Simsite3D Software Installation Guide version 3.0

Protein Structural Analysis & Design Lab
MSU
J. Van Voorst and L. Kuhn

This guide shows how to install and configure the SimSite3D software[1] tools. Text in <angular brackets> should be replaced by actual names provided by the user (without the angular brackets). Text in [square brackets] denotes optional parameters to be replaced by actual names provided by the user. For help with installation and usage, or to provide feedback on the software, please contact Jeffrey Van Voorst (vanvoor4@msu.edu) or Leslie Kuhn (KuhnL@msu.edu).

1 Setting up the SimSite3D environment

The following steps show how to configure SimSite3D. There are four increasingly user specific levels where the variables used by the SimSite3D tools may be set. The first level is the system wide configuration file /etc/simsite3d/simsite3d.conf. The second level is the local configuration file ~/.simsite3d/simsite3d.conf. The third level includes the SimSite3D variables that are set in your environment (e.g. global environment variables and the variables set in your .cshrc file). The final level checked is the options set on the command line when you create a sitemap or run a binding site search. Please note that any variable set in a more user specific level will overwrite all previously assigned values to that variable.

(Note: the examples for setting environment variables assume your shell is the tcsh shell. If you use a significantly different shell, such as bash, please consult the documentation for setting environment variables in your shell or consult your systems administrator).

A. Setting the SIMSITE3D_INSTALL_DIR Environment Variable

There is one exception to the levels of the SimSite3D environment. The environment variable **$SIMSITE3D_INSTALL_DIR** must be set either by the system or in your local environment (or both). **$SIMSITE3D_INSTALL_DIR** may not be set in an SimSite3D configuration file (i.e. an simsite3d.conf file). You may set **$SIMSITE3D_INSTALL_DIR** locally by editing your .cshrc file and adding the following line:

```
setenv SIMSITE3D_INSTALL_DIR </path/to/SimSite3D/installation> (the directory
       where SimSite3D tools and configuration files are to be installed or are
       installed , e.g., /opt/SimSite3D_Software).
```

B. Setting the System Defaults in an SimSite3D Configuration File

The system default values for the SimSite3D runtime environment may be set in the /etc/simsite3d/simsite3d.conf file; likewise, local values for the SimSite3D variable may be set in ~/.simsite3d/simsite3d.conf. An simsite3d.conf file takes the format of one variable name per

[1] Note: The name for SimSite3D at MSU is SimSite3D.

line. The format of a line is the variable name followed by one or more whitespace characters (such as spaces or tabs) and followed by the value of the variable. Both the '#' and the '%' characters are used to denote comments, and all characters after the first comment character on a line will be ignored.

The following variables are supported. Each variable (except for **$SIMSITE3D_SCRATCH_DIR**) must have an assigned value before creating sitemaps and searching. The format of the following list is the name of the variable followed by a comment explaining the expected directory to list as the value of the variable.

- **$SIMSITE3D_DBASE_SITES**: directory holding the searchable database of SimSite3D database sitemaps (files)
- **$SIMSITE3D_DBASE_LIGS**: directory holding the ligand files (in mol2 format) corresponding to the SimSite3D database sitemaps
- **$SIMSITE3D_DBASE_PROTS**: directory holding the protein crystal structure files corresponding to the SimSite3D database sitemaps
- **$SIMSITE3D_DIVERSE_SITES**: directory holding the diverse set of sitemaps (distributed as part of SimSite3D) used to normalize the scores SimSite3D assigns to hits
- **$SIMSITE3D_DIVERSE_LIGS**: directory holding the ligands corresponding to the diverse set of sitemaps
- **$SIMSITE3D_SCRATCH_DIR**: if this variable is set, use the given directory as the base directory for temporary files and directories used by SimSite3D (during the creation of sitemaps and running searches)
- **$SIMSITE3D_PROJ_OUTPUT**: directory to hold the results of your searches, and if **$SIMSITE3D_SCRATCH_DIR** is **not** set, the given directory will also be used for the creation of temporary directories and files. Note that by default, the results of a search will be in a subdirectory of **$SIMSITE3D_PROJ_OUPUT**.

An example simsite3d.conf file, named simsite3d.conf.example, may be found in the **$SIMSITE3D_INSTALL_DIR**/SimSite3DSoftParams directory.

If you prefer to use the configuration files as specified in this section and will not be using environment variables you may skip over section **C**.

