\c\c\2n\c\...	6603514 :...		0.0
3711453		⚑	O=C(c\2[nH]n\c\1...	2594755 1		0.0
3711499		⚑	O=C\4OCCC4(OC(...	2997660 1	16 3 3	0.0

Figure 7.10.: A table with two sticky columns. Note that the scrollbar at the bottom does not include the sticky columns.

7.2.6. Selecting Molecules

The selection of molecules (a row in the table) works slightly different than with common tables. The only change we made is that the current selection will not be cleared when you select a new molecule by clicking on the corresponding row in the table. So you can select/unselect multiple molecules by consecutive clicking on them.

Additionally you can select many molecules in a row when you hold down the `Shift` key while clicking on the first and the last molecule you want to select (which is the common behavior that you may know from other programs).

7.2.7. Writing and Editing Comments

The comment editor for molecules can be opened by using the context menu.

7.2.8. Placeholders in Table Cells

The table can be used to access all molecule information which is stored in the database. To conserve memory these information are loaded just when they should be shown. This may cause a little delay in displaying the content of the table, which can be noticed at fast scrolling over a long distance. During the loading process the table cells are filled with three dots ("...") as placeholders. These placeholders are replaced by the real content as soon as it is available.

7.3. Dendrogram

The dendrogram view shows the result of a hierarchical clustering. A hierarchical clustering works on a (sub)set of molecules and a couple of properties. In the beginning, the clustering algorithm puts each molecule into one cluster. Then it searches for two molecules (clusters) with the smallest distance between each other based on the chosen properties. When found, they are merged into a new cluster. This will be repeated until only one cluster remains. The result is a tree which represents the merge history like shown in Fig. 7.11.

The dendrogram view shows this result, allows the user to navigate in the hierarchical tree structure and to highlight different clusters based on their relation.

7.3.1. The View in Detail

The dendrogram view is separated in four parts, shown in Fig. 7.12. Each part is marked by a surrounding color as explained below:

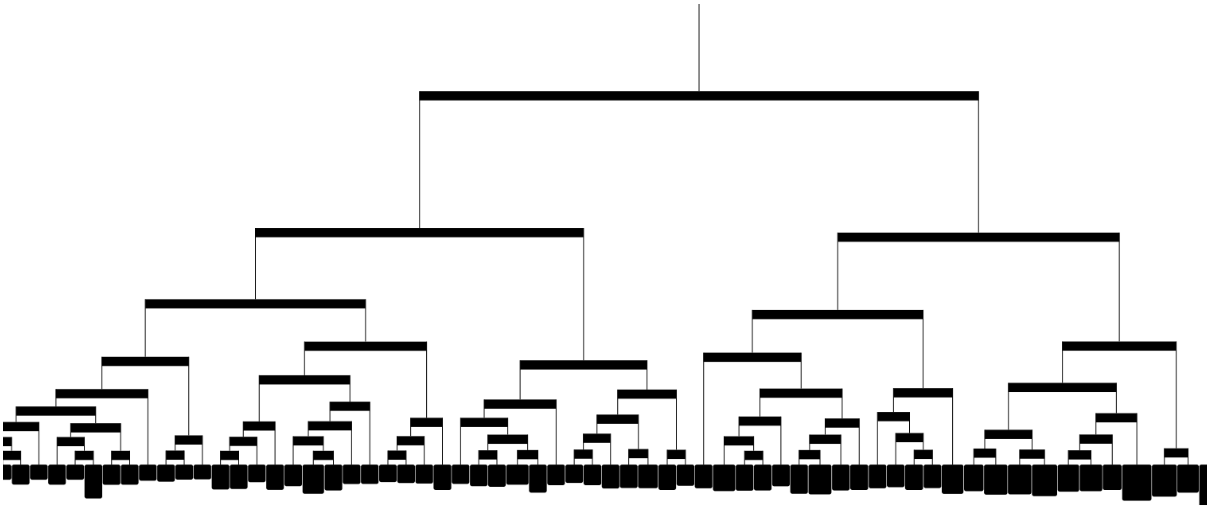
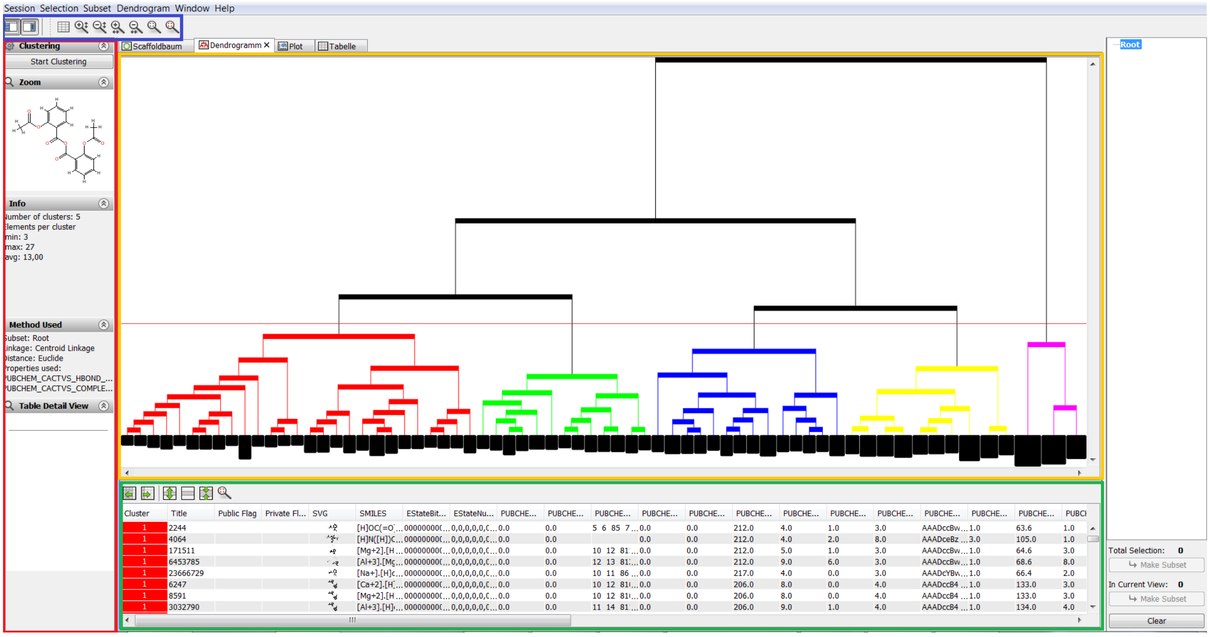


Figure 7.11.: The tree frame

- the tree frame in which the result is shown (marked yellow)
- the sidebar (marked red)
- the toolbar (marked blue)
- a special representation of the table view (marked green)



umber of clusters: 5
lements per cluster
min: 3
max: 27
avg: 13,00

ubset: Root
inkage: Centroid Linkage
istance: Euclide
roperties used:
UBCHEM_CACTVS_HBOND_...
UBCHEM_CACTVS_COMPLE...

