
FILTER
Database preparation
Property calculation

version 2.0.2

OpenEye Scientific Software, Inc.

March 24, 2009

9 Bisbee Court, Suite D
Santa Fe, NM 87508
www.eyesopen.com
support@eyesopen.com

Copyright © 1997-2005 by OpenEye Scientific Software, Santa Fe, New Mexico. All rights reserved.

All rights reserved. This material contains proprietary information of OpenEye Scientific Software. Use of copyright notice is precautionary only and does not imply publication or disclosure.

The information supplied in this document is believed to be true but no liability is assumed for its use or the infringement of the rights of others resulting from its use. Information in this document is subject to change without notice and does not represent a commitment on the part of OpenEye Scientific Software.

This package is sold/licensed/distributed subject to the condition that it shall not, by way of trade or otherwise, be lent, re-sold, hired out or otherwise circulated without OpenEye Scientific Software's prior consent, in any form of packaging or cover other than that in which it was produced. No part of this manual or accompanying documentation, may be reproduced, stored in a retrieval system on optical or magnetic disk, tape, CD, DVD or other medium, or transmitted in any form or by any means, electronic, mechanical, photocopying recording or otherwise for any purpose other than for the purchaser's personal use without a legal agreement or other written permission granted by OpenEye.

This product should not be used in the planning, construction, maintenance, operation or use of any nuclear facility nor the flight, navigation or communication of aircraft or ground support equipment. OpenEye Scientific software, shall not be liable, in whole or in part, for any claims arising from such use, including death, bankruptcy or outbreak of war.

Windows is a registered trademark of Microsoft Corporation. Apple and Macintosh are registered trademarks of Apple Computer, Inc. AIX and IBM are registered trademarks of International Business Machines Corporation. UNIX is a registered trademark of the Open Group. RedHat is a registered trademark of RedHat, Inc. Linux is a registered trademark of Linus Torvalds. Alpha is a trademark of Digital Equipment Corporation. SPARC is a registered trademark of SPARC International Inc.

SYBYL is a registered trademark of TRIPOS, Inc. MDL is a registered trademark and ISIS is a trademark of MDL Information Systems, Inc. SMILES, SMARTS, and SMIRKS may be trademarks of Daylight Chemical Information Systems. Macromodel is a trademark of Schrödinger, Inc. Schrödinger, Inc may be a wholly owned subsidiary of the Columbia University, New York.

Python is a trademark of the Python Software Foundation.

Java is a trademark or registered trademark of Sun Microsystems, Inc. in the U.S. or other countries.

“The forefront of chemoinformatics” is a trademark of Daylight Chemical Information Systems, Inc.

Other products and software packages referenced in this document are trademarks and registered trademarks of their respective vendors or manufacturers.

CONTENTS

1	Introduction	1
1.1	Overview	1
1.2	History	1
1.3	Customization	2
1.4	Polar surface area	2
1.5	pKa normalization	3
1.6	LogP	3
1.7	LogS	3
1.8	Aggregators	4
1.9	Lipinski and Hydrogen-Bonds	4
1.10	Normalization	5
1.11	Reagent Selection	5
1.12	Pharmacokinetics	5
1.13	Dyes	5
1.14	Structural and Chemical features	6
1.15	Functional Groups	6
1.16	Accumulation of rules	6
1.17	Additional Filters	7
2	Theory	8
2.1	The Rant!	8
2.2	Filtering Principles	9
2.3	Variations of Filtering	9
3	Usage	11
3.1	Command Line Interface	11
3.2	Example executions	15
4	The Param File	17
5	The Filter Files	18
5.1	Filter file format	18

A	Release Notes	25
A.1	Filter 2.0	25
A.2	Filter 1.0	28
B	Lead-like filter	30
C	Drug-like filter	36
D	Example Substructure hits	42
	Bibliography	51

Introduction

1.1 Overview

Filter is a program for eliminating inappropriate or undesirable compounds from a large set before you begin to use them in modelling studies. Filter attempts to remove all of the compounds you wouldn't want to suggest to a medicinal chemist as a potential hit. Though there are many general principles, this exercise is obviously case dependent, depending on ease of the assay, intended target, personal bias of the modeller & medicinal chemist, strengths of your company, etc. To match this need, Filter comes with a default filter that encapsulates many of the standard filtering principles, such as removal of unstable, reactive and toxic moieties. In addition, Filter allows you to customize the filtering criteria to fit your needs.

Filter's criteria for passing or failing a given molecule fall into three categories: physical properties, atomic & functional-group content, and molecular graph topology. The simple physical properties include molecular weight, topological polar surface area (TPSA), logP, and aqueous solubility. The filters also include absolute and relative content of heteroatoms as well as limits on the number of a very wide variety of functional groups. The graph topology filters address issues concerning the number and size of ring systems, the flexibility of the molecule and the size and shape of non-ring chains.

Filter allows easy tabulation of all of the data it generates in filtering molecules. Filter can be used to generate a tab-separated file for easy import into a spreadsheet. This function will allow the user to combine the values dynamically for a variety of purposes, including, but not limited to, determining which filter values best fit their need.

As a convenience to users, the Filter product comes with a second application, molprop, that simply calculates the logP, logS and PSA of a molecule. Molprop can attach the data to the molecule as SDData (sdf format), as part of the comment (mol2, sdf format), as generic data (oeb format), or simply write it to a log file.

1.2 History

When filter began in 2000, it was designed simply to remove compounds with reactive or otherwise undesirable functional groups. Over the years, the understanding of lead-like and drug-like compound selection has advanced. In addition, with the publication of Lipinski's "Rule of 5" [1], more and more

pharmacokinetic properties have been pushed earlier into the virtual screening process.

In its current version, Filter still provides the basic functional group selection, but it now also has many additional database preparation features. It allows selection based upon several physical properties (MW, LogP, LogS, PSA), assay counter-indicators (aggregators and dyes), PK (ABS, Veber, Egan, Lipinski). It allows database preparation in the form of setting pKa states, normalizations (tautomers & dative or hypervalent states). Finally, it provides excellent cheminformatics quality-control in the form of valence-state validation, aromaticity perception, implicit hydrogen perception and bond-order perception. Filter is a one-stop database preparation tool aimed at generating databases suitable for high-throughput virtual screening.

Finally, we want to point out that in the virtual screening world, time is of the essence. Algorithms for preliminary database preparation should not take extremely large amounts of time. Because of this, we have taken that philosophy that all the calculations included in Filter should be 2D or graph-based algorithms. While this does occasionally limit the technology we are willing to include in Filter, we feel it allows us to deliver a product that is appropriate for the task of virtual-screening database preparation.

1.3 Customization

It is the nature of database compound selection that the filters to be used are both dependent on the task at hand as well as the experiences and expertise of the user and the audience. For this reason, the compound filters used by Filter are fully customizable. Not only can the cutoff values of any of the default filter parameters be modified to any value, but individual filters can be turned off altogether simply by removing or commenting-out the lines referring to those filters in the filter file. For instance, a user can easily eliminate all of the functional group filtering simply by removing all the RULE lines from a filter file. Alternately, if a user only wants to calculate LogP and filter based on a maximum of 5.75, a filter file containing only a single line "MAX_XLOGP 5.75" is sufficient.

1.4 Polar surface area

Filter implements the topological polar-surface area (TPSA) algorithm developed by Ertl et al [6]. In Ertl's publication, use of TPSA both with and without accounting for phosphorus and sulfur surface-area is reported. However, evidence shows that in most PK applications one is better off not counting the contributions of phosphorus and sulfur atoms toward the total TPSA for a molecule. Filter's implementation of TPSA allows either inclusion or exclusion of phosphorus and sulfur surface area with the default being not including it (see PSA_USE_SandP in the filter file). One should be aware that TPSA values are mildly sensitive to the protonation state of a molecule. If the pkaNorm parameter is false, the TPSA value is calculated using the input structure and if pkaNorm is true, the TPSA is calculated using the pKa normalized molecular structure.

1.5 pKa normalization

Filter has a rule-based system to set the ionization state of input molecules. This feature is controlled with the pKaNorm command-line parameter. If pKaNorm is true, the program attempts to set each molecule to its most energetically favorable ionization state for pH=7.4. The rule-based nature of this calculation allows it to be quite fast. Further, despite being rule-based, this approach takes into account many secondary charge interactions. While more advanced levels of theory can be found for predicting ionization states, we feel this method is very well suited to virtual-screening database preparation. However, for hit-to-lead or lead optimization we would recommend a higher level of theory for both primary pKa estimation (Hammet-Taft or experiment) and formal-charge to formal-charge interactions (Poisson-Boltzmann). Finally, given the level of detail at which it is practice to carry out calculations for a high-throughput, consistency in ones approach is a significant virtue.

1.6 LogP

Filter includes an implementation of the XLogP algorithm [5]. We chose the XLogP in part because its atom-type contribution allows calculation of the XLogP contribution of any fragment in a molecule and allows minimal corrections to a simple additive form to calculate the LogP of any molecule made from combinations of fragments. Further, though the method contains many many free parameters, its simple linear form allows for ready interpretation of the model and most of the parameters in the model make rational sense.

Unfortunately, when implementing the original algorithm as published, we found several difficult details. First, the internal-hydrogen bond term was calculated using a single 3D conformation. We found this both arbitrary and unnecessary. This arbitrary 3D calculation has been replaced with a 2D approach to recognition of common internal-hydrogen bonds. In our tests, this 2D method worked comparably to the published 3D algorithm. Next, we found the training set had a few subtle atom-type inconsistencies. We corrected both of these problems and refit to the original XLogP training data. This implementation gives results that are quite similar to the original XLogP algorithm, so we call it OEXLogP to distinguish it from the original method.

1.7 LogS

The work of Yalkowsky at Arizona [12] has resulted in what is now called the "generalized solvation equation." It states that the solubility of a compound can be broken into two steps, first the melting for the pure solid to pure liquid and second, the phase transfer from pure liquid into water. For many small organic molecules, this second step is somewhat related to LogP. Because of this relation we choose to explore the use of the XLogP atom-types in solubility prediction. The expectation was that this might provide an approximate though robust and fast method for calculating solubility. We fit the XLogP atom-types to a training set of nearly 1000 public solubilities. From this we derived a linear model for solubility. The model is extremely fast and is useful for classifying compounds as insoluble, poorly soluble, slightly soluble, moderately, soluble or very soluble. The model is notable for the difficulty it has predicting solubilities for compounds with ionizable groups. Further, it is not suitable for the PK

predictions that come late in a project. However, it is useful for eliminating compounds with severe solubility problems early in the virtual-screening process.

1.8 Aggregators

Recent work in the Shoichet lab [2, 3] has demonstrated the importance of small-molecule aggregation in medium and high-throughput assays. Because these small-molecule aggregates can sequester some proteins, they give the appearance of being active inhibitors. There are now several hundred published aggregators in addition to a published QSAR model for predicting aggregation propensity. Filter offers the ability to eliminate any of the known aggregators as well as the ability to eliminate compounds that are predicted to be aggregators using the QSAR model. In our experience, the published QSAR model for predicting aggregators is quite aggressive. It occasionally identifies compounds that are known to be genuine small-molecule inhibitors of a specific protein. While in most cases this model can be useful, until more definitive work is published, we feel you should gain some experience with the model and judge its performance for yourself.