C. Configuring SimSite3D using Environment Variables

Environment variables may used to specify values for the SimSite3D runtime variables (parameters). Any variable set in your environment will take precedence over the variables set in the simsite3d.conf files. The list of supported variables and their expected values are given in section **B**.

As an example, suppose you want your output in most cases to be placed in the directory /home/uid/simsite3d/search_results; to set this in your environment, you would place the following line in your .cshrc file (in your home directory):

```
setenv SIMSITE3D_PROJ_OUTPUT </home/uid/simsite3d/search_results>
```

Then, in the window(s) in which you'll be installing the software and generating or comparing sitemaps run the following command to update your environment:

```
source ~/.cshrc
```

D. File and Directory Permissions

One item to consider is the permissions on the directories used by SimSite3D. SimSite3D users require "cd" and read permission for all SimSite3D directories (listed in section **B**) and require read permissions for all files in the SimSite3D directories. In addition, a user generating a searchable sitemaps database (e.g. a user who wishes to build all sitemaps from the proteins in **$SIMSITE3D_DBASE_PROTS** and ligands in **$SIMSITE3D_DBASE_LIGS** and storing them in **$SIMSITE3D_DBASE_SITES**) will require write access to the **$SIMSITE3D_DBASE_SITES** directory and the ability to create files in **$SIMSITE3D_DBASE_SITES**.

2 Building and installing the SimSite3D Software tools

After setting the **$SIMSITE3D_INSTALL_DIR** environment variable to the desired installation directory, a default installation of SimSite3D software can be done automatically by running the install.py script (from the SimSite3D source directory). The install script will install SimSite3D in **$SIMSITE3D_INSTALL_DIR** and will install the tools in the directory **$SIMSITE3D_INSTALL_DIR**/bin:

```
./install.py  (The preceding "./" is important for the system to locate the
     file, even when it is in your working directory.)
```

NOTE: If you are planning to build SimSite3D Software on a system where the default for GCC is to build 64 bit executables and you desire to build 32 bit executables, you can achieve this by editing the file configure.ac (in the simsite3d_software directory) and adding "-m32" to the CFLAGS line.

Use gzip and tar to unpack the scoring normalization dataset files, `data/diverse_sitemaps.tbz2` and `data/diverse_ligands.tbz2`, into **$SIMSITE3D_DBASE_SITES** and **$SIMSITE3D_DBASE_LIGS** respectively.

For details on custom installations, please see the Appendix section II.

3 Other Considerations

File naming convention

The SimSite3D protein, ligand and sitemap file naming scheme is:

- XXXXXX_p.pdb for proteins (e.g., `HIV-pr_K41R_V64I_0200758_LIG_1_p.pdb)`
- XXXXXX_l.mol2 for ligands (e.g., `HIV-pr_K41R_V64I_0200758_LIG_1_l.mol2)`
- Sitemaps files generated by SimSite3D corresponding to the ligand binding site (`e.g., LIG_1`) in the given protein, where XXXXXX is the Pfizer crystal structure name
    - XXXXXX_s.csv is the sitemap labels file (e.g., `HIV-pr_K41R_V64I_0200758_LIG_1_s.csv)`

    - XXXXXX_s.pdb contains the sitemap interaction points
    - XXXXXX_a.pdb holds the atoms from XXXXXX_p.pdb which have a corresponding sitemap interaction point (in XXXXXX_s.pdb)
    - XXXXXX_rad.pdb contains those residues from the protein (XXXXXX_p.pdb) that have at least one heavy atom within 4.5 Å of any atom in the ligand (XXXXXX_l.mol2.)
- Ligand files generated by SimSite3D during a search have the following naming
    - XXXXXX_NNNNN_l.mol2 is the ligand from the database structure XXXXXX in the reference frame of the query pocket given by the Nth hit from XXXXXX's pocket to the query pocket; where NNNNN is N written as 5 digits (zero padded) (e.g, `HIV-pr_K41R_V64I_0200758_LIG_1_00012_f.mol2`)
    - XXXXXX_NNNNN_f.mol2 is the fragment(s) of the database ligand (from XXXXXX); that is the portion of XXXXXX_NNNNN_l.mol2 that "fit" in the query's pocket.

Simultaneous SimSite3D searches

A number of external protein-ligand scoring function programs do not allow a user to provide a name for the output file(s). If multiple searches (search_sitemaps runs) are conducted simultaneously and such a protein-ligand scoring function is used, the search_sitemaps runs must be carried out with different **$SIMSITE3D_PROJ_OUTPUT** directories or the results might be undefined. As an example, DrugScore 1.2 uses the ligand filename as part of the DrugScore output filenames; if two or more searches are run at the same time and in the same directory the searches might overwrite or delete their neighbors' output files.

