BioPharmics Platform Manual
Version 5.197

Documentation Technical Team
BioPharmics Division, Optibrium Ltd.

BioPharmics™
a division of optibrium™

A BIOPHARMICS LLC PUBLICATION

Published by BioPharmics LLC. BioPharmics LLC is a division of Optibrium, Ltd., Cambridge, UK.

CONTENTS

FOREWORD AND RELEASE NOTES

This manual covers the usage of the BioPharmics Platform Modules: Tools (sf-tools.exe), Docking (sf-dock.exe), Similarity (sf-sim.exe), Affinity (sf-quansa.exe), and xGen (sf-xgen.exe) through their command-line interfaces. The patented ForceGen approach for conformer generation, implemented within the Tools module, is the fastest and most accurate approach currently available on normal small molecules and also on macrocyclic compounds. The new eSim technique also represents a breakthrough in 3D molecular similarity computation. The eSim methodology underpins many aspects of docking and affinity prediction.

Release Notes

Version 5.197

Minor changes only: Tools Module has improved perception/preservation of configurationally variable ring substituents and a new visualization for distance violations in NMR-restrained conformational search/profiling; Docking Module has improved parsing/matching of HET code names in PDB file processing and a new prototype method for covalent docking.

Detailed notes can be found here, and the respective chapters will cover significant changes with specific examples.

Version 5.193

Minor changes only: 1) More robust parsing for oddly produced MOL2 files; 2) improved behavior with poorly formed SMILES that fail to indicate protons for bracketed atoms and a new parameter to control final energy window (`-en_final`) without affecting search strategy; 3) QuanSA minor bug fix for pose caching during model building; and 4) the sf-dock `rms_list` command now requires a `<max_n>` parameter to control how many poses will be considered.

Version 5.191

Minor changes only: 1) Bug fixes in fgen3d regarding illegal SMILES strings (e.g. atoms making 11 bonds) to eliminate crashes when parsing poorly curated molecular databases; 2) fixes to the PSIM command parsing to reduce brittleness in protein/ligand list specification; 3) additional example of cross-scaffold QuanSA affinity prediction

added to the manual; 4) updates to the xGen PyMol GUI to overcome changes in PyMol v2.6; and 5) inclusion of Mac binaries for statistical utility commands.

Version 5.189
Minor changes only: 1) a technical change to recognize certain types of ring chirality that is not detectable by topology (affects `regen3d`); 2) addition of a final conformer compression option for ForceGen (-nfinal) to allow for conformer pool size control following full conformer search; 3) improvements to RMSD calculations for evaluation of docking quality; and 4) a method to automatically define bounding boxes given collections of aligned ligands.

Version 5.186
The primary changes relate to the inclusion of a new chapter in this manual on Advanced Applications, which include PROTAC docking, a GUI for real-space ligand refinement using xGen, and bound ligand strain estimation for complex molecules. In addition, a number of different methods for binding-site definition in the Docking module have been implemented.

Version 5.173
Minor version release with full examples and manual update. One substantive change: multiple ligand alignment has been improved with respect to cross-platform behavior using a normalization within pose-clique scoring. To *turn off* this new default behavior, in the Similarity module use `-me_norm` and in the QuanSA module use `-clnorm`. A bug in score parsing within the docking pose family method has been fixed, resulting in improved pose rankings, especially when no prior ligand knowledge is being used.

Version 5.164
Minor version release: Sim Module fixed memory leak when using -poscon; Tools Module implemented new behavior for torsional and positional restraints along with the ability to turn off inclusion of torsional restraints in SFDB conformer files.

Version 5.162
Tools Module: 1) improvement in enforcement of torsional restraints during fgen3d (and regen3d) procedures; 2) fixed issue with parsing variant SDF files with non-standard tags; 3) added tautomer recognition for imide/amide proton shift; and 4) increased max SDF tag line width to 2000 characters. Docking Module: 1) changed default PDB ligand parsing size to 100 heavy atoms; and 2) fixed bug in reporting of polar component of docking scores. QuanSA Module: fixed issue with reading named molecules where molecules are repeated within an SFDB. ESim and xGen Modules: no significant user-facing changes. Additional minor bug fixes within all Modules.

Version 5.142
Minor additions and changes within all Modules. Please see the beginning of each specific chapter for details.

Version 5.125
Minor additions and changes within the Tools, Similarity, and QuanSA Modules. Please see the beginning of each specific chapter for details.

Version 5.114
Minor version release: fixed minor bugs in Tools modules along with minor bug fixes and enhancements within the eSim module.

Version 5.111
Minor version release.

1. New behavior in the sf-tools profile command:
   (a) If a reference molecule is provided, the atom name and residue information (if present) in the reference molecule structure will be copied over to the input molecule so that the key output files (e.g. <outprefix>-

match.mol2) will contain those annotations. This can be helpful in workflows that involve peptidic ligands and NMR restraints.

(b) In cases with very large numbers of conformers in the molecule to be profiled (> 5000), calculation of nearest-neighbor RMSD and average RMSD to all other conformers is skipped. This avoids poor scaling behavior in situations where conformational profile reports are desired for extremely large conformer pools. NOTE: it is not recommended to generate pools of such size, as we have yet to identify a use-case where agnostic conformational search yielding pools of more than 2000 conformers (e.g. from -pextrm preparation) is appropriate.

2. Addition of ext_sfdb_sdf command to sf-tools: extracts the contents of an SFDB to SDF format, with per-conformer energy values reported. This can be used, for example, instead of the profile command in situations where NMR constraints are not being used, but where knowing the MMFF94sf energy value for each conformer is desired.

3. Addition of compress_rms and compress_n utilities to sf-tools:

   (a) The former uses the RMSD threshold from the -rms option to compress the input SFDB or molecule list. RMSD calculations are done with symmetry correction, so that the resulting pools are non-redundant. The command will produce fewer or the same number of conformations as provided in the input.

   (b) The latter uses the RMSD threshold from the -strict_rms option and the number of conformers provided by the -nconfs option. All conformers that are redundant according to -strict_rms are discarded. Then, the redundancy threshold is progressively increased until the resulting conformer pool has fewer or equal members than -nconfs.

4. Addition of the comp_ensemble command to sf-tools: provides statistics of the closeness by RMSD of two conformer ensembles. Automatically identifies macrocyclic ring systems.

5. Improvements to thread-safety for PSIM and GrindPDB operations. The new command grindpdblist is multi-threaded and much faster than running grindpdb serially. The psim_align_all function is now multi-threaded, with near linear speedup with increased computing cores.

6. Ligand kekulization and protonation is more robust and consistent, providing cleaner operation on 3D structures lacking protons.

7. Added -min_output option to sf-dock (analogous to sf-sim)

8. Added -min_2way to sf-sim (turns on +two_way and sets min corresponding output score)

The full set of examples have been run with the v5.111 Linux binary on the reference platform.

Version 5.101
Minor version release: the 5.101 release introduces the xGen module, which implements a new approach to real-space fitting of ligands into X-ray density maps. The full set of examples have been run with the v5.101 Linux binary on the reference platform, and the examples have been amended to include the xGen module.

Version 5.001
Major version release: the full integration of more sophisticated force field and partial charge modeling into the BioPharmics Platform is now complete, with all aspects of conformer search, docking, molecular similarity, and affinity prediction now benefitting from uniform technical underpinnings. The 5.001 release introduces relatively minor changes within each of the modules relative to the latest revision of the v4.5 release.

1. The Tools module includes improvements in ligand Kekulization, constrained minimization, and constraint-based conformer search. Default behavior has changed for heuristic protonation (heuristic protonation is turned on by default rather than assuming that formal charges in .smi or .sdf files are intended).

2. The QuanSA module incorporates improvements in the manner in which crystallographic data can be used to guide model building as well as improvements in the model refinement process when adding new training molecules.

3. The Sim module contains improvements in multiple-ligand alignment.

4. The Docking module now implements a blended approach to electrostatics, combining a Coulombic approach with the pre-existing directional hydrogen-bonding method, and the eSim method has been integrated into the docking process and pose family generation process.

The full set of examples have been run with the v5.001 Linux binary on the reference platform.

