Sieve

- Version 3.0.4 -

Last update: November 4, 2010

silicos

Table of contents

1. Introduction

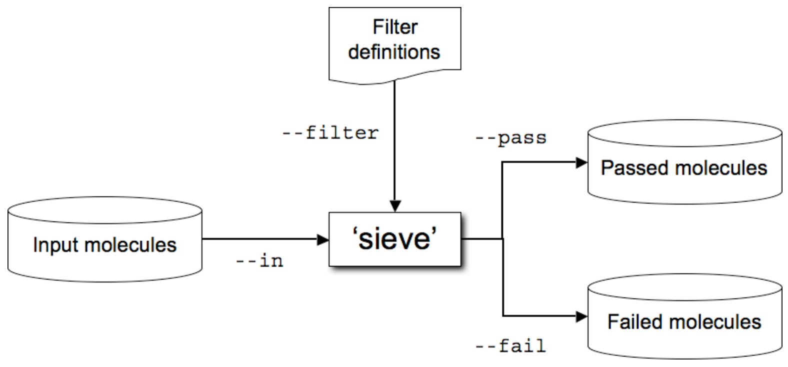
'Sieve' is a program for filtering out molecules with unwanted properties. It is based on the Open Babel open source C++ API for rapid calculation of molecular properties.

The program comes with a number of pre-programmed molecular properties that can be used for filtering. These properties include, amongst others:

- Physicochemical parameters, such as logP, topological polar surface area criteria, number of hydrogen bond acceptors and donors, and Lipinski's rule-of-five;
- Graph-based properties, including ring-based parameters and rotatable bond criteria;
- Selection criteria by means of smarts patterns;
- Similarity criteria;
- Three-dimensional distances between user-definable fragments.

Sieve is a command line-driven program that is instructed by means of command line options and a user-definable filter file. It is by means of this 'filter' file that the user can define the actual filter criteria to be used. The following figure describes the actual data flow:

Figure 1. The input and output flow of molecules and associated files in the program 'sieve'. Command line options are also indicated.



With a set of molecules as input, the program Sieve categorizes these input molecules into 1) a set of molecules that fulfill all criteria as defined in the filter definition file (passed molecules), and 2) a set of molecules that do not fulfill at least one of the defined filter criteria (failed molecules).

The program 'sieve' can also be run in tabulate mode where it functions as a property calculator. In this mode, the requested properties of all input molecules are simply calculated and tabulated. No filtering is done in this run mode. This option is useful for database characterization and for optimization of the filter parameters.

2. Usage

2.1. *Command line interface*

'Sieve' is run from the command line as follows:

```
> sieve [options]
```

Options can be required or optional. If one or more of the required options is missing, the program stops and displays an error message:

```
> sieve

ERROR:
Command line option '--in' is missing.
```

Specifying the '`-h`' or '`--help`' option generates a help message:

```
> sieve -h

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  SIEVE v2.4.0 | Jul 16 2010 13:37:54

  -> GCC:         4.0.1 (Apple Inc. build 5490)
  -> Open Babel:  2.2.99

  Copyright (C) 2005-2010 by Silicos NV

  This file is part of the Open Babel project.
  For more information, see <http://openbabel.sourceforge.net/>

  This program is free software; you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation version 2 of the License.

  This program is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License for more details.
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


TASK:

  Sieve is a tool to filter out molecules according user-defined criteria.

REQUIRED OPTIONS:

  --in <file>
    Specifies the file containing the input molecules. The format of the
    file is specified by the file extension. Gzipped files are also read.

  --pass <file>
    Specifies the file to which the molecules are written that pass the filtering.
    The format of the file is specified by the file extension. Gzipped
    files are also read.

  --filter <file>
    Specifies the file in which the filter criteria have been defined.

OPTIONAL OPTIONS:

  --fail <file>
    Specifies the file to which the molecules are written which do not
    pass the filter criteria. The format of the file is specified
    by the file extension. Gzipped files are also read.

  --salts
    This flag directs the program to keep all separate fragments instead of
    stripping away all but the largest fragments before the filtering takes
    place. By specifying this option, this stripping is not performed and
    ensures that all salt counterions are taken into account in the filtering
    process.
```

```
--rename
  This flag directs the program to rename the title of each molecules
  into a increasing digit reflecting the sequence of the molecule in
  the input file. Existing titles are overwritten.

--tab
  This flag directs the program to calculate all properties listed in
  the filter definition file without applying any filtering. This option
  can be used for database characterisation. The calculated parameters
  are written to standard output.

--noLog
  This flag specifies whether verbose logging should be switched off. If not
  specified, then for each molecule a message is written to standard error
  whether the molecule passes or fails the filter criteria. This behaviour
  can be switched off with this command line option. However, even
  when this option is specified, information is still written to standard error,
  but reduced to a large extend.

-h  --help

-v  --version
```

The following sections describe in detail the different options.

2.2. Required command line options

2.2.1. Input molecules

The name of the file containing the input molecules is specified using the '`--in`' option:

```
> sieve --in filename.ext [other_options]
```

The input file contains one or more molecules specified as a set of connection tables according specific molecular formats. The format of these connection tables is specified by the input filename extension. The allowed input formats are those that are supported by Open Babel. Zipped-formats are also allowed.

2.2.2. Passed molecules

The name of the file to which the molecules are written that have successfully passed all the filter criteria is specified using the '`--pass`' option:

```
> sieve --pass filename.ext [other_options]
```

The pass output file contains the molecules specified as a set of connection tables in a specific molecular format. The format is specified by the filename extension. The supported output formats are those that are supported by Open Babel. Zipped-formats are also allowed.

If the tabulate run mode has been requested (see below), then this output option is not required. Instead, in this situation the option is simply ignored by the program.

2.2.3. Filter criteria

This option specifies a filter file describing the filter criteria that should be applied to all input molecules. The name of the file is specified using the '`--filter`' option:

```
> sieve --filter filename [other_options]
```

A detailed overview of the specific format of this file and a list of all parameters is provided in Section 3.

2.3. *Optional command line options*

2.3.1. Failed molecules

Molecules that fail for at least one of the filter criteria are not written to the pass output file but to an other file. This file is optional. The name of this specific file is specified with the '`--fail`' options:

```
> sieve --fail failed.smi [other_options]
```

The fail output file contains the molecules specified as a set of connection tables in a specific molecular format. The format is specified by the filename extension. The supported output formats are those that are supported by Open Babel. Zipped-formats are also allowed.

If the tabulate run mode has been requested (see below), then this output option is not processed.

2.3.2. Property tables

During a normal 'sieve' run, a file of input molecules is filtered based on a set of user-definable molecular property criteria. This default behavior is modified when the tabulate run mode has been selected by specifying the '`--tab`' option. In this case, the majority of the requested properties are still calculated for each molecule, but the results of the properties are simply written to standard output without performing a filtering step. The purpose of this option is to generate a wide range of molecular properties that can be used for database characterization and statistical analysis, as well as for optimization of the user-definable fragment patterns.

When the '`--tab`' option has been specified, the molecules are not written to the '`--pass`' and '`--fail`' files; instead, with the '`--tab`' option specified, these latter two options are not required and are not even processed by the program.

Property tables are calculated for the majority of properties, with the exception of a few. Table 1 provides a summary of the properties that are supported during a tabulate run.

Table 1. Summary of the properties that are supported in the tabulate run mode.

Property rule	**Number of rules allowed**	**Supported by `--tab` mode**	**Remark**
Title rule	Unlimited (at least one should match)	Yes	The molecular title is tabulated
Element rules	Maximum 1	Yes	The molecular formula is tabulated
Topological property rules	Maximum 1	Yes	-
Physical property rules	Maximum 1	Yes	-

Property rule	**Number of rules allowed**	**Supported by `--tab` mode**	**Remark**
Fragment rules	Unlimited (all should match)	Yes	-
Similarity rules	Unlimited (all should match)	Yes	-
Similarity stack rules	Unlimited	No	-
Distance rules	Unlimited (all should match)	No	-
Sdf-tag rules	Unlimited (all should match)	Yes	-

2.3.3. Salts

Before any property calculation and filtering step takes place in sieve, each molecule is cleaned by removing all small fragments that are present in the molecular connection table. Such small fragments are often salt fragments. Implementation-wise, salts are first removed, and then the actual filtering is taking place on the remaining part of the molecular entity. This prevents molecules to be filtered out due to unwanted salt fragments that should have been removed in first instance.

The default salt-stripping step can be suppressed by specifying the '`--salts`' option, in which case no salt stripping occurs. In this case, all calculations and filtering steps are performed on the entire molecule together with all salt moieties that might be present in the molecular connection table.

2.3.4. Rename

Molecular titles are sometimes absent or misleading. In such cases, it might be useful to rename the title of each molecule into a number reflecting the sequence of the molecule in the input file. This feature is optional and is initiated with the '`--rename`' option. If this option is not provided, then renaming will not occur.

2.3.5. Logging

When the program is run in the normal filtering mode (not the tabulate mode), output messages are written to standard error to indicate whether or not the molecule fails or passes the filter criteria. For large input files, this could lead to large logging files. Therefore, a command line option is provided to turn off this default behaviour. When the '`--noLog`' command line option is specified, only a limited number of messages are written to standard error.

3. Filter parameters

The filter file (‘`--filter`’) specifies the limits on all of the calculated physical properties, and also specifies functional group substructures that should be included in the filtering step.

Rules are specified by a keyword that is optionally followed by some limits or specifications. Words can be separated by white space. Blanc lines, or lines starting with ‘`#`’ of ‘`//`’, are ignored. These latter can be used as comment lines.

Property rule statements may be repeated in a single filter definition file. Unless indicated otherwise (see Table 1 and each of the appropriate sections), in this case only the first encountered property line is included for the filtering.

The eight general types of rules that may occur in a filter file are:

- *include* rule (1 specific rule)
- *title* rule (1 specific rule)
- *element* rules (2 specific rules)
- *topological property* rules (47 specific rules)
- *physical property* rules (8 specific rules)
- *fragment* rules (3 specific rules)
- *sdf-data rules* (2 specific rules)
- *distance* rules (1 specific rule)

Together, these lead to a total of 65 rules that are specified by specific keywords. Each of these are detailed in the following sections.

3.1. Include rules

INCLUDE

Include rules are specified with the `INCLUDE` keyword, and define the name of a file which should be included starting at that position. The purpose of these include rules is to enable nesting of different filter files into one large file. For example, one could prepare a file in which definitions of unwanted functional groups are specified, and another file in which definitions on drug-like physicochemical parameters are specified. With the `INCLUDE` keyword, one can merge these two files in one large set of rules.

Include rules are specified by `INCLUDE` keyword followed by the name of the file that should be included:

```
INCLUDE /Users/hans/Filters/Druglike.sieve
INCLUDE /Users/hans/Filters/Clean.sieve
```

Only complete filenames are allowed. For example, the following will generate an error message and will cause the program to halt:

```
INCLUDE ~/Filters/Clean.sieve
```

An unlimited number of `INCLUDE` keywords are allowed and the keywords may be nested as well. If more than one `INCLUDE` keyword is provided, all of the keywords are processed in the order that these are defined in the corresponding filter file(s).

3.2. *Title rules*

TITLE

Title rules filter the input molecules based on the presence of their molecular titles. In sdf-files, this molecular title is specified as the first line of each molecular entry, while in smiles entries such title could optionally be specified following the smiles specification separated by whitespace.

Title rules are specified with the `TITLE` keyword. Molecules can have only one title at most.

Multiple title rules, each specifying different title strings, may be specified. In this case, each of the different title rules will be checked for a valid match against each of the input molecules. If a match has been found between the molecule and one of the different title rules, the molecule is flagged to pass for this criterion. If no match could be found between the molecule and all of the different title rules, the molecule is flagged to fail.

The title itself should be enclosed by double quotes to allow titles to contain spaces:

```
TITLE “Molecule title”
```

If quotes are to be treated as being part of the title, a backslash sign should precede the quotes:

```
TITLE “Molecule title \” with backslash”
```

In case the tabulate mode has been selected for running ‘sieve’ (command line option ‘`--tab`’), only the specification of the keyword without the actual title string is sufficient. When run in this mode, the program will write the molecular titles to standard output.