Total Selection: 0
In Current View: 0

Cluster	Title	Public Flag	Private Fl...	SVG	SMILES	EStateBit...	EStateNu...	PUBCHE...	PUBCHE...	PUBCHE...	PUBCHE...	PUBCHE...	PUBCHE...	PUBCHE...	PUBCHE...	PUBCHE...	PUBCHE...	PUBCHE...	PUBCHE...	PUBCI
1	2244				[H]OC(=O'...	00000000(...	0,0,0,0,0,(...	0.0	0.0	5 6 85 7...	0.0	0.0	212.0	4.0	1.0	3.0	AAADccBw...	1.0	63.6	1.0
1	4064				[H]N([H])C...	00000000(...	0,0,0,0,0,(...	0.0	0.0		0.0	0.0	212.0	4.0	2.0	8.0	AAADceBz ...	3.0	105.0	1.0
1	171511				[Mg+2].[H ...	00000000(...	0,0,0,0,0,(...	0.0	0.0	10 12 81...	0.0	0.0	212.0	5.0	1.0	3.0	AAADccBw...	1.0	64.6	3.0
1	6453785				[Al+3].[Mg...	00000000(...	0,0,0,0,0,(...	0.0	0.0	12 13 81...	0.0	0.0	212.0	9.0	6.0	3.0	AAADccBw...	1.0	68.6	8.0
1	23666729				[Na+].[H]c...	00000000(...	0,0,0,0,0,(...	0.0	0.0	10 11 86 ...	0.0	0.0	217.0	4.0	0.0	3.0	AAADcYBw...	1.0	66.4	2.0
1	6247				[Ca+2].[H]...	00000000(...	0,0,0,0,0,(...	0.0	0.0	10 12 81...	0.0	0.0	206.0	8.0	0.0	4.0	AAADccB4 ...	1.0	133.0	3.0
1	8591				[Mg+2].[H ...	00000000(...	0,0,0,0,0,(...	0.0	0.0	10 12 81...	0.0	0.0	206.0	8.0	0.0	4.0	AAADccB4 ...	1.0	133.0	3.0
1	3032790				[Al+3].[H]...	00000000(...	0,0,0,0,0,(...	0.0	0.0	11 14 81...	0.0	0.0	206.0	9.0	1.0	4.0	AAADccB4 ...	1.0	134.0	4.0

Figure 7.12.: The whole view

7.3.2. Sidebar

Fig. 7.12 shows the dendrogram sidebar (marked red).

- The “Start Clustering” button opens the configuration dialog for the clustering

- The “Zoom” element shows the molecule under the mouse in detail
- The “Info” element shows statistical data about the currently chosen clustering
- The “Method Used” element shows which configuration was used to generate the actual dendrogram
- The “Table Detail” element is equal to its correspondent in the real table view

7.3.3. Tree Frame

In this part of the view you can see an actual dendrogram like shown in Fig. 7.11. The molecules on the bottom are represented by black boxes because the zoom level is too small to show the real images. The cluster selection bar shown in red in the middle is draggable and divides the subtrees below it into different clusters. Each cluster separated this way is painted in a different own color. The Info panel in the sidebar informs about the number of clusters created this way and their size. With the mouse wheel or the keys “+” and “-“ the view can be zoomed in and out. At a closer zoom level the black boxes at the bottom are replaced by the real molecule images like shown in Fig. 7.12 (marked yellow).

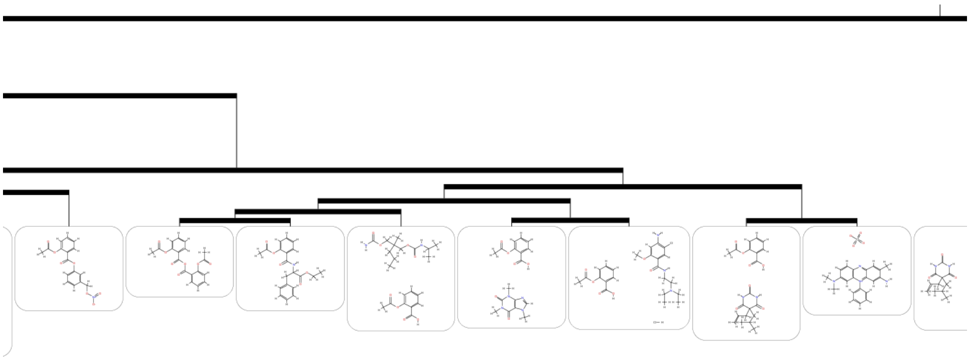


Figure 7.13.: Zoomed in

The zoom is separated into vertical and horizontal zoom. This is necessary, because the tree usually has a very small height and a huge width. So when zooming with the mouse wheel, the view scales the images and than adapts the tree above to fit in the window. Zooming while holding the `Ctrl` key will scale only the tree, not the image. In addition to click on an image to add/remove it from the selection, clicking on an inner node of the tree will result in selecting all leaves below it, if at least one leaf is unselected and deselect them otherwise.

7.3.4. Toolbar

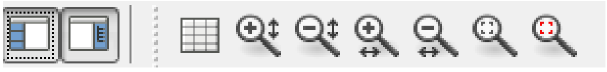


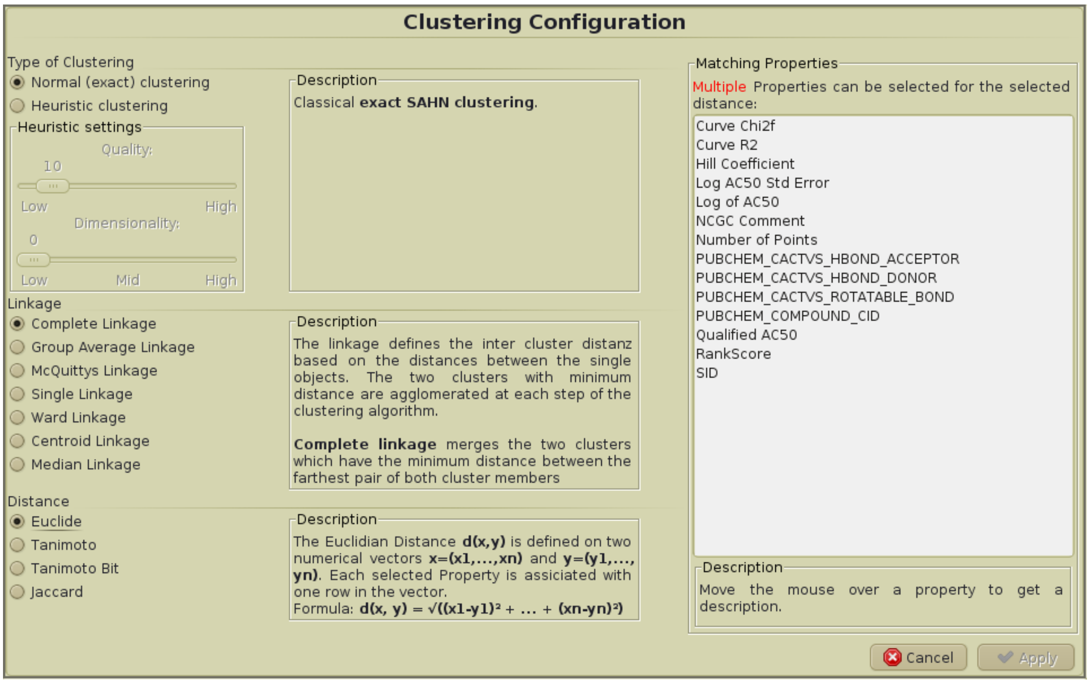
Figure 7.14.: The toolbar

The two buttons on the left shown in Fig. 7.14 are the standard items to show/hide the side- and subset bar. The third button from the left shows/hides a special table below the dendrogram. The next four buttons change the vertical and horizontal zoom of the dendrogram. The second button from the right fits the dendrogram into the available space. The button at the right zooms the view so that the whole selection is visible.

7.3.5. Table

The shown table below the dendrogram is the standard table view with one additional function, see *Sec. 7.2: Table*. The first column shows in which subclusters the selection bar in the dendrogram has divided the molecules. An example is shown in Fig. 7.12 (marked green).

7.3.6. Clustering Configuration Panel



Clustering Configuration

Type of Clustering
● Normal (exact) clustering
○ Heuristic clustering

Heuristic settings
Quality: 10 (Low – High)
Dimensionality: 0 (Low – Mid – High)

Description
Classical **exact SAHN clustering**.