Pre-Version 5
The Version 4 BioPharmics/Surflex Platform releases introduced the ForceGen, eSim, and QuanSA methodologies to complement and extend prior work in docking, molecular similarity, and affinity prediction. Version 4 also marked the beginning of integration of MMFF94sf into all modules along with exploitation of multi-core computing architectures using OpenMP.

Note:
There may be minor variations between the figures shown in the manual and the precise results shown in the software distribution. There are no statistically significant differences, but, for example, the N^{th} ranked solution indicated in the manual may correspond more closely to the $(N-1)^{st}$ in the actual distribution. The variations are due to small algorithmic changes across minor version increments as well as cross-platform and compiler differences.

The SFDB file format makes use of an open-source compression method called LZ4.

```
LZ4 - Fast LZ compression algorithm
Copyright (C) 2011-present, Yann Collet.

BSD 2-Clause License (http://www.opensource.org/licenses/bsd-license.php)

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

    * Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above
copyright notice, this list of conditions and the following disclaimer
in the documentation and/or other materials provided with the
distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

You can contact the author at :
 - LZ4 homepage : http://www.lz4.org
 - LZ4 source repository : https://github.com/lz4/lz4
```

CHAPTER 1

PROGRAM INSTALLATION

The Surflex platform contains separate modules for docking (Surflex-Dock, which also incorporates algorithms for protein pocket detection and comparison), molecular similarity (Surflex-Sim), QSAR (QuanSA), and real-space X-ray density fitting (xGen). These modules are all supported by the Surflex-Tools module, which implements algorithms for ligand preparation and manipulation. Each module has a simple command-line interface. The manuals have been combined into this single document, because the interface style and semantics are closely related across the applications. This short chapter addresses program installation.

1.1 DISTRIBUTION STRUCTURE

The distribution is divided into three directories: this document is found under the "doc" directory, with binary executable code under "bin," and examples described within the manual under "examples." The executables for all platforms are in "bin". Directly under that directory, named binary files for version and platform exist. For convenience, the executables are also copied to standard names under architecture-specific directories (e.g. "bin/linux/sf-dock.exe"). Users can either copy the contents of the architecture-specific bin directory to someplace on their executable path or they can add the proper bin directory to their path. Also within "bin" are directories called "data" and "util" which contain relevant data files and utility programs/scripts, respectively. The contents of the examples directories will be described in succeeding chapters.

1.2 LICENSING FILES

The Surflex modules are typically licensed as a bundle, with binary executable access to all modules on Linux, Windows, and Mac platforms. Under this type of license, you will have received a licensing file (typically called surflex_bin.lic). The Tools, Dock, Sim, xGen, and QuanSA modules can all be run with these license files. A valid

licensing file is required for the Surflex modules to run. The licenses are set up to be simple to install and use, requiring no license server or node locking. The licensing agreement that governs use of the software addresses any limitations on acceptable use and installation.

The *preferred* method for identifying the location of the license file and for other provided data is to set an environment variable called SURFLEX_ROOT to the directory where the distribution is location (e.g. "/share/packages/SF-Full-Distribution" on Linux or Mac or something like "C:\Packages\SF-Full-Distribution" on Windows). The license file should be placed in this folder. In a multi-user environment, it may be advisable to put the distribution and license file in a central location, to be accessed by any authorized user.

Another method, suited for individual users, is to place the license file in a user's HOME directory. If the environment variable "HOME" is set, and the license file is in that directory, it will be found. This works well on Windows, Mac, and Linux (for Windows, one needs to explicitly set the HOME environment variable). Note that it is still preferable to set the SURFLEX_ROOT directory as instructed above so that other files can be found automatically.

For backwards compatibility, an older method of directly pointing to a license file is provided by setting an environment variable called SURFLEXLIC to the pathname of the file (e.g. "/share/packages/surflex/surflex_bin.lic"). If none of these methods finds a license file, Surflex will look for the license in the following directories: C: (the normal root of a Windows file system), /usr/local/bin, and in the current directory in which Surflex is invoked.

In the following chapters, for each module, the overall command-line parameters will be listed first. Following that, use of each module will be detailed (the Surflex-Dock chapter will include detailed description of the shared commands and options among the modules). Examples relevant to each program will complete each chapter. These examples will refer to those provided in the standard binary distribution.

1.3 NOTES ON RECOMMENDED PROTOCOLS

In making use of *any* Surflex module, molecular preparation is critical. In particular, molecules must be protonated as expected in the relevant physiological condition. Commands are provided to aid in molecular preparation of ligands and aspects of protein preparation as well, but details of protonation variants for metal-containing enzymes and specific tautomer choices should be carefully considered by the user and addressed manually if need be. For each module, there are recommended protocols, which should be followed to obtain results that are similar to those found in the relevant published work. For example, when running either Surflex-Dock or Surflex-Sim, the recommended protocol is to select a parameter scheme for the intended application. For virtual screening, this will usually involve the **-pscreen** or **-pfast** options, and for optimal pose prediction one will select the **-pgeom** or **-pquant** options.

The Surflex-Tools module provides ligand preparation tools for 2D to 3D conversion, protonation, and conformational elaboration including sophisticated ring search of both non-macrocycles and macrocycles. The ligand preparation tools are controlled by parameter selection regimes for the intended use of the molecules: **-pscreen** for virtual screening, **-pgeom** for pose prediction, and **-pquant** for affinity and pose prediction within the QuanSA module. Faster variants also exist and are described later.

For QuanSA in particular, special care must be taken with molecule names so that data provided on molecular activities matches with the structures provided. It is advisable to make use of the recommended directory structure and naming conventions when using the QuanSA module.

1.4 NOTES ON USE OF MULTIPLE THREADS

All Surflex modules now make use of multiple computing cores by default. This can result in very significant speedups on modern hardware. The reference architecture for production is a many-core Xeon-based Ubuntu Linux workstation. The default number of threads to be used is nominally 36 in all modules, unless the user explicitly sets the ‘‘OMP_THREAD_LIMIT’’ environment variable, in which case the user choice will be used.

The recommended manner of user control for core utilization is to set the environment variable OMP_THREAD_LIMIT. Generally, it is best to limit the total number of threads to the total number of physical computing cores on the target hardware. For example, on a 36 physical-core workstation, the following line ‘‘export OMP_THREAD_LIMIT=36’’

should be present in the `.bashrc` file. By limiting thread utilization in this manner, flexibility is afforded the algorithms in dynamically making use of teams of computing threads within different procedures (including nested ones).

One can change thread utilization from the default value by using the `-nthreads` argument within each module to specify a particular number of computing threads to be used. This can be greater or less than the default value, but the operating system may set an upper limit that cannot be over-ridden. We are experimenting with different strategies for resource utilization, and changes in thread utilization behavior will occur over time.

1.5 PREPARATION FOR DIFFERENT OPERATING SYSTEMS

The BioPharmics Platform is primarily supported as a set of command-line modules, though integration with multiple graphical modeling platforms is planned (e.g. Surflex-Dock is currently accessible via the MOE product of CCG and the pose generation module within the StarDrop product of Optibrium). For any modern Linux variant, the standard tools available within any shell will suffice. Some of the more extensive scripts within the use-case examples rely on standard commands such as *wget*, *gzip and gunzip*, *tar*, *awk*, *head*, *tail*, *sort*, and *grep*. Of these, only *wget* and *gunzip* are practical requirements, as they support the downloading and processing of large numbers of PDB files.

The Linux Intel OMP library `libiomp5.so` has been added to the distribution (under the `bin/linux` directory), and it may be required on certain Linux variants. If loading errors are encountered, this library should be added to a directory within the user's LD_LIBRARY_PATH (or the LD_LIBRARY_PATH should be defined to include wherever the `libiomp5.so` file is). Also, if the error `"invalid $N$ use detected"` is encountered, this library should be used in place of the default GNU OMP dynamic library. The libraries loaded by the Surflex platform can be identified by running `"lsof -p <pid>"` where the process ID of a running Surflex module is given.