3.3. *Element rules*

Element rules are used to specify restrictions on the allowed or unwanted elements within the molecules.

In the current version of ‘sieve’, two specific element rules have been implemented:

- ONLY_ELEMENTS
- EXCLUDED_ELEMENTS

Element rules are specified by the appropriate keyword followed by a list of element symbols. The specification of the elements should be separated by white space, and can be written in lowercase, uppercase, or a combination of both. Quotes are not allowed:

```
ONLY_ELEMENTS H C N O Br I S
EXCLUDED_ELEMENTS C N O S
```

For each of the two specific element rules, only one of each is allowed in the filter file. If more than one identical rule keyword is provided, only the first keyword will be processed correctly and a message will be printed to warn the user that more than one identical keyword has been encountered. All the information provided by the subsequent identical rules in the filter file is neglected, as exemplified here:

```
ONLY_ELEMENTS H C N O Br I S
ONLY_ELEMENTS Cl
```

In this example, the molecules are filtered on the presence of the H, C, N, O, Br, I and S elements, as defined by the first `ONLY_ELEMENTS` rule. In the next example, both element rules are included for the filtering since the two specific keywords are different:

```
ONLY_ELEMENTS H C N O Br I S
EXCLUDED_ELEMENTS Cl
```

In case the tabulate mode has been selected for running 'sieve' (command line option '`--tab`'), then only the specification of the keyword without the elements is sufficient. When run in this mode, the program writes the calculated molecular formula to standard output.

ONLY_ELEMENTS

This specific element rule specifies the list of elements that are allowed in the input molecules. Molecules consisting of additional elements that have not been specified by this rule are flagged to fail. Implicit hydrogen atoms are taken into account for the filtering.

In the following example, only molecules that contain elements out of the H, C, N, O, Br, I, or S list are flagged for passing:

```
ONLY_ELEMENTS H C N O Br I S
```

Molecules with, for example, a fluor atom would be rejected in this example.

EXCLUDED_ELEMENTS

This rule specifies the elements that are not allowed in the input molecules. Molecules consisting of elements that are specified by this rule are flagged for failure. Implicit hydrogen atoms are taken into account during the filtering step.

In the following example, molecules that contain one or more halogens are rejected:

```
EXCLUDED_ELEMENTS F Cl Br I
```

3.4. Topological property rules

Topological property rules specify limits on topology-derived properties. These rules are specified by the appropriate *keyword* followed by a *minimum* and a *maximum* numerical limit for the calculated property:

```
KEYWORD minimum maximum
```

The *keyword* specifies the topological property and the *minimum* and *maximum* numerical values specify the limit criteria that are used for the filtering. The *minimum* and *maximum* limits are inclusive, meaning that all molecules of which the property value is larger or equal than *minimum* AND smaller or equal than *maximum* are flagged for success for that particular property.

In cases where a *minimum* or *maximum* limit is hard to specify, for example when one does not want to specify a certain limit, a '*' character may be substituted. In the next example, all molecules with a minimum of 30 non-hydrogen atoms are passed through:

```
ATOMS 30 *
```

In the next example, all molecules with more than 20 heavy atoms are filtered out:

```
ATOMS * 20
```

which is similar to specifying:

```
ATOMS 0 20
```

In the current version of Sieve, 47 specific topological property rules have been implemented:

- ATOMS
- CARBONS
- HETERO_ATOMS
- HETERO_CARBON_RATIO
- HALIDES
- HALIDE_FRACTION
- BONDS
- ROTATABLE_BONDS
- RIGID_BONDS
- FLEXIBILITY
- CHIRAL_CENTERS
- HBOND_ACCEPTORS
- HBOND_DONORS
- LIPINSKI_ACCEPTORS
- LIPINSKI_DONORS
- FORMAL_CHARGES
- TOTAL_FORMAL_CHARGE
- RINGS
- ATOMS_IN_SMALLEST_RING
- ATOMS_IN_LARGEST_RING
- RING_FRACTION
- AROMATIC_RINGS
- ATOMS_IN_SMALLEST_AROMATIC_RING
- ATOMS_IN_LARGEST_AROMATIC_RING
- AROMATIC_RING_FRACTION
- AROMATIC_OVER_TOTAL_RING_FRACTION
- NONAROMATIC_RINGS
- ATOMS_IN_SMALLEST_NONAROMATIC_RING

- ATOMS_IN_LARGEST_NONAROMATIC_RING
- NONAROMATIC_RING_FRACTION
- RINGSYSTEMS
- ATOMS_IN_SMALLEST_RINGSYSTEM
- ATOMS_IN_LARGEST_RINGSYSTEM
- RINGSYSTEM_FRACTION
- RINGS_IN_SMALLEST_RINGSYSTEM
- RINGS_IN_LARGEST_RINGSYSTEM
- SIDECHAINS
- ATOMS_IN_SMALLEST_SIDECHAIN
- ATOMS_IN_LARGEST_SIDECHAIN
- SIDECHAIN_FRACTION
- CORES
- ATOMS_IN_CORE
- CORE_FRACTION
- BRIDGES
- ATOMS_IN_SMALLEST_BRIDGE
- ATOMS_IN_LARGEST_BRIDGE
- BRIDGE_FRACTION

For each of these 47 topological property rules, only one of each is allowed in the filter file. If more than one identical rule keyword is encountered, only the first keyword will be processed correctly and a message will be printed to warn the user that more than one identical keyword has been encountered. All the information provided by the subsequent identical rules in the filter file is neglected.

When the program is run in tabulate mode (with the '`--tab`' command line option) to have only the properties calculated but not filtered, then the *minimum* and *maximum* limits are not required and need not to be specified.

In the following sections, all topological property rules are detailed.

3.4.1. Atoms

ATOMS

Specifies the number of non-hydrogen atoms (the so-called heavy atoms) as filter criterion.

In the following example, all molecules with ≥10 and ≤20 heavy atoms are accepted for this criterion, while all other molecules are rejected:

```
ATOMS 10 20
```

CARBONS

Specifies the number of carbon atoms as filter criterion.

In the following example, all molecules with ≥10 and ≤20 carbon atoms are accepted for this criterion, while all other molecules are rejected:

```
CARBONS 10 20
```

HETERO_ATOMS

Specifies the number of heteroatoms as filter criterion. Heteroatoms are defined as the atoms that are not hydrogen or carbon.

In the following example, all molecules with ≥10 and ≤20 heteroatoms are accepted, while all other molecules are rejected:

```
HETERO_ATOMS 10 20
```

HETERO_CARBON_RATIO

Specifies the ratio of the number of heteroatoms over the number of carbon atoms. Heteroatoms are defined as all atoms that are not hydrogen or carbon. For example, the `HETERO_CARBON_RATIO` for chloroform ($CHCl_3$) is 3 / 1 = 3.

If there are no carbon atoms, then a value of zero is returned and the molecule is marked to fail for this criterion.

In the following example, all molecules with a hetero-over-carbon ratio of ≥0.5 and ≤1.5 are accepted, while all other molecules are rejected:

```
HETERO_CARBON_RATIO 0.5 1.5
```

HALIDES

Specifies the number of halide elements as filter criterion. A halide is defined as a F, Cl, Br, or I atom.

In the following example, all molecules with more than three halides are flagged to fail:

```
HALIDES 0 3
```

HALIDE_FRACTION

Specifies the ratio of the halide weight over the total molecular weight as filter criterion. A halide is defined as a F, Cl, Br, or I atom. Implicit hydrogen atoms are taken into account for the calculation of the total molecular weight.

If the total molecular weight is zero, then a value of zero is returned and the molecule is flagged to fail for this criterion.

In the following example, all molecules that have a halide weight over the total weight ratio between 0.5 and 0.9 are flagged to pass this criterion:

```
HALIDE_FRACTION 0.5 0.9
```

3.4.2. Bonds

BONDS

Specifies the number of bonds between non-hydrogen atoms (the so-called heavy atoms) as filter criterion.

In the following example, all molecules with ≥10 and ≤20 bonds are accepted, while all other molecules are rejected:

```
BONDS 10 20
```

ROTATABLE_BONDS

Specifies the number of rotatable bonds as filter criterion. A rotatable bond is defined as follows:

- bond is not double; and
- bond is not triple; and
- bond is not a primary amide ($O=C-NH_2$); and
- bond is not a secondary amide (or $O=C-NH_1$).

and excluding:

- bonds in ring; and
- end standing bonds; and
- bonds to hydrogen atoms.

In the following example, molecules with more than 10 rotatable bonds would be excluded:

```
ROTATABLE_BONDS 0 10
```

RIGID_BONDS

Specifies the number of rigid bonds as filter criterion. A rigid bond is defined as follows:

- bond is double; or
- bond is triple; or
- bond is a secondary amide ($O=C-NH_1$);

but excluding:

- end standing bonds; and
- bonds to hydrogen atoms; and
- bonds in ring.

In the following example, molecules with more than 10 rigid bonds would be excluded by this filter criterion:

```
RIGID_BONDS 0 10
```

FLEXIBILITY

Flexibility is defined as the ratio of the number of rotatable bonds (as defined by the `ROTATABLE_BONDS` property) over the total number of bonds (as defined by the sum of the `ROTATABLE_BONDS` and `RIGID_BONDS` properties). The flexibility is always between 0 and 1, with 0 in the extreme case when the molecule contains no rotatable bonds, and 1 being the theoretical case when all bonds are rotatable.

When the molecule contains no bonds, then a value of zero is returned and the molecule is filtered according the specified criteria.

When the molecule contains no non-hydrogen atoms, a value of zero is returned and the molecule is flagged to fail for this criterion.

In the following example, molecules with a flexibility between 0.3 and 0.5 are passed along:

```
FLEXIBILITY 0.3 0.5
```

3.4.3. Chirality

CHIRAL_CENTERS

Specifies the number of chiral centers as filter criterion.

In the following example, all molecules with less than three chiral centers are flagged for acceptance, while all other molecules are rejected:

```
CHIRAL_CENTERS 0 2
```

3.4.4. Hydrogen bonds

HBOND_ACCEPTORS

Specifies the number of hydrogen bond acceptors as filter criterion. A hydrogen bond acceptor is defined as an atom with one of the following properties:

- an aromatic nitrogen with no hydrogen atoms connected, no amide nitrogen, and not carrying a positive charge; or
- an aliphatic nitrogen with no hydrogen atoms connected and not carrying a positive charge; or
- any oxygen atom that is not carrying a positive charge; or
- a thionyl (=S) sulfur atom;

but excluding:

- amide nitrogen atoms.

In the following example, all molecules with less than 11 hydrogen bond acceptors are allowed, while all other molecules are rejected:

```
HBOND_ACCEPTORS 0 10
```

HBOND_DONORS

Specifies the number of hydrogen bond donors as filter criterion. A hydrogen bond donor is defined as an hydrogen atom with one of the following properties:

- each hydrogen bonded to a nitrogen; or
- each hydrogen bonded to an oxygen; or
- each hydrogen bonded to a sulfur.

According these definitions, a nitrogen with one connected hydrogen is counted as one hydrogen bond donor, while a nitrogen with two hydrogen atoms connected is counted as two hydrogen bond donors.

In the following example, all molecules with less than six hydrogen bond donors are flagged to pass this criterion with success, while all other molecules are flagged for failure:

```
HBOND_DONORS 0 5
```

LIPINSKI_ACCEPTORS

Specifies the number of Lipinski's hydrogen bond acceptors as filter criterion. A Lipinski hydrogen bond acceptor is defined as any nitrogen or oxygen atom regardless of the number of connected hydrogen atoms.

In the following example, all molecules with less than 11 Lipinski hydrogen bond acceptors are accepted, while all other molecules are rejected:

```
LIPINSKI_ACCEPTORS 0 10
```

LIPINSKI_DONORS

Specifies the number of Lipinski's hydrogen bond donors as filter criterion. A Lipinski hydrogen bond donor is defined as an hydrogen atom with one of the following properties:

- each hydrogen connected to a nitrogen; or
- each hydrogen connected to an oxygen.