1.9 Lipinski and Hydrogen-Bonds

The work of Lipinski [1] introduced the application of simple filter-like rules to roughly predict late-stage PK properties, in particular oral bioavailability. Unfortunately, Lipinski's "Rule-of-Five" has come into the common vernacular to such a large degree that some of the specific details are often lost in the commotion. There are two critical examples of this. First, Lipinski used "violation of 2 rules" to categorize compounds. In the subsequent analysis, a significant difference in the two populations of molecules was detected, however, little analysis of the importance of a single violation was done. However, many now consider a single violation to be bad and two violations to be worse. At this writing, we don't know of evidence to support this. Second, Lipinski used well-codified and well-understood yet imprecise definitions of "hydrogen-bond donors" and "hydrogen-bond acceptors" in his classification model. While this makes the algorithm quite understandable and easy to implement, it sometimes causes confusion with those who prefer more refined definitions of hydrogen-bond donors and acceptors.

To address the first problem, Filter allows users to set the number of Lipinski failures required to reject a model. In keeping with the original publication, the default value is 2. To address the second problem, Filter calculated two kinds of hydrogen-bond donors and acceptors. In the Lipinski calculation, the published definitions are used (donor count is the number nitrogen or oxygen atoms with at least one hydrogen attached and acceptors are the number of nitrogens and oxygens). For calculation of the number of donors and acceptors in the molecule for the sake of chemical properties, a more complex algorithmic approach is taken. This approach identifies the donors and acceptors outlined in the work of Mills and Dean [13] and also in the book by Jeffrey [14].

1.10 Normalization

In addition to the pKa normalization mentioned above, Filter allows any number of additional molecular normalizations. Since normalizations are usually specific to a particular company or site, Filter provides the ability for users to input normalizations, such as the nitro tautomer state, but doesn't provide default implementations. Each normalization can be input as a SMIRKS reaction definition (one reaction per line) in a text file and entered into filter via the 'normalize' command-line parameter. All normalization transformations that can be appropriately applied to each molecule will be applied before each subsequent step in the filtering process.

1.11 Reagent Selection

While virtual screening of whole molecules is the most common task for which Filter is used, we recognize that reagent selection for small linear library synthesis or large combinatorial library synthesis is still a necessary task at many pharmaceutical companies. While the remainder of the Filter program either adjusts molecules or eliminates molecules, the "selection" command-line parameter allows a user to specify a specific functional group that is required in each of the molecules. If a user is hoping to identify a set of acyl-halide reagents, they can use the selection parameter to require that each compound have exactly one acyl-halide. In addition they might want to modify the filter to exclude functional groups (such as primary amines) that may be acceptable for typical lead-like molecules, but aren't acceptable for the specific reagent the user has in mind.

1.12 Pharmacokinetics

Throughout the last decade, several attempts at developing fast-approximate QSAR models for bioavailability have been published. The first of these was Lipinski's work [1] (see above) and it has been followed by work at Pharmacopia [10], Abbott [8] and GSK [9]. Each of these algorithms has been implemented in Filter. The simplest and probably most trusted is the work of LogP and PSA used by Egan. The most recent work by Martin, the Abbott Bioavailability Score (ABS) appears to be a refinement of the first generation models and is designed specifically to categorize a molecule's probability of having a bioavailability >10% in rats.

1.13 Dyes

Years ago, many high-throughput assays could be interfered with by colored molecules. While assay technology has continued to advance and often this is not a problem, we recognize that most molecules that are dyes are not the type of molecules that are commonly carried forward in lead-development projects. Therefore, we continue to include a pattern-based filter for dye molecules. While these patterns occasionally identify molecules that are considered acceptable by some users, in general, they identify molecules that the majority of chemists would rather not see at the top of their virtual-screening hitlists.

1.14 Structural and Chemical features

There are a number of important structural and chemical features of molecules that it is desirable to limit for virtual screening, but don't fall into any of the other categories. These include simple measures such as molecular weight, ring count, ring-system size, size of non-ring structures, length of unbranched chains, hetero-atom fraction, halide fraction, formal charges and rotatable bonds. It also includes slightly more complex algorithms for hydrogen-bond donors and acceptors as well as chiral centers.

1.15 Functional Groups

Functional group removal remains at the heart of the Filter algorithm. Filter's filter files include functional groups that fall into several categories including: reactive or labile groups, undesirable groups, generally acceptable groups, protecting groups, and user-derived groups. While reasonable defaults are provided for each functional group in all of the categories, it is unusual to find an experienced computational or medicinal chemist who agrees with all of the default values. We strongly encourage you to examine the filter file at least once during your initial use and either reassure yourself that the default values are reasonable or develop a custom filter file of your own design. The filter files contain only functional group names. If there is any confusion regarding the definitions, please refer to the appendix for documentation of examples of each group.

1.16 Accumulation of rules

Filter contains numerous rules that judge the quality of molecules on many different facets. When examined individually, each of these rules seems quite reasonable and even profitable. However, when each molecule is tested against hundreds of individual filters, the fraction of molecules that pass all the filters can be surprisingly small. Sometimes less than 50% of vendor databases pass the filters. If this is unacceptable we recommend you examine the predicted aggregator, solubility and Veber filters. In our experience, these are the most common failures. The best method for examining this dilemma is to use a tab-separated table file (use table parameter) with failure value flagged with an asterix (tableflag parameter) and also to examine the log file. The combination of these two methods allows quick adjustment of the filter file to generate the desired database size and properties.

To demonstrate this principle in a tangible way, we carried out filtering on 141 of the best-selling non-antibiotic prescription drugs from 2005. We designed two filters. The first filter adjusts each value so that it just spans the range of the properties for the 144 compounds, and thus this filter allows all 141 compounds to pass. The second filter is very similar to the first. However, for each value, rather than spanning the entire range, it's properties are set to cover from the 2.5th percentile to the 97.5th percentile. The differences between these two filters are often both in reasonable ranges. For instance, the full range of molecular weight spans 130 to 781, while the 2.5th percentile is 145 and the 97.5th percentile is 570. However, the remarkable result is that when the later filter is used, only 75 of the 141 molecules pass the filter! This demonstrates how slight changes to many filters can lead to a significant reduction in the number of compounds that pass all of the filters. Both these example filters are available in the data directory of the distribution (filter_blockbuster.txt and filter_2.5_blockbuster.txt).

1.17 Additional Filters

For this release, we have included three new experimental filters in addition to the default “lead-like” (`filter_lead.txt`) and “drug-like” (`filter_drug.txt`) filters. The filters `filter_alldrug.txt`, `filter_blockbuster.txt` and `filter_2.5_blockbuster.txt` can all be found in the data directory of the filter distribution. The `filter_alldrug.txt` filter is by far the most lenient filter available (though users are free to generate their own). This filter was designed to be the minimum filter that would pass all the molecules found in the “small molecule drug” file from the DrugBank web site (<http://redpoll.pharmacy.ualberta.ca/drugbank>) [15]. Unfortunately, this filter stands more as an example of the futility of this type of approach than as a practical filter. In order to pass every one of these molecules, including some that would not be acceptable for modern project work, many of the individual filters must be set to unreasonable values. For instance, the molecular weight range is 30 to 1500 and the hetero-atom count range is 1 to 60!

In order to generate more useful filters, rather than using a large list of drug molecules, we focussed on the best-selling, non-antibiotic, prescription drugs. We compiled a list of approximately 140 compounds. We designed two filters around this set of compounds, one that passes all of the compounds and another that is slightly more discriminating. For details of this second filter, please see “Accumulation of rules” above. Here we’ll focus on the first and most useful of these filters, `filter_blockbuster.txt`. The physical property values in this filter are quite good. However, the functional group filters in this filter are probably too restrictive as 140 compounds is not sufficient to span all acceptable functionality. We expect that combining the physical property limits from `filter_blockbuster.txt` with the functional group filters from `filter_lead.txt` or `filter_drug.txt` may provide a very good filter. In future releases we plan to include a default filter that is optimized along these lines. For this release, that experimentation is left to the user. Despite the caveat concerning overly strict functional group filters, we expect that even without modification, `filter_blockbuster.txt` is a useful filter.

Theory

2.1 The Rant!

Nearly every computational tool used in early drug discovery yields statistically predictive, rather than absolutely definitive results. In nearly every case, prudence demands that one consider the causes of false-positives and false-negatives and make an attempt to optimize the area under the receiver-operator curve (ROC) for the computational tool. However, there are well known methods for improving statistical predictions of this nature that are independent of the absolute false-positive and false-negative rates. These methods include filtering the population to which a test will be applied. By applying a test to smaller populations that only contain molecules appropriate for the specific application at hand, the negative impact of the false-positive rate on the predictive results can be dramatically improved.

A familiar example from the medical world will serve to illustrate this principle. Assume we have a test for the presence of the new “foo virus” which has an exceptional ROC curve with false-positive and false-negative values (1/1000 and 1/1000 respectively). Let us assume that the foo-syndrome, caused by the foo virus, effects 1 person in 20,000. If we gave this test to 100,000 people from the general population, we would expect five to actually have the foo syndrome. With this test, there is only a 0.05% percent chance that any of them would not be detected (i.e. be a false-negative). However, we would expect there to be 100 false positive test results. Thus of the 105 total positive test results, only 4.8% would actually have the foo syndrome (positive predictive value = 4.8%).

Alternatively, we could start by using very simple screening before applying the test. We first eliminate people who do not have any risk factors for contracting the foo virus. Next we may eliminate people whose blood is incompatible with the test for the foo virus. Further, we may want to eliminate people who acknowledge that they will refuse treatment for the foo virus even if we determine that they do have it. By these admittedly simple screens, we apply the test for the foo virus to a much smaller group with a decidedly higher prevalence of the virus. For instance, after the filtering, we may be left with a group of only 1000 people who have a 1 in 200 chance of having the syndrome. Now, we still have the same five people who actually have the disease, but we only expect one false positive test. Suddenly, there are six positive tests, and 83% of them actually have the syndrome! This is reflected in a much more reasonable (83%) positive predictive value.

Bringing the discussion back to drug design. If we have a ligand-based design tool such as ROCS, we can imagine that the receiver-operator curve may have a false positive rate as low as 1 in 10000. For this exercise, lets assume no false negatives. When using that to identify 50 inhibitors from a database

of 2.5 million available compounds, we'd identify 300 potential inhibitors, and 5 out of every 6 of these would be a false positive (positive predictive value of 17%)! If we first run filter and eliminate 65% of the 2.5 million compounds, this leaves us with 875000 compounds to push through ROCS. There will be about 88 false positives to go with the 50 true positives and the positive predictive value will increase over two-fold with relatively little work.

2.2 Filtering Principles

We can apply the same principles to screening for lead candidates, regardless of whether it is virtual screening or high-throughput screening. While both are reasonable screens, each can be plagued by very low positive-predictive values (despite low false-positive rates), particularly when applied to all available compounds, or large virtual libraries. We want to use simple filtering techniques to focus the set of compounds passed on to more computationally intensive screening methods. First, we can consider filtering based on functional groups. Generally speaking, there are toxic and reactive functional groups that we simply don't want to consider (alkyl-bromides, metals etc). Next, there are functional groups that aren't strictly forbidden, but are not desired in large quantities. For instance, parafluoro-benzene, or trifluoromethyl have specific purposes, but heavily fluorinated molecules can be eliminated.