For Windows, the recommended way to run Surflex command-line modules is within the Cygwin environment (see www.cygwin.com). The minimal base packages are sufficient, and it is recommended to install the 64-bit version of Cygwin. Users will need to ensure that the Cygwin binary directory is on their executable path. The easiest way to set this up is to search for "environment variables" within the Control Panel and edit the system path by appending the Cygwin binary directory to the end of the path list. It is also wise to set a sensible "HOME" environment variable. Licensing files can be placed in the HOME folder, and binary files can be copied to the Cygwin binary folder.

For Mac, setup is slightly simpler than for Windows, because the modern MacOS is built on top of a Unix variant. The recommendation is to follow these steps:

```
# Open a Terminal window using the Terminal app

# Install minimal command-line and development tools:
> xcode-select --install              # Accept the default suggestions.

# Install the HomeBrew package manager (single line):
> /usr/bin/ruby -e "$(curl -fsSL
    https://raw.githubusercontent.com/Homebrew/install/master/install)"

# Install openmp library support:
> brew install libomp

# Install the web utility wget:
> brew install wget

# Install the plotting utility gnuplot with AquaTerm:
> brew install Caskroom/cask/aquaterm
> brew install gnuplot --with-aquaterm --with-qt4
```

Note: it is *required* to install the OpenMP library support, as multi-core processing is now enabled and present within all BioPharmics Platform modules.

For Mac architectures that rely on ARM-based processors (e.g. the M2 CPU), some additional work is required to set up the libomp library. The simplest fix, which should work for most situations is to copy `libomp.dylib` (located in the binary Mac distribution folder `bin/mac/`) into `/usr/local/opt/`. That will allow the OS to correctly locate the OMP library required by the binary executable files.

A more complex but more "correct" fix can be accomplished as follows:

```
# Homebrew needs to be installed in two places on Apple silicon
#   1)  /opt/homebrew for ARM64 (default location on Apple silicon)
#   2) /usr/local/ for Intel code (the compiled BioPharmics Platform Mac binaries are Intel)

# Open a shell window and run the following:
> arch -x86_64 zsh
> cd /usr/local && mkdir homebrew
> curl -L https://github.com/Homebrew/brew/tarball/master | tar xz --strip 1 -C homebrew

# OR (within a shell window)
> /usr/sbin/softwareupdate --install-rosetta --agree-to-license
> arch -x86_64 /bin/bash -c "$(curl -fsSL
    https://raw.githubusercontent.com/Homebrew/install/master/install.sh)

# Then install libomp:
> arch -x86_64 /usr/local/homebrew/bin/brew install libomp

# Last, create a soft link or copy the libomp folder to /usr/local/opt/
```

On a Mac, the best place to put Surflex license files is within your home directory. The best place to put executable files is in "/usr/local/bin" which is already on the default executable path on modern Apple laptops and desktops.

1.6 MOLECULAR VISUALIZATION

All of the figures in this document were produced using PyMol, which can be obtained free of charge. Within the examples, PyMol .pml files can be used to generate visualizations of results as are shown in the Figures. The scripts access PyMol via the alias "`pym`" which causes the command "`pymol -Q`" to be executed.

Within the .pml files, a number of PyMol aliases are used. The following `.pymolrc file` defines them:

```
set valence, 1
set stick_radius=0.20
alias z, zoom visible, 3, state = -1
alias s, show sticks; hide (h. and (e. c extend 1))
alias hp, hide (h. and (e. c extend 1))
alias unlabel, label visible, ""
alias square, viewport 500, 500
alias thin, set stick_radius=0.10
alias thick, set stick_radius=0.20
alias all, set all_states, on
alias one, set all_states, off
```

Plans for more complete integration of the various Surflex modules into user-friendly GUI-based modeling suites is underway.

One PyMol GUI (for xGen ligand refinement) is now distributed with the BioPharmics Platform (see the Advanced Applications Chapter for details). It was initially developed for PyMol version 2.4. Later versions (including the v2.6 LTS PyMol release), contain a misnamed DLL on Windows. The program `mtz2ccp4_px.exe` requires an Intel FFT library and looks for `mkl_rt.dll`. Unfortunately, the PyMol distribution's updated library is named `mkl_rt.2.dll`. Attempting to load an MTZ file into PyMol results in nothing happening. This can be fixed by navigating to the following folder in the PyMol installation: `Schrodinger/PyMOL2/Library/bin/` and renaming `mkl_rt.2.dll` to `mkl_rt.dll`.

One can test the fix by loading an MTZ file into PyMol and looking for the omit maps that are provided in standard PDB-deposited structures. Another method is to execute the `mtz2ccp4_px` command in a shell/console window, as follows to see a usage message:

```
> cd ~/AppData/Local/Schrodinger/PyMOL2/Library/bin
> ./mtz2ccp4_px.exe
Usage: mtz2ccp4_px.exe <space_group> <a> <b> <c> <alpha> <beta> <gamma> <reso_high> ...
```

CHAPTER 2

TOOLS MODULE TECHNICAL MANUAL

Preparation of ligand and protein structures is critical to produce sensible and reliable predictive molecular modeling results. The most crucial aspect is protonation, but issues of internal energetics can also be quite important. With respect to protonation, all BioPharmics Platform programs expect molecules to be protonated as expected in the relevant physiological condition (and including explicit non-polar hydrogen atoms). The Tools module provides utilities that aid in preparation (described below). However, aspects such as tautomer choice and special protonation states for metal chelation moieties are the responsibility of the user.

The Tools module provides ligand-focused methods for SMILES string to 3D conversion, protonation, ring search, and conformational elaboration. Validation of 3D structure generation and conformational elaboration has been published, with very extensive comparative validation on non-macrocycles and macrocycles [1, 2].

Because different levels of exploration in conformational space are possible, ligand preparation should be done with the *application* in mind. By default, ligand preparation is done in a fast but thorough enough manner to make for accurate pose predictions (the `-pgeomf` parameter set). Optionally, users may choose to prepare ligands for rapid screening using the `-pscreen` or `-pfast` parameter sets (the latter of which does not explore new ligand ring conformations). For quantitative affinity prediction (or particularly detailed pose prediction), the `-pquant` parameter set should be used. These different protocols affect the total number of initial conformations that will be produced with the sampling procedure, as well as how deep the elaboration of flexible ring systems will be. Note that all of the other Surflex modules will perform *additional* conformational exploration, certainly including local optimization and possibly including more significant variation.

Multi-threaded computation is ubiquitous in the Tools module. By default, during conformational search (`forcegen` or `ligprep` commands), the module will allow the operating system to decide how many computation threads to allocate. Generally, for simple molecules, setting the number of threads equal to the number of physical hardware cores will produce the fastest per-molecule times (e.g. `-nthreads 36` on a 36-core workstation). For complex molecules (e.g. macrocycles), setting the number of threads to be equal to the number of usable threads is best (e.g.

`-nthreads 72` on a 36-core workstation capable of hyperthreading with 2 threads per core). This varies somewhat depending on machine architecture and operating system, so benchmarking different thread counts is recommended.

However, maximal speed per molecule is different that *maximum throughput*, which is the total number of molecules that can be processed in a fixed time by fully utilizing the compute capabilities of a given architecture. If maximal throughput is desired (e.g. when preparing a large database), then running multiple parallel single-threaded calculations (e.g. using the `-multiproc` option) is best.

An option for making use of NMR data in constraining macrocycle search has been added (the `-molconstraint` option) along with a command for profiling a conformer pool against a set of NMR constraints (the `profile` command). These features are now in *beta* release and are being refined with collaborators.

2.1 SURFLEX-TOOLS COMMAND LINE INTERFACE

Note that there may be minor variations between the figures shown in the manual and the precise results shown in the software distribution. There are no statistically significant differences, but, for example, the Nth ranked solution indicated in the manual may correspond more closely to the (N-1)st in the actual distribution. The variations are due to small algorithmic changes across minor version increments as well as cross-platform and compiler differences.