According these definitions, a nitrogen with one connected hydrogen is counted as one hydrogen bond donor, while a nitrogen with two hydrogen atoms connected is counted as two hydrogen bond donors

In the following example, all molecules with less than six Lipinski hydrogen bond donors are accepted, while all other molecules are rejected:

```
LIPINSKI_DONORS 0 5
```

3.4.5. Charges

FORMAL_CHARGES

This keyword specifies the total number of atoms bearing a formal charge as filter criterion. Correct assignment of formal charges is the responsibility of the user as 'sieve' is not modifying the input molecules.

In the following example, only molecules that have no atoms bearing a formal charge are passed successfully:

```
FORMAL_CHARGES 0 0
```

TOTAL_FORMAL_CHARGE

This keyword specifies the total molecular formal charge as filter criterion. Correct assignment of formal charges is the responsibility of the user as 'sieve' is not modifying the input molecules.

In the following example, only molecules with a neutral formal charge are passed successfully:

```
TOTAL_FORMAL_CHARGE 0 0
```

3.4.6. Rings

RINGS

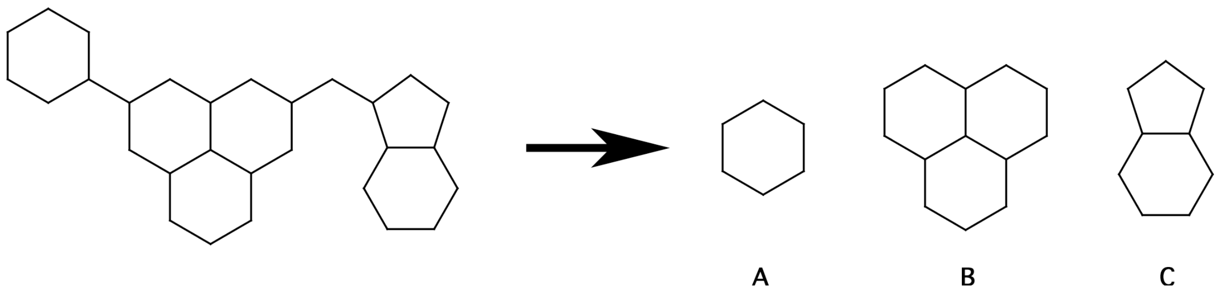
Specifies the total number of rings as filter criterion. Rings are detected by a 'smallest-subset-of-smallest-rings' (SSSR) algorithm, as implemented by OpenBabel and based on the blue-obelisk:findSmallestSetOfSmallestRings algorithm. Figure 2 illustrates the concept of rings and ringsystems.

If the input molecule has no rings then a value of zero is returned and the specified *minimum* and *maximum* parameters determine whether the molecule will be passed or not.

In the following example, molecules with five to six rings would be passed:

```
RINGS 5 6
```

Figure 2. The concept of rings and ringsystems illustrated by example. In this example, the molecule consists of three ringsystems (A-C) and six rings. The number of atoms in each of these three ringsystems varies between 6 (ringsystem A) and 13 (ringsystem B), while the number of atoms in the rings varies between five and six.



ATOMS_IN_SMALLEST_RING

Specifies a limit on the number of atoms in the smallest ring. With reference to the example above (Figure 2), the smallest ring is the five-membered ring in ringsystem 'C', which contains 5 atoms.

When the molecule has no rings, filtering is not performed and a value of zero is returned.

In the following example, only molecules with no rings, or with the smallest ring consisting of 5 to 6 atoms, are successfully passed:

```
ATOMS_IN_SMALLEST_RING 5 6
```

ATOMS_IN_LARGEST_RING

Specifies a limit on the number of atoms in the largest ring. With reference to the example above (Figure 2), the largest ring is the six-membered ring in ringsystems 'A', 'B' or 'C', each containing 6 atoms.

When the molecule has no rings, filtering is not performed and a value of zero is returned.

In the following example, only molecules with no rings, or with the largest ring consisting of 5 to 6 atoms, are flagged to pass this criterion:

```
ATOMS_IN_SMALLEST_RING 5 6
```

RING_FRACTION

Is calculated as the number of ring atoms divided by the total number of atoms. The outcome is always a number between 0 and 1, both numbers inclusive. Hydrogen atoms are not taken into account for the calculation.

If the input molecule has no rings then a value of zero is returned and the specified *minimum* and *maximum* parameters determine whether the molecule will be passed or not.

When the molecule contains no non-hydrogen atoms, a value of zero is returned and the molecule is marked to fail for this criterion.

In the following example, molecules with no rings, or with a maximum of 30% of all non-hydrogen atoms being part of rings, are passed through:

```
RING_FRACTION 0 0.3
```

AROMATIC_RINGS

Specifies the total number of aromatic rings as filter criterion. Rings are detected by a 'smallest-subset-of-smallest-rings' (SSSR) algorithm, as implemented by OpenBabel and based on the blue-obelisk:findSmallestSetOfSmallestRings algorithm. Figure 2 illustrates the concept of rings and ringsystems.

If the input molecule has no rings then a value of zero is returned and the specified *minimum* and *maximum* parameters determine whether the molecule will be passed or not.

In the following example, molecules with one to two aromatic rings would be passed:

```
AROMATIC_RINGS 1 2
```

ATOMS_IN_SMALLEST_AROMATIC_RING

Specifies a limit on the number of atoms in the smallest aromatic ring.

When the molecule has no rings, filtering is not performed and a value of zero is returned.

In the following example, only molecules with no rings, or with the smallest aromatic ring consisting of 5 to 6 atoms, are successfully passed:

```
ATOMS_IN_SMALLEST_AROMATIC_RING 5 6
```

ATOMS_IN_LARGEST_AROMATIC_RING

Specifies a limit on the number of atoms in the largest aromatic ring.

When the molecule has no rings, filtering is not performed and a value of zero is returned.

In the following example, only molecules with no rings, or with the largest aromatic ring consisting of 5 to 6 atoms, are successfully passed:

```
ATOMS_IN_LARGEST_AROMATIC_RING 5 6
```

AROMATIC_RING_FRACTION

Is calculated as the number of aromatic ring atoms divided by the total number of atoms. The outcome is always a number between 0 and 1, both numbers inclusive. Hydrogen atoms are not taken into account for the calculation.

If the input molecule has no rings then a value of zero is returned and the specified *minimum* and *maximum* parameters determine whether the molecule will be passed or not.

When the molecule contains no non-hydrogen atoms, a value of zero is returned and the molecule is marked to fail for this criterion.

In the following example, molecules with no rings, or with a maximum of 30% of all non-hydrogen atoms being part of aromatic rings, are passed through:

```
AROMATIC_RING_FRACTION 0 0.3
```

AROMATIC_OVER_TOTAL_RING_FRACTION

Is calculated as the number of aromatic ring atoms divided by the total number of rings. The outcome is always a number between 0 and 1, both numbers inclusive.

If the input molecule has no aromatic rings then a value of zero is returned and the specified *minimum* and *maximum* parameters determine whether the molecule will be passed or not.

When the molecule contains no non-hydrogen atoms or contains no rings, a value of zero is returned and the molecule is marked to fail for this criterion.

In the following example, molecules with no rings, or with a maximum of 30% of all rings being aromatic, are passed through:

```
AROMATIC_OVER_TOTAL_RING_FRACTION 0 0.3
```

NONAROMATIC_RINGS

Specifies the total number of non-aromatic rings as filter criterion. Rings are detected by a 'smallest-subset-of-smallest-rings' (SSSR) algorithm, as implemented by OpenBabel and based on the blue-obelisk:findSmallestSetOfSmallestRings algorithm. Figure 2 illustrates the concept of rings and ringsystems.

If the input molecule has no rings then a value of zero is returned and the specified *minimum* and *maximum* parameters determine whether the molecule will be passed or not.

In the following example, molecules with one to two non-aromatic rings would be passed:

```
NONAROMATIC_RINGS 1 2
```

ATOMS_IN_SMALLEST_NONAROMATIC_RING

Specifies a limit on the number of atoms in the smallest non-aromatic ring.

When the molecule has no rings, filtering is not performed and a value of zero is returned.

In the following example, only molecules with no rings, or with the smallest non-aromatic ring consisting of 5 to 6 atoms, are successfully passed:

```
ATOMS_IN_SMALLEST_NONAROMATIC_RING 5 6
```

ATOMS_IN_LARGEST_NONAROMATIC_RING

Specifies a limit on the number of atoms in the largest non-aromatic ring.

When the molecule has no rings, filtering is not performed and a value of zero is returned.

In the following example, only molecules with no rings, or with the largest non-aromatic ring consisting of 5 to 6 atoms, are successfully passed:

```
ATOMS_IN_LARGEST_NONAROMATIC_RING 5 6
```

NONAROMATIC_RING_FRACTION

Is calculated as the number of non-aromatic ring atoms divided by the total number of atoms. The outcome is always a number between 0 and 1, both numbers inclusive. Hydrogen atoms are not taken into account for the calculation.

If the input molecule has no rings then a value of zero is returned and the specified *minimum* and *maximum* parameters determine whether the molecule will be passed or not.

When the molecule contains no non-hydrogen atoms, a value of zero is returned and the molecule is marked to fail for this criterion.

In the following example, molecules with no rings, or with a maximum of 30% of all non-hydrogen atoms being part of non-aromatic rings, are passed through:

```
NONAROMATIC_RING_FRACTION 0 0.3
```

NONAROMATIC_OVER_TOTAL_RING_FRACTION

Is calculated as the number of non-aromatic ring atoms divided by the total number of rings. The outcome is always a number between 0 and 1, both numbers inclusive.

If the input molecule has no non-aromatic rings then a value of zero is returned and the specified *minimum* and *maximum* parameters determine whether the molecule will be passed or not.

When the molecule contains no non-hydrogen atoms or contains no rings, a value of zero is returned and the molecule is marked to fail for this criterion.

In the following example, molecules with no rings, or with a maximum of 30% of all rings being non-aromatic, are passed through:

```
NONAROMATIC_OVER_TOTAL_RING_FRACTION 0 0.3
```

RINGSYSTEMS

Specifies a limit on the number of ringsystems in a cyclic molecule. A ringsystem is defined as the entire set of rings that remain after deleting all the atoms and bonds that do belong to at least one ring. Figure 2 illustrates the concept of ringsystems.

In the following example, only molecules with four ringsystems will be flagged for passing successfully:

```
RINGSYSTEMS 4 4
```

If the input molecule has no rings then a value of zero is returned and the filtered according the specified parameters.

ATOMS_IN_SMALLEST_RINGSYSTEM

Specifies a limit on the number of atoms in the smallest ringsystem. The quantification of 'smallest ringsystem' is based on the number of atoms in the ringsystem. With reference to the example above (Figure 2), the smallest ringsystem is ringsystem 'A', which contains 6 atoms.

When the molecule has no rings, filtering is not performed and a value of zero is returned.

In the following example, only molecules with no rings, or with the smallest ring consisting of 5 to 6 atoms, are passed successfully:

```
ATOMS_IN_SMALLEST_RING 5 6
```

ATOMS_IN_LARGEST_RINGSYSTEM

Specifies a limit on the number of atoms in any of the largest ringsystem. The quantification of 'largest ringsystem' is based on the number of atoms in the ringsystem. With reference to the example above (Figure 2), the largest ringsystem is 'B', containing 13 atoms.

When the molecule has no rings, filtering is not performed and a value of zero is returned.

In the following example, only molecules with no rings, or with the largest ring consisting of up to 7 atoms, are passed:

```
ATOMS_IN_LARGEST_RING * 7
```

RINGSYSTEM_FRACTION

Is calculated as the number of ringsystem atoms divided by the total number of atoms. The outcome is always a number between 0 and 1, both numbers inclusive. Hydrogen atoms are not taken into account for the calculation. This filter produces exactly the same result as the `RING_FRACTION` filter.

If the input molecule has no rings then a value of zero is returned and the specified *minimum* and *maximum* parameters determine whether the molecule will be passed or not.

When the molecule contains no non-hydrogen atoms, a value of zero is returned and the molecule is marked to fail for this criterion.