Beyond simple functional group filtering, one can consider both simple and complex physical properties which can be used to characterize the kinds of compounds one would like to keep and those one would like to eliminate. These properties attempt to consider "drug-likeness", such as bio-availability, solubility, toxicity, and synthesizability even before the primary high-throughput or virtual screening, which primarily are geared toward detecting potency alone. The best known of the physical property filters is Lipinski's "rule-of-five", which focuses on bioavailability [1]. However, many other physical properties, such as solubility, atomic content, ring structures, and surface area ratios can also be considered. We provide algorithms for calculating many of these properties, and apply them with filters based on literature studies.

Finally, one should eliminate the types of compounds that can be troublesome at later stages. For instance, Shoichet's aggregating compounds often produce false positives that can waste enormous resources if they were identified by virtual or high-throughput screening [2, 3]. Similarly (and sometimes concomitantly) dyes can appear to be inhibitors by interfering with colorimetric or fluorometric assays or binding non-specifically to the target protein.

2.3 Variations of Filtering

Different types of filters are appropriate under different circumstances. Very early in a project, when little or no SAR is available, very strict drug-like filters can be applied. This prevents a project team from spending chemistry resources pursuing difficult compounds that may not be modifiable to introduce appropriate properties. However, when considering compounds for purchase for HTS, different filters can be applied. Oprea, et al, pointed out that the best molecules for initial HTS are smaller and less functionalised than drugs, but with some activity [4]. Therefore, strict lead-like filters can be applied to ensure that hits identified from HTS have sufficient "room" for elaboration into (usually larger and more highly functionalised) leads. However, when SAR suggests that particular compounds or series may yield

valuable information, filtering criteria can be loosened, because the secondary screens (QSAR models, similarity to known actives) that are being applied are effective in detecting useful compounds. Reflecting back on the medical analogy, this is the case where an improved primary screen with a dramatically improved false-positive rate (say 1 in 100,000) can be safely applied to a larger population without terrible effects on the positive-predictive value. In the Filter program, we provide examples of both "heavy" (restrictive) and "light" (lenient) filters.

Usage

3.1 Command Line Interface

Executing Filter with no arguments will result in:

```
prompt> filter
```

```
Filter version 2.0.1
  OEChem version 1.4.2, 20070115
  Platform: microsoft-win32-g++3.3-i586
  OpenEye Scientific Software, Inc.
```

No argument specified on the command line

Required parameters:

```
-in : Input filename
-out : Output filename
```

For more help type:

```
filter --help'
```

A description of the command line interface can be obtained by executing filter with the `-help` option.

```
prompt> filter --help
```

will generate the following output:

```
Help functions:
  filter --help simple      : Get a list of simple parameters
  filter --help all        : Get a complete list of parameters
  filter --help <parameter> : Get detailed help on a parameter
  filter --help html       : Create an html help file for this program
```

If you desire to see all of the command-line options use `-help all`.

```
prompt> filter --help all
```

will generate the following output:

```
Filter version 2.0.1, 20070115
  OEChem version 1.4.2, 20070115
  Platform: microsoft-win32-g++3.3-i586
```

OpenEye Scientific Software, Inc.

Complete parameter list

filter

```

-dots : Write a dot to the terminal for every 500 cpds processed
-fail : Write failed molecules to file
-filter : Filter to use in filtering
-in : Input filename
-info : Write to specified info file (overrides -prefix)
-interval : Update info file every N molecules
-log : Write to log file
-newrule : File listing additional filter rules and SMARTS
-normalize : SMIRKS file for normalization of molecules
-out : Output filename
-param : Filter control parameter file
-pkanorm : Apply a neutral pH model
-prefix : Prefix all generic file output with this string
-salt : Molecule file of salts to remove instead of calling strip salts
-sdtag : Add all filter parameters as SD Tags
-select : Smarts pattern to select for groups
-table : Write complete output in table form
-tableFlag : Flag failure values in table form with an asterix
-typecheck : Check for broken molecules (impossible valence states)

```

3.1.1 Required Parameters

- in** File containing one or more molecular connection tables from which you would like to remove the non-medicinal compounds. File formats are discussed in the subsequent section.
- out** File to fill with the molecules that pass all of the specified filters. File formats are discussed in the subsequent section.

3.1.2 Molecular File Formats

Filter can read and write a variety of molecular file formats. The file format is automatically interpreted from the filename suffix.

File type	Extension
SMILES	.smi .ism .can .smi.gz .ism.gz .can.gz
SDF	.sdf .mol .sdf.gz .mol.gz
MOL2	.mol2 .mol2.gz
PDB	.pdb .ent .pdb.gz .ent.gz
MacroModel	.mmod .mmod.gz
OEBinary v2	.oeb .oeb.gz
Old OEBinary	.bin

Old OEBinary format can be read but not written by Filter. Gzipped OEBinary version 2 is the recommended output format.

Filter is also capable of piping formatted input and output. The simple “-” can be used in place of a file name to indicate `std::cin` or `std::cout` with the default SMILES format.

```
prompt> filter -in - -out -
```

This execution will run Filter with `std::cin` as the input with SMILES format. It will also open `std::cout` with SMILES format as output. However, the use of “-” does not allow control of the file format.

To control the format of `std::cin` and `std::cout` one may use the file extensions without a preceding filename.

```
prompt> filter -in .ism -out .oeb.gz
```

This executes Filter with the input from `std::cin` formatted in isomeric SMILES and the output sent to `std::cout` in gzipped OEBinary version 2 format.

3.1.3 Optional Parameters

- filter** This optional parameter specifies a filter file to be used in place of the default filter. If only simple additions to the default filters are desired, please see the `-newrule` parameter. The file format for this file is described in the chapter on parameter files. There are two special reserved strings for this parameter. If the `-filter` parameter is “lead”, then the default lead-based filter will be used. If the `-filter` parameter is “drug”, then the default drug-based filter will be used. [default = lead]
- fail** This specifies an optional molecular output file where the molecules that fail to pass the filter will be written. If this parameter is specified, then every molecule from `-in` will either be written to `-out` or to `-fail`. [default = null]
- prefix** For an execution of the Filter program, three general purpose files are written in addition to the output file specified on the command-line. These files are the “info”, “log” and “param” files. Normally, they all begin with the prefix “filter”. However, this can be overridden with the `-prefix` parameter. This is particularly useful if you want to run multiple filter jobs in the same directory without overwriting files.
- info** Normally, Filter writes an “info” file during the progress of any execution. At regular intervals during the execution, the info file is updated to reflect the most recent progress. If you are interested in seeing the progress of a run, it is best to either use the `-dots` parameter or look at the info file. Normally, the info file is saved as `filename.info` where “filename” is the prefix specified by the `-prefix` flag. However, one may use the `-info` flag to specify an info file with a separate name. [default = null]
- newrule** This optional parameter can specify a file that contains filter rules to supplement the default filter or the filter specified with the `-filter` parameter. This parameter can be used to extend the functional group list used to filter. New filter rules can also be added directly to the filter file specified with `-filter`. [default = null]
- typecheck** This boolean flag controls whether the valence states of atoms will be checked. This check identifies molecules that are poorly specified, or represent nonsensical chemical states. For ex-

ample, an oxygen with eight hydrogens attached or a carbon with a +9 formal charge would be rejected. [default = true]

- select** This parameter is a SMARTS string that allows a user to require a specific functional group or substructure be present in all molecules that pass the filter. This feature is particularly useful for identification of reagents for library design. Selection items can also be added directly to the filter file specified with `-filter`. The command-line argument only allows specification of the SMARTS pattern, and exactly one copy of that functional group is required. If a user wants to specify a selection SMARTS with minimum and maximum number of occurrences other than 1, then they can use a SELECT statement inside the filter file. [default = null]
- log** This flag determines where the logging information is written. The logging information includes a listing of the filter used, followed by a one line comment about why each molecule failed, or if it passed, an assessment of the probability that the compound lies in drug-like space (see Chapter 6, CMV).
- table** This flag specifies a file for a tab-separated format table that includes all of the filter data. These data files are ready for import into a spreadsheet program for easy examination. Each column of the table includes one of the filter categories (such as "Molecular Weight") and each row of the table corresponds to a single molecule. The table contains complete entries for all of the molecules in the input file regardless of whether they pass or fail. NOTE: Setting this flag will cause the program to slow down. [default = null]
- tableFlag** This flag specifies that *if* a table is being written, any values in the table that would cause a molecule to fail a filter will be flagged with an asterix. This provides a means of seeing all the filters a molecule might fail, as the log file typically only provides the first failure. [default = false]
- interval** This is the interval at which data is written to the filter.info file. The filter.info file contains running totals that are relevant to a filter run. Examining the filter.info file is the best means of checking on the progress of a filter execution. If this flag is 50, then the filter.info file is re-written every 50 molecules. [default = 5000]
- param** This flag specifies a parameter file that contains all of the command-line parameters in a simple text file. The parameter file is automatically written to "filter.param" with every execution. It is a record of the input that was used and can be used to rerun the exact same process. It can also be altered by hand to modify a prior execution. If a parameter is set in the param file and on the command line, the command line setting takes precedence. More information is available in the chapter on param files below.
- pkanorm** This boolean flag determines whether compounds will be modified to reflect a pH=7.4 model. Notice, this will modify the molecules permanently. [default = true]
- normalize** This flag indicates an optional SMIRKS file. This file should contain the set of reactions you wish to use to normalize the connection table of your molecules. Please note: These reactions are applied before the filtering process and can *significantly* slow the filtering process. [default = null]
- salt** This flag specifies a molecule file that you consider to be salts. If any molecular entries contain multiple disconnected fragments, then any fragment contained in the "salt" file will be removed. If

no file is specified, or if there are multiple disconnected fragments in a molecule record that are not in the salt file, then the first largest remaining fragment will be retained and all others discarded. [default = null]

- sdtag** This boolean flag indicates whether you want the molecular properties used for the filtering run (see `-filter`) to be attached to output molecules as SD tag data. This parameter will only work for `.sdf` or `.oeb` formats. [default = false]
- dots** Boolean flag that determines whether filter writes a single dot (.) to the terminal (stdout) for every 500 compounds that are processed.