This is the command-line help listing of Surflex-Tools:

```
BioPharmics Platform Version 5.197

Usage: surflex-tools <options> <command> args

  [LIGAND PREP PARAMETER SELECTION CHOICES]
    POSE PREDICTION: Normal preparation mode
      -pgeomf      Pose accuracy parameter set, fixed max ensemble size 250 [DEFAULT]
      -pgeom       Deeper ring search + macrocycles, max 250
    VIRTUAL SCREENING: Recommended modes
      -pscreen     Screening parameter set, variable max 50/120 [RECOMMENDED]
      -pfast       Screening parameter set, max 50
      -pfastf      Extremely fast screening preparation, max 50
    AFFINITY or DETAILED POSE PREDICTION: Recommended modes
      -pquant      Deep ring+conformer search including macrocycles, max 1000
      -pquantf     Deep conformer search (macrocycles off), max 1000

  [LIGAND COMMANDS]
    smiles         SmilesString molname
    smiles_list    SmilesFile outprefix
    fgen3d         Smiles_or_SDF outprefix
                   Following options apply to smiles/smiles_list/fgen3d:
      +reprot        (+-)  Protonate using heuristics for physiological pH
      -chelate       (+-)  Deprotonate chelation motifs (e.g. hydroxamic acids)
      -molid    (SDF_Tag)  Specify molecule ID tag (e.g. "Compound ID" def: mol title)
      -failthresh  (7.00)  Energy per atom failure threshold for 3D generation
      -3dfast        (+-)  Use faster 3D generation

    forcegen       PathList_or_MolArchive outprefix (AKA ligprep)
      +-macrocyc     (+-)  Macrocycle searching (OFF, +macrocyc OR -pgeom OR -pquant --> ON)
      +-findbeta     (+-)  Look for and enforce beta sheet h-bonding (def: OFF)
      -nfinal        (*)  Final max ForceGen output confs (default -nconfs value)
      -en_final      (*)  Final energy window (default -enthresh value)
      -nconfs        (250) Depth of ForceGen search and upper bound on output confs per
          molecule
      +-dumpall      (+-)  Dump all non-redundant conformers, even high energy (default OFF)
      +ring          (+-)  Ring search
      -ringrounds    (3)    Max rounds for ring elaboration
      -enthresh      (10.0) The energy window above minimum for conformational searching
      -rms           (0.25) Redundancy RMS for conformers
      -strict_rms    (0.10) Strict RMS for conformers
```

```
    -enum_chiral   (0)  Max number of unspecified chiral centers to enumerate
    -max_enum      (8)  Max number of alternate isomers to keep based on energy/diversity
    -racemize_list (none) File with molecule name list to be forced as racemic if chiral
    -racemize      (+-)  Force racemic mixtures for all chiral molecules

    -multiproc npc pnum  NPC processor run, current processor is PNUM (e.g. 8 1-8)
                         Affects behavior of forcegen/smiles_list/fgen3d
    -nthreads      (*)   Maximum number of threads to use

  prot           mol_or_mollist output_prefix
    +fp            (+-) Full reprotonation
    +reprot        (+-) Protonate using heuristics for physiological pH
    +mark_chiral   (+-) Explicitly mark chirality as specified by 3D structure
    +misc_remin    (+-) Minimization after protonation
    -chelate       (+-) Deprotonate chelation motifs (e.g. hydroxamic acids)

  min            mol_or_mollist output_prefix (SF_charging AND constrained minimization)
    -pospen        (5.0)   Penalty for deviating from input geometry (kcal per Angstrom^2)
    -pwiggle       (0.2)   Wiggle room for from input geometry
    +mark_chiral   (+-) Explicitly mark chirality as specified by 3D structure
  energy         molecule_file output_prefix

  fgen_deep      PathList_or_MolArchive outprefix
    -nlevels       (4)    Levels of deep search
    -levelwidth    (12)   Breadth of deep search
    -maxwidth      (12)   Max tries for breadth of deep search
    -maxtime       (12.0) Max time of deep search (hours)
    -en_window     (20.0) Final energy window
    -deep_divrms   (0.10) Final RMSD diversity cutoff
    -deep_clrms    (1.00) Clustering RMSD

[MOLECULE SEARCHING CONSTRAINTS]
    -torcon      <frags>   Molecular fragments (multi-mol2) to constrain conformation
      -torpen      (0.05)  Penalty for torsional deviation (kcal per deg^2)
      -twiggle     (5.0)   Amount of free wiggle with zero penalty (degrees)
      -sfdb_torcon         Turn OFF inclusion of torcon terms in output SFDB (default ON)
    -mmdielectric  (80.0)  Dielectric constant for internal ligand energetic calculations
    -molconstraint <cfile> Constraint file on distances and torsions (e.g. from NMR)
      +-qmin               Use new approach for distance constraints on confusable
                           protons. Approximates 1/r^6 averaging (default OFF).


[MISCELLANEOUS COMMANDS]
  extract_sfdb   input.sfdb outprefix
  subset_sfdb    namelist input.sfdb outprefix
  make_sfdb      SearchedMolList outprefix
  ext_sfdb_sdf   input.sfdb outprefix
  ext_sfdb_1sdf  input.sfdb outprefix
  combine_sfdb   SFDBList outprefix [Combines SFDBs for a SINGLE MOL into one SFDB]
                                    [-en_window and -deep_divrms]
  enwin_sfdb     input.sfdb <energy_window> <rmsd> outprefix [reduces a single-mol SFDB
      by energy and RMSD]
  ransel_sfdb    input.sfdb proportion outprefix
  apply_poscon   poscon-frag input.sfdb out-prefix
  ran_archive    mol2archive ranarchive-prefix
  regen3d        mol2archive newarchive-prefix (may be used with -torcon)
  profile        conf_pool.sfdb ref_conf.mol2 out-prefix
    +pcenter     Turn ON centering of ref conf (default OFF)
  rms_conflist   inmol2archive.sfdb goldmol2archive.<mol2/sfdb> out-prefix
  comp_ensemble  inmol2archive.<mol2/sfdb> goldmol2archive.<mol2/sfdb> out-prefix
  comp_ens_min   inmol2archive.<mol2/sfdb> goldmol2archive.<mol2/sfdb> out-prefix
  reorder        mol2archive proportion outputarchive
  get            mol2archive molname outmolname
  mget           mol2archive molnamelist outmolarchive
  mgetnum        mol2archive molnumberlist outmolarchive
```

```
105   info            mol
      rms             mol1 mol2
107   splitmols       mol2archive outprefix [splits the archive into prefix-molnames.mol2]
      mergemols       mol2archive outprefix [merges all mols in archive into a single mol]
109   prettysdf       mol2archive outprefix [strips unneeded H and FLATTENS molecules]
      parse_sdf       sdfarchive  outprefix [Parse tagged data in SDF]
111     -molid (SDF_Tag) Specify molecule ID tag (e.g. "Compound ID" def: mol title)
      adjust_diel     conf_pool.sfdb start_diel end_diel outprefix
113   bm_ensemble     conf_pool.sfdb [ref_conf.mol2 OR "none"] outprefix
        -bm_nbest     (100)  Number of confs to fully minimize
115     -bm_ntweak    (5)    Number of tweaks to make
        -bm_twsize    (0.01) Size of random tweaks
117     -bm_rms       (0.01) Redundancy elimination RMSD
      compress_rms    InMolList_or_SFDB outprefix (option: -rms <val>)
119   comp_macrms     InMolList_or_SFDB outprefix (option: -rms <val>)
      compress_n      InMolList_or_SFDB outprefix (options: -nconfs <val> -strict_rms <val>)
121   bound_energy    conf_pool.mol2 [xgen prefix OR "none"] outprefix
        -bwiggle      (0.1)  Wiggle room deviation input geometry
123     -bwigpen      (1.0)  Penalty for deviating from input geometry (kcal per Angstrom^2)
      unbound_energy conf_pool.sfdb outprefix
```

All commands should be typed lower-case. Molecular output is generally in Sybyl mol2 format, and this is the preferred input file format as well, but MDL mol/sdf files will also work.

The Tools module is primarily used for ligand preparation: 2D to 3D conversion and conformer elaboration. Conformer generation is both fast and accurate (for details, see and references [1, 2]) for different tasks (screening, pose prediction, and affinity prediction) and for different types of small molecules including macrocycles.