In the following example, molecules with no rings, or with a maximum of 30% of all non-hydrogen atoms being part of a ringsystem, are passed through:

```
RINGSYSTEM_FRACTION 0 0.3
```

RINGS_IN_SMALLEST_RINGSYSTEM

Specifies a limit on the number of rings in the smallest ringsystem. The quantification of 'smallest ringsystem' is based on the number of rings in the ringsystem. This is different to the `ATOMS_IN_SMALLEST_RINGSYSTEM` filter, where the number of atoms is used as quantification. With reference to the example above (Figure 2), the smallest ringsystem is ringsystem 'A', which contains one ring. Rings are detected by a 'smallest-subset-of-smallest-rings' (SSSR) algorithm, as implemented by OpenBabel and based on the blue-obelisk:findSmallestSetOfSmallestRings algorithm.

When the molecule has no rings, filtering is not performed and a value of zero is returned.

In the following example, only molecules with no rings, or with the smallest ringsystem consisting of one ring, are passed successfully:

```
RINGS_IN_SMALLEST_RINGSYSTEM 1 1
```

RINGS_IN_LARGEST_RINGSYSTEM

Specifies a limit on the number of rings in the largest ringsystem. The quantification of 'largest ringsystem' is based on the number of rings in the ringsystem. This is different to the `ATOMS_IN_LARGEST_RINGSYSTEM` filter, where the number of atoms is used as quantification. With reference to the example above (Figure 2), the largest ringsystem is ringsystem 'C', which contains three rings. Rings are detected by a 'smallest-subset-of-smallest-rings' (SSSR) algorithm, as implemented by OpenBabel and based on the blue-obelisk:findSmallestSetOfSmallestRings algorithm.

When the molecule has no rings, filtering is not performed and a value of zero is returned.

In the following example, only molecules with no rings, or with the largest ringsystem consisting of one ring, are passed successfully:

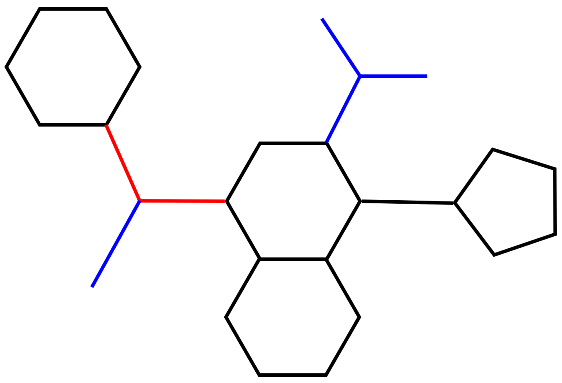
```
RINGS_IN_LARGEST_RINGSYSTEM 1 1
```

3.4.7. Molecular skeleton

SIDECHAINS

Specifies the total number of sidechains as filter criterion. Sidechains are calculated in an iterative fashion by virtually removing all atoms having a valence of one, until no more such atoms remain. The set of all removed atoms compose the sidechain atoms. Hydrogen atoms are not taken into account for the calculation. Figure 3 below illustrates the concept of sidechains on an example molecule.

Figure 3 The concept of 'sidechains', 'bridges', and 'cores' illustrated on a virtual molecule. In this example, the molecule contains two 'sidechains' (colored in blue). In addition, the molecule contains one 'bridge' (shown in red). Finally, the 'core' of the molecule is composed of the black lines (the ringsystems) in combination with the 'bridges' (red lines).



The molecule shown in Figure 3 contains two sidechains. Hydrogen atoms are excluded.

If the input molecule has no rings then a value of zero is returned and the specified *minimum* and *maximum* parameters determine whether the molecule will be passed or not.

In the following example, only molecules with five sidechains would be passed:

```
SIDECHAINS 5 5
```

ATOMS_IN_SMALLEST_SIDECHAIN

Specifies a limit on the number of atoms in the smallest sidechain of the molecule. In the example above in Figure 3, the smallest sidechain contains 1 atom. Hydrogen atoms are excluded from the calculation.

When the molecule has no rings, filtering is not performed and a value of zero is returned.

In the following example, only molecules with either no sidechains, or molecules where the smallest sidechain may consist between 1 and 3 atoms, would be passed:

```
ATOMS_IN_SMALLEST_SIDECHAIN 1 3
```

ATOMS_IN_LARGEST_SIDECHAIN

Specifies a limit on the number of atoms in the largest sidechain of the molecule. Hydrogen atoms are excluded from the calculation. This criterion returns 0 if the molecule contains no sidechains, and in this case the filtering is not applied. In the example above in Figure 3, the largest sidechain contains 3 atoms.

When the molecule has no rings, filtering is not performed and a value of zero is returned.

In the following example, only molecules with either no sidechains, or molecules where the largest sidechain may consist between 1 and 3 atoms, would be passed:

```
ATOMS_IN_LARGEST_SIDECHAIN 1 3
```

SIDECHAIN_FRACTION

Specifies the fraction of sidechain atoms over the total atom count as filter criterion. Hydrogen atoms are excluded from the calculation.

If the input molecule has no rings then a value of zero is returned and the specified *minimum* and *maximum* parameters determine whether the molecule will be passed or not.

When the molecule contains no non-hydrogen atoms, a value of zero is returned and the molecule is marked to fail for this criterion.

In the following example, molecules of which at least half of its atoms are sidechain atoms are passed through:

```
SIDECHAIN_FRACTION 0.5 *
```

CORES

Specifies the total number of core fragments as filter criterion. Cores are the remaining atoms when all side chains have been removed (Figure 3). According the definition of a core, molecules can contain either a single core or no core at all.

If the input molecule has no rings then a value of zero is returned and the specified *minimum* and *maximum* parameters determine whether the molecule will be passed or not.

In the following example, only molecules having a single core would be passed:

```
CORES 1 1
```

ATOMS_IN_CORE

Specifies a limit on the number of atoms in the core of the molecule. Hydrogen atoms are excluded from the calculation.

When the molecule does not contain a core, then a value of zero is returned and filtering is not performed.

In the following example, only molecules with either no core, or molecules with a core consisting of exactly 22 atoms would be passed:

```
ATOMS_IN_CORE 22 22
```

CORE_FRACTION

Specifies the fraction of core atoms over the total atom count as filter criterion. Hydrogen atoms are excluded from the calculation.

This criterion returns zero if the molecule contains no cores, and the filtering is applied according the limits specified as criteria.

When the molecule contains no non-hydrogen atoms, a value of zero is returned and the molecule will be marked to fail for this criterion.

In the following example, molecules of which at least half of its atoms are core atoms are passed through:

```
CORE_FRACTION 0.5 *
```

BRIDGES

Specifies the total number of bridge fragments as filter criterion. Hydrogen atoms are excluded. Bridges are calculated as the remaining atoms when all sidechains and ring systems have been removed (Figure 3).

If the input molecule has no rings then a value of zero is returned and the specified *minimum* and *maximum* parameters determine whether the molecule will be passed or not.

In the following example, only molecules having two or three bridges would be passed successfully:

```
BRIDGES 2 3
```

ATOMS_IN_SMALLEST_BRIDGE

Specifies a limit on the number of atoms in the smallest bridge of the molecule. Hydrogen atoms are not counted.

When the molecule does not contain a bridge a value of zero is returned and filtering is not performed.

In the following example, only molecules with either no bridge, or molecules in which the smallest bridge contains between one and three atoms would be passed:

```
ATOMS_IN_SMALLEST_BRIDGE 1 3
```

ATOMS_IN_LARGEST_BRIDGE

Specifies a limit on the number of atoms in the largest bridge of the molecule. Hydrogen atoms are not counted.

When the molecule does not contain a bridge a value of zero is returned and filtering is not performed.

In the following example, only molecules with either no bridge, or molecules in which the largest bridge contains between one and three atoms would be passed:

```
ATOMS_IN_LARGEST_BRIDGE 1 3
```

BRIDGE_FRACTION

Specifies the fraction of bridge atoms over the total atom count as filter criterion. Hydrogen atoms are excluded.

A value of zero is returned when the molecule contains no bridges, and the filtering is applied according the limits specified as parameters.

When the molecule contains no non-hydrogen atoms, a value of zero is returned and the molecule will be marked to fail for this criterion.

In the following example, molecules of which at least half of its atoms are bridge atoms are passed through:

```
BRIDGE_FRACTION 0.5 *
```

3.5. *Physical property rules*

Physical property rules specify limits on a number of physical properties. Physical property rules are specified in a manner identical to the topological property rules, *i.e.* using the appropriate *keyword* followed by a *minimum* and a *maximum* limit for the particular property:

```
KEYWORD minimum maximum
```

In the current version of 'sieve', eight specific physical property rules have been implemented:

- MOLWT
- LOGP
- LOGS
- TPSA
- LIPINSKI_VIOLATIONS
- ANDREWS_ENERGY
- LIGAND_EFFICIENCY
- ABSORPTION

For each of these eight physical property rules, only one of each is allowed in the filter file. If more than one identical rule keyword is provided, only the first keyword will be processed correctly and a message will be printed to warn the user that more than one identical keyword has been encountered. All the information provided by the subsequent identical rules in the filter file is neglected.

When the program is run in tabulate mode (with the '`--tab`' command line option) to have only the properties calculated but not filtered, then the *minimum* and *maximum* limits are not required and need not to be specified.

In the following sections, the eight physical property rules are detailed.

MOLWT

Specifies the molecular weight as filter criterion. In the following example, all molecules with a molecular weight ≥300 and ≤500 are accepted, while all other molecules are rejected:

```
MOLWT 300 500
```

LOGP

Specifies a limit on the calculated logP, which is the negative logarithm of the octanol/water distribution of the compound. Compounds with a preference for fatty environments will have a large logP, while the logP of more polar compounds will be lower or negative. The algorithm for calculating the logP is based on Silicos' implementation as described in Section 5.1.

In the following example, only molecules with a calculated logP ≤5 are passed through:

```
LOGP * 5
```

LOGS

Specifies a limit on the calculated logS, which is the negative logarithm of the water solubility of the compound. Soluble compounds will have a large logS, while rather insoluble compounds will have a small or negative logS. The algorithm for calculating the logP is based on a proprietary implementation of Silicos and is explained in detail in Section 5.2.

In the following example, only molecules with a calculated logS ≤-5 are passed through:

```
LOGS * -5
```

TPSA

Specifies a limit on the topological polar surface area. The algorithm for calculating the topological polar surface area is based on the Ertl implementation,[1] and is expressed in squared Å.

In the following example, only molecules with a calculated topological polar surface area ≤150 are passed through:

```
TPSA 0 150
```

LIPINSKI_VIOLATIONS

This parameter specifies a limit on the number of violations against Lipinski's rule-of-five. This rule-of-five is defined as:

- Number of Lipinski's hydrogen bond acceptors should be ≤10 (calculated with the `LIPINSKI_ACCEPTORS` keyword);
- Number of Lipinski's hydrogen bond donors should be ≤5 (calculated with the `LIPINSKI_DONORS` keyword);
- Calculated logP should be ≤5 (calculated with the `LOGP` keyword);
- Molecular weight should be ≤500 (calculated with the `MOLWT` keyword).

In the following example, only molecules with a maximum of one violation against Lipinski's rule-of-five are passed through:

```
LIPINSKI_VIOLATIONS 0 1
```

ANDREWS_ENERGY

This parameter specifies limits on the predicted molecular binding energy calculated according a modified version of the method of Andrews and coworkers[2].

[1] Ertl, P. *et al.* (2000) 'Fast calculation of molecular polar surface area as a sum of fragment-based contributions and its application to the prediction of drug transport properties', *J. Med. Chem.* **43**, 3714-3717.

When using this filter parameter, it is important that the correct ionization state of the molecules is used. For example, as the Andrews' energy depends heavily on the correct protonation state of primary amines, carboxylic acids and phosphate groups, care should be taken that these moieties are input in their correct ionization form. 'Sieve' does not take care of that automatically.

In the following example, only molecules with an Andrews' energy between 10 and 15 are passed through:

```
ANDREWS_ENERGY 10 15
```

LIGAND_EFFICIENCY

The ligand efficiency of a molecule is normally defined as being the total binding energy (expressed in kcal/mol units) for a given protein target divided by the number of non-hydrogen atom of the ligand. The ligand efficiency provided in this context is the maximal value that might be expected for a given compound in case of optimal interaction with the protein. For implementation details, see Section 5.4.