3.2 Example executions

This section has a series of example Filter command-line executions. Each example is followed by a brief description of its behavior.

```
prompt> filter drugs.smi drugs.oeb.gz
prompt> filter -in drugs.smi -out drugs.oeb.gz
```

These two commands will yield identical results. These execute Filter with the default parameters. The file `drugs.smi` is opened in SMILES format for input, and the output is written to the file `drugs.oeb.gz` in gzipped OEBinary version 2 format.

```
prompt> filter -in drugs.smi -out drugs.sdf -filter myfilter
```

This command is the same as the one above except for the `-filter` flag. It executes Filter with the parameters found in the `myfilter` file. The file `drugs.smi` is opened in SMILES format for input, and the output is written to the file `drugs.oeb.gz` in gzipped OEBinary version 2 format.

```
prompt> filter -param myparameters drugs.smi drugs.oeb.gz
prompt> filter drugs.smi drugs.oeb.gz -param myparameters
```

The first of these two commands will yield exactly the same results as the example above. `drugs.smi` will be mapped to the `-in` flag and `drugs.oeb.gz` will be mapped to the `-out` flag begin the second to last and last command-line arguments respectively. Unfortunately, the second of these two commands, will fail to parse because the implicit input and output arguments are not the final two arguments in the list.

```
prompt> filter -in drugs.smi -out drugs.oeb.gz -table drugs.table
```

This executes Filter on the file `drugs.smi` and writes molecules that pass the filter to the file `drugs.oeb.gz`. It also writes the all of the filter data to the tab-separated value file `drugs.table`.

```
prompt> cat maybridge.05-1.sdf | filter -in .sdf -out .ism|omega .ism m.oeb.gz
```

This command presumes that you have a `sd` format file called "maybridge.05-1.sdf". That file is 'cat'ed and piped to the filter program. The `-in .sdf` flag indicates that filter should read `.sdf` format from `std::in`. Since no `-filter` flag is specified, the default filter will be used. The `-out .ism` flag indicates filter will write isomeric smiles format to `std::out`. The output would then be piped into `omega`.

```
prompt> filter -in drugs.smi -out drugs.oeb.gz -select '[N;$(*-a)]'
```

This command will filter the compounds in `drugs.smi` with the default filter and write the output to `drugs.oeb.gz`. It also requires that molecules contain exactly one instance of the aniline substructure defined by the SMARTS pattern "[N;\$(*-a)]".

The Param File

Filter's command-line interface can be efficiently run using the `-param` command line parameter followed by the name of a parameter file. Param files are simply files that contain one command-line parameter on each line. Every execution of filter generates a `.param` file called *filter.param* (unless the name is overridden with either the `-param` flag or the `-prefix` flag). This file contains all of the parameters used by Filter. Further, this file can be used in subsequent Filter runs with the `-param` flag either with or without user modifications.

The following execution will generate a `.param` file called "filter.param" which is listed below. This file could be edited and used with subsequent Filter runs.

```
prompt> filter -in drugs.smi -out drugs.sdf -table drugs.table
```

Listing of "filter.param":

```
#Interface settings

#filter :
  #-fail (Not set, no default)
  #-filter (Not set, no default)
  -in drug.smi
  -interval 5000 (default)
  -log filter.log (default)
  #-newrule (Not set, no default)
  #-normalize (Not set, no default)
  -out drug.sdf
  #-param (Not set, no default)
  -pkanorm true (default)
  #-salt (Not set, no default)
  -sdtag false (default)
  #-select (Not set, no default)
  -table drugs.table
  -typecheck true (default)
```

```
prompt> filter -param filter.param
```

This would result in exactly the same execution as the one which generated the file above.

The Filter Files

Filter is no longer dependent on any parameter files. However, there are two parameter files a user can provide if they would like to override or augment the default parameter sets. We recommend that you do this by modifying the provided default (`filter_example` and `newrule_example`) files rather than starting from scratch.

There are two primary files a user may choose to provide. The first is the filter file. It provides acceptable limits for all of the physical properties and functional groups in the default filter. The second is the newrule file. If you have a filter you like, but would like to augment it with a set of additional rules, these can be added with a newrule file.

5.1 Filter file format

The four types of statements that can occur in a filter file are physical property limits, rules, newrules, and selections. The statements should occur one-per-line in the filter file.

5.1.1 Customizing Filters

If the appropriate line is not in the filter file, or the value is false, the respective measure will not be used in filtering and its value will not be included in any table-based output.

5.1.2 Physical property limits

Simple properties

There are a large number of physical property limits. They occur as three fields on a line. For example:

```
MIN_HETEROATOMS 2 "Minimum number of heteroatoms"
```

The first field is the property keyword, the second field is the value assigned to that keyword, and the third field is a brief informational message. There are a fixed number of physical property keywords. No additional physical property keywords can be added by the user. The current keywords and a brief definitions of each are included below.

Molecular Weight Isotopic molecular weight

MIN_MOLWT
MAX_MOLWT

Heavy Atom Count Number of non-hydrogen atoms

MIN_NUM_HVY
MAX_NUM_HVY

Carbon Count Number of carbons

MIN_CARBONS
MAX_CARBONS

Hetero-Count Number of non-carbon and non-hydrogen atoms

MIN_HETEROATOMS
MAX_HETEROATOMS

Hetero-Atom to Carbon Ratio Hetero Count/Carbon Count

MIN_Het_C_Ratio
MAX_Het_C_Ratio

Chiral Count Number of chiral atoms

MIN_CHIRAL_CENTERS
MAX_CHIRAL_CENTERS

Hydrogen-bond Acceptors Sum of a) degree 2, aromatic, non-positive nitrogens, b) electron rich or negative, valence less than 4, non-aromatic nitrogens, c) negatively charged or not electron withdrawn and neutral oxygens, and d) degree 1, double bonded, electron rich sulfur.

MIN_HBOND_ACCEPTORS
MAX_HBOND_ACCEPTORS

Hydrogen-bond Donors Sum of hydrogen atoms on Nitrogen, Oxygen and Sulfur atoms.

MIN_HBOND_DONORS
MAX_HBOND_DONORS

Lipinski Acceptors Number of Nitrogens + Oxygens

MIN_LIPINSKI_ACCEPTORS
MAX_LIPINSKI_ACCEPTORS

Lipinski Donors Number of Hydrogens attached to Nitrogen or Oxygen

MIN_LIPINSKI_DONORS
MAX_LIPINSKI_DONORS

Halide Fraction Percent of molecular weight from halides

MIN_HALIDE_FRACTION
MAX_HALIDE_FRACTION

Formal Count Number of atoms with a formal charge (excludes dative)

MIN_COUNT_FORMAL_CRG
MAX_COUNT_FORMAL_CRG

Formal Sum Sum of formal charges

MIN_SUM_FORMAL_CRG
MAX_SUM_FORMAL_CRG

Connected Non-Ring Considers sets of contiguous (bonded) non-ring atoms

MIN_CON_NON_RING
MAX_CON_NON_RING

Unbranched Chains The size of unbranched non-ring chains

MIN_UNBRANCHED
MAX_UNBRANCHED

Total Functional Group Count Total number of functional groups (not just those recognized by filter patterns). Does not count any ring-systems as functional groups. Degree 1 atoms heteroatoms, particularly those with double bonds or dative bonds are considered part of ring systems and do not count as a functional group.

MIN_FCNGRP
MAX_FCNGRP

Ring Systems Number of ring systems (contiguous systems of ring atoms and bonds)

MIN_RING_SYS
MAX_RING_SYS

Ring Size Maximum size of any single ring system

MIN_RING_SIZE
MAX_RING_SIZE

Rotor Count Number of rotatable bonds. Allows optional adjustment for aliphatic rings following the method of Oprea [4].

MIN_ROT_BONDS
MAX_ROT_BONDS
ADJUST_ROT_FOR_RING

Rigid Count Number of rigid bonds

MIN_RIGID_BONDS
MAX_RIGID_BONDS

LogP

Filters logP calculation is a derivative of the published XLogP algorithm (see introduction for details). There are two associated filters

XLogP Calculated LogP

```
MIN_XLOGP      -5.0      "Minimum XLogP"
MAX_XLOGP      6.0      "Maximum XLogP"
```

Solubility

Solubility is also a filter, but rather than a quantitative cutoff, categories are used. The 6 allowable categories include: insoluble, poorly, moderately, soluble, very, and highly. These categories are keywords used in the filter files as follows.

Solubility Calculated solubility class

```
MIN_SOLUBILITY moderately "minimum solubility"
```

Pharmacokinetic predictors

Several secondary filters that are built upon published combinations of simpler properties are included. These include Lipinski [1], Veber(GSK) [9], Martin (Abbott)[8] and Pharmacopia (Egan)[10]. The Lipinski filter allows one to specify the number of allowed violations (the published work allows compounds to pass with a single violation but not multiple violations). All other filters in this category simply allow a boolean flag of whether the filter should be applied.

All of these properties are used for filtering in the default filters.

The Veber, Lipinski, Martin, and Egan (pharmacopia) filters are turned on by inclusion of one of the following lines in a filter file:

psa Peter Ertl's Topological Polar surface area (Phosphorus and Sulfur area is optional)

```
PSA_USE_SandP  false      "Count S and P as polar atoms"
MIN_2D_PSA     0.0        "Minimum 2-Dimensional (SMILES) Polar Surface Area"
MAX_2D_PSA     150.0     "Maximum 2-Dimensional (SMILES) Polar Surface Area"
```

GSK/Veber Veber's measure of bioavailability (PSA > 140 or Rotatable bonds >10) [9]

```
GSK_VEBER      true       "PSA>140 or >10 rot bonds"
```

Lipinski Violations Number of Lipinski violations. A single Lipinski violation is considered acceptable[1]

```
MAX_LIPINSKI   1          "Maximum number of Lipinski violations"
```

Abbott/Martin Yvonne Martin's Abbott Bioavailability Score. This is reported as a probability that F>10% in rats. (ABS)[8]

```
MIN_ABS 0.5 "Minimum probability F>10% in rats"
```

Pharmacopia/Egan “Egan egg” measure of bioavailability (LogP >5.88 or PSA > 131.6)[10]

```
PHARMACOPIA true "LogP > 5.88 or PSA > 131.6"
```

Aggregators

Aggregators are small molecules that can interfere with assay results by sequestering protein in an aggregation of small molecules in solution. They appear to have activity in many assays, but in fact are usually not specific inhibitors of the protein in question. Includes two measures of whether a molecule is one of the aggregators defined by Shoichet et. al. [2, 3] The first measure is whether the molecule is an exact match to one of the approximately 400 published aggregators. The second measure is whether the molecule hits in Shoichet's QSAR model for predicting aggregators.

Aggregators Whether a compound is known or predicted to aggregate in concentrations common in virtual screening.

```
AGGREGATORS true "Eliminate known aggregators"
PRED_AGG true "Eliminate predicted aggregators"
```

5.1.3 Elemental filters

There are two separate filters for processing elements. This process is carried out in two stages to allow the filter to treat atoms in the counter-ion portion of a molecule separately from the atoms in the primary portion of the molecular record.

The format of the two fields is a keyword followed by a comma delimited list of atomic symbols.

```
ALLOWED_ELEMENTS H,C,N,O,F,S,Cl,Br
```

```
ELIMINATE_METALS Sc,Ti,V,Cr,Mn,Fe,Co,Ni,Cu,Zn,Y,Zr,Nb,Mo,Tc,Ru,Rh,Pd,Ag,Cd
```

First, any compounds with the atoms indicated in ELIMINATE_METALS fail to pass the filter. Next, the OEChem function `OETheFunctionFormerlyKnownAsStripSalts` is called. Finally, compounds with atoms other than those specified by ALLOWED_ELEMENTS are eliminated. For instance, this allows organic molecules that are complexed with silver to be eliminated based on their metal chelate even though they themselves are acceptable while at the same time eliminating a sulphate counter-ion from another molecule before it leads to elimination of the acceptable cationic molecule.