Linkage
● Complete Linkage
○ Group Average Linkage
○ McQuittys Linkage
○ Single Linkage
○ Ward Linkage
○ Centroid Linkage
○ Median Linkage

Description
The linkage defines the inter cluster distanz based on the distances between the single objects. The two clusters with minimum distance are agglomerated at each step of the clustering algorithm.

Complete linkage merges the two clusters which have the minimum distance between the farthest pair of both cluster members

Distance
● Euclide
○ Tanimoto
○ Tanimoto Bit
○ Jaccard

Description
The Euclidian Distance **d(x,y)** is defined on two numerical vectors **x=(x1,...,xn)** and **y=(y1,...,yn)**. Each selected Property is assiciated with one row in the vector.
Formula: **d(x, y) = √((x1-y1)² + ... + (xn-yn)²)**

Matching Properties
Multiple Properties can be selected for the selected distance:
Curve Chi2f
Curve R2
Hill Coefficient
Log AC50 Std Error
Log of AC50
NCGC Comment
Number of Points
PUBCHEM_CACTVS_HBOND_ACCEPTOR
PUBCHEM_CACTVS_HBOND_DONOR
PUBCHEM_CACTVS_ROTATABLE_BOND
PUBCHEM_COMPOUND_CID
Qualified AC50
RankScore
SID

Description
Move the mouse over a property to get a description.

Figure 7.15.: The clustering config dialog

After clicking the start clustering button in the sidebar, a configuration panel as in Fig. 7.15 will show up. Here, the user can choose from two clustering algorithms and select their parameters:

7.3.6.1. Clustering Algorithms

Normal (exact) clustering: Classical exact SAHN clustering.

> WARNING: The exact clustering can work on 2^{16} (65.536) molecules at a max, because of technical restrictions in Java and memory limitation. Some of the parameters (*Euclide* in combination with *Ward, Median* or *Centroid*) may work on more molecules, but this is experimental and should only be used with caution. Please note that these restrictions do not apply to the heuristic algorithm.

Heuristic clustering: The heuristic SAHN clustering algorithm should be used if the runtime or the memory consumption by the exact clustering is to high. Empiric tests showed that this algorithm can be considered useful for subsets with a size larger that 10000 molecules. Two parameters are selectable: quality and dimensionality. Both have a direct influence on quality and speed. The dimensionality should be set according to the dimensionality of the data. Low for a dimensionality between 1 and 7, Mid 7-13 and High > 13. If unsure start with a low dimensionality and use a higher value if the quality is insufficient regardless of which quality is used.

7.3.6.2. Linkage & Distance

The linkage method defines the way in which the distance between clusters is calculated and controls which pair should be merged.

- **Complete Linkage:** Merges the two clusters which have the minimum distance between the farthest pair of both cluster members.
- **Group Average Linkage:** Merges the two clusters with the minimum distance of the unweighted cluster average (UPGMA).
- **McQuittys Linkage**: Merges the two clusters with the minimum distance of the weighted cluster average (WPGMA).
- **Single Linkage:** Merges the two clusters with the minimum distance between the nearest pair of both cluster members.
- **Ward Linkage:** Merges the two clusters that lead to minimal variance increase.
- **Centroid Linkage**: Merges the two clusters with the minimum distance of the unweighted centers (UPGMC).
- **Median Linkage:** Merges the two clusters with the minimum distance of the weighted cluster centers (WPGMC).

The accepted types of properties depend on the distance computation.

- **Euclide:** Any number of numerical properties.
- **Tanimoto:** Exactly one property which has to be a binary fingerprint.
- **Tanimoto Bit:** Same as Tanimoto but works on bit strings not on character strings.
- **Jaccard:** Works on one fingerprint property with feature counts on each position.

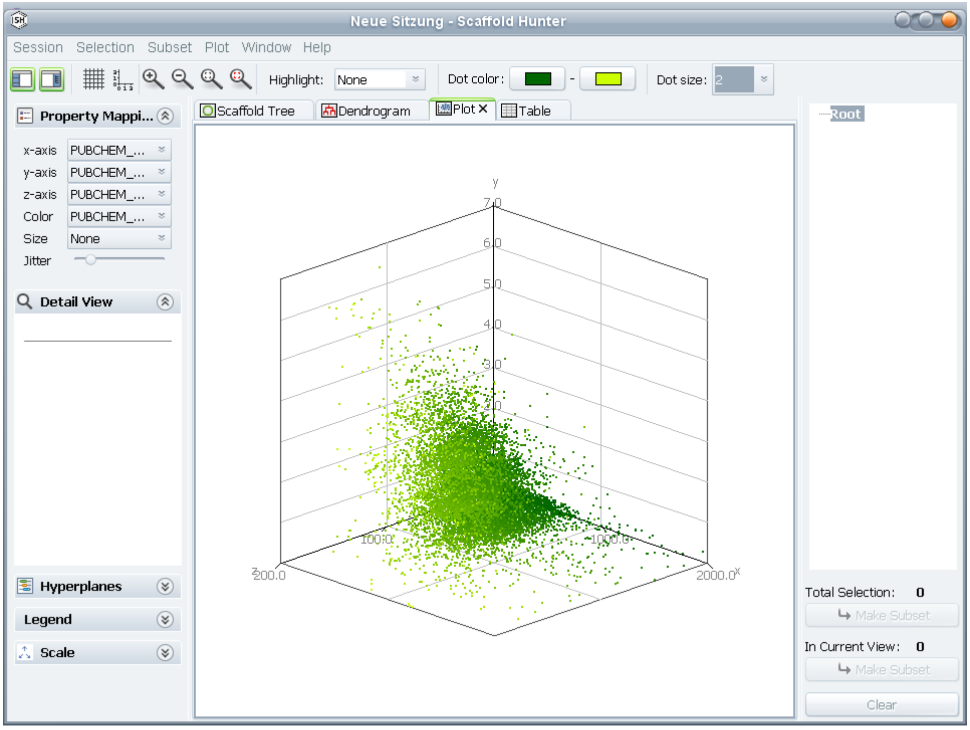
7.4. Plot

The plot view allows you to view numerical molecule data in form of two- or three-dimensional scatter-plots. Any numerical data that is stored in the database can be displayed by linking them either to one of the three spatial axes or to the color and size attributes of a dot in the diagram.

7.4.1. Choosing the Data to be Displayed

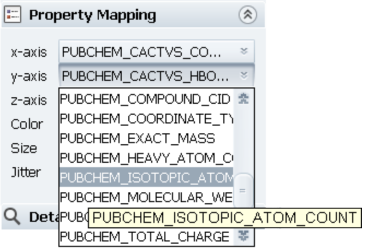
The first thing you want to do after opening a plot view is to select the data that should be displayed. This can be done with the *Property Mapping* panel in the sidebar. Here you find a combo box for each of the three spatial axes, for the dot color and the dot size, which allows you to chose from numerical molecule properties. Additionally there is a slider to apply a jitter to the data on the spatial axes.

Please note that the data has to be loaded from the database when the mapping is changed. This can cause a slight delay before the diagram is displayed. The mapping of data to axes is also shown in the *Legend panel* of the sidebar. There you can also see the domain of each axis.



Highlight: None
Dot color: -
Dot size: 2
x-axis PUBCHEM_...
y-axis PUBCHEM_...
z-axis PUBCHEM_...
Color PUBCHEM_...
Size None
Jitter
Total Selection: 0
In Current View: 0

Figure 7.16.: The plot view



x-axis PUBCHEM_CACTVS_CO...
y-axis PUBCHEM_CACTVS_HBO...