In the following example, only molecules with a predicted ligand efficiency higher than 0.37 (the mean of the training set) are passed successfully through:

```
LIGAND_EFFICIENCY 0.37 *
```

ABSORPTION

This parameter specifies a limit on the predicted passive intestinal absorption (PAI) of drugs. The PAI is calculated according a paper of Egan and coworkers, and is based on a multivariate analysis of the molecular topological polar surface area and the calculated logP.[3] The calculated absorption property returns only two possible values:

- '1' ('true') if the predicted PAI for the compound is higher than 90%. In the original publication,[3] these compounds are labeled as 'WAbs' ('well-absorbed');
- '0' ('false') if the predicted PAI for the compound is lower than 30%. This corresponds to the 'PAbs' ('poorly-absorbed') compounds of the original publication.[3]

In the following example, only molecules, which are predicted, to be well absorbed will be passed through:

```
ABSORPTION 1 1
```

To retrieve only compounds that are not passively absorbed, the following statement may be used:

```
ABSORPTION * 0
```

[2] Andrews, P.R.; Craik, D.J.; Martin, J.L. (1984) 'Functional group contributions to drug-receptor interactions', *J. Med. Chem.* **27**, 1648-1657.

[3] Egan, W.J. *et al.* (2000) 'Prediction of drug absorption using multivariate statistics', *J. Med. Chem.* **43**, 3867-3877.

which is similar to:

```
ABSORPTION 0 0
```

3.6. *Fragment and similarity filters*

Fragment rules are specifications to define limits on the presence of user-definable molecular substructures within the input molecules. These substructures are defined by means of a smarts pattern, and the user can put limits on the number of occurrences that these particular fragments are allowed in each molecule. When counting the number of occurrences of a particular substructure in a target molecule, only unique matches are taken into account. A unique match is defined as one that does not cover the identical atoms that a previous match has covered. For instance, searching for the 'c1ccccc1' pattern in phenol will only yield one unique match.

In addition to specifying limits on the number of substructures present in the input file molecules, the user can also define similarity rules to specify criteria on the required minimum or maximum Tanimoto similarity to a reference fragment or reference molecule.

If a match has been found between the molecule and a fragment or similarity rule, then the molecule is passed for this criterion. If a match cannot be found between the molecule and the fragment or similarity rule, the molecule is flagged to fail for this rule. Multiple fragment rules, each specifying different fragment or similarity cutoffs, may be specified. In this case, each of the different rules will be checked for a valid match against each of the input molecules. A molecule is only passed successfully when it succeeds for *all* of the specified rules. This is different to the implementation of the `TITLE` keyword where multiple keywords may also be provided, but where a molecule is passed successfully when matching *at least one* of the title keywords.

In the current version of Sieve, three fragment and similarity rules have been implemented:

- FRAGMENT
- SIMILARITY
- SIMILARITY_STACK

In the following sections, these three fragment property rules are described in more detail.

FRAGMENT

The general syntax of fragment rule is:

```
FRAGMENT name smarts minimum maximum
```

`FRAGMENT` is the required keyword to specify a fragment rule. The *name* field specifies a user-definable name for the fragment, and the *smarts* field specifies the substructure in smarts terminology. The *minimum* and *maximum* limits define the allowed number of occurrences of the specific fragment in the molecule. These latter two fields are not required when the program is run in tabulate mode ('`--tab`' command line option).

In order to filter out molecules having less than one and more than two substituted phenyl rings, the following fragment rule could be specified:

```
FRAGMENT phenyl c1ccccc1 1 2
```

Unsubstituted phenyl rings are specified as:

```
FRAGMENT phenyl c1[cH][cH][cH][cH][cH]1 1 2
```

Filtering out all ester functionalities would required a fragment definition like this:

```
FRAGMENT ester [O;X2;H0][C;X3]=O 0 0
```

If an invalid smarts pattern is provided that the program cannot process, an error is written out and the program halts:

```
-> FRAGMENT  f1==============================
*** Open Babel Error  in SMARTSError
  SMARTS Error:
c1cxccc1
   ^

f1 "c1cxccc1": Failed parsing
```

As already mentioned, more than one fragment rule may be specified in the filter file. However, fragment rules having a name that is identical to an earlier defined rule name generate an error message.

SIMILARITY

The general syntax of similarity rules is:

```
SIMILARITY name smiles minimum maximum
```

`SIMILARITY` is the required keyword to specify a similarity rule. The *name* field specifies a user-definable name for the fragment, and the *smiles* field specifies the reference molecule or fragment in SMILES terminology. The *minimum* and *maximum* limits define the range of allowed Tanimoto similarity between the input molecules and the specified fragment.

When the program is run in tabulate mode (with the '`--tab`' command line option) to have only the similarity calculated but not filtered, then the *minimum* and *maximum* limits are not required and need not to be specified.

In order to reject molecules that have a Tanimoto similarity of less than 0.75 to a phenyl ring as reference, the following similarity rule could be specified:

```
SIMILARITY phenyl c1ccccc1 0.75 *
```

If an invalid smiles pattern is provided that the program cannot process, an error is written out and the program halts:

```
-> FRAGMENT  f1============================
*** Open Babel Error  in SMARTSError
  SMARTS Error:
c1cxccc1
   ^

f1 "c1cxccc1": Failed parsing
```

As already mentioned, more than one similarity rule may be specified in the filter file. However, similarity rules with names that are identical to earlier defined rules generate an error message.

SIMILARITY_STACK

The general syntax of similarity stack rules is:

```
SIMILARITY_STACK name smiles minimum maximum
```

`SIMILARITY_STACK` is the required keyword to specify a similarity stack rule. The *minimum* and *maximum* limits define the range of the required Tanimoto similarity between the input molecules and each of the molecules in the similarity stack. The *name* field specifies a user-definable name for the fragment, and the *smiles* field specifies the reference molecule or fragment in SMILES terminology.

With similarity stacks, the user can direct the program to evaluate the similarity of the input molecule against both a set of user-defined fragments and the molecules that have previously successfully passed the similarity stack filter criteria. The implementation of a similarity stack can be clarified using an example based on the following input parameters:

```
SIMILARITY_STACK benzene c1ccccc1 * 0.5
SIMILARITY_STACK phenol c1ccccc1O * 0.5
```

For this example, the similarity stack is initially loaded with both the 'benzene' and 'phenol' fragments as defined in the filter file. The required *minimum* and *maximum* Tanimoto cutoff criteria are set to 0.0 and 0.5, respectively, thereby focusing on diversity rather than similarity. The program starts by calculating the Tanimoto similarity between the first input molecule and each of the fragments on the similarity stack (in this example, the stack is loaded with the 'benzene' and 'phenol' fragments). If *all* of the calculated similarities between the input molecule and each of the fragments on the similarity stack all obey the specified cutoff limits, the input molecule is added to the similarity stack and the molecule is written to the passed molecules file. However, if *at least one* of the calculated similarities fails against the specified cutoff limits, then the molecule is not added to the similarity stack and is written to the failed molecules file. These series of steps are repeated until all input molecules have been processed accordingly.

If an invalid SMILES pattern is provided that the program cannot process, an error is written out and the program halts:

```
-> FRAGMENT  f1============================
*** Open Babel Error  in SMARTSError
  SMARTS Error:
c1cxccc1
   ^
```

```
f1 "c1cxccc1": Failed parsing
```

Multiple SIMILARITY_STACK rules with the same name will lead to an error message with subsequent halt of the program.

In the case when different cutoff values have been specified on the different SIMILARITY_STACK lines, then the values of the last SIMILARITY_STACK specification are taken as the limits for the newly created molecules on the similarity stack. In the example below, the minimum and maximum Tanimoto cutoff values for all new molecules that are added to the stack during the run will be set to 0.1 and 0.4, respectively, since these are the limits that were specified at the last SIMILARITY_STACK line. This means that, in order for a input molecule to get passed through the filter criteria, the molecule should have a Tanimoto similarity between 0 and 0.5 for benzene, and a similarity between 0.1 and 0.4 for phenol and all other new molecules on the stack.

```
SIMILARITY_STACK benzene c1ccccc1 * 0.5
SIMILARITY_STACK phenol c1ccccc1O 0.1 0.4
```

Similarity stack calculations are not performed when the program is run in tabulate mode (‘--tab’).

3.7. *Distance filters*

Distance rules define limits on the three-dimensional distances between a set of user-definable fragments or patterns. These patterns are defined by means of a SMARTS definition. Distance rules are fully specified using the combination of two sets of keywords, namely the PATTERN and DISTANCE keywords. If a DISTANCE keyword is specified, then at least two PATTERN keyword should be specified as well. However, PATTERN keywords may be specified without a DISTANCE keyword. In this case, only the presence of the specified patterns is checked without actually taking into account any distance limits.

Distance filters are not applied when ‘sieve’ is run in tabulate mode.

PATTERN

The general syntax of the PATTERN rule is:

```
PATTERN name smarts [center_index_1 … center_index_n]
```

PATTERN is the required keyword to specify a pattern rule. The *name* field specifies a user-definable name for the pattern, and the *smarts* field specifies the substructure in SMARTS terminology. The optional *center_index_1* up to *center_index_n* variables specify which atoms of the smarts pattern should be used to calculate the geometrical center of the pattern. Counting starts from 1, so the first atom in the *smarts* definition has index 1, and the last atom in the *smarts* string has an index that is equal to the number of atoms in the *smarts*. Indices lower than 1, or higher than the number of atoms in the *smarts*, will lead to an error and halt of the program. If the index variables are not provided, then the geometrical centrum of the entire *smarts* is used to calculate the distances.

In the following example a pattern is defined that matches phenyl rings. Since no optional indices are specified, the geometrical center is calculated from all six atoms:

```
PATTERN phenyl c1ccccc1
```

which is equal to specifying:

```
PATTERN phenyl c1ccccc1 1 2 3 4 5 6
```

By providing indices to the keyword line, one can specify which atoms should be included in the calculation of the geometrical center:

```
PATTERN phenyl c1ccccc1 1
```

Of course, this example should be used with care since it is unpredictable how the actual smarts pattern will be matched on a phenyl ring. A better example is the following:

```
PATTERN methoxy [OH]C 1
```

In this case, the geometrical center is put on the hydroxyl oxygen, while in the following specification the entire methoxy fragment (oxygen and carbon atom) is used to calculate the geometrical center from:

```
PATTERN methoxy [OH]C 1 2
```

which is equivalent to

```
PATTERN methoxy [OH]C
```

If an invalid smarts pattern is provided that the program cannot process, an error is written out and the program halts:

```
-> FRAGMENT  f1==============================
*** Open Babel Error  in SMARTSError
  SMARTS Error:
c1cxccc1
   ^

f1 "c1cxccc1": Failed parsing
```

Multiple `PATTERN` rules with the same name will lead to an error message with subsequent halt of the program.

When the defined pattern is not present in the molecule, then the molecule will not pass the filter and will be written to the failed molecules file.

DISTANCE

The general syntax of the `DISTANCE` rules is:

```
DISTANCE pattern1 pattern2 minimum maximum
```

`DISTANCE` is the required keyword to specify a distance rule. The *pattern1* and *pattern2* fields specify the two patterns for which the geometrical distance should be calculated, and this distance is compared to the specified *minimum* and *maximum* fields. If the actual distance is smaller than *minimum* or larger than *maximum* then this particular molecule is rejected and subsequently written to the failed molecules file. As with all other filter rules, the *minimum* and *maximum* values may be left unspecified by working with '*' characters.

When the `DISTANCE` keyword is referring to a undefined pattern name, an error message is generated and the program halts:

```
ERROR: DISTANCE line with undeclared pattern name 1: phen
```

The same happens when the first and second specified pattern names are referring to the same pattern:

```
ERROR: DISTANCE line with duplicate pattern names (1 == 2): phenyl phenyl
```

For more technical information on the implementation of the fitting and matching process, the reader is referred to Section 5.5.