5.1.4 Functional group rules

Rules statements set the limits for the maximum number of the specified type of functional group that may be allowed in the molecule. For example:

```
RULE 0 acid_halide
```

The first field of a rule statement is the word RULE in all capital letters. The second field is a number indicating the maximum number of the group allowed in a molecule. The third field is the functional

group keyword. Functional-group keywords are case sensitive. The functional groups for which rules may be generated include:

acetal, acid_halide, acyclic_NCN, acyclic_NS, Acyl_cyanides, Acylhydrazide, alcohol, aldehyde, alkene, alkyaniline, alkyl_halide, Alkyl_phosphate, alkyne, alphahalo_amine, alpha_halo_ketone, amide, aminal, amine, amino_acid, anhydride, aniline, arenesulfonyl, aryl_halide, azide, aziridine, azo, azocyanamides, benzyl_ether, benzyloxycarbonyl_CBZ, beta_azo_carbonyl, beta_carbonyl_quat_nitrogen, beta_halo_carbonyl, carbamate, carbamic_acid, carbodiimide, carbonate, carboxylic_acid, cation_C_Cl_I_P_or_S, chloramidines, cyanohydrins, cycloheximide_derivatives, cyclopropyl, cytochalasin_derivatives, dioxane_6MR, dioxolane_5MR, di_peptide, disulfide, dithioacetal, dye, enamine, enol_ether, epoxide, ester, ether, fluorenylmethoxycarbonyl_Fmoc, guanidine, halide, halo_alkene, halo_amine, halopyrimidine, hemiacetal, hemiaminal, hemiketal, hetero_hetero, HOBT_esters, hydrazine, hydrazone, hydroxamic_acid, hydroxylamine, imine, iodine, iodoso, iodoxy, isocyanate, Isonitrile, isothiocyanate, ketal, ketone, lactam, lactone, Lawesson_s_reagent, malonic, methoxyethoxymethyl_MEM, methyl_ketone, michael_acceptor, Monensin_derivatives, nitrile, nitro, nitroso, Nitroso, N_methoyl, nonacylhydrazone, noxide, N_P_S_Halides, NS_beta_halothyl, organometallic, oxalyl, oxaziridine, oxime, oxygen_cation, Paranitrophenyl_esters, pentafluorophenyl_esters, perhalo_ketone, peroxide, phenol, phosphanes, phosphinic_acid, phosphonamide, phosphonic_acid, phosphonic_ester, phosphonylnitrile, phosphoramides, phosphoranes, phosphoric_acid, phosphoric_ester, phosphoryl, phosphoryl, phthalimides_PHT, polyenes, primary_amine, propiolactones, quinone, saponin_derivatives, SCN2, secondary_amine, squalestatin_derivatives, sulfide, sulfinimine, sulfinylthio, sulfonamide, sulfone, sulfonic_acid, sulfonic_ester, sulfonimine, sulfonyl_halide, sulfonylnitrile, sulfonylurea, sulfoxide, t_butyl(dimethyl)silyl_TBDMS, t_butyl(diphenyl)silyl_TBDPS, t_butyl_ether, t_butoxycarbonyl_tBOC, terminal_vinyl, tertiary_amine, tetrahydropyran_THP, thioamide, thiocarbamate, thioester, thiol, thiourea, Triacyloxime, triazine, tricarbo_phosphene, triflates, triisopropylsilyl_TIPS, trimethylsilyl_TMS, and urea.

5.1.5 New rules

New rules specify additional functional groups or substructures that may be used with the default Filter. They must specify a substructure definition in the form of a SMARTS in addition to the substructure name and maximum limit. For example:

```
NEWRULE norborane 1 C1CC2CCC1C2
```

The first field is the NEWRULE keyword. The second field defines the name associated with the substructure (primarily for logging purposes). The third field indicates the maximum number of the substructure that can be allowed. The fourth field is the SMARTS string for the substructure, norborane in this case. This example rule would indicate that molecules with a single norborane substructure would be allowable, but that those with 2 or more norboranes would be eliminated.

Newrules that have a name that is identical with one of the original rules take precedence over the original rule.

5.1.6 Selection statements

The select statement allows a filter file to specify the required number of substructures in order to be able to pass the filter. These statements are similar to new rules except that they list a required range for passing the filter rather than the range for failing to pass the filter. For example:

```
SELECT amine 1 1 [N;!$(*-* [#6;#1]);!(a);!(*=,#*)]
```

The first field is the SELECT keyword. The second field indicates the name for the selection (again for logging purposes). The third field is the minimum number of substructures required to be in the molecule. The fourth field is the maximum number of substructures allowed in the molecule. The fifth field is the substructure defined by a SMARTS pattern. The example requires that molecules contain exactly one amine. This version of filter only accepts a single SELECT statement. Any complex boolean substructure statements can be incorporated directly into the SMARTS. If multiple SELECT statements occur in a filter file, only the final one will be applied.

Release Notes

A.1 Filter 2.0

March 2009 2.0.2

This is a minor bug fix release that updates filter to OpenEye's standardized distribution system.

Bug Fixes and Refinements:

1. FILTER installations now support having multiple versions and platform architectures in the same directory structure.
2. FILTER installations on Microsoft Windows platforms now have documentation available in the start menu.
3. Filter installations on Microsoft Windows platforms now have OpenEye Command Prompts available from the start menu. These prompts set the appropriate environment variables for running OpenEye software.

January 2007 2.0.1

This release is a broad-sweeping bug fix release with a dramatically improved introduction in the documentation. It includes many simple and complex bug fixes including; conceptual clarifications of several algorithms, fixes and improvements to the data representation, several advances in the pKa model, and significantly more documentation. While this is only a bug-fix release, it represents a significant step-forward in the refinement of the product. We hope you find it suits your needs.

Bug Fixes and Refinements:

1. Added three additional filters that are designed around the best-selling prescription drugs
2. Fixed two bugs in the algorithm for recognizing hydrogen-bond donors and acceptors
3. Abbott Bioavailability Score corrected to be reported as a probability
4. Aggregator function augmented to include more than 250 additional known aggregators
5. Predicted aggregator QSAR model introduced to further enhance aggregator recognition capability

6. XLogP and LogS corrected to always examine the hypervalent form of molecules (as are used in the training set)
7. Improved data reporting and filter control for the pharmacokinetic properties
8. Fixed category reporting for solubility calculation
9. Changed the default TPSA calculation to ignore rather than include P and S surface area
10. Renamed and clarified aliphatic_group filter to “maximum connected non-ring atoms”
11. Fixed typos in names beta_carbonyl_quat_nitrogen and t_butoxycarbonyl_tBOC
12. Updated chloramidine filter name to the more common imidoyl_chloride
13. Fixed error in beta_azo_carbonyl pattern
14. Made patterns for cytochalasin, monensin, squalestatin and saponin more specific
15. Clarified the exclusion patterns for acyclic_NCN and SCN2
16. Dramatically improved the specificity and range of the michael_acceptor pattern
17. Implemented an algorithm for “total functional group count” that takes account of all functional groups rather than only those included in the filter patterns
18. Fixed error in alkyl_halide pattern that limited it to bromine and iodine
19. Fixed multiple patterns that matched their function group multiple times because of symmetry
20. Limited enamine definition to exclude exceedingly delocalized instances
21. Added -tableFlag parameter to mark all failure values in the table file with an asterisk
22. Fix t_butyl_ether pattern to avoid matching t_butyl_alcohols
23. Removed dependence of rigid bond count upon implicit or explicit hydrogens
24. Extended the unique flag to allow values of “true”, “false” or a filename used to pre-load a file of uniques
25. Cleaned-up table data for the select parameter
26. Corrected problem that MIN_HALIDE_FRACTION and ALLOWED_ELEMENTS could not properly be removed from the filter file
27. Fixed bug so NEWRULE lines with trailing white space are not ignored
28. Fixed crash bug that occurred when a filter file was specified that was not a valid file
29. Added pKa for phenol groups with significant electron withdrawing groups
30. Improved pKa value for barbituate-like compounds

31. Clarified alpha-acyl carbon deprotonation
32. Removed 1,2,3 triazole pKa rule
33. Clarified tertiary amine rule
34. Improved efficiency for applying pKa rules to multi-conformer molecules

October 2005 2.0.0

This is the first official "2.0" version of filter. It includes many bug-fixes and feature expansions based on the long beta period. Some of these include higher level filtering flags such as Lipinski and other similar measures. This release also includes a pKa model that was missed in the beta release as well as non-pH normalization capabilities. This version makes exporting filtering data to other programs facile with both table and sdtag data export.

New Features:

1. Neutral pKa model optionally applied to all molecules
2. Support for molecular graph normalization
3. Optional salt file can be used to remove salts from database
4. Calculated properties can be attached to molecules as SD tag data
5. All filtering data can be saved in tab-separated value format
6. Filter on fractional halide weight
7. Removal of known aggregator molecules
8. Veber bioavailability filtering
9. Lipinski violation filtering
10. Abbott/Martin bioavailability filtering
11. Pharmacopia bioavailability filtering
12. Solubility class filtering
13. Chiral center count filtering

February 2005 2.0b1

This is the first version of Filter based on OEChem. As such, it does not alter the graph of your molecules. This is a departure from Filter 1.x, that was based on OELib and applied a pH model to the molecules in an attempt to fix them in protonation state for pH=7.0.

This version of filter has also eliminated the need for cumbersome external data files while still allowing user control over the filtering process. Several new functional groups have been added to the filters and more realistic filter values have been used for several functional groups.

High-level filters such as “total number for functional groups” and “carbon to heteroatom ratio” have been added to prevent unfunctionalized molecules from slipping through the filters.

New Features:

1. Based on OEChem
2. Superior *Data Integrity*
3. Valence checking (broken molecule filter)
4. Substructure selection (for selection of filtered reagent lists)
5. Not dependent on external data files
6. Improved filters and filter properties
7. >500 molecules/second with SMILES I/O
8. Support for additional platforms
9. Documentation
10. Improved command-line interface including command-line help

A.2 Filter 1.0

February 2001 1.0.1, 1.0.2

This fixes several minor bugs and adds solubility prediction and tabulated output.

New Features:

1. Solubility prediction based on fitting to XlogP-like atom types.
2. Table based output of filtering data

December 2000 1.0

This initial version of filter is based on OELib. It is distinct from previous non-commercial versions of filter in that it incorporates XLogP and PSA, provides extensive default filters and uses physical properties in addition to functional groups.

New Features:

1. OpenEye’s general version of XlogP.
2. Approximate polar surface area.
3. Filtering based on physical properties including: molecular weight, ring size and number, allowed elements rotatable bonds, hydrogen-bond donors and acceptors, formal charges and chromophores.