Figure 7.17.: The Property Mapping panel

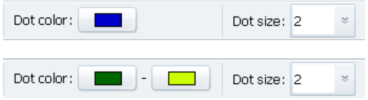
7.4.2. Dot Color and Size

The default color and size for dots can be changed at the toolbar. These values are used when no data is mapped to the color/size properties. As soon as you change the property mapping the elements in the toolbar change to let you enter an interval for the color/size of the dots.

7.4.3. Hyperplanes

Sometimes you do not want to see the complete set of data, but a small part of it. This is where the concept of hyperplanes becomes useful. Hyperplanes allow you to define thresholds for an axis, that means a minimum and a maximum value that a data must have to be displayed as dot in the diagram. They can be applied to any of the 5 dimensions (x, y, z, color and size). When hyperplanes are set the associated threshold values are shown in the diagram as black lines.

Hyperplanes can be set and adjusted using the Hyperplanes panel in the sidebar. For each of the 5 possible dimensions there is a slider that lets you change the upper and the lower threshold value. For a better control the values can also be entered as numbers in the text fields.



Dot color: 🟦 Dot size: 2

Dot color: 🟩 - 🟨 Dot size: 2

Figure 7.18.: The button for the color and the combo box for the dot size. The upper figure shows the default behavior, when no data is mapped. The lower figure shows the behavior when data is mapped to color/size.

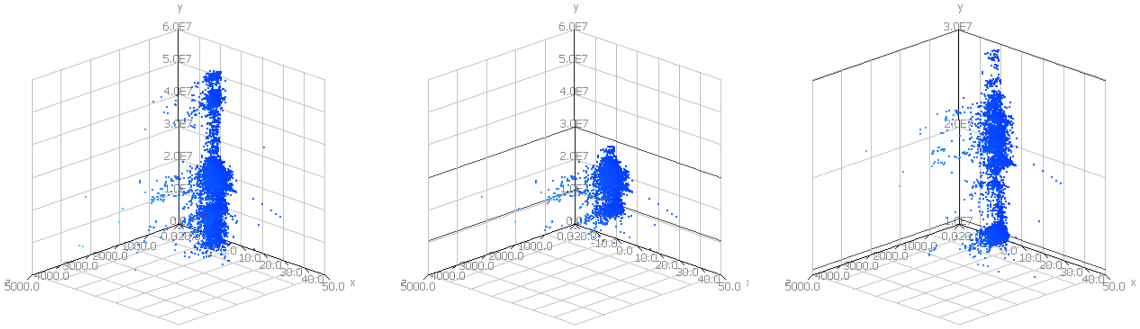
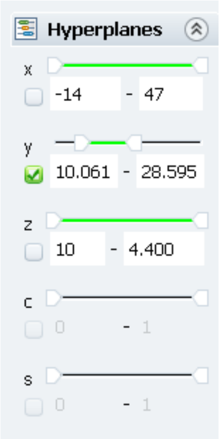


Figure 7.19.: The use of hyperplanes in a diagram: Without hyperplanes (left figure), with hyperplanes at the y-axis (middle) and with the y-axis adjusting to the hyperplanes (right figure).



Hyperplanes

x
☐ -14 - 47

y
☑ 10.061 - 28.595

z
☐ 10 - 4.400

c
☐ 0 - 1

s
☐ 0 - 1

Figure 7.20.: The hyperplane panel

While the axes always try to adjust themselves to the domain that is mapped this behavior may be inappropriate when using hyperplanes. By default a hyperplane does not influence the domain of an axis, it just defines which values should be shown and which not. To let an axis adjust according to the visible range that is defined by the hyperplanes, click the checkbox for the axis in the panel.

7.4.4. Zooming, Scrolling and Rotating the Diagram

The diagram can be zoomed by using the buttons in the toolbar or by using the mouse wheel. When using the mouse wheel the diagram will be zoomed with keeping the focus at the current position of the mouse pointer. Furthermore each axis can be scaled independently by using the buttons in the Scale panel of the sidebar. In case the diagram becomes too large to fit in the window you can scroll around by clicking in the diagram with the left mouse button and drag it around. Rotating the diagram works similar, just press the right mouse button while dragging. Rotation only works in 3D-mode.

The *Fit Graph* button in the toolbar always resets the zoom, scroll position and rotation angles. Zooming and scrolling the diagram to show the current selection can be done by clicking the button “Zoom to current selection” in the toolbar.

7.4.5. Highlighting, Selecting and Picking

To avoid confusion with the dot colors the current selection and flags (public and private) are not shown in the diagram by default. To make them visible you have to set the accordant highlighting mode, which can be done with the combo box in the toolbar. When the highlighting mode is set to “Selection” then the dots that represent the molecules in the current selection are shown in red. Setting the highlighting mode to “Public flag” shows the molecules with a public flag in green, while the highlighting mode “Private flag” shows the molecules with a private flag in blue.

These features can also be set and removed from a molecule in the plot view, according to Table 7.1.

Effect	**Action**
add a molecule to or remove it from the selection	place the mouse pointer over the dot and click the left mouse button
add multiple molecules to the selection	hold the `Shift` key and the left mouse button down while drawing a box around the dots
remove multiple molecules from the selection	hold the `Ctrl` key and the left mouse button down while drawing a box around the dots
toggle the public flag of a molecule	place the mouse pointer over the dot and click the middle mouse button
toggle the private flag of a molecule	place the mouse pointer over the dot and click the middle mouse button while pressing the `Shift` key

Table 7.1.: Plot view Mouse Actions

Please note that the highlighting mode adjusts automatically to the action that is performed.

In any case, no matter if you perform one of these actions or not, the molecule that belongs to the dot under the mouse pointer is shown in the *Detail View* panel in the sidebar.

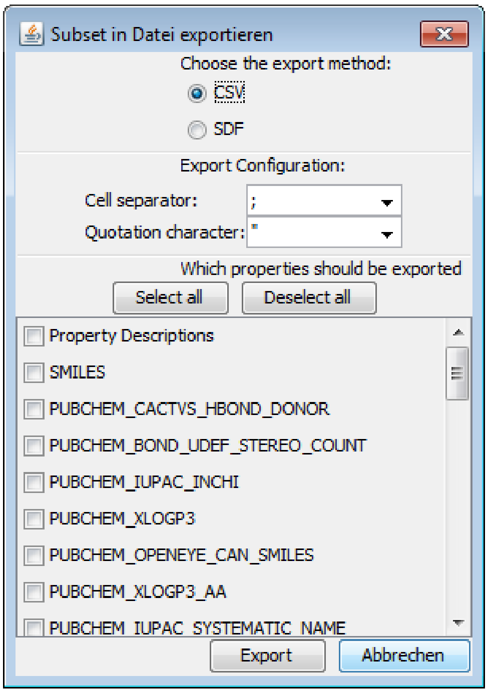
7.4.6. Hiding the Axis Ticks and Grids

To toggle display of the axis ticks and grids there are two buttons in the toolbar:

- this button toggles the display of the ticks
- this button toggles the display of the grids

8. Export

Right clicking on a subset offers the possibility to export this subset. At the moment CSV and SDF export is supported. Starting the export will open the dialog shown in Fig. 8.1.



Subset in Datei exportieren

Choose the export method:

● CSV

○ SDF

Export Configuration:

Cell separator: ;

Quotation character: "

Which properties should be exported

☐ Property Descriptions
☐ SMILES
☐ PUBCHEM_CACTVS_HBOND_DONOR
☐ PUBCHEM_BOND_UDEF_STEREO_COUNT
☐ PUBCHEM_IUPAC_INCHI
☐ PUBCHEM_XLOGP3
☐ PUBCHEM_OPENEYE_CAN_SMILES
☐ PUBCHEM_XLOGP3_AA
☐ PUBCHEM_IUPAC_SYSTEMATIC_NAME

Figure 8.1.: Export Dialog

If CSV is chosen, the cell separator and the quotation character can be defined. After choosing the export format, the collection of properties which should be exported can be defined. Clicking the export button will open a file chooser dialog to define the location and the name of the export.