3.8. *Sdf-data rules*

Sdf-data rules are specifically implemented to filter sdf-files based on the content and presence of certain property tags. In an sdf-file, these property tags are specified by means of a property field name and its corresponding value:

```
> <TAGNAME>
value
---- blank line ----
```

The property name starts with a '>' sign and the tag name itself is enclosed in angle brackets. A blank line terminates each property entry. Molecules can have multiple tags with the same name, although such a situation is rather uncommon.

If a match has been found between the molecule and a sdf-tag rules, then the molecule is passed for this criterion. If a match cannot be found between the molecule and a sdf-tag rule, the molecule is flagged to fail for this rule. Multiple sdf-tag rules, each specifying different sdf-tags, may be specified. In this case, each of the different rules will be checked for a valid match against each of the input molecules. A molecule is only passed successfully when it succeeds for *all* of the specified sdf-tag rules. This is different to the implementation of the TITLE keyword where multiple keywords may also be provided, but where a molecule is passed successfully when matching *at least one* of the title keywords.

In the current version of 'sieve', two sdf-tag rules have been implemented:

- SDFTAG
- SDFTAG_VALUE

SDFTAG

This rule specifies the number of allowed property tags with the given name. The actual value of the property tag is not used in this rule:

```
SDFTAG <reg.no> 1 6
```

The example above passes those molecules that contain between one and six `<reg.no>` property tags. Molecules in sdf-files with more than six or less than one `<reg.no>` property tags are filtered out. Please note that the property tag name should be enclosed by angle brackets. This allows the use of blanks in the property tag names.

SDFTAG_VALUE

This rule specifies limits on the actual value of the given property tag. The property tag name should be enclosed by angle brackets. This allows the use of blanks in the property tag names.

When run in standard non-tabulate mode, all values are treated and filtered as real numbers. If the tag-value cannot be converted to a real number, the compound gets flagged as being failed. Also, in the case the specified sdf-tag cannot be found for a given molecule, the molecule gets flagged as being failed.

When the program is run in tabulate mode ('`--tab`' command line option), the tag-value is printed out without any conversion applied. This means that in this case strings could also be output. In cases where the specified sdf-tag cannot be found for a given molecule, a value of zero is printed out.

The following example passes through all molecules in a sdf-file that contains `<reg.no>` property tags of which the actual value is between 1.0 and 6.5:

```
SDFTAG_VALUE <reg.no> 1 6.5
```

There is no check on the sdf-tag parameter itself, hence multiple `SDFTAG_VALUE` rules with identical sdf-tag specifications but different *min* and *max* values could be present in the filter file. In those cases, if one of these identical rules fails the filter criteria, then the molecule is flagged as failed. Only when all sdf-tag rules are passed successfully then the molecule is flagged as passed.

4. Example filter file

4.1. CMC likeness

The following example provides the physicochemical definitions for compounds with CMC-like properties. The CMC-v2007 was used and only molecules with 1-59 non-hydrogen atoms of H, C, O, N, F, Cl, I, Br, S, or P were included. All molecules were ionized according a pH of 7.0. The spread was calculated using three times the standard deviation of the population:

```
ONLY_ELEMENTS                H C O N F Cl Br I S P
ATOMS                                   1        59
CARBONS                                 1        40
HETERO_ATOMS                            1        19
HETERO_CARBON_RATIO                   0.0       1.1
HALIDES                                 0         4
HALIDE_FRACTION                       0.0       0.2
ROTATABLE_BONDS                         0        20
RIGID_BONDS                             0        46
FLEXIBILITY                           0.0       0.7
CHIRAL_CENTERS                          0        10
HBOND_ACCEPTORS                         0        13
HBOND_DONORS                            0         9
LIPINSKI_ACCEPTORS                      0        15
LIPINSKI_DONORS                         0         9
FORMAL_CHARGES                          0         4
TOTAL_FORMAL_CHARGE                    -3         3
RINGSYSTEMS                             0         4
ATOMS_IN_SMALLEST_RING                  3         7
ATOMS_IN_SMALLEST_RINGSYSTEM            3        20
ATOMS_IN_LARGEST_RINGSYSTEM             3        20
RING_FRACTION                         0.2       1.0
SIDECHAINS                              0        10
ATOMS_IN_SMALLEST_SIDECHAIN             1         9
ATOMS_IN_LARGEST_SIDECHAIN              1        16
SIDECHAIN_FRACTION                    0.0       0.8
CORES                                   0         1
ATOMS_IN_CORE                           3        35
CORE_FRACTION                         0.2       1.0
BRIDGES                                 0         2
ATOMS_IN_SMALLEST_BRIDGE                1         7
ATOMS_IN_LARGEST_BRIDGE                 1         7
BRIDGE_FRACTION                      0.00      0.25
MOLWT                                   0       750
LOGP                                  -11        14
LOGS                                  -11         3
TPSA                                    0       240
LIPINSKI_VIOLATIONS                     0         3
ANDREWS_ENERGY                        -22        42
LIGAND_EFFICIENCY                     0.0       0.8
ABSORPTION                              0         1
```

5. Physical properties

5.1. LogP

The implementation of the logP prediction in 'sieve' is based on a proprietary fragment- and property-based method. For this, the logP is calculated from:

$$\text{logP} = \text{sum}(p_i * w_i) \text{ with } i = 0 \text{ to } 34$$

with p_i being the value of property i, and w_i being the corresponding weight. A list of the defined fragments and properties with corresponding weights is provided in Table 2.

Table 2. Summary of the properties with corresponding factors for the calculation of logP. The properties are indicated by the corresponding filter name or by a smarts pattern.

i	**Property**	**w**$_i$
0	Count of F	+1.570
1	Count of Cl	+1.688
2	Count of [Br,I]	+1.452
3	Count of [$([NH2;!+;X3;v3]),$([NH3;+;X4;v4])]	-1.531
4	Count of [$([NH1;!+;X3;v3]),$([NH2;+;X4;v4])]	-0.311
5	Count of [$([NH1;!+;X2;v3]),$([NH2;+;X3;v4])]	-0.594
6	Count of [$([NH0;!+;X2;v3]),$([NH1;+;X3;v4])]	+0.545
7	Count of [$([NH0;!+;X1;v3]),$([NH1;+;X2;v4])]	-0.820
8	Count of [$([NH0;+]);!$([N+](=O)[O-]);!$([N+]=[N-])]	-2.424
9	Count of [N+]=[N-]	+1.686
10	Count of [$([n;!+]),$([nH;+])]	-0.065
11	Count of [nH0;+]	-1.357
12	Count of [OH0;X2;v2]	+0.215
13	Count of [$([OH1;!-;X2;v2]),$([OH0;-;X1]);!$([N+](=O)[O-]); !$(C(=O)[OH]);!$(C(=O)[O-])]	-0.857
14	Count of [$([OH0;X1;v2]);!$([N+](=O)[O-]);!$(S=O);!$(P=O); !$(C(=O)[OH]);!$(C(=O)[O-])]	-0.214
15	Count of O~P(~O)(~O)~O	+0.665
16	Count of [$(P);!$(O~P(~O)(~O)~O)]	+0.841
17	Count of [SH0;X2;v2]	-0.547
18	Count of [$([SH1;X2;v2]),$([SH0;-])]	-1.609
19	Count of O=S	+0.351
20	Count of [CH3;!R]	+0.128
21	Count of [CH2;!R]	+0.215
22	Count of [CH1;!R]	+0.132
23	Count of [CH0;!R]	+0.065
24	Count of [cH0]	+0.139
25	Count of [cH1]	-0.080
26	Count of [CH1;R]	-0.197

27	(Intercept)	-0.717
28	ATOMS	+0.142
29	HETERO_ATOMS	-1.600
30	RINGSYSTEMS	+0.246
31	HETERO_CARBON_RATIO	+0.538
32	MOLWT	+0.006
33	RING_FRACTION	+1.736
34	TPSA	+0.076

The coefficients were obtained from least squares fitting against 23,455 experimental logP values from the PHYSPROP database. Only compounds with a molecular weight between 200 and 600 Da and with experimental logP values between -2.6 and 8.1 were used in the fitting process to limit the contribution of outliers and less drug-like molecules.

A plot of the calculated versus the experimental logP values is given in Figure 4. Residual standard error is 1.17 on 23,415 degrees of freedom.

Figure 4. Scatter plot of the entire set of experimental logP values versus calculated logP. Only compounds with a molecular weight between 200 and 600 Da were used in this scatter plot.

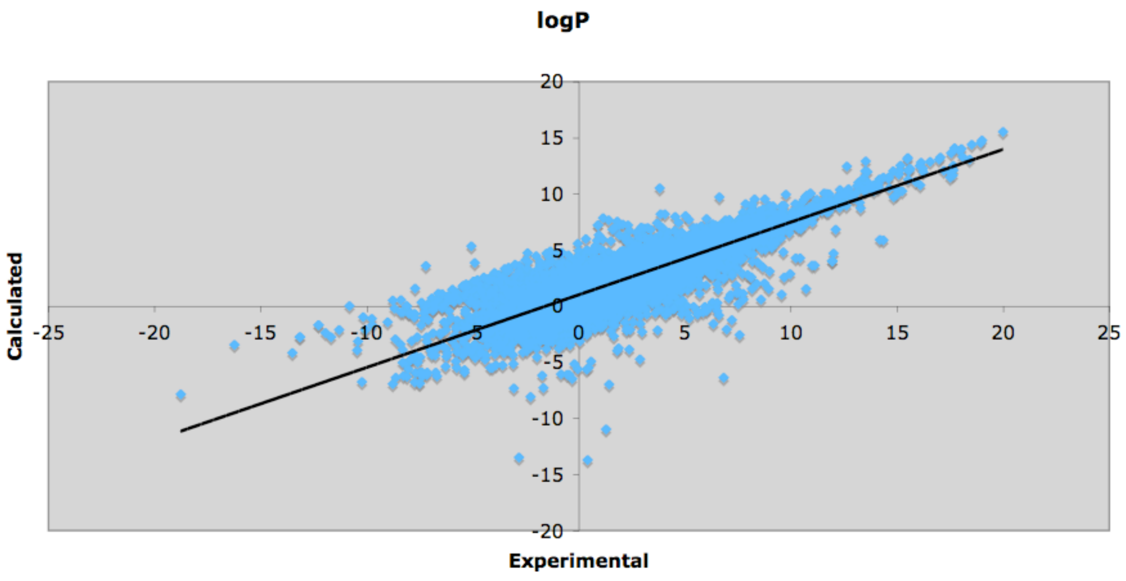
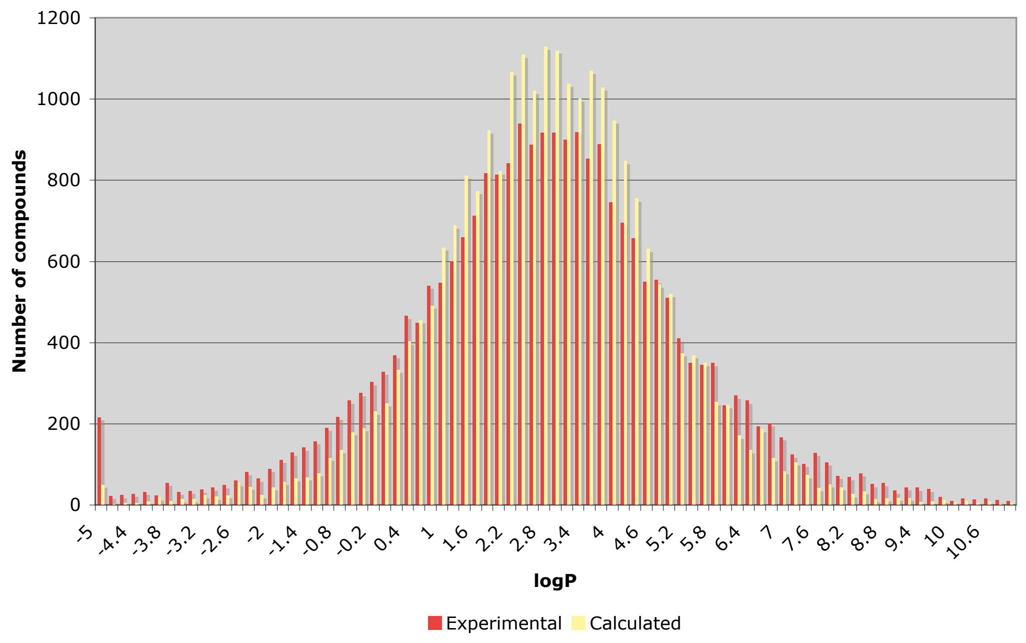


Figure 5. Comparison of the distribution of the experimental and calculated logP values for the entire set of 24,701 compounds. The means of the experimental and calculated distributions are 2.75 and 2.79, respectively, while the corresponding standard deviations are respectively 2.69 and 2.04.