Depending on the intended application, there are seven user-selectable parameter modes:

1. *-pscreen*: recommended for preparation of all but the very largest databases for virtual screening (50 or 120 max conformers per molecule depending on ligand flexibility)
2. *-pfast*: an alternative to `-pscreen` where a reduction of the total number of conformers per molecule is important (50)
3. *-pfastf*: very fast conformational search, recommended for preparing extremely large compound databases (50 conformers max per molecule)
4. *-pgeomf (DEFAULT)*: appropriate for fast geometric sampling of ligands (250)
5. *-pgeom*: for more thorough geometric studies, including macrocycles (250)
6. *-pquantf*: for preparation of molecules in affinity prediction workflows (1000)
7. *-pquant*: for more accurate preparation of molecules (including macrocycles) in affinity prediction workflows (1000)

2.2 PRIMARY CHANGES IN CURRENT VERSION

General notes about the current version can be found in the Release Notes in the Foreword to this manual. Detailed notes can be found here.

2.3 2D (SMILES OR SDF) TO 3D MOLECULAR STRUCTURES

Prior to the Version 4.0 release, the BioPharmics Platform relied upon users having other means to produce 3D structures of ligands. With many different protocols, frequent problems were observed, principally with fidelity to specified chirality and with high-energy sub-structures such as cis-amide configurations outside of rings. In order

to address these issues, and to provide an all-in-one solution for many small-molecule modeling needs, 2D to 3D structure conversion has been introduced. The SMILES format has become ubiquitous, both from the perspective of widely used programs (e.g. ChemDraw) and databases (e.g. ChEMBL). Because the format is also unambiguously specified with respect to chirality and formal charge, it is the *preferred* format supported for producing 3D coordinates *de novo*.

However, 2D SDF formatted files are also supported. Care must be taken when using hash/wedge notations for specifying chirality, especially in complex bridged ring systems. The meaning of the hashed bond is that the atom at the wide end is *away* from the viewer (i.e. into the paper or display), and the meaning of the filled wedge is that the atom is *toward* the viewer. Rotation of the orientation of a 2D molecule will not affect the chirality specification. However, flipping a molecule vertically or horizontally or moving the location of an atom attached to a chiral center such that it changes the geometric ordering *will* change the chirality specification. Note also that SDF files may or may not record an indication that the drawn geometry of double bonds is an explicit specification of their geometry.

Currently, only tetrahedral chirality and cis/trans configurations of individual double bonds are currently supported. Less common types of chirality are not currently supported (e.g. tetrahedral allene-like systems, square planar centers, or trigonal bipyramidal centers). The 3D structure generation feature includes extensive control over the interpretation of chirality within molecules, with support for enumerating unspecified chiral centers as well as indicating whether specific molecules are pure racemic mixtures when containing more than a single chiral atom.

Ligand preparation, especially regarding protonation and tautomer choice can be critical. Surflex-Tools *does not* currently offer an automatic means to enumerate tautomers. Users must provide all expected tautomeric variations of a molecule as separate molecule files. In cases where protons are absent (e.g. in 3D SDF files from the PDB), Surflex-Tools provides an automatic means to add protons to ligands, and effort is made to heuristically choose protonation states for common acids and bases such that they will be protonated sensibly assuming a physiological pH. For example, primary, secondary, and tertiary amines will receive formal charges of 1 and will have an extra proton added, as will groups such as amidines. Carboxylic acids will be deprotonated, with a formal charge of -1. However, there are cases of importance where specific moieties may have a context dependent behavior. For example, terminal sulfonamides on ligands acting as inhibitors of zinc-containing enzymes (as seen with carbonic anhydrase) will generally have a charge of -1, with the remaining H-N-SO2-R torsion being quite free to rotate. The user is responsible for addressing unusual protonation states.

Note that because MMFF is the core forcefield, only small molecules with atoms including [C N O S P H F Cl I and Br] can be processed (and proteins may also include chelated ions).

The `smiles` command is used as follows:

```
# Directory: examples/tools/ligand_2d_3d_conversion

> sf-tools.exe -reprot smiles "CC(=O)OC1=CC=CC=C1C(O)=O" aspirin-neutral
> sf-tools.exe -reprot smiles "CC(=O)OC1=CC=CC=C1C([O-])=O" aspirin-charged
> sf-tools.exe smiles "CC(=O)OC1=CC=CC=C1C(O)=O" aspirin

# KEY OUTPUT FILES:
#   aspirin-neutral.mol2   Neutral acetylsalicylic acid
#   aspirin-charged.mol2   Deprotonated acetylsalicylic acid
#   aspirin.mol2           Deprotonated acetylsalicylic acid
```

The default behavior of the command is to respect the specified formal charges within the given SMILES string and to fill the valence of each heavy atom in the conventional manner. The first two invocations of the command produce two different 3D structures based on the difference in the specified formal charge on the acidic oxygen atom. The third invocation takes a nominally neutrally charged specification and produces the deprotonated form because the `+reprot` flag is the default behavior. At physiological pH, it is expected that the molecule will be seen in its negatively charged state. The output molecules from the three smiles commands are shown in Figure 2.1.

The proprietary algorithm that implements 3D structure generation does not rely on distance geometry or templates, rather making use of serial application of an iteratively more stringent molecular forcefield that has special terms to enforce chirality constraints and other heuristic conformational preferences. We have done extensive quality control to manually verify correct chirality from input SMILES in many cases, especially complex ones with multiple fused

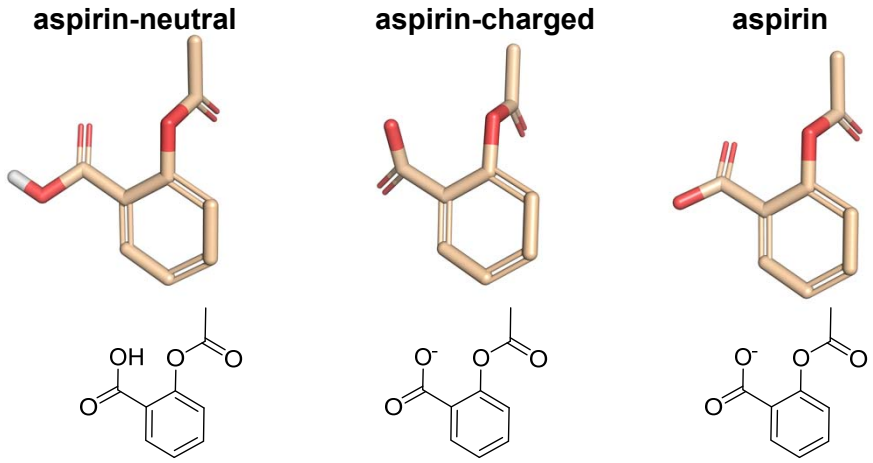


Figure 2.1 Aspirin structures resulting from the three example `smiles` commands.

and/or bridged ring systems. In cases where we have found a discrepancy between the 3D structure generated and a definitive 2D depiction, the problem has been traced to the SMILES string itself. However, cases in which incorrect structure generation appears to occur should be reported. Note that cases in which some chiral centers are specified and others are not will produce correct configurations for the specified centers but *arbitrary* configurations for the unspecified ones.