After completing the export, the program will show a success message.

Bibliography

[1] Bernhard Dick, Thorsten Flügel, Henning Garus, Michael Hesse, Philipp Kopp, Philipp Lewe, Dominic Sacré, Till Schäfer, and Thomas Schmitz. Endbericht PG 552 - Drug Hunting. Technical report, 2011.

[2] Adalbert Gorecki, Anke Arndt, Arbia Ben Ahmed, Andre Wiesniewski, Cengizhan Yücel, Gebhard Schrader, Henning Wagner, Michael Rex, Nils Kriege, Philipp Büderbender, Sergej Rakov, and Vanessa Bembenek. Endbericht PG 504 - ChemBioSpaceExplorer. Technical report, 2007.

[3] Marcus A. Koch, Ansgar Schuffenhauer, Michael Scheck, Stefan Wetzel, Marco Casaulta, Alex Odermatt, Peter Ertl, and Herbert Waldmann. Charting biologically relevant chemical space: a structural classification of natural products (sconp). *Proceedings of the National Academy of Sciences of the United States of America*, 102(48):17272–17277, November 2005.

[4] Ansgar Schuffenhauer, Peter Ertl, Silvio Roggo, Stefan Wetzel, Marcus A. Koch, and Herbert Waldmann. The scaffold tree - visualization of the scaffold universe by hierarchical scaffold classification. *Journal of Chemical Information and Modeling*, 47(1):47–58, 2007.

A. Keyboard Shortcuts

	Key	Action
Zoom	+/-	Zoom in/out
	0	Fit graph / Table: Normalize rows
	s	Fit selection / Table: Scroll to first selected molecule
Selection	Ctrl+A	Select all molecules/scaffolds
	Ctrl+Shift+A	Deselect all molecules/scaffolds
	Ctrl+I	Invert selection
	Ctrl+D	Deselect all molecules in the current views subset
	Ctrl+N	Make subset from selected molecules
	Ctrl+Shift+N	Make subset from selected molecules in current view
View	Ctrl+W	Close current view
	Ctrl+PageUp	Show next view
	Ctrl+PageDown	Show previous view
	ALT+←	Toggle display of sidebar
	ALT+→	Toggle display of subset

Table A.1.: Global shortcuts (view-independent)

	Mouse	Action
Selection	Left-click on molecule/scaffold	Toggle selection of molecule/scaffold
	Shift + Left-click + Drag	Select all molecules/scaffolds in selection rectangle
	Ctrl + Left-click + Drag	Deselect all molecules/scaffolds in selection rectangle
	Left-click on tree node	Toggle selection of subtree (dendrogram view only)
Flags	Middle-click on molecule/scaffold	Add/remove public flag for molecule/scaffold
	Shift + Middle-click on molecule/scaffold	Add/remove private flag for molecule/scaffold
Zoom	WheelUp / WheelDown	Zoom in/out
	Ctrl + WheelUp / WheelDown	Zoom in/out vertically (dendrogram view only)
Navigation	Left-click + Drag	Move view area (table view only)
	Right-click + Drag	Rotate 3D-graph (plot view only)
	Right-click on molecule/scaffold	Open context menu (all views; except plot view)

Table A.2.: Mouse actions (global and view-specific)

	Key	Action
Navigation	→	Move cursor clockwise
	←	Move cursor counter-clockwise
	↑	Move cursor to first child
	↓	Move cursor to parent
	C	Focus cursor
Radii	Ctrl+↑	Increase radii
	Ctrl+↓	Decrease radii
	F	Toggle radii lock
Expanding	Enter	Open/close children
	Ctrl+Enter	Open/close entire subtree
	E	Expand next level/Reduce to level
	Ctrl+E	Expand all nodes
	Ctrl+Shift+E	Reset to default expand level
Visual	Space	Select/deselect scaffold
	M (pressed)	Show molecules
	Ctrl+M	Toggle permanent display of molecules
	Ctrl+P	Configure property mappings...
Node scaling	PageUp	Scale up cursor scaffold
	PageDown	Scale down cursor scaffold
	Alt+PageUp	Scale up selected scaffolds
	Alt+PageDown	Scale down selected scaffolds
	N	Normalize cursor scaffold
	Alt+N	Normalize selected scaffolds
	A	Normalize all nodes
Layout	L	Switch to linear layout
	B	Switch to balloon layout
	R	Switch to radial layout

Table A.3.: Scaffold tree view shortcuts

Key	Action
Ctrl+G	Toggle display of grid
Ctrl+T	Toggle display of ticks

Table A.4.: Plot view shortcuts

Key	Action
Ctrl+T	Toggle display of table
Ctrl + Plus	Zoom in vertically
Ctrl + Minus	Zoom out horizontally

Table A.5.: Dendrogram view shortcuts

B. How to Write Plugins

In this chapter you will learn the basic parts needed to write new plugins. Feel free to write new import or calc plugins or to expand the existing ones. We are always happy to get new plugin (versions) submitted.

B.1. Import Plugins

B.1.1. What is needed to write a new import plugin

To write a new import plugin you need the Scaffold Hunter source code. The plugins basis is held in the `edu.udo.scaffoldhunter.plugins.dataimport`. You should put every new plugin inside the `edu.udo.scaffoldhunter.plugins.dataimport.impl.PLUGINNAME` package.

B.1.2. Basic parts of a plugin

Every import plugin consists of three base classes:

PluginSettingsPanel At first a class on base of `PluginSettingsPanel`. This is a `JPanel` which shows the configuration options of the plugin in the Import Dialog. It has methods to get and set the current configuration options and those which have been set in the past.

PluginResults The `PluginResults` are built on base of a plugin run and contain all results of the specific plugin with a given configuration. At first there are some basic results, such as the number of results, name of the rows in resulting data, which of those rows maybe numeric, and finally iterables consisting of the molecules.

ImportPlugin The import plugin itself is the interface to Scaffold Hunter. It gives instances of the `PluginSettingsPanel` and the `PluginResults` back, has a method to test if a configuration will give useable results or fail in the beginning and has some basic information, as name and an UID.

Next to those classes there are the `Arguments`, which is a simple `Object` consisting of a current configuration and a class implementing the `Serializeable` interface which contains data, which is saved into the database. It could be used to save older settings.

B.1.3. Writing a simple plugin

The source tree already consists of a very simple plugin, it is named `DummyImportPlugin`. In this part you will get a step by step guide whose result will be such a simple plugin, which can generate a simple error message and gives two empty molecules back.

B.1.4. First Version - An empty configuration panel

The source of the first version can be found in the `edu.udo.scaffoldhunter.plugins.dataimport.impl.example1` package. This example contains everything that is needed to be listed in the Import Dialog. With this example you are not able to go further through the import process, it does not give back all needed parts. Lets look at the important parts of the source.

B.1.4.1. ImportPlugin.java

@PluginImplementation The first important thing is the Annotation `@PluginImplementation`. This is needed by the used plugin framework to recognize this class as a plugin. If this line is missing, the plugin will not be listed in the import dialog.

extends AbstractImportPlugin Your import plugin has to inherit from `AbstractImportPlugin`. If instead of this you only go and implement the `ImportPlugin` interface there will be a wrong name in the list of import sources in the import dialog.

getTitle() The `getTitle()` method returns a short name of the plugin. This is also the name which is listed in the import dialog.

getID() In `getID()` your plugin should return a unique name of the plugin, this is for example used by Scaffold Hunter to match saved properties for the plugins. During the development process of Scaffold Hunter all Plugins used the form `CLASSNAME_VERSION` that should be unique enough.

getDescription() The `getDescription()` method returns a description about for what the import plugin can be used, like "This is an import plugin which can be used to import data from our internal webfrontend".

getSettingsPanel(settings,arguments) Here we return only an empty `SettingsPanel`. This will change in the third example.

getResults(arguments) In the first example we don't have a result object yet.

checkArguments(arguments) Inside of the `checkArguments(arguments)` method you should check if a plugin run will definitely fail, then you return an error message, otherwise null. In this example we return a message to interrupt the import process. Otherwise the Example1 plugin would cause errors in the future.

B.1.5. Second Version - We give a simple result

In the second step we will add an `ImportPluginResult` object, which will give back one molecule without a structure but with two properties (one will be numerical).

B.1.5.1. Example2ImportPluginResults.java

As a new part in the second Example we have a `PluginResults` class. This class has to implement the `PluginResults` interface. Let us go through the new methods.

getSourceProperties() In the `getSourceProperties()` method you have to give back a Map of `PropertyNames`. The Map can also include `PropertyDefinitions`, which could build the base for a more detailed way of defining the type of the property, which should be implemented in further versions of Scaffold Hunter, most times a null-value for the `PropertyDefinition` part is sufficient. So we generate a new Set with the two property names "title" and "number".

getProbablyNumeric() The `getProbablyNumeric()` method gives back a Set of strings containing the property names of those properties which contain numeric values. It is used in the mapping dialog to automatically select which properties should be treated as numbers. In this example it is the property "number".

getMolecules() This method contains the main plugin task. It returns the `Iterable` which contains the new molecules. Here we create a No Notifying Molecule (`NNMolecule`). The usage of `NNMolecule` in place of `Molecule` has a very high positive impact on the import speed. We add our two properties to the molecule and put it into a simple List which we return. When you write your own plugin you will probably write your own class implementing the `Iterable` interface.

getNumMolecules() The `getNumMolecules()` method returns the number of molecules which will be imported. We built one `Molecule` so we return 1.

addMessageListener(listener), removeMessageListener(listener) The `add/removeMessageListener` methods will be used to give fault messages during the import process, so we just ignore them now.

B.1.5.2. Example2ImportPlugin.java

In the plugin itself there are only small changes.

getResults(arguments) The created `PluginResults` implementation is returned.

checkArguments(arguments) We return something, so do not generate an error message and return null.

B.1.6. Third Version - Settings panel

The third example adds a very simple Settings Panel to the plugin where we can type in the molecule title and an error message for the `checkArguments(arguments)` method.

B.1.6.1. Example3PluginArguments.java

At first there is a new Class, `Example3PluginArguments`, which holds the arguments for a single plugin run. This is a very simple class with three fields:

- `boolean` error : Will be true, if the plugin should generate an error message in the `checkArguments` method.
- `String` errorMessage : The message which will be given back if the plugin generates an error message.
- `String` moleculeTitle : The content of the title property in the molecule.

B.1.6.2. Example3PluginSettingsPanel.java

The second new class is the `Example3PluginConfigurationPanel`. So we are able to fill the configuration panel within the import dialog with content.

Example3PluginSettingsPanel(arguments) The constructor first checks if the ConfigurationPanel got an arguments object, this happens when you select an item in the import jobs list of the import dialog. If it does not get an `Example3PluginArguments` object it generates a new one with the default values. Afterwards the different parts of the Panel are generated, with the values from the arguments object and then some formatting is done.

getSettings() We do not safe any settings, so this method returns null. If you want to have saveable settings generate a new class that implements `Serializeable` which holds those settings. An instance of this class with the content which should be saved has to be returned here.

getArguments() The `getArguments()` method returns the current settings being made in a `Example3PluginArguments` object. So here we read the `JCheckBox` state, and the content of the two `JTextField` instances and put them to the corresponding fields in the returned object.

B.1.6.3. Example3ImportPluginResults.java

In the Results class we only had to add the arguments and built the molecule title property on it. So first there is a new private field `Example3ImportPluginArguments` which holds the arguments for the plugin run.

Example3ImportPluginResults(arguments) The new constructor just sets the internal arguments field to the supplied arguments. When you write a plugin you should put some initialization here, like opening database connections, counting of molecules, etc.

getMolecules() In the `getMolecules()` method we set the title property according to the `moleculeTitle` field in the arguments.

B.1.6.4. Example3ImportPlugin.java

getSettingsPanel(settings,arguments) Here our new `SettingsPanel` is returned and we cast the arguments into the right type.

getResults(arguments) Same for the results, they now await arguments, so the class gets them.

checkArguments(arguments) Now we are able to check, if a plugin run will "succeed" so we either give the error message from the arguments if the user wants it or return null.

B.1.7. Fourth and last version - Output a message during the import

At this point you are able to build configurations, check for an error at the beginning of the import and return molecules. There is only one last part missing, the possibility to send Messages during the import. This is realized in the `Example4ImportPlugin`.

B.1.7.1. Example4ImportPluginArguments.java

At first we added a new configuration option to switch the message on or off. It is named `generateMessage`.

B.1.7.2. Example4ImportPluginSettingsPanel.java

In the Settings panel the default value for the `generateMessage` is added. Furthermore there is a `JCheckBox` added, to switch the Message on or off.

B.1.7.3. Example4ImportPluginResults.java

In the `Example4ImportPluginResults` some changes have been made. First there is a LinkedList which holds the listeners which are registered with the results.

addMessageListener(listener) The `addMessageListener(listener)` method now adds the given listener to the `messageListeners` list.

removeMessageListener(listener) The `removeMessageListener(listener)` method now removes the given listener from the `messageListeners` list.

getMolecules() The `getMolecules()` method has been rewritten. Instead of a simple List it now returns a self written `Iterable<Molecule>`, which still gives back our simple Molecule. But in the `getNext()` method of the included `Iterator<Molecule>` the argument generateMessage is checked and if we should send a Message a new instance of the type `edu.udo.scaffoldhunter.model.data.Message` is generated which consists of a Message saying that the Molecule structure on base of a SMILES string could not be generated. You only need to give a type of the message to the constructor of the message, the name can be empty and the other two arguments null, they are set in other parts of the import process. Some MessageTypes are already defined in the `edu.udo.scaffoldhunter.model.dataimport.MergeMessageTypes` class. Two of them should be useful for import plugins:
If you need other MessageTypes just implement them using the MessageType interface.

Name	Description
`MOLECULE_BY_SMILES_FAILED`	Can’t build Molecule on base of SMILES
`MOLECULE_BY_MOL_FAILED`	Can’t build Molecule on base of MOL

B.2. Calculation Plugins

B.2.1. What is needed to write a new calculation plugin

To write a new calc plugin you need the Scaffold Hunter source code. The calc plugin interfaces and helper classes are stored in the following packages:

- `edu.udo.scaffoldhunter.plugins`
- `edu.udo.scaffoldhunter.plugins.datacalculation`
- `edu.udo.scaffoldhunter.model.data`
- `edu.udo.scaffoldhunter.model.datacalculation`

Your plugin should be placed in a new package called `edu.udo.scaffoldhunter.plugins.datacalculation.impl.PLUGINNAME`, where PLUGINNAME should be replaced by the name of your plugin.

B.2.2. Basic parts of a plugin

Every import plugin consists of three base classes:

PluginSettingsPanel At first a class on base of `PluginSettingsPanel` from `edu.udo.scaffoldhunter.plugins`. This is a `JPanel` which shows the configuration options of the plugin in the Calc Dialog (See Fig. 5.12). It has methods to get and set the current configuration options and those which have been set in the past.

CalcPluginResults The `CalcPluginResults` class is constructed during a plugin run and contains all results of the specific plugin with a given configuration. With help of the `CalcPluginResults` class the calc plugin tells the plugin system which properties where calculated and supplies an `Iterable` over all molecules (the calculated properties are attached to those molecules).

CalcPlugin The calc plugin itself is the interface to Scaffold Hunter. It returns instances of the `PluginSettingsPanel` and the `CalcPluginResults`, has a setter method to get notified about existing properties and has getter methods which provide basic information like the title, description and unique identifier of the plugin.

Next to those classes there is a `PluginArguments` class, which is a simple `Object` representing a plugin configuration and a `CalcPluginSettings` class implementing the `Serializable` interface which contains data, that is saved into the database. It could be used by the plugin to save and retrieve settings like e.g. an arguments history.

B.2.3. Writing a simple plugin

In the next sections we will – in a step-by-step guide – develop a plugin that reads an existing numerical property, adds or subtracts 1.0 from the property and saves the new value into a new property. The plugin is configurable and has the ability to send messages to the GUI if something goes wrong.

B.2.4. First version - Create a plugin that does nothing

The first plugin version can be found in `edu.udo.scaffoldhunter.plugins.datacalculation.impl.example1`. It implements just the basic things, while having no real functionality. It can be selected and executed, but behaves transparently, thus doesn't calculate anything.

B.2.4.1. Example1CalcPlugin.java

@PluginImplementation The first important thing is the `@PluginImplementation` annotation above the class definition. This is needed by the plugin framework to recognize this class as a plugin. If this line is missing, the plugin will not be listed in the calc dialog.

extends AbstractCalcPlugin Your calc plugin has to inherit from `AbstractCalcPlugin`. This abstract class implements the `ImportPlugin` interface for you and also overrides the `toString()` method so that your plugin title is shown correctly in the list of calc plugins in the calc dialog.

getTitle() The `getTitle()` method returns a short name of the plugin. This is also the name which is listed in the calc dialog.

getID() In `getID()` your plugin should return a unique name of the plugin, this is for example used by Scaffold Hunter to match saved properties for the plugins. During the development process of Scaffold Hunter all plugins used the form `CLASSNAME_VERSION` that should be unique enough.

getDescription() The `getDescription()` method returns a description about for what the calc plugin can be used, like ”This is a calc plugin which can be used to calculate the xyz-fingerprint”.

setAvailableProperties(availableProperties) As we do not need to know something about existing properties yet, we leave this method blanc. This will change in the fourth version.

getSettingsPanel(settings,arguments) As we do not need any configuration yet, we just return an empty instance of `SettingsPanel`. This will change in the third version.

getResults(arguments,molecules,msgListener) The given `molecules` parameter is an `Iterable` over the available molecules. As we do not want to calculate any property nor modify any molecule, we return a class implementing `CalcPluginResults` interface, which will return the `molecules` parameter unchanged. It also returns an empty `Set` of `PropertyDefintitions`, which declares we have no properties to be added to Scaffold Hunter.

B.2.5. Second version - ’Calculate’ a new property

The second version of the plugin can be found in `edu.udo.scaffoldhunter.plugins.datacalculation.impl.example2`. Here, we change our last plugin version, so that it creates and saves a property for all given molecules. We don’t really calculate something, we just create a numerical property with value 1.0.

B.2.5.1. Example2CalcPlugin.java

Example2CalcPlugin() In the constructor of `Example2CalcPlugin`, we create a new `PropertyDefinition` and store it in a member variable named `propDef`. `propDef` describes the characteristics of the property we want to add to every molecule. Therefore we set the property type to be a numerical property. See Table B.1 to learn which property types are available. We also set a title, a description and a key. The title is a short description of the property, where as the description is a sentence describing the property in detail. The property key is used for internal processing and written in uppercase letters by convention. It should be as unique as possible. Additionally, we set the property definition to be mappable (this means it can be mapped on a visual feature in the main program) and define it as molecule property (by saying it is not a scaffold property).

getResults(arguments,molecules,msgListener) Here we change our custom `CalcPluginResults` implementation: In the `getMolecules()` method we not simply return the `Iterable` over the available molecules like in the last version. Instead we transform all molecules with a custom transform function first, and return the transformed molecules. The transform function will do all the work like calculating a property and adding it to the molecule. Read *Sec. B.2.5.2: Example2CalcPluginTransformFunction.java* to see what is does in our example.
In the `getCalculatedProperties()` method we return a `Set` which contains the `propDef` we created before. This notifies the plugin system we want to add this property.

PropertyType	Description
`NumProperty`	An ordinary numerical property.
`StringProperty`	An ordinary string property.
`BitStringFingerprint`	A bit fingerprint represented by a string of 1 and 0 (chars).
`BitFingerprint`	A bit fingerprint that interprets every bit of a string as a bit. This is logically identical to BitStringFingerprint but has less memory consumption.
`NumericalFingerprint`	A fingerprint that consists of many numerical values. A NumericalFingerprint is a simple string with integer values separated by a comma: int,int,...

Table B.1.: Property Types

B.2.5.2. Example2CalcPluginTransformFunction.java

implements Function<Molecule, Molecule> Our transform function needs to implement the `Function` interface and we want to transform from molecule to molecule.

Example2CalcPluginTransformFunction(propDef) In the constructor of the `Example2CalcPluginTransformFunction` we simply save the given `PropertyDefinition` parameter in a member variable named `propDef`. We will need this in the `apply()` function.

apply(molecule) The `apply()` functions gets one molecule as input parameter, and returns the transformed molecule. We just insert a mapping from the `propDef` (our `PropertyDefinition`) to the value 1.0 to the molecules property map. Then we return the molecule. The plugin system will read this map and save the property.

B.2.6. Third version - Make the plugin configurable

The third version of the plugin can be found in `edu.udo.scaffoldhunter.plugins.datacalculation.impl.example3`. We now want to make the plugin configurable. The user should choose whether the 'calculated'/added property is set to 1.0 or −1.0, by enabling or disabling a checkbox. Therefore we will extend `PluginSettingsPanel` by a checkbox and introduce a `CalcPluginArguments` class to store the state of the checkbox.

B.2.6.1. Example3CalcPluginArguments.java

The `Example3CalcPluginArguments` class just has a `boolean` member variable encoding the state of the checkbox. Additionally it has a getter and a setter method for this variable.

B.2.6.2. Example3CalcPluginSettingsPanel.java

Example3CalcPluginSettingsPanel(arguments) The `Example3CalcPluginSettingsPanels` constructor saves a reference to the given `arguments` parameter. It also creates a checkbox which is initialized to the state stored in the `arguments` parameter and adds it to the panel. Additionally it creates an `ActionListener` which reacts on changes of the checkbox state and updates the corresponding `boolean` value in the `arguments`.

getArguments() The `getArguments()` method simply returns the `arguments`.

B.2.6.3. Example3CalcPluginTransformFunction.java

The `Example3CalcPluginTransformFunction` constructor is changed so that it saves a reference to the new `arguments` parameter.

apply(molecule) The `apply()` method now determines the property value based on the saved `arguments`.

B.2.6.4. Example3CalcPlugin.java

In the calc plugin itself there are just a few changes. The following methods changed:

getSettingsPanel(settings,arguments) The `getSettingsPanel()` method creates a new `Example3CalcPluginArguments` instance, if the `arguments` parameter is `null`. You should always initialize your arguments in this way. Afterwards a new `Example3CalcPluginSettingsPanel` is instantiated with the `Example3CalcPluginArguments` as parameter and returned.

getResults(arguments,molecules,msgListener) In the `getResults()` method there is just one small change: The `arguments` parameter is casted and passed to the `Example3CalcPluginTransformFunctions` constructor.

B.2.7. Fourth version - Use existing properties for calculation

The fourth version of the plugin can be found in `edu.udo.scaffoldhunter.plugins.datacalculation.impl.example4`. In this version we want to read existing numerical properties and let the user select one. For every molecule our plugin creates a new property which is the same as the selected one, but 1.0 or -1.0 (based on the users choice) is added to the property value.

B.2.7.1. Example4CalcPluginArguments.java

A new variable which saves the property chosen by the user is added together with corresponding getter and setter methods to the `PluginArguments` from the last version.

B.2.7.2. Example4CalcPluginSettingsPanel.java

The `PluginSettingsPanel` from the last version is extended to show a `JList` with all numerical property definitions. A list selection listener is used to update the `Example4CalcPluginArguments` with the property definition selected by the user.

B.2.7.3. Example4CalcPluginTransformFunction.java

The `apply()` method was adjusted so that the value of the chosen property is read from the input molecule, 1.0 or -1.0 is added and the resulting new property value is appended to the property map of the output molecule.

B.2.7.4. Example4CalcPlugin.java

In comparison to the last version there are several small changes in `Example4CalcPlugin`. The constructor was deleted and the creation of the property definition moved to the `getResults()` method.

setAvailableProperties(availableProperties) In the `setAvailableProperties()` method the `availableProperties` are saved as a member variable. Please note that the `setAvailableProperties()` method is the first method called by the plugin system after instantiation of the plugin. For this reason the plugin is able to use this information when creating a `SettingsPanel` in the `getSettingsPanel()` method.

getSettingsPanel(settings, arguments) The only change made in the `getSettingsPanel()` method is that the `availableProperties` are passed to the constructor of the `Example4CalcPluginSettingsPanel`.

getResults(arguments,molecules,msgListener) The `getResults()` method is now responsible for creation of the `PropertyDefinition` stored in `propDef`. `propDef`s key, title and description attributes are set dynamically based on the corresponding attributes of the chosen input property stored in the parameter `arguments`.

B.2.8. Fifth and last version - Display a message during calculation

The fifth and last version of the plugin can be found in `edu.udo.scaffoldhunter.plugins.datacalculation.impl.example5`. Here we will enable the plugin to send messages to the GUI in case that the input property chosen by the user is not defined for a molecule.

B.2.8.1. Example5CalcPlugin.java

In comparison to the last plugin version, there is just one small change: The `getResults(arguments, molecules,msgListener)` method now passes the `msgListener` parameter to the constructor of the `Example5CalcPluginTransformFunction`.

B.2.8.2. Example5CalcPluginTransformFunction.java

All the message handling is done in the `Example5CalcPluginTransformFunction`. In its constructor we therefore read the new `msgListener` parameter and save it as a member variable with the same name. In the `apply()` method we will then use the `msgListener` member variable to send a message.
But first we need more details about the message system integrated into the calc plugin system. Because the `msgListener` variable references an object implementing the `MessageListener` interface, we are able to send a `Message` object by calling `msgListener.receiveMessage(message)`. Suited for the needs of calc plugins there is the `CalcMessage` class which extends the `Message` class. `CalcMessage` has two similar constructors allowing to create a message with a desired `MessageType`, a molecule title and (optionally) a property definition. An object implementing the `MessageType` interface defines a message string and a message icon. For convenience there is an enum `CalcMessageTypes` with some predefined `MessageTypes`, which can be used for your message. See Table B.2 to learn what types you can use. Depending on the used `MessageType` implementation some or all of the given attributes of a message are presented to the user in a tree-like manner shown in Fig. 5.13.

apply(molecule) In the `apply()` method an else clause (which sends the message) was inserted after the if-block (which does the calculation). So in case the chosen property is not defined for the given molecule, a `CalcMessage` is created and sent to the GUI. Note that the molecule title needed to construct the message is read from the molecule properties map, by asking for the key `CDKConstants.TITLE`.

B.2.8.3. Example5CalcPluginArguments.java

This class remains completely unchanged.

B.2.8.4. Example5CalcPluginSettingsPanel.java

This class remains completely unchanged.

Enum constant	Message text
`PROPERTY_NOT_PRESENT`	<molecule title>: source property <property definition title> needed for calculation was not present, thus no value calculated
`CALCULATION_ERROR`	<molecule title>: an error occurred in the calculation plugin, thus no value calculated

Table B.2.: Calc Message Types

B.2.9. Use transform options to handle frequent tasks

If you write a calc plugin that takes the structural information (e.g. 2D graph structure) of a molecule into account, then you may want to give the user the opportunity to use transform options. Transform options are a kind of preprocessing on all molecules before the calc plugin operates on them. See *Sec. 5.2.2.1: Transform Options* for more information.

Using transform options is rather simple. First change your `PluginArguments` class so that it inherits from `AbstractCalcPluginArguments`, which can be found in `edu.udo.scaffoldhunter.plugins.datacalculation`. The second thing you have to do, is to integrate an instance of `CalcPluginTransformOptionPanel` (can be found in `edu.udo.scaffoldhunter.plugins.datacalculation`) in your custom `PluginSettingsPanel`. As the name suggests, you can add the `CalcPluginTransformOptionPanel` like any other Swing container somewhere to your custom panel. When instantiating `CalcPluginTransformOptionPanel`, you need to pass your `PluginArguments` instance to its constructor. After following the instructions above your plugin will show a panel with transform options and depending on the users choice the molecules are transformed automatically before they are processed by the plugin.

C. FAQ - Frequently Asked Questions

C.1. How can I prevent other users from getting access to my data?

If you conduct a study with confidential data and do not want any other user to access Scaffold Hunters internal database, then use a `MySQL` server installed on your local computer. You can also use the connection type `HSQLDB` in the Fig. 4.2, and save the database file on a secure place somewhere on your harddisk.

C.2. The tooltip window annoys me!

In order to disable the Tooltip window select `Session` → `Preferences` → `General Configuration` from Scaffold Hunters menu bar and disable the checkbox labeled "enable tooltips". Instead of disabling the tooltip, you can adjust the other tooltip settings as well. It is possible to change the maximum size of the structure image shown in the tooltip or to select the tooltip popup delay.

C.3. I cannot use all clustering options – there are no properties available for some distance measures!

Some distance measures are only applicable on molecule fingerprints. Read *Chap. 5.2.2: Calculation of Properties* to learn how to calculate additional properties like chemical fingerprints.

C.4. I have lost my password, how can I access my data?

As Scaffold Hunter saves your password encrypted, there is no possibility to restore your password. But it is possible to create a new password. First you need a tool to manage `MySQL` databases (for example: `MySQL`-Workbench[1]). Start Scaffold Hunter and go to the StartDialog. Create a new user and note username and password. Open the database management tool of your choice and connect to the database using the data you used in the Connection Dialog shown on Fig. 4.2. After you have connected to the database, you will probably see a long list of database tables. Select the table `profiles` and edit the table data. Search for the two rows showing your old and your newly created username. Copy the data of the fields `password` and `salt` from the row corresponding the new username to the fields in the row corresponding your old username. Submit your changes and exit the management tool. Now start Scaffold Hunter and login with your old username and the newly created password. Thats it.

WARNING: Be careful while editing the database. Please make a backup of the database prior to editing. Only follow the above instructions if you exactly know what you are doing.

---

[1] `http://www.mysql.de/products/workbench/`