5.2. LogS

The implementation of the logS prediction in 'sieve' is based on a simple fragment-based method. For this, the *logS* is calculated from:

$$logS = 0.898 + 0.104 \text{ sqrt}(MOLWT) + w_i c_i$$

with w_i and c_i being the respective weights and counts for fragment i, and *MOLWT* the molecular weight of the compound. A list of the defined fragments with corresponding weights is provided in Table 3.

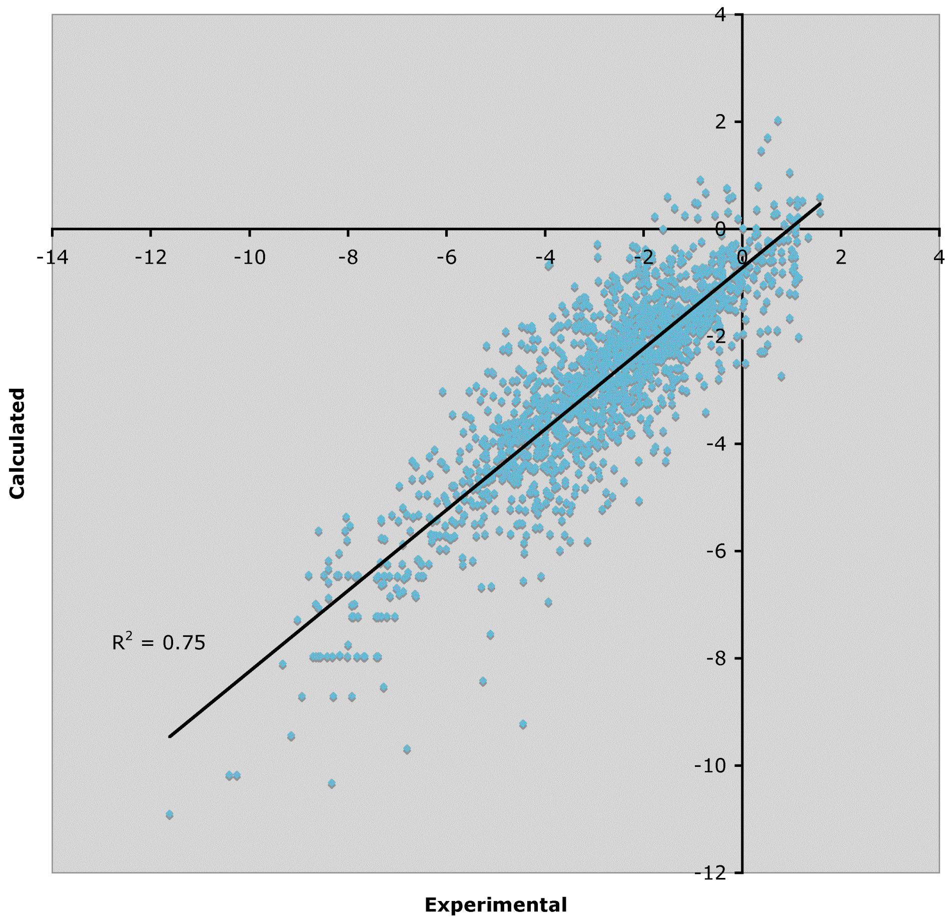
Table 3. Summary of the applied fragments with corresponding factors for the calculation of logS.

i	Property	w_i
0	Count of [NH0;X3;v3]	0.715
1	Count of [NH2;X3;v3]	0.411
2	Count of [nH0;X3]	0.825
3	Count of [OH0;X2;v2]	0.315
4	Count of [OH0;X1;v2]	0.148
5	Count of [OH1;X2;v2]	0.630
6	Count of [CH2;!R]	-0.356
7	Count of [CH3;!R]	-0.339
8	Count of [CH0;R]	-0.219
9	Count of [CH2;R]	-0.231
10	Count of [ch0]	-0.376
11	Count of [ch1]	-0.224
12	Count of F	-0.217

i	Property	w_i
13	Count of Cl	-0.497
14	Count of Br	-0.580
15	Count of I	-0.515

A plot of the calculated versus the experimental logS values is given in Figure 6.

Figure 6. Scatter plot of the experimental versus calculated logS values.



5.3. *TPSA*

The topological polar surface area implementation in 'sieve' is based on the work of Peter Ertl and coworkers.[4]

5.4. *Ligand efficiency*

The ligand efficiency in this version of 'sieve' is a proprietary version calculated using a fragment-based approach, in which the coefficients were obtained by fitting against 857 experimentally determined binding free

[4] Ertl, P.; Rohde, B.; Selzer, P. (2000) 'Fast calculation of molecular polar surface area as a sum of fragment-based contributions and its application to the prediction of drug transport properties', *J. Med. Chem.* **43**, 3714-3717.

energies (http://www.bindingmoad.org/) and the calculated ligand efficiencies thereof.

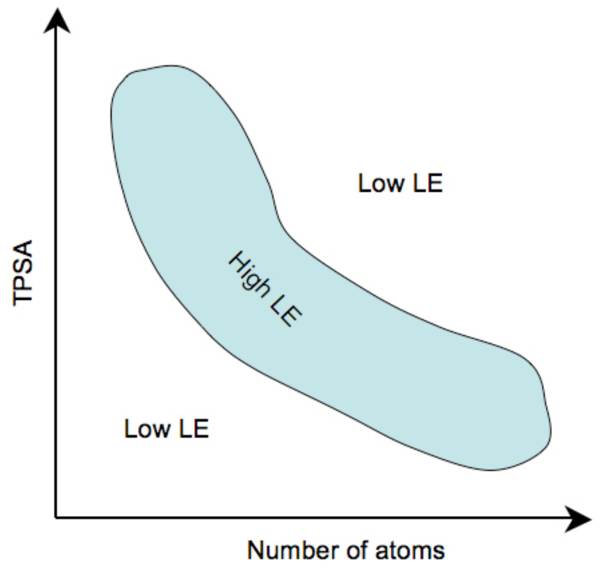
The predicted ligand efficiency is calculated from the following equation:

$$LE = -0.0130\, HBA + 0.0849\, HCR + 0.0036\, TPSA - 0.0140\, TPSA/\ln(ATOMS) - 0.3156\, \ln(ATOMS)$$

with *LE* being the calculated ligand efficiency, *HBA* the number of hydrogen bond acceptors as calculated by the HBOND_ACCEPTORS filter, *HCR* the hetero/carbon ratio as calculated by the HETERO_CARBON_RATIO filter, *TPSA* the topological polar surface area as calculated by the TPSA filter, and *ATOMS* being the total number of non-hydrogen atoms as calculated by the ATOMS filter.

Analysis of this regression equation shows that the ligand efficiency is influenced by a subtle equilibrium between the number of atoms and the topological polar surface area. For smaller ligands, optimal ligand efficiency is achieved when the topological polar surface area is increased, while for larger compounds the reverse is true (Figure 7). For optimal ligand efficiency, small fragments should be as polar as possible, while the larger, more drug-like molecules, should be rather lipophilic. In addition to this equilibrium between the number of atoms and topological polar surface area, ligand efficiency is also increased by a larger hetero atom/carbon ratio, but increasing the number of hydrogen bond donor atoms rather than increasing the acceptor atoms should only increase this hetero atom/carbon ratio.

Figure 7. Subtle equilibrium between the topological polar surface area (TPSA), the number of atoms, and the ligand efficiency (LE).



The experimental versus predicted ligand efficiencies are plotted in Figure 8. Distributions of the two populations are shown in Figure 9.

Figure 8. Experimental versus predicted ligand efficiencies as calculated by the method in Sieve. Residual standard error is 0.1075 on 851 degrees of freedom. R^2 equals 0.42.

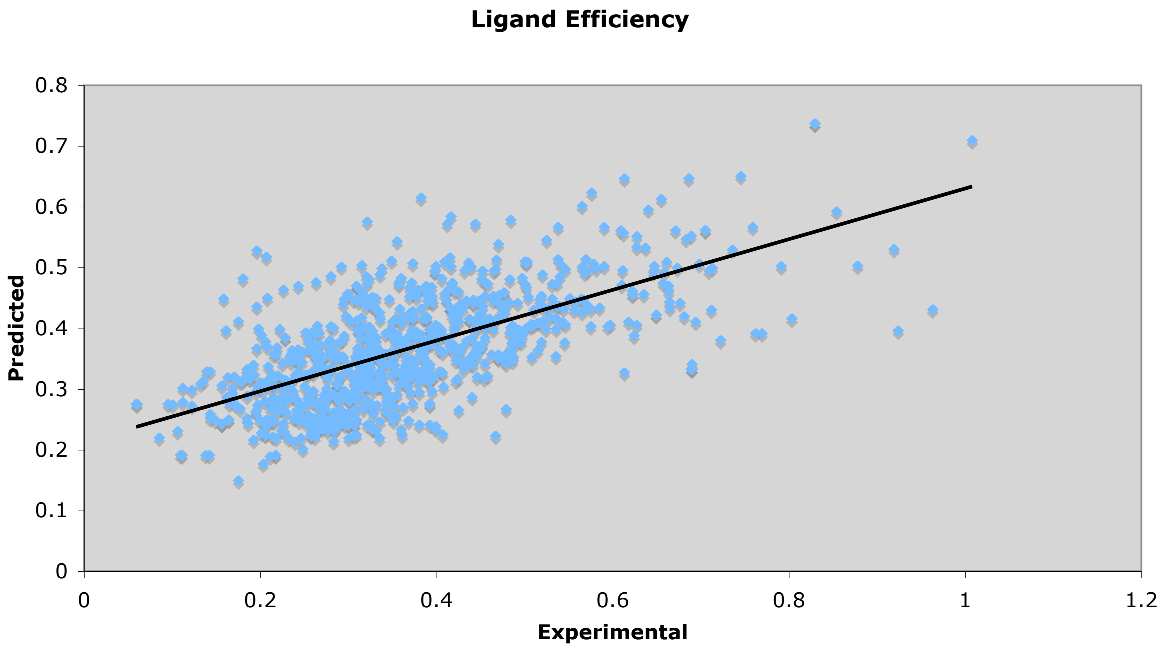
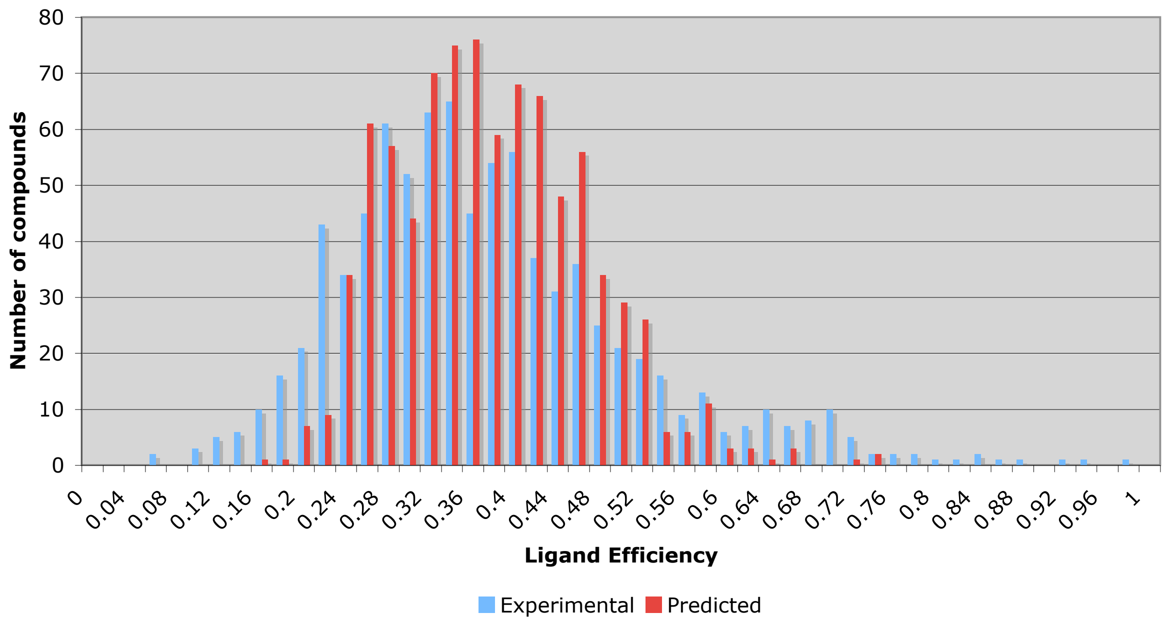
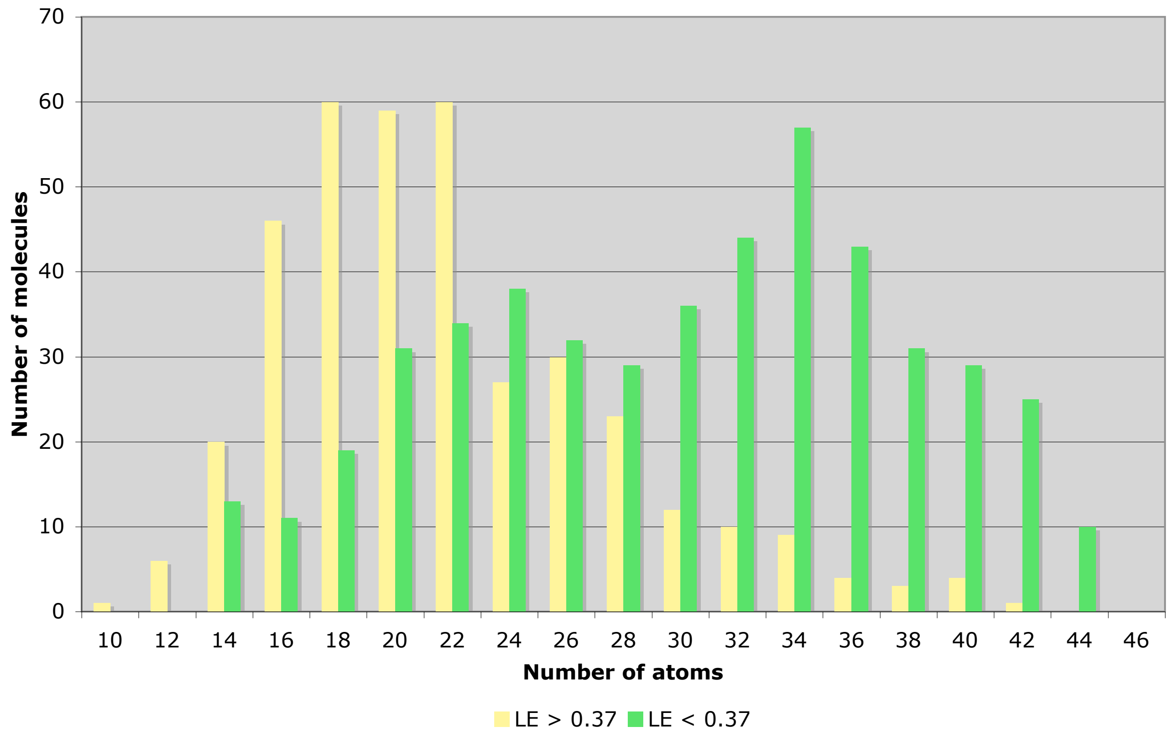