-
4. Filter based on function groups including: acid halide, aldehyde, alkyl halide, anhydride, azide, azo, di-peptide, long chains, michael-acceptors, beta-halo-carbonyl, nitro, peroxide, phosphonic acid, sulfonic acid, triphenyl-phosphene, epoxide, sulfonyl halide, aziridine, imine, oxalyl, thiol, urea, aniline, aryl halide, carbamate, ester, ether, hydrazine, hydrazone, hydroxylamine, nitrile, sulfide, sulfone, sulfoxide, thiourea, thioamide

Lead-like filter

```
#!/*****
#Copyright (C) 2000-2005 by OpenEye Scientific Software, Inc.
#*****/
#This file defines the rules for filtering multi-structure files based on
#properties and substructure patterns.
MIN_MOLWT      150      "Minimum molecular weight"
MAX_MOLWT      440      "Maximum molecular weight"

MIN_NUM_HVY    10      "Minimum number of heavy atoms"
MAX_NUM_HVY    25      "Maximum number of heavy atoms"

MIN_RING_SYS   0       "Minimum number of ring systems"
MAX_RING_SYS   3       "Maximum number of ring systems"

MIN_RING_SIZE  0       "Minimum atoms in any ring system"
MAX_RING_SIZE  20      "Maximum atoms in any ring system"

MIN_CON_NON_RING  0     "Minimum number of connected non-ring atoms"
MAX_CON_NON_RING  15    "Maximum number of connected non-ring atoms"

MIN_FCNGRP     0       "Minimum number of functional groups"
MAX_FCNGRP     12      "Maximum number of functional groups"

MIN_UNBRANCHED 0       "Minimum number of connected unbranched non-ring atoms"
MAX_UNBRANCHED 3       "Maximum number of connected unbranched non-ring atoms"

MIN_CARBONS    5       "Minimum number of carbons"
MAX_CARBONS    23      "Maximum number of carbons"

MIN_HETEROATOMS 2      "Minimum number of heteroatoms"
MAX_HETEROATOMS 12     "Maximum number of heteroatoms"

MIN_Het_C_Ratio 0.10   "Minimum heteroatom to carbon ratio"
MAX_Het_C_Ratio 1.1    "Maximum heteroatom to carbon ratio"

MIN_HALIDE_FRACTION 0.0  "Minimum Halide Fraction"
MAX_HALIDE_FRACTION 0.5  "Maximum Halide Fraction"

#count ring degrees of freedom = (#BondsInRing) - 4 - (RigidBondsInRing) - (BondsSharedWithOther)
#must be >= 0, from JCAMD 14:251-265,2000.
```

```
ADJUST_ROT_FOR_RING      true      "BOOLEAN for whether to estimate degrees of freedom in rings"

MIN_ROT_BONDS            0          "Minimum number of rotatable bonds"
MAX_ROT_BONDS            10         "Maximum number of rotatable bonds"

MIN_RIGID_BONDS          0          "Minimum number of rigid bonds"
MAX_RIGID_BONDS          25         "Maximum number of rigid bonds"

MIN_HBOND_DONORS         0          "Minimum number of hydrogen-bond donors"
MAX_HBOND_DONORS         4          "Maximum number of hydrogen-bond donors"

MIN_HBOND_ACCEPTORS      0          "Minimum number of hydrogen-bond acceptors"
MAX_HBOND_ACCEPTORS      6          "Maximum number of hydrogen-bond acceptors"

MIN_LIPINSKI_DONORS      0          "Minimum number of hydrogens on O & N atoms"
MAX_LIPINSKI_DONORS      5          "Maximum number of hydrogens on O & N atoms"

MIN_LIPINSKI_ACCEPTORS   0          "Minimum number of oxygen & nitrogen atoms"
MAX_LIPINSKI_ACCEPTORS   10         "Maximum number of oxygen & nitrogen atoms"

MIN_COUNT_FORMAL_CRG     0          "Minimum number formal charges"
MAX_COUNT_FORMAL_CRG     3          "Maximum number of formal charges"

MIN_SUM_FORMAL_CRG       -2         "Minimum sum of formal charges"
MAX_SUM_FORMAL_CRG       2          "Maximum sum of formal charges"

MIN_CHIRAL_CENTERS       0          "Minimum chiral centers"
MAX_CHIRAL_CENTERS       4          "Maximum chiral centers"

MIN_XLOGP                 -5.0      "Minimum XLogP"
MAX_XLOGP                  4.0      "Maximum XLogP"

#choices are insoluble<poorly<moderately<soluble<very<highly
MIN_SOLUBILITY            moderately "Minimum solubility"

PSA_USE_SandP             false     "Count S and P as polar atoms"
MIN_2D_PSA                0.0       "Minimum 2-Dimensional (SMILES) Polar Surface Area"
MAX_2D_PSA                150.0     "Maximum 2-Dimensional (SMILES) Polar Surface Area"

AGGREGATORS              true      "Eliminate known aggregators"
PRED_AGG                  true      "Eliminate predicted aggregators"

#secondary filters (based on multiple primary filters)
GSK_VEBER                 true      "PSA>140 or >10 rot bonds"
MAX_LIPINSKI              1          "Maximum number of Lipinski violations"
MIN_ABS                   0.5        "Minimum probability F>10% in rats"
PHARMACOPIA              true      "LogP > 5.88 or PSA > 131.6"

ALLOWED_ELEMENTS          H,C,N,O,F,S,Cl,Br
ELIMINATE_METALS          Sc,Ti,V,Cr,Mn,Fe,Co,Ni,Cu,Zn,Y,Zr,Nb,Mo,Tc,Ru,Rh,Pd,Ag,Cd

#acceptable molecules must have <= instances of each of the patterns below

#specific, undesirable functional groups

RULE 0 quinone
```

```
RULE 0 pentafluorophenyl_esters
RULE 0 paranitrophenyl_esters
RULE 0 HOBt_esters
RULE 0 triflates
RULE 0 lawesson_s_reagent
RULE 0 phosphoramides
RULE 0 beta_carbonyl_quat_nitrogen
RULE 0 acylhydrazide
RULE 0 cation_C_Cl_I_P_or_S
RULE 0 phosphoryl
RULE 0 alkyl_phosphate
RULE 0 phosphinic_acid
RULE 0 phosphanes
RULE 0 phosphoranes
RULE 0 imidoyl_chlorides
RULE 0 nitroso
RULE 0 N_P_S_Halides
RULE 0 carbodiimide
RULE 0 isonitrile
RULE 0 triacyloxime
RULE 0 cyanohydrins
RULE 0 acyl_cyanides
RULE 0 sulfonylnitrile
RULE 0 phosphonylnitrile
RULE 0 azocyanamides
RULE 0 beta_azo_carbonyl
RULE 0 polyenes
RULE 0 saponin_derivatives
RULE 1 cytochalasin_derivatives
RULE 4 cycloheximide_derivatives
RULE 1 monensin_derivatives
RULE 1 squalestatin_derivatives
```

#functional groups which often eliminate compounds from consideration

```
RULE 0 acid_halide
RULE 0 aldehyde
RULE 0 alkyl_halide
RULE 0 anhydride
RULE 0 azide
RULE 0 azo
RULE 0 di_peptide
RULE 0 michael_acceptor
RULE 0 beta_halo_carbonyl
RULE 0 nitro
RULE 0 oxygen_cation
RULE 0 peroxide
RULE 0 phosphonic_acid
RULE 0 phosphonic_ester
RULE 0 phosphoric_acid
RULE 0 phosphoric_ester
RULE 0 sulfonic_acid
RULE 0 sulfonic_ester
RULE 0 tricarbo_phosphene
RULE 0 epoxide
RULE 0 sulfonyl_halide
```

```
RULE 0 halopyrimidine
RULE 0 perhalo_ketone
RULE 0 aziridine
RULE 1 oxalyl
RULE 0 alphahalo_amine
RULE 0 halo_amine
RULE 0 halo_alkene
RULE 0 acyclic_NCN
RULE 0 acyclic_NS
RULE 0 SCN2
RULE 0 terminal_vinyl
RULE 0 hetero_hetero
RULE 0 hydrazine
RULE 0 N_methoyl
RULE 0 NS_beta_halothyl
RULE 0 propiolactones
RULE 0 nitroso
RULE 0 iodoso
RULE 0 iodoxy
RULE 0 noxide
```

```
#groups of molecules
```

```
RULE 0 dye
```

```
#functional groups which are allowed, but may not be wanted in high quantities
#common functional groups
```

```
RULE 6 alcohol
RULE 4 alkene
RULE 4 amide
RULE 4 amino_acid
RULE 2 amine
RULE 4 primary_amine
RULE 4 secondary_amine
RULE 4 tertiary_amine
RULE 2 carboxylic_acid
RULE 6 halide
RULE 0 iodine
RULE 2 ketone
RULE 4 phenol
RULE 1 imine
RULE 1 methyl_ketone
RULE 1 alkylaniline
RULE 4 sulfonamide
RULE 1 sulfonylurea
RULE 0 phosphonamide
RULE 0 alphahalo_ketone
RULE 0 oxaziridine
RULE 1 cyclopropyl
RULE 2 guanidine
RULE 0 sulfonimine
RULE 0 sulfinimine
RULE 1 hydroxamic_acid
RULE 0 phosphoryl
```

```
RULE 0 sulfinylthio
RULE 0 disulfide
RULE 0 enol_ether
RULE 0 enamine
RULE 0 organometallic
RULE 0 dithioacetal
RULE 1 oxime
RULE 0 isothiocyanate
RULE 0 isocyanate
RULE 3 lactone
RULE 3 lactam
RULE 1 thioester
RULE 1 carbonate
RULE 0 carbamic_acid
RULE 1 thiocarbamate
RULE 0 triazine
RULE 1 malonic
```

#other functional groups

```
RULE 2 alkyne
RULE 4 aniline
RULE 4 aryl_halide
RULE 2 carbamate
RULE 3 ester
RULE 5 ether
RULE 1 hydrazone
RULE 0 nonacylhydrazone
RULE 1 hydroxylamine
RULE 2 nitrile
RULE 2 sulfide
RULE 2 sulfone
RULE 2 sulfoxide
RULE 1 thiourea
RULE 1 thioamide
RULE 1 thiol
RULE 2 urea
```

```
RULE 0 hemiketal
RULE 0 hemiacetal
RULE 0 ketal
RULE 1 acetal
RULE 0 aminal
RULE 0 hemiaminal
```

#protecting groups

```
RULE 0 benzyloxycarbonyl_CBZ
RULE 0 t_butoxycarbonyl_tBOC
RULE 0 fluorenylmethoxycarbonyl_Fmoc
RULE 1 dioxolane_5MR
RULE 1 dioxane_6MR
RULE 1 tetrahydropyran_THP
RULE 1 methoxyethoxymethyl_MEM
RULE 2 benzyl_ether
RULE 2 t_butyl_ether
```

RULE 0 trimethylsilyl_TMS
RULE 0 t_butyldimethylsilyl_TBDMS
RULE 0 triisopropylsilyl_TIPS
RULE 0 t_butyldiphenylsilyl_TBDPS
RULE 1 phthalimides_PHT
RULE 2 arenesulfonyl