4 External Protein-Ligand Scoring Functions

To aid in the evaluation of hits, an external protein-ligand scoring function can be used to give an initial approximation on the compatibility of the the database ligand fragment for the query binding site, considering the atomic interactions rather than sitemap (pharmacophore-like) correspondences.

Specifying Available Protein-Ligand Scoring Functions:

The interface to the protein-ligand scoring functions is via a comma separated text file (ext_prot_lig_score_fcns.conf) in the **$SIMSITE3D_INSTALL_DIR**/SimSite3DSoftParams directory. Each non comment line in the protein-ligand scoring functions file should have exactly 3 fields or it will be ignored. The fields are:

1. The first field is to provide a unique name to the scoring function as it should be specified on the command line to search_sitemaps. If two lines have the same value (name) in the first field, only the first line with that name will be used.
2. This field is used to specify (internally to SimSite3D) which type of scoring function will be called. SimSite3D 3.0 supports DrugScore and has an SFCscore interface that needs a bit more testing.
3. The last field holds the explicit command line to run the protein-ligand scoring function where the protein and ligand file names are replaced with the variables

$PROTEIN and **$LIGAND** respectively. As an example, to run DrugScore in our lab we might use

```
“/soft/linux/drugscore/bin/drugscore.lnx PAIR 1b38_atp_p.pdb
     1b38_atp_l.mol2”
```

to score ATP with respect to cdk2 in the PDB structure 1b38. In this case, the command to place in this field for our DrugScore line would be

```
“/soft/linux/drugscore/bin/drugscore.lnx PAIR $PROTEIN $LIGAND”
```

The first scoring function specified in the ext_prot_lig_score_fcns.conf file is taken to be the default external scoring function. Besides the supported scoring functions, there is a special scoring function with the name “NONE” in the first field; the “NONE” scoring function implies that no protein-ligand scoring function is to be used.

Supported Protein-Ligand Scoring Functions:

The following protein-ligand scoring functions are supported

1. NONE: This scoring function is a place holder used to indicate the desire to not use an external protein-ligand scoring function.
2. DrugScore 1.2 PAIR.
3. SFCscore

Adding New Protein-Ligand Scoring Functions:

Adding additional protein-ligand scoring functions requires parsing of the scoring function output files. To do so, add a line to the ext_prot_lig_score_fcns.conf file and write a C++ derived class of the C++ class ExternalScoringFunction that can determine the name of the score file (from the new protein-ligand scoring function) and parse the score file to obtain the score(s).

Notes about SFCscore:

Based on discussions between the Protein Structural Analysis and Design Lab at Michigan State and the Pocket Mining Group in CADD at Pfizer Ann Arbor, the “default” SFCscore .spf file included with SFCscore will result in SFCscore scoring protein-ligand interactions in approximately the same as manner as DrugScore 1.2 PAIR scores protein-ligand interations. In particular, both the default SFCscore and DrugScore 1.2 PAIR scoring functions consider protein-ligand contacts alone. One is not limited to using a specific .spf file with SimSite3D, and one method of supporting multiple SFCscore scoring methods would be to create a .spf file for each method. Note: at the present SimSite3D calls the scoring functions for each ligand fragment (rather than concatenating ligands in the same coordinate file).

Appendix I: SimSite3D Software operating system and software requirements

SimSite3D Software has been built and tested using the following tools:

Tool	***Versions***

Tool	***Versions***
GNU Automake	1.6.3, 1.9.6, 1.10
GNU Autoconf	2.57, 2.59, 2.61
GNU Make	3.79.1, 3.81, 3.81
GNU libtool	1.4.3, 1.5.22, 1.5.24
Perl	5.8.0, 5.8.8, 5.8.8
Python	2.2.2, 2.4.3, 2.5
GCC	3.2.2, 4.1.2, 4.1.2
Operation Systems (Linux Distributions)	Redhat 9, RHEL v5 WS, Gentoo
DrugScore	1.2

Appendix II: Custom installation notes

To force the building of 32 bit binaries on 64 bit machines, edit the `configure.ac` file and add the "-m32" option to the CFLAGS parameter.

A custom build of the C/C++ tools requires the following commands (in the order given) to use the auto configuration tools.