Figure 9. Distribution of the predicted and calculated ligand efficiencies that were used in the training of the model. The mean of the experimental distribution is 0.37 with a standard deviation of 0.14, while the mean of the predicted distribution is also 0.37 with a standard deviation of 0.15.



An alternative approach to extract compounds that might have an intrinsic large ligand efficiency is to focus on the number of heavy atoms. This approach has the advantage that it is not based on an underlying mathematical model that has been trained on a rather limited dataset with possibly restricted predictive power. The distribution of the number of atoms for molecules having a smaller or larger experimental ligand efficiency than the mean is shown in Figure 10. From this figure, it is clear that there is significant difference to be noted between the two distributions. For the molecules with an experimental ligand efficiency larger than the mean ligand efficiency (> 0.37), the distribution peaks around 18-22 heavy atoms, while for the molecules with ligand efficiency < 0.37 the maximum is around 34 heavy atoms. A total of 28 atoms seems to be the optimal cutoff value to separate both distributions.

Figure 10. Histogram showing the distribution of the number of atoms as a function of the experimental ligand efficiency (LE). The yellow bars is the distribution of all molecules having an experimental LE that is larger than the mean (> 0.37), while the green distribution is of all molecules with an experimental LE smaller than 0.37.



Based on these observations, an interesting alternative approach could be to include a filter based on the number of atoms, for example keeping only those molecules in which the total number of heavy atoms is between 12 and 28. The accuracy of this approach in extracting molecules having a ligand efficiency larger than 0.37 is somewhat less than with the `LIGAND_EFFICIENCY` filter approach, but the predictive power might be comparable (Table 4).

Table 4. Comparison between the two approaches to estimate intrinsic ligand efficiency.

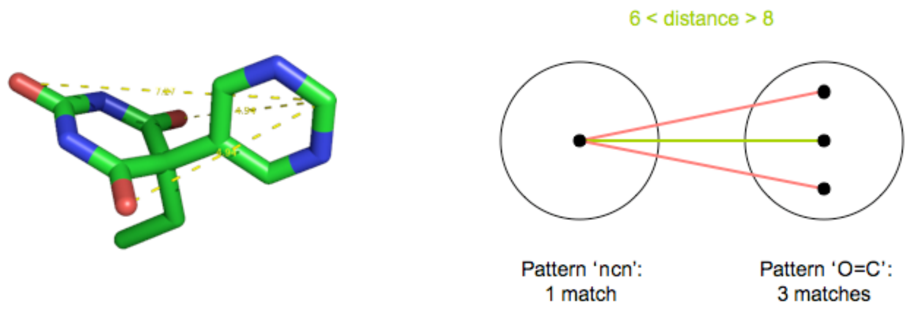
		Predicted LE			Number of atoms	
		< 0.37	> 0.37		> 27	< 27
LE	< 0.37	**44.3%**	13.0%		**34.9%**	21.4%
	> 0.37	9.8%	**32.6%**		5.8%	**35.5%**

5.5. Distance filters

Distance filters are implemented as geometrical distance constraints between user-specified patterns. These patterns are implemented as smarts queries. Since a given pattern may have more than one occurrence in a given molecule, a specific procedure is needed to ensure consistency between the different distance constraints. In the following sections, this procedure is highlighted in more detail. For consistency, the following nomenclature is used:

- 'pattern': a smarts representation of the user-defined substructure;
- 'match': a match is found when a pattern is found in a given molecule. There could be multiple matches per pattern, or none (Figure 11).

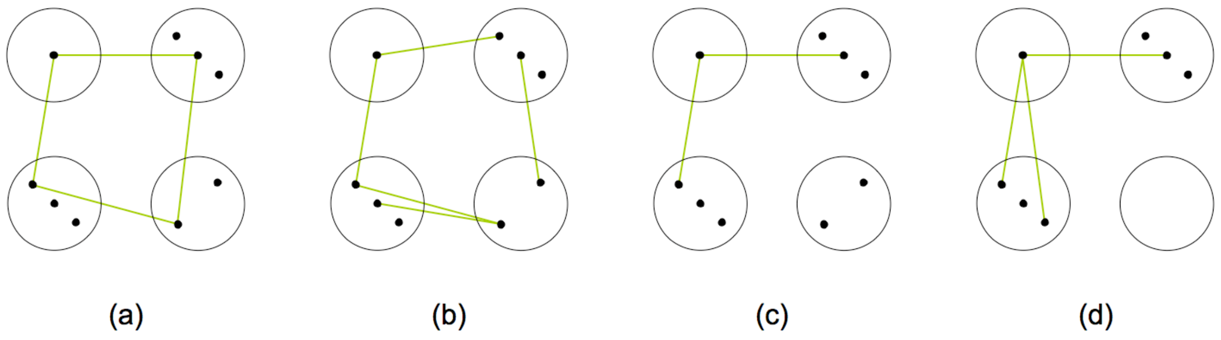
Figure 11. Example showing the relation between pattern and matches. In the example given in this figure, the molecule has three occurrences of a carboxyl group, and therefore three matches of the 'C=O' pattern. In addition, a single match was found for the 'ncn' pattern. Between the matches of both patterns, 1 x 3 three distance can be calculated, and of which only one distance falls within the limits of 6-8 Å. Since at least one distance is fitting within the user-defined limits, the distance criterion can be considered to fulfill the criteria.



A distance constraint between two patterns is obeyed if the distance between the two patterns falls within the defined limits of the distance constraint. When multiple matches are found for a given pattern, it is sufficient for a distance to be obeyed if the distance between at least one of the matches is falling within the limits (Figure 11).

With three or more patterns per molecule, the situation becomes slightly more complicated. In this case, only the matches that obey all distances are considered in the final criteria (Figure 12).

Figure 12. The four distance constraints are along the edges of the squares (not shown), and the distance that obey the user-defined criteria are shown in green. (a) Example that fulfills are distance constraints and therefore considered to be a 'pass'. (b) Example in which the distance constraints between the four patterns are fulfilled, but because non-consistent matches are involved it is considered to be a 'fail'. (c) Example of a 'fail' since two of the four distance constraints are not fulfilled. (d) Another example of a 'fail' since for one pattern there are no matches.



6. Revision history

6.1. *Version 1.9*

No additional rules have been added, only bug fixing:-

- Corrected smarts implementation errors in the TPSA calculations;
- Corrected the generation of incorrect labels when semantical errors where used for the ONLY_ELEMENTS and EXCLUDED_ELEMENTS rules.

6.2. *Version 2.0*

Major rewrite of the code. A number of modifications have been introduced:

- The filters that are involved in ring detection have been modified since the OEChem API does not support a SSSR-detection algorithm;
- Modification of the semantics for FRAGMENT, SIMILARITY, and SIMILARITY_STACK filters;
- Added the LIGAND_EFFICIENCY filter;
- Modification of the command line arguments;
- Modification of the logP, logS, and Andrews energy implementations.

6.3. *Version 2.1*

Addition of the distance filter criteria, with implementation of the PATTERN and DISTANCE keywords.

6.4. *Version 2.2*

- Added the '`--noLog`' command line option;
- Removed the requirement that a fail file should always be provided when run in normal mode.
- Added the SDFTAG and SDFTAG_VALUE keywords.

6.5. *Version 2.3*

- Added the TITLE keyword.
- Added the BONDS keyword.
- Fixed some potential memory leak issues in logP, logS, and TPSA properties

6.6. *Version 3.0*

- Porting to the Open Babel C++ API

- Added the '`--rename`' command-line option.

6.6.1. *Version 3.0.1*

- Included suggestions by Chris Morley: new fingerprint.h file, solved an error regarding the '`--rename`' command-line option, and corrected some spelling errors.

6.6.2. *Version 3.0.2*

- Code optimisation of the read and write functions.

6.6.3. *Version 3.0.3*

- Added '#include <stdlib.h>' lines to define the exit() function.

6.6.4. *Version 3.0.4*

- Included a StripSalts() function (defined in 'stripSalts.cpp' and 'stripSalts.h') to temporarily replace the OBMol::StripSalts() method from Open Babel 2.3.0. This method contains a bug that causes the system to crash when OBSmartsPattern::Match() is called subsequently to calling the OBMol::StripSalts() method. The bug has been fixed as of Open Babel trunk revision 4289 (November 3, 2010), but is not yet included in the 2.3.0 stable release. As soon as a new stable release gets out, then the function can again be replaced by the original OBMol::StripSalts() method, and the correspodning stripSalts.* files can be removed.

- Added a 'FAQ.txt' file containing answers to FAQ's picked up from the Open babel mailing list.