Drug-like filter

```
#!/*****
#Copyright (C) 2000-2005 by OpenEye Scientific Software, Inc.
#*****/
#This file defines the rules for filtering multi-structure files based on
#properties and substructure patterns.
MIN_MOLWT      200      "Minimum molecular weight"
MAX_MOLWT      600      "Maximum molecular weight"

MIN_NUM_HVY    15      "Minimum number of heavy atoms"
MAX_NUM_HVY    35      "Maximum number of heavy atoms"

MIN_RING_SYS   0       "Minimum number of ring systems"
MAX_RING_SYS   5       "Maximum number of ring systems"

MIN_RING_SIZE  0       "Minimum atoms in any ring system"
MAX_RING_SIZE  20      "Maximum atoms in any ring system"

MIN_CON_NON_RING  0     "Minimum number of connected non-ring atoms"
MAX_CON_NON_RING  15    "Maximum number of connected non-ring atoms"

MIN_FCNGRP     0       "Minimum number of functional groups"
MAX_FCNGRP     18      "Maximum number of functional groups"

MIN_UNBRANCHED 0       "Minimum number of connected unbranched non-ring atoms"
MAX_UNBRANCHED 6       "Maximum number of connected unbranched non-ring atoms"

MIN_CARBONS    7       "Minimum number of carbons"
MAX_CARBONS    35      "Maximum number of carbons"

MIN_HETEROATOMS 2      "Minimum number of heteroatoms"
MAX_HETEROATOMS 20     "Maximum number of heteroatoms"

MIN_Het_C_Ratio 0.10   "Minimum heteroatom to carbon ratio"
MAX_Het_C_Ratio 1.0    "Maximum heteroatom to carbon ratio"

MIN_HALIDE_FRACTION 0.0  "Minimum Halide Fraction"
MAX_HALIDE_FRACTION 0.5  "Maximum Halide Fraction"

#count ring degrees of freedom = (#BondsInRing) - 4 - (RigidBondsInRing) - (BondsSharedWithOther)
#must be >= 0, from JCAMD 14:251-265,2000.
```

```
ADJUST_ROT_FOR_RING      true      "BOOLEAN for whether to estimate degrees of freedom in rings"

MIN_ROT_BONDS            0          "Minimum number of rotatable bonds"
MAX_ROT_BONDS            20         "Maximum number of rotatable bonds"

MIN_RIGID_BONDS          0          "Minimum number of rigid bonds"
MAX_RIGID_BONDS          35         "Maximum number of rigid bonds"

MIN_HBOND_DONORS         0          "Minimum number of hydrogen-bond donors"
MAX_HBOND_DONORS         6          "Maximum number of hydrogen-bond donors"

MIN_HBOND_ACCEPTORS      0          "Minimum number of hydrogen-bond acceptors"
MAX_HBOND_ACCEPTORS      8          "Maximum number of hydrogen-bond acceptors"

MIN_LIPINSKI_DONORS      0          "Minimum number of hydrogens on O & N atoms"
MAX_LIPINSKI_DONORS      5          "Maximum number of hydrogens on O & N atoms"

MIN_LIPINSKI_ACCEPTORS   0          "Minimum number of oxygen & nitrogen atoms"
MAX_LIPINSKI_ACCEPTORS   10         "Maximum number of oxygen & nitrogen atoms"

MIN_COUNT_FORMAL_CRG     0          "Minimum number formal charges"
MAX_COUNT_FORMAL_CRG     3          "Maximum number of formal charges"

MIN_SUM_FORMAL_CRG       -2         "Minimum sum of formal charges"
MAX_SUM_FORMAL_CRG       2          "Maximum sum of formal charges"

MIN_CHIRAL_CENTERS       0          "Minimum chiral centers"
MAX_CHIRAL_CENTERS       4          "Maximum chiral centers"

MIN_XLOGP                 -5.0      "Minimum XLogP"
MAX_XLOGP                  6.0      "Maximum XLogP"

#choices are insoluble<poorly<moderately<soluble<very<highly
MIN_SOLUBILITY             moderately "Minimum solubility"

PSA_USE_SandP             false     "Count S and P as polar atoms"
MIN_2D_PSA                 0.0      "Minimum 2-Dimensional (SMILES) Polar Surface Area"
MAX_2D_PSA                 150.0    "Maximum 2-Dimensional (SMILES) Polar Surface Area"

AGGREGATORS               true     "Eliminate known aggregators"
PRED_AGG                   true     "Eliminate predicted aggregators"

#secondary filters (based on multiple primary filters)
GSK_VEBER                 true     "PSA>140 or >10 rot bonds"
MAX_LIPINSKI               1          "Maximum number of Lipinski violations"
MIN_ABS                    0.5        "Minimum probability F>10% in rats"
PHARMACOPIA               true     "LogP > 5.88 or PSA > 131.6"

ALLOWED_ELEMENTS          H,C,N,O,F,S,Cl,Br
ELIMINATE_METALS          Sc,Ti,V,Cr,Mn,Fe,Co,Ni,Cu,Zn,Y,Zr,Nb,Mo,Tc,Ru,Rh,Pd,Ag,Cd

#acceptable molecules must have <= instances of each of the patterns below

#specific, undesirable functional groups

RULE 0 quinone
```

```
RULE 0 pentafluorophenyl_esters
RULE 0 paranitrophenyl_esters
RULE 0 HOBt_esters
RULE 0 triflates
RULE 0 lawesson_s_reagent
RULE 0 phosphoramides
RULE 0 beta_carbonyl_quat_nitrogen
RULE 0 acylhydrazide
RULE 0 cation_C_Cl_I_P_or_S
RULE 0 phosphoryl
RULE 0 alkyl_phosphate
RULE 0 phosphinic_acid
RULE 0 phosphanes
RULE 0 phosphoranes
RULE 0 imidoyl_chlorides
RULE 0 nitroso
RULE 0 N_P_S_Halides
RULE 0 carbodiimide
RULE 0 isonitrile
RULE 0 triacyloxime
RULE 0 cyanohydrins
RULE 0 acyl_cyanides
RULE 0 sulfonylnitrile
RULE 0 phosphonylnitrile
RULE 0 azocyanamides
RULE 0 beta_azo_carbonyl
RULE 0 polyenes
RULE 0 saponin_derivatives
RULE 1 cytochalasin_derivatives
RULE 4 cycloheximide_derivatives
RULE 1 monensin_derivatives
RULE 1 squalestatin_derivatives
```

#functional groups which often eliminate compounds from consideration

```
RULE 0 acid_halide
RULE 0 aldehyde
RULE 0 alkyl_halide
RULE 0 anhydride
RULE 0 azide
RULE 0 azo
RULE 0 di_peptide
RULE 0 michael_acceptor
RULE 0 beta_halo_carbonyl
RULE 0 nitro
RULE 0 oxygen_cation
RULE 0 peroxide
RULE 0 phosphonic_acid
RULE 0 phosphonic_ester
RULE 0 phosphoric_acid
RULE 0 phosphoric_ester
RULE 0 sulfonic_acid
RULE 0 sulfonic_ester
RULE 0 tricarbo_phosphene
RULE 0 epoxide
RULE 0 sulfonyl_halide
```

```
RULE 0 halopyrimidine
RULE 0 perhalo_ketone
RULE 0 aziridine
RULE 1 oxalyl
RULE 0 alphahalo_amine
RULE 0 halo_amine
RULE 0 halo_alkene
RULE 0 acyclic_NCN
RULE 0 acyclic_NS
RULE 0 SCN2
RULE 0 terminal_vinyl
RULE 0 hetero_hetero
RULE 0 hydrazine
RULE 0 N_methoyl
RULE 0 NS_beta_haloethyl
RULE 0 propiolactones
RULE 0 nitroso
RULE 0 iodoso
RULE 0 iodoxy
RULE 0 noxide
```

```
#groups of molecules
```

```
RULE 0 dye
```

```
#functional groups which are allowed, but may not be wanted in high quantities
#common functional groups
```

```
RULE 6 alcohol
RULE 4 alkene
RULE 4 amide
RULE 4 amino_acid
RULE 2 amine
RULE 4 primary_amine
RULE 4 secondary_amine
RULE 4 tertiary_amine
RULE 2 carboxylic_acid
RULE 6 halide
RULE 0 iodine
RULE 2 ketone
RULE 4 phenol
RULE 1 imine
RULE 1 methyl_ketone
RULE 1 alkylaniline
RULE 4 sulfonamide
RULE 1 sulfonylurea
RULE 0 phosphonamide
RULE 0 alphahalo_ketone
RULE 0 oxaziridine
RULE 1 cyclopropyl
RULE 2 guanidine
RULE 0 sulfonimine
RULE 0 sulfinimine
RULE 1 hydroxamic_acid
RULE 0 phosphoryl
```

```
RULE 0 sulfinylthio
RULE 0 disulfide
RULE 0 enol_ether
RULE 0 enamine
RULE 0 organometallic
RULE 0 dithioacetal
RULE 1 oxime
RULE 0 isothiocyanate
RULE 0 isocyanate
RULE 3 lactone
RULE 3 lactam
RULE 1 thioester
RULE 1 carbonate
RULE 0 carbamic_acid
RULE 1 thiocarbamate
RULE 0 triazine
RULE 1 malonic
```

#other functional groups

```
RULE 2 alkyne
RULE 4 aniline
RULE 4 aryl_halide
RULE 2 carbamate
RULE 3 ester
RULE 5 ether
RULE 1 hydrazone
RULE 0 nonacylhydrazone
RULE 1 hydroxylamine
RULE 2 nitrile
RULE 2 sulfide
RULE 2 sulfone
RULE 2 sulfoxide
RULE 0 thiourea
RULE 1 thioamide
RULE 1 thiol
RULE 2 urea
```

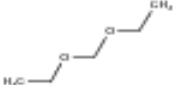

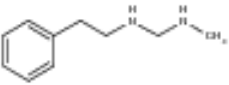

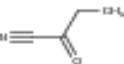
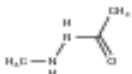



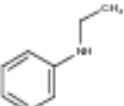

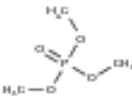

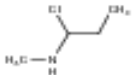
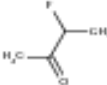
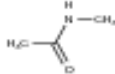
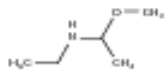

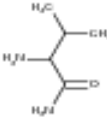
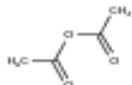
```
RULE 0 hemiketal
RULE 0 hemiacetal
RULE 0 ketal
RULE 1 acetal
RULE 0 aminal
RULE 0 hemiaminal
```


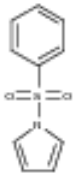

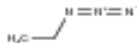

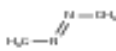

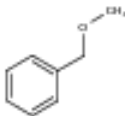
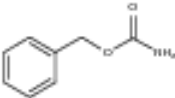
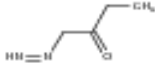
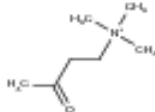
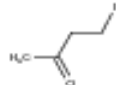
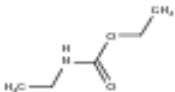
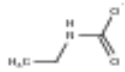

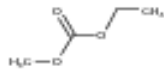

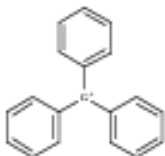


#protecting groups

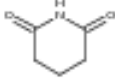

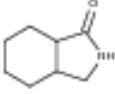


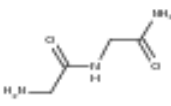


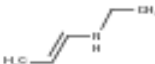
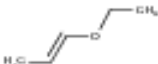

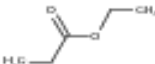
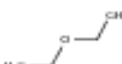
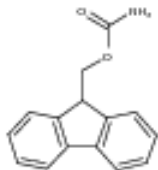
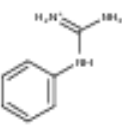