- aclocal
- libtoolize –copy –force
- automake -a -c
- autoconf
- ./configure –prefix=**$SIMSITE3D_INSTALL_DIR**
- make install

When performing a custom installation, please copy the python utilities in the python directory to the desired location and recursively copy the python directory to the **$SIMSITE3D_INSTALL_DIR**`/python` directory. Use gzip and tar to unpack the scoring normalization dataset files, `data/diverse_sitemaps.tgz` and `data/diverse_ligands.tgz`, into **$SIMSITE3D_DBASE_SITES** and **$SIMSITE3D_DBASE_LIGS** respectively. Copy the datafiles from the `params` directory to **$SIMSITE3D_INSTALL_DIR**`/SimSite3DSoftParams`.

The C++ and Python code for SimSite3D is designed to be relatively portable and should build with minimal effort on any recent Linux distribution. In addition, the C++ code is expected to build on any recent *nix distribution except possibly for the Timer class and scandir call (neither of which are essential for the computational portion of the SimSite3D code).

Appendix III: Changes Between Versions

Version 3.0:

- Additional changes exist in the code to pursue a variety of ideas alternative methods. We are evaluating whether the computational cost incurred is reasonable with respect to the increase in accuracy, as well as tuning these methods. They include: MSMS surface complementarity assessment of binding sites; pose clustering to group transformations (dockings) for a given query/database sitemap pair that are so similar as to be reasonably represented by a single orientation (requires a bit of debugging and much tuning); energy scoring of matched hydrogen bonds; and projection of Hbond points to spherical caps, similar to the work of M. Rarey, B. Kramer, F. Lengauer, and G. Klebe (1996), “A Fast Flexible Docking Method Using an Incremental Construction Algorithm”, *JMB*, 261:470-489.
- Surface complementarity prototype compares MSMS surfaces by 4 values. The first step is to align the database surface to the query surface using a transformation saved in the triangle matching step. Next, all database vertices that are farther than 1.0 Å from the query surface (triangle patch) are removed, and all faces (triangles) missing 1 or more vertices are removed from the database triangular mesh. The sum of the areas of the remaining database triangles is used to estimate the amount of complementary surface area between the two surfaces. The RMSE of the remaining database vertices to the query surface is used to estimate “goodness” of fit. The number of remaining database points gives a bit more coarse value than the estimate surface area of how “large” the matched surface patches are. The number of remaining faces in the database mesh as opposed to the number of points in the database mesh gives an indication of how fragmented the matched surface patches are.
- The matching of points to spherical caps is similar to that in the 1996 Flexx paper, but we are trying to line up query protein donors and acceptors to database protein donors and acceptors (in orientations similar to the query protein). The authors in Flexx used the spherical caps to determine if protein atoms could make hbonds with ligand atoms.
- Updated the default scoring method to use the weights attained in the latest round of training and testing, corresponding to the addition of hydrophobic spheres, which changed the relative sampling and weighting of hydrophobic versus polar matches.
- Added the ability to generate random perturbations (alignments) from the original orientation of the target sitemap (uses a poisson-like distribution rather than using a strictly uniform distribution). Using a poisson-like distribution helps to generate samples that are some distance from each other to help sample the entire space without using as large a number of samples as required if sampling from a uniform distribution.
- Added the ability to score sitemaps as given (i.e. as aligned by an outside method).

- Additional scoring methods -- energy-based scoring of matched H-bonding groups -- needs better placement of probe points before being pursued further
- Hydrophobic shells around hydrophobic atoms added as a way to more thoroughly sample hydrophobic surface
- Fixed handling of HETATMs to be a bit more standard
- Updated the python class utils.py to reflect the current environment variable processing
- The method of BKP Horn is used to determine the transformation (quaternion + translation vector) for the 3 pt alignments of sitemaps. This eliminates the need to support a routine to find eigen values/vectors in the case of 3 pt alignments.
- The Timer class now uses sigaction() in the place of signal(). The main reason for the change is sigaction is the preferred method of signal handling and is more portable from the Unix perspective.

Version 2.7.2:

- Cleaned up misc. files that were no longer used.
- Added support for metals in sitemaps
- Repaired issue with fragmenting mol2 files containing multiple substructures.
- Atoms which were labeled as HETATMs in the input PDB files should now be correctly labeled as HETATMs in any output PDB files.
- Having the names of runtime variables in the config files not prefixed with SIMSITE3D_ was deemed a possible place of confusion. The example config files and parsing was updated accordingly.
- Blank chainIDs and insertion codes are omitted (not represented by a character) from the label in the sitemap labels file. If an insertion code is not blank, it is appended to the residue number; if a chainID is blank, "__" appears instead of "_C_", where C is given as an example of a nonblank chainID.
- Added support for water molecules to be considered as part of the protein when generating a sitemap.
- Adjusted the scoring functions templates {do you mean parsing here, or how the scoring function lines are specified by the user?}
- Template points now have a unique label in the corresponding csv file.
- Added a field to the output for a hit. Field 6 now contains the ligand fragment binary string indicating which atoms in the ligand are/are not part of the fragment

Version 2.7.1:

- Fixed a copy-n-paste bug in method used to load environment variables
- Programs no longer support loading of a conf file from command line
- conf files are no longer required

- system conf file is now /etc/simsite3d/simsite3d.conf
- local conf file is now ${HOME}/.simsite3d/simsite3d.conf
- Write Ligand id as mol2 Molecule Name: mol2File::write() now puts the "struct_id" + _NNNNN as the molecule name (instead of "*****"). mol2File::write() now places the atom-type + orbit in a 6 char column.
- Allow mol2 Files to be in dos Format: We cannot assume that the input files (structures, ligands, etc) will be created only on *nix platforms. As such, the mol2File class had to be modified to ignore the '\r' Windows includes to denote the end of a line.
- General Handling of Runtime Environment: The environment is handled in a somewhat general fashion so that it is relatively easy to add or change environment and configuration variables. Of course, this has nothing to do with the actual implementation of the ideas the variables represent. Added conf file, environment variables and conf file on cmdline
- Updated Cmdline Flags: Renamed some of the SimSite3D Software cmdline parameters (flags) to better fit the updated cmdline options
- Ligand Fragmenting Update: Ligand fragments with few than N heavy atoms are discarded. If no ligand fragments remain, the alignment is discarded. Looking at a radius of 1.5Å around template points shows that (especially for pockets) a number of small holes exist in the "interior" of the template volume. Bumping up to 2.0Å seems to fill in the holes and should not increase the volume much. However, this method is relatively costly and is not ideal. If we could use an ellipse or set of bounding boxes to represent the template volume, it is likely we would be better off.
- Sitemap Atoms File: fixed bug that was putting additional atoms in the sitemap atoms file.
- Normalization Dataset: added environment variables (and the associated conf entries) for the diverse dataset and ligands.
- Update Filenames in Sitemap Labels Files: sitemap .csv files only have the name of the files and not the path to the ligand and protein files. In addition, we assume that all of the sitemap files are in the same directory. Given that assumption we do not need to write the _s.pdb and _a.pdb and _rad.pdb in the XXXXXX_s.csv file.
- Repaired Installation Script: repaired the install.py script so that it does not install the auto_gen_sitemaps.pl script, does not install the diverse sitemaps and does not blow away the python and examples directories if it is installing to the source dir.
- External Prot-Lig SF Changes: changed the way that the external scoring functions are called. The explicit command line is listed in external_scoring_functions.txt with **$PROTEIN** and **$LIGAND** denoting where SimSite3D will stuff in the chosen protein and ligand files.

Version 2.7.0:

- Sitemap Volume: Ligand hydrogen atoms are ignored when using a ligand to define the volume of a sitemap.
- Match Prints: Given the sitemap points of a query binding pocket, each hit has an associated binary string (in the score file). A 1 in a position indicates the hit has a similar interaction point which corresponding to the given query point. The match prints can be used as another ranking method besides the SimSite3D score for the hit. In particular, hierarchical clustering techniques can use the match prints to generate a dendogram of the hits based on the satisfied sitemap points of the query pocket.
- Sitemap Files: The sitemaps now include the protein atoms making the interactions associated with the sitemap points and labels for the sitemap points. This additional information is placed into 2 additional files (for a total of 3 files per sitemap).
    - The "original" sitemap file is relatively unchanged from previous versions of SimSite3D. This file is denoted as the "sitemap points" file and is in PDB format.
    - The "sitemap labels" file contains additional information about the sitemap and is used to tie the points file to the atoms file. The main use for this file outside of SimSite3D is to provide the labels given to the sitemap points. More information in stored in this file and depending on development needs, this file is the one which is mostly likely to change in the future.
    - The "sitemap atoms" file contains those protein atoms that gave rise to the sitemap points and is a subset of the protein PDB file.
- Ligand Files: The ligand fragment file associated with a hit is now generated by keeping only those ligand atoms which are within interaction distance of a query pocket protein atom giving rise a query sitemap point and do not have significant overlap with any protein atom.
- Working/Output Directory: At the present, the Linux distributions used to test SimSite3D require execute permissions on a particular in order to "cd" into that directory. In addition, SimSite3D makes use of several temporary files that need read/write access. An environment variable and a command line parameter were added to direct SimSite3D as to which directory to build the results directory and to create/delete temporary files.
- The examples directory is now recursively copied (by install.py) to **$SIMSITE3D_INSTALL_DIR** so that the installation files of SimSite3D need not be accessible by users.

Version 2.6.1:

- External Scoring Functions: The options column in external_scoring_fuctions.txt was split into two columns. Column 3 is for those options which must or may precede the protein and ligand on the command line for the external scoring function. Column 6 is for the options which must or may follow the protein and ligand on the command line (e.g. SFCscore requires that the user given .sfc file be specified as the last option to

SFCscore. In the case of SFCscore, the path to the .sfc file should be the last item in column 6 of the SFCscore line in external_scoring_functions.txt).

Version 2.6:

- Ligand Fragmentation: The method to handle the fragmentation of ligands has changed from considering flexible bonds as cleavable to a brute-force ring search. The current handling is to keep all rings for which at least one bond of the ring should be kept. The ligand fragmentation method is a preliminary one which is based on relatively loose constraints used to determine if a ligand atom is in the query pocket.
- External Scoring Functions: To be able to handle a variety of external protein-ligand scoring functions or none at all, the method has been generalized a bit. For an explanation please see the section on external scoring functions.
- Sitemap Headers: Sitemaps need not be regenerated, but to allow for comments which are compatible with those expected by tools such as MovIt, the comments have been changed from using the '%' and '#' characters to "REMARK 10".
- Normalization of Query Sitemaps: In the interest of keeping the number of steps for a sitemap search to a minimum, the query sitemaps are normalized at creation time and the statistics are saved in the generated sitemap file. Because of the current computational load to normalize a sitemap, the perl script (auto_gen_sitemaps.pl) to generate the database sitemaps does not normalize the database sitemaps.
- C++ Program Defaults: At the request of some, the C++ programs now have default values for most of the command line parameters. Adding the defaults, allows the use of C++ programs interchangeably with the Perl wrapper scripts and allows for more flexibility in running a search. Ideally, the more obscure parameters and those which modify the sitemap parameters should be left to their default values. Those parameters which are deemed the most useful to change are explained in detail in the sections describing the C++ program search_sitemaps.

Version 2.5:

- Ligand fragmentation: for each hit to a given query, SimSite3D will write out 2 .mol2 files. One file contains the ligand, corresponding to the given hit, as it exists in the database, but with a rigid transformation applied to move the ligand to the coordinate space of the query pocket (sitemap). The second file contains the subset of atoms and bonds representing the portion(s) of the ligand hit that fall inside the query pocket[2]. This fragmentation is an initial implementation and is not expected to be ideal in every case. In particular, the ligand can only be cut at flexible points. Thus, rigid fragments of ligands are included or excluded on an all or nothing basis (e.g. for a phenyl ring the entire ring is included if any one of the atoms falls inside the query pocket).

[2] The volume of a query pocket is defined as a ball. The center of the ball is given by the average position or centroid of the query sitemap. The radius of the ball is taken to be the distance from the centroid to the point of the sitemap which is farthest away from the centroid plus a buffer of 0.5 Å.

- DrugScore: after an SimSite3D search, each hit is given a DrugScore based on the database ligand fragment (for that hit) and the query protein structure. The DrugScore field gives another method of ranking the order of the hits (besides the SimSite3D orientation score).
- Search results: to help organize the results of a search, SimSite3D will create a subdirectory in the directory where the search was executed. This subdirectory will hold the results (.out file) of the search and a directory holding the ligand fragments.
- Results file: A header was added to help reproduce runs when developing and to help compare results across different version of SimSite3D. Each result (line) has an additional field for the DrugScore (or empty if DrugScore is not used) for the database ligand fragment in the query pocket. At present, the profile time field has been dropped.
- Ligand fragments directory: For each line in the results file there are 2 .mol2 files written to the ligands subdirectory of the results directory (for more detail please refer to the ligand fragmentation section above).
- Sitemaps: the sitemap files have been added a header listing the parameters which were used to generate them. This header allows the developers and testers to compare sitemaps between the different versions of SimSite3D.