The drugs morphine, atropine, tetracycline, testosterone, and taxol all represent challenging structures, both for correct 2D to 3D conversion as well as for generation of quality ring conformations. The `smiles_list` command allows for processing multiple SMILES strings, specified one per line in a file with a name as the second field on each line. If the molecule name field is omitted, molecules will be named based on their sequence in the files. The command runs as follows:

```
# Directory: examples/tools/ligand_2d_3d_conversion
# FILE CONTENTS: Example-Smiles (each line: <SMILES> <molname>)
CN1C2CCC1CC(C2)OC(=O)[C@H](CO)C3=CC=CC=C3 atropine
[H][C@@]12OC3=C(O)C=CC4=C3[C@@]11CCN(C)[C@]([H])(C4)[C@]1([H])C=C[C@@H]2O morphine
...

> sf-tools.exe smiles_list Example-Smiles example

# NOTE: The fgen3d command will convert .smi and .sdf files
> sf-tools.exe fgen3d Drugs-Smiles.smi drugs

# KEY OUTPUT FILES:
#   example.mol2        Set of single 3D conformers for each SMILES string
#   drugs.mol2          Set of single 3D conformers...

# Look at the 3D examples
> pym disp.pml
```

Inspection of the 3D conformers (Figure 2.2) shows that the correct structures were generated. Note that while the particular configurations are not unreasonable, they are generally not optimal, especially for complex ring systems. The 2D to 3D conversion produces structures at a local energy minimum (not a global minimum). For example, the dmp structure within the examples has a relatively high energy ring conformation from the 3D generation procedure. The next major section will discuss procedures for conformer elaboration, which include comprehensive ring conformer search. The `examples/tools/ligand_2d_3d_conversion` folder contains a file called "Drugs-Smiles" which consists of nearly 1500 SMILES strings corresponding to a large fraction of the unique small-molecule ingredients

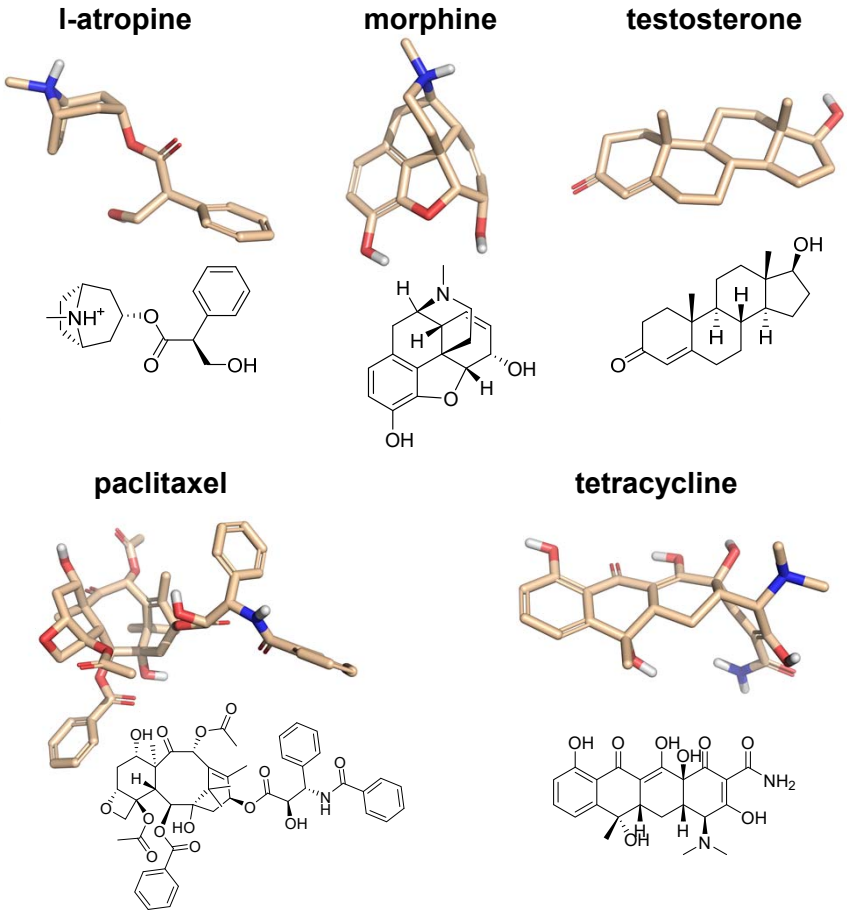


Figure 2.2 Five of the structures resulting from the example `smiles_list` command.

from the FDA substance registration system. Compared with other sources for such data, we have found these SMILES strings to be relatively accurate.

Using a single computing core, for the smaller drugs, 3D structure generation takes tenths of a second (e.g. dopamine). For medium-sized drugs, the timing is close to one second (e.g. tolterodine or sildenafil). For the largest of the drugs (e.g. vancomycin), the process can take longer, due to both the number of atoms and the complexity of meeting all of the chirality and ring constraints. The procedure is multi-core capable, and times will be much faster when using multiple computing cores. Also, the process of conformer elaboration, which nearly always follows 3D structure generation, will tend to dominate the computational cost.

2.3.1 SDF to 3D Molecular Structures

The `fgen3d` command will process either SMILES input (specified with the filename suffix ".smi") or SD/SDF input (specified by the suffixes ".sdf" or ".sd"). In the former case, behavior is exactly as with the `smiles_list` command just described. In the latter case, the SD file is processed, but the 2D coordinates are used as a starting point for structure generation. In many cases, this yields a low-energy and configurationally correct structure very quickly. However, the process reverts to a *de novo* approach in cases where the first try fails. Formal charges specified within the file will be used if the `-reprot` option is indicated; otherwise, the molecule will be protonated using heuristics to produce reasonable structures for physiological pH.

2.3.2 Chirality Specification

Both SMILES and SDF formats support explicit specification of tetrahedral chirality and the configurations of double bonded atoms. The SMILES format is unambiguous, making use of parity indications. The SDF format is more complex, especially for tetrahedral centers. The configurations may be specified either using parity or using the hash/wedge bond convention with the structure as drawn in 2D. Either or both types of indications may be present in an SD file, and in cases where they disagree, it is the hash/wedge indication that dominates. For double bonds, the *only* way to specify configuration is by how the molecule is drawn in 2D. In addition to that, it is possible to mark a double bond as having unspecified chirality when both isomers are intended. Depending on the sketching program, unexpected differences may exist between the drawn molecule and the exported SDF file.

When producing 3D structures from SMILES or SD input, Surflex-Tools produces mol2 formatted output with special properties fields that record the configurations of the specified chiral centers and double bond configurations. Only a single conforming isomer is produced by the 3D structure generation process, but additional isomers may be enumerated subsequently. The configurational specifications are interpreted by the `forcegen` command according to options selected by the user (see next Section).

2.3.3 Protonation Given a 3D Starting Point

The typical use case involves beginning from a 2D starting point, as above. However, there may be cases where a 3D conformer is provided whose protonation state is partial. Adding protons in such cases is often a tricky proposition, because file formats such as SYBYL mol2 do not generally specify formal charges, so hybridization states must often be inferred from geometries (similarly for SDF files produced by many procedures, e.g. those used by the RCSB PDB). A protonation procedure is provided for such cases.

Use of the protonation procedure is as follows:

```
# Directory: examples/tools/conformer_generation/cdk2
> sf-tools.exe -fp     -misc_remin prot pdblig-2xnb.sdf v1-2xnb
> sf-tools.exe -reprot -misc_remin prot pdblig-2xnb.sf v2-2xnb
> sf-tools.exe -reprot             prot pdblig-2xnb.sdf v3-2xnb
> sf-tools.exe                     prot pdblig-2xnb.sdf v4-2xnb  # DEFAULT

> sf-tools.exe prot cdk2-mols-noprot.mol2 prot-cdk2

# KEY OUTPUT FILES:
#   v[1-4].mol2     Protonation variations for the ligand of 2XNB.
#   prot-cdk2.mol2  All 80 CDK2 molecules, now protonated.
```

Note that ligand preparation using the `forcegen` command, as described below, must be performed *after* protonation of the ligands. Protonation occurs automatically using the `smiles`, `smiles_list`, or `fgen3d` commands but may be explicitly required when beginning from 3D ligand structures.

By default, when using the `prot` command, it is assumed that the 3D structure input is fully specified in terms of chiral configuration by the conformation that is given. So, the procedure will explicitly mark the given configurations as being specified so that a subsequent `forcegen` command will not enumerate nominally unspecified centers. This behavior is modulated by the `(+-)mark_chiral` flag (default is +). Also in the default mode of operation, full heuristic protonation will be carried out (see Figure 2.3). Selecting neutral protonation or turning off minimization can be accomplished as well by using the by `-fp` and `-misc_remin` command-line options.

2.3.4 Constrained Minimization and SF Charges

If ligands have been obtained from other workflows, it may be desirable to perform a constrained minimization in order to compared the geometry that Surflex obtains relative to the pre-existing conformations. Also, when employing ligands as the target of eSim similarity screening, it is important to calculate partial charges using the Surflex electronegativity equalization methodology. The `min` command is similar in syntax to the `prot` command. It is used as follows:

```
# Directory: examples/tools/conformer_generation/cdk2
```

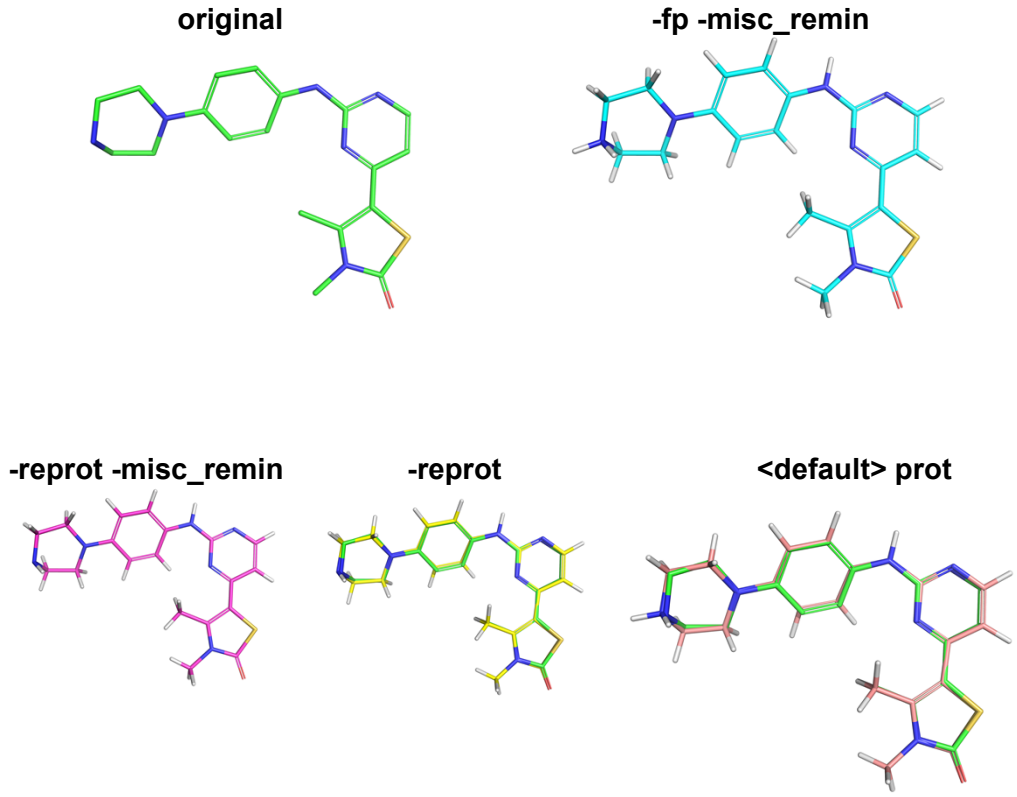


Figure 2.3 Four examples of variations of protonation of a CDK2 inhibitor. By default (lower right), full heuristic protonation and constrained minimization is performed. Protonation toward neutral formal charge can be specified by `-reprot`, and constrained minimization may be turned off by `-misc_remin`. By default, all pre-existing protons are removed and then protons are added. Existing protons can be preserved by specifying the `-fp` flag. This can be useful if the particular atom ordering and numbering is relevant to analysis.

```
> sf-tools.exe -fp -misc_remin prot pdblig-2xnb.sdf v1-2xnb

> sf-tools.exe min v1-2xnb.mol2 lig2xnb
> sf-tools.exe -pospen 100.0 min v1-2xnb.mol2 hardcon

# KEY OUTPUT FILES:
#   lig2xnb-sfcharge.mol2 Identical conformers as input, with partial charge calculated
#   lig2xnb-min.mol2      Charged and minimized conformers self-constrained
#   hardcon-min.mol2      Charged and minimized conformers tightly self-constrained
#   *-log                 Log files providing information about minimization process
```

If the user desires no constraint on the minimization, the `-pospen` argument should be specified as 0.0.

2.4 LIGAND PREPARATION: CONFORMER ENSEMBLES

The BioPharmics Platform shares a common aspect for all ligand calculations: ligand conformational exploration using the ForceGen method implemented within Surflex-Tools. There are four primary reasons for establishing this step: 1) to ensure that molecules to be processed by the Dock/Sim/QuanSA modules can be correctly handled by the MMFF94sf forcefield; 2) to establish baseline energetic expectations for each molecule against which strain estimates can be made; 3) to centralize ring/conformation search in a single fast and easy-to-use place; and 4) to facilitate use of the extremely fast similarity-based optimizations available within the Surflex-Sim and Surflex-QuanSA modules,

which require conformational elaboration prior to the respective procedures. This feature also includes extensive control over the interpretation of chirality within molecules, with support for enumerating unspecified chiral centers as well as indicating whether specific molecules are pure racemic mixtures when containing more than a single chiral atom.

The speed and accuracy of the ForceGen approach has been documented in two published papers, each with extensive comparisons to other methods using public benchmarks [1, 2]. Both on "normal" small molecules and on complex macrocycles, ForceGen is both faster and more accurate than alternative methods such as OMEGA from OpenEye Scientific Software and ConfGen/MacroModel/PrimeMCS from Schrodinger Inc.

Control of the `forcegen` procedure is done using parameter selection schemes: `-pgeom/f` for prediction of geometric molecular configurations (the overall default is `-pgeomf` if no other option is specified), `-pscreen/-pfast/f` for rapid virtual screening of molecules (the recommended mode is `-pscreen`), and `-pquant/f` for affinity prediction within QuanSA or for especially high accuracy pose prediction (for non-macrocycles the recommendation is `-pquantf`, otherwise `-pquant`). Each of these basic schemes may be modified with respect to the maximal number of output conformations, the energy window of allowable internal strain, and the number of ring conformational variants to be generated and/or used.

The `forcegen` command is used as follows (using the output from the SMILES string processing discussed above):

```
# Directory: examples/tools/conformer_generation/forcegen_simple

> sf-tools.exe -pgeom forcegen example.mol2 pgmols
> sf-tools.exe extract_sfdb pgmols.sfdb ExampleMols/pg

# Note tha tht ext_sfdb_sdf command is analogous to extract_sfdb
# but it will create individual SDF files rather than mol2 files
# The ext_sfdb_1sdf command will create a *single* SDF file with
# the entire SFDB. Each molecule has tags for molecule ID, conformer
# number, energy, etc.

# Look at the 3D examples
> pym disp.pml

# KEY OUTPUT FILES:
#   pgmols.sfdb                 (SFDB file containing all conf. ensembles)
#   ExampleMols/pgeom-list      (List of pathnames to multi-conf molecule files)
#   ExampleMols/pgeom-*.mol2    (Elaborated conformers)
```

The input archive of molecules is searched, with their conformations placed into multi-mol2 files prefixed by the specified string. So, a molecule with the name "atropine" in the molecular file name as input will result in two output files: (i) ExampleMols/pgeom-ring-atropine.mol2, the ring conformations that can serve as input to dynamic pose elaboration in Surflex-Dock and Surflex-Sim, and (ii) ExampleMols/pgeom-atropine.mol2 which contains up to 250 conformers and is input, for example, in 3D similarity calculations. The parameter schemes control the overall behavior of the forcegen procedure:

- `-pfast/f`: For rapid screening. Performs limited ring search, assuming that the provided conformer or conformers are reasonable starting points. Identifies lowest energy ring conformers of the input ligand and uses those from which to elaborate freely rotatable bonds (up to two ring variants are used for `-pfast` and just one for `-pfastf`). Produces a maximum of 50 conformations per molecule. The overall energy window is 10.0 kcal/mol (same default for all modes). On typical drug-like compound databases, the `-pfast` approach requires 1–3 seconds per molecule, with the `-pfastf` approach typically requiring less than 1 second per molecule median time and about 1.5 seconds per molecule mean time;

- `-pscreen`: Very similar to `-pfast` except that conformer pools are limited to either 50 or 120 depending on molecular flexibility. For more flexible molecules, slightly deeper ring search is allowed. This mode is nearly as fast as `-pfast` on drug-like small molecule databases but produces better ensembles for more complex molecules. Essentially, it is a hybrid mode between `-pfast` and `-pgeomf`.

- `-pgeom/f`: Maximum ensemble size is 250. Deeper ring search is done in this mode. For `-pgeomf`, this allows for 4 ring variants and for `-pgeom` 10 such variants. In the `-pgeomf` mode, macrocycles are not searched by default but can be minimally searched by specifying `+macrocyc`. In the `-pgeom` mode, deeper ring search is performed, both on macrocycles and non-macrocycles. The `-pgeomf +macrocyc` mode produces better macrocyclic conformational ensembles than many other methods in a few minutes per molecule. The `-pgeom` mode produces better macrocycle results than all but the most exhaustive alternative methods, typically in roughly 10 minutes per molecule. The `-pgeomf` mode typically takes 2–8 seconds per molecule (non-macrocycles), and it is well-suited to pose prediction with docking or ligand-based methods.

- `-pquant/f`: Yet deeper ring search compared with `-pgeom/f` and up to 1000 conformers are produced for each ligand. For detailed work with macrocycles, the `-pquant` mode is recommended. It produces substantially better conformational ensembles than the `-pgeom` mode at a computational cost of about a factor of 3. The `-pquant` mode can be time-consuming, and for non-macrocycles, the `-pquantf` mode is recommended (e.g. for QuanSA).

- Recommended options:

  `-nfinal`: Can be used to *decrease* the total number of conformers generated. The maximum number of conformers is controlled by the overall parameter scheme (e.g. `-pquant` = 1000). This can be used to compress the results of deeper search protocols (e.g. `-nfinal 100 -pquant`).

  `-nconfs`: To *increase* the ensemble size of faster/shallower search protocols, the `-nconfs` parameter can be used (e.g. `-nconfs 100 -pfast`). However, increasing the ensemble size for a fast search protocol will produce *poorer* results than decreasing the size for a more thorough down to the same maximal size. If one wants high-quality (but smaller) conformer pools, `-nfinal 100 -pquant [or -pgeom]` is recommended over `-nconfs 100 -pfastf`.

  `-enthresh`: Modifies the size of the energy window, which affects both the search strategy and the final default energy window. The default value is 10.0 kcal/mol, but macrocyclic ligands will include double the `-enthresh` value (so, a default of 20.0 kcal/mol).

  `-en_final`: Modifies the final energy window above the discovered nominal global minumum, which is controlled by the `-enthresh` parameter. For example, searching a macrocycle with the `-pquant` parameter scheme can be modified to decrease the final energy window by specifying `-en_final 7.0 -pquant`.

Figure 2.4 shows the effect of ligand preparation using the `-pgeom` protocol on four challenging ring systems. The ring-search method does not use templates nor distance geometry, rather making use of combinations of "ring-bends" applied to each ring system, in a direct analogy to the forces acting on molecules at physiological temperatures in condensed phase. Atropine contains a seven-membered ring system bridged by a tertiary amine with an exocyclic substituent. The Surflex-Tools approach automatically identified the protonated nitrogen as being anomeric and produced alternate orientations for the methyl and proton substituents, which can be very important, especially in considering the relative alignments of such ligands. More subtle is that the carbon with the exocyclic substituent is *not* topologically chiral (the ring system itself is symmetric), but the specification indicates that the methoxy is "down" relative to the bridging functionality, so only one form is produced.

A lot of creativity is displayed in the construction of different ring systems, both by chemists and in nature. A good example is the seven-membered cyclic urea that forms the core of a series of relatively rigid HIV-protease inhibitors. The lowest-energy conformation (pink carbons) is relatively flat, but the system can bend forward and backward without excessive strain. The two cases of testosterone and tetracycline, both natural products with four fused rings, show an interesting contrast. The particular staggered fusion geometry for the six-membered rings in testosterone coupled with a five-membered ring leads to a fairly rigid structure with only limited ability to flex up or down from the lowest-energy state. In contrast, the linear fusion of four six-membered rings in tetracycline yields very significant flexibility.

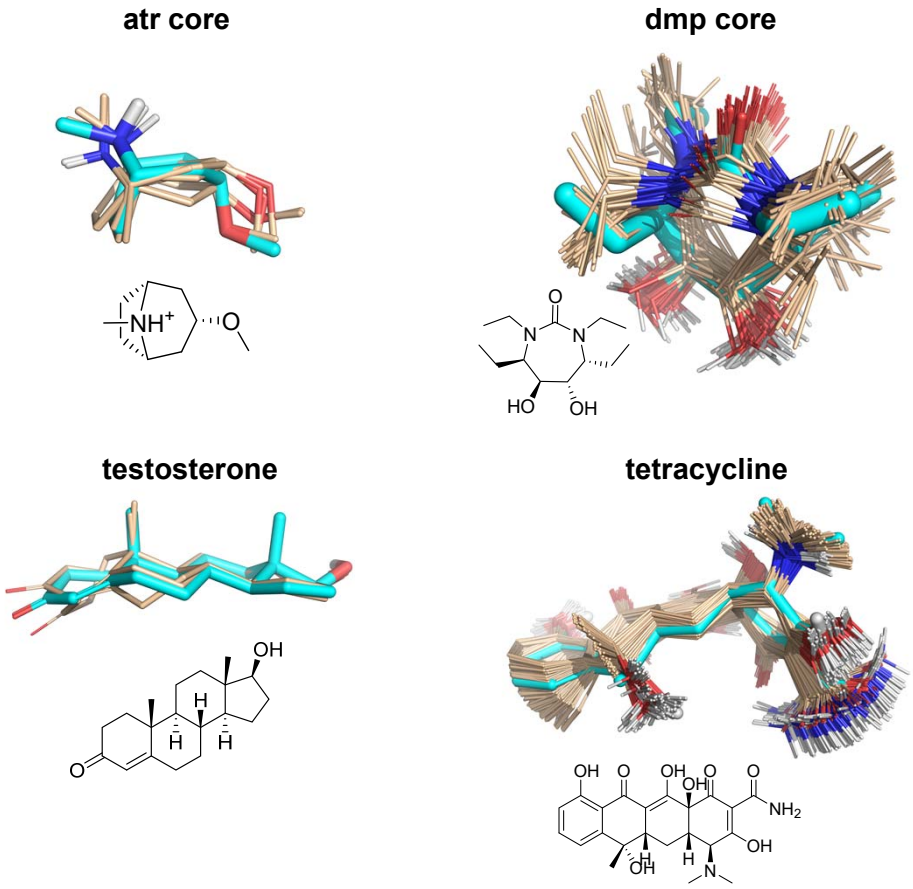


Figure 2.4 Four examples of ring elaboration in the `forcegen` output pgeom-ring-*.mol2 files: the atropine core, the seven-membered cyclic urea ring system (dmp core) of a potent HIV-protease inhibitor, the relatively rigid steroid core of four fused rings in testosterone, and the significantly flexible four-ring system of tetracycline. The lowest energy conformer for each is highlighted in cyan. For the dmp core, the lowest energy conformer is rotated 90 degrees to show that the ring conformation is correct.

2.4.1 Chirality Enumeration

Enumeration of unspecified chiral centers or proper accommodation of racemic compounds is controlled by several user settable options to the ForceGen protocol. The `-enum_chiral` option (default 0) specifies the maximum number of unspecified chiral centers to enumerate during ligand preparation. Note that in many cases, certain chirality combinations result in either impossible or extremely high energy conformations. Consequently, chiral enumeration treats the variants as conformational alternatives, essentially as part of the conformational elaboration process, subject to both redundancy and energetic pruning. Chiral enumeration includes enumeration of unspecified tetrahedral centers as well as unspecified double bonds.

Note that if a user has specified that a large number of chiral centers are to be enumerated, then the total number of possible isomeric alternatives can become very large (e.g. 1024 possibilities for 10 chiral centers). The maximum number of isomers is capped by the `-max_enum` parameter (default 8). Generally, if a large number of centers are unspecified, it is quite likely that a data entry mistake has been made (a common example is not specifying the chiral configurations within steroid scaffolds). Absent a data entry mistake, the mixture of isomers is sufficiently complex that one should not rely on an enumeration scheme to reliably identify what the active isomer(s) may be.

Racemic mixtures are treated as a special case because the configurational energy of enantiomers is identical. So, rather than producing enantiomeric variants prior to