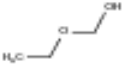
```
RULE 0 benzyloxycarbonyl_CBZ
RULE 0 t_butoxycarbonyl_tBOC
RULE 0 fluorenylmethoxycarbonyl_Fmoc
RULE 1 dioxolane_5MR
RULE 1 dioxane_6MR
RULE 1 tetrahydropyran_THP
RULE 1 methoxyethoxymethyl_MEM
RULE 2 benzyl_ether
RULE 2 t_butyl_ether
```

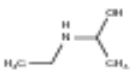

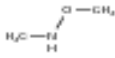
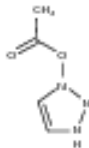
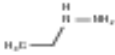
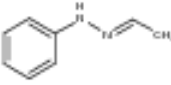
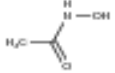





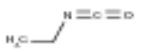

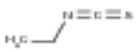
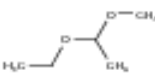
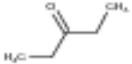



RULE 0 trimethylsilyl_TMS
RULE 0 t_butyldimethylsilyl_TBDMS
RULE 0 triisopropylsilyl_TIPS
RULE 0 t_butyldiphenylsilyl_TBDPS
RULE 1 phthalimides_PHT
RULE 2 arenesulfonyl

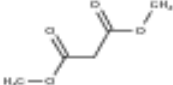
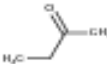
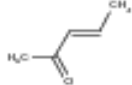
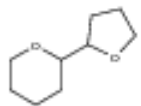

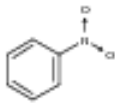
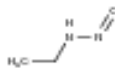

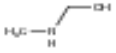
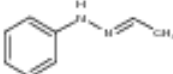




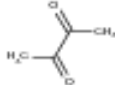



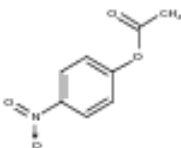
Example Substructure hits

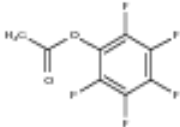





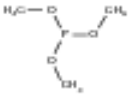

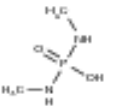

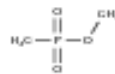
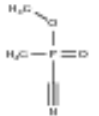
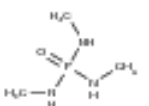


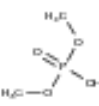
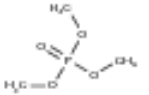
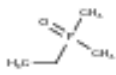

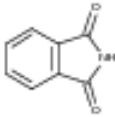
<p>acetal</p> 	<p>acid_halide</p> 	<p>acyclic_NCN</p> 	<p>acyclic_NS</p> 
<p>Acyl_cyanides</p> 	<p>Acyhydrazide</p> 	<p>alcohol</p> 	<p>aldehyde</p> 
<p>alkene</p> 	<p>alkylaniline</p> 	<p>alkyl_halide</p> 	<p>Alkyl_phosphate</p> 
<p>alkyne</p> 	<p>alphahalo_amine</p> 	<p>alphahalo_ketone</p> 	<p>amide</p> 
<p>aminal</p> 	<p>amine</p> 	<p>amino_acid</p> 	<p>anhydride</p> 

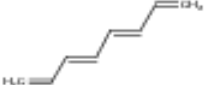
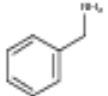

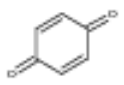
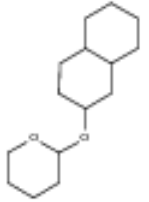
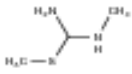
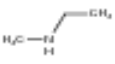


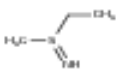
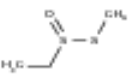
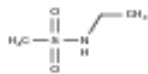
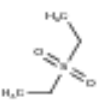

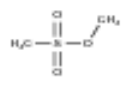
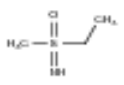

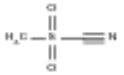
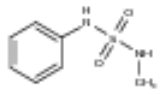
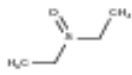
<p>aniline</p> 	<p>arenesulfonyl</p> 	<p>aryl_halide</p> 	<p>azide</p> 
<p>aziridine</p> 	<p>azo</p> 	<p>Azocyanamides</p> 	<p>benzyl_ether</p> 
<p>benzyloxycarbonyl_CBZ</p> 	<p>beta_azo_carbonyl</p> 	<p>Beta_carbonyl_quart_nitr</p> 	<p>beta_halo_carbonyl</p> 
<p>carbamate</p> 	<p>carbamic_acid</p> 	<p>Carbodiimide</p> 	<p>carbonate</p> 
<p>carboxylic_acid</p> 	<p>Cation_C_Cl_I_P_or_S</p> 	<p>Chloramidines</p> 	<p>Cyanohydrins</p> 

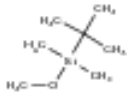
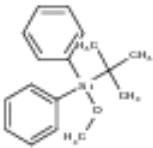
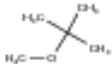
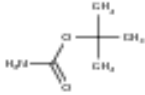
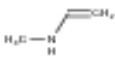
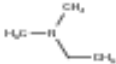

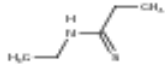
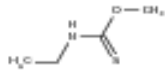
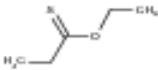

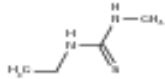
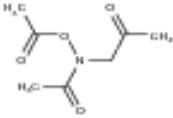
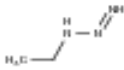

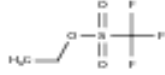
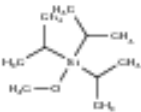
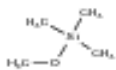
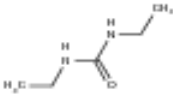
Cycloheximide_derivative 	cyclopropyl 	Cytochalasin_derivatives 	dioxane_6MR 
dioxolane_5MR 	di_peptide 	disulfide 	dithioacetal 
enamine 	enol_ether 	epoxide 	ester 
ether 	fluorenylmethoxycarbonyl 	guanidine 	halide 
halo_alkene 	halo_amine 	halopyrimidine 	hemiacetal 

hemiaminal	hemiketal	hetero_hetero	HOBT_esters
			
hydrazine	hydrazone	hydroxamic_acid	hydroxylamine
			
imine	iodine	iodoso	iodoxy
			
isocyanate	isonitrile	isothiocyanate	ketal
			
ketone	lactam	lactone	Lawesson_s_reagent
			

malonic 	methoxyethoxymethyl_M 	methyl_ketone 	michael_acceptor 
Monensin_derivatives 	nitrile 	nitro 	nitroso 
Nitroso 	N_methyl 	nonacylhydrazone 	noxide 
N_P_S_Halides 	NS_beta_halothyl 	organometallic 	oxalyl 
oxaziridine 	oxime 	oxygen_cation 	Paranitrophenyl_esters 

<p>Pentafluorophenyl_esters</p> 	<p>perhalo_ketone</p> 	<p>peroxide</p> 	<p>phenol</p> 
<p>Phosphanes</p> 	<p>Phosphanes</p> 	<p>Phosphanes(Phosphite)</p> 	<p>Phosphinic_acid</p> 
<p>phosphoramidate</p> 	<p>phosphonic_acid</p> 	<p>phosphonic_ester</p> 	<p>Phosphonylnitrile</p> 
<p>Phosphoramides</p> 	<p>Phosphoranes</p> 	<p>phosphoric_acid</p> 	<p>phosphoric_ester</p> 
<p>phosphoric_ester</p> 	<p>phosphoryl</p> 	<p>Phosphoryl</p> 	<p>phthalimides_PHT</p> 

<p>Polyenes</p> 	<p>primary_amine</p> 	<p>propiolactones</p> 	<p>quinone</p> 
<p>Saponin_derivatives</p> 	<p>SCN2</p> 	<p>secondary_amine</p> 	<p>Squalestatin_derivatives</p> 
<p>sulfide</p> 	<p>sulfinimine</p> 	<p>sulfinylthio</p> 	<p>sulfonamide</p> 
<p>sulfone</p> 	<p>sulfonic_acid</p> 	<p>sulfonic_ester</p> 	<p>sulfinimine</p> 
<p>sulfonyl_halide</p> 	<p>Sulfonylnitrile</p> 	<p>sulfonylurea</p> 	<p>sulfoxide</p> 

t_butyl(dimethylsilyl)_TBD	t_butyl(diphenylsilyl)_TBDP	t_butyl_ether	t_butyloxycarbonyl_tBOC
			
terminal_vinyl	tertiary_amine	tetrahydropyran_THP	thioamide
			
thiocarbamate	thioester	thiol	thiourea
			
Triacyloxime	triazine	tricarbo_phosphene	Triflates
			
triisopropylsilyl_TIPS	trimethylsilyl_TMS	urea	
			

BIBLIOGRAPHY

- [1] Lipinski, C., et al., "Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings.", *Adv. Drug Deliv. Rev.*, 23:3, 1997.
- [2] McGovern S.L., et al., "A specific mechanism of nonspecific inhibition", *J. Med. Chem.*, 46:4265-72, 2003;
- [3] Seidler J., et al., "Identification and prediction of promiscuous aggregating inhibitors among known drugs", *J. Med. Chem.*, 46:4477-86, 2003.
- [4] Oprea, T., "Property distribution of drug-related chemical databases.", *J. Comput. Aid. Mol. Des.*, 14:251-264, 2000.
- [5] Wang, R., Ying, F., and Lai, L., "A new atom-additive method for calculating partition coefficients", *J. Chem. Inf. Comput. Sci.*, 37:615-621, 1997.
- [6] Ertl, P., Rohde, B., and Selzer, P., "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, 2000.
- [7] Clark, D.E., "Rapid calculation of polar molecular surface area and its application to the prediction of transport phenomena. 1. Prediction of intestinal absorption", *J. Pharm. Sci.*, 88:807-814, 1999.
- [8] Martin, Y.C., "A bioavailability score", *J. Med. Chem.*, 48:3164-3170, 2005.
- [9] Veber, D.F., Johnson, S.R., Cheng, H.Y., Smith, B.R., Ward, K.W., Kopple, K.D., "Molecular properties that influence the oral bioavailability of drug candidates", *J. Med. Chem.*, 45:2615-2623, 2002.
- [10] Egan, W.J., Merz, K.M., Baldwin, J.J., "Prediction of drug absorption using multivariate statistics", *J. Med. Chem.*, 43:3867-3877, 2000.
- [11] Rishton, G.M., "Nonleadlikeness and leadlikeness in biochemical screening", *Drug. Disc. Today*, 8:86-96 2003.
- [12] Yalkowsky, S.H., Valvani, S.C., "Solubility and partitioning 1: Solubility of nonelectrolytes in water", *J. Pharm. Sci.* 69:912 1980.

-
- [13] J.E.J. Mills and P.M. Dean, “Three-dimensional hydrogen-bond geometry and probability information from a crystal survey”, *J Comput-Aided Mol Des*, 10:607, 1996.
- [14] George A. Jeffrey, An Introduction to Hydrogen Bonding, Oxford University Press, 1997.
- [15] Wishart, D.S., “DrugBank: a comprehensive resource for in silico drug discovery and exploration”, *Nucleic Acids Res.* 1:34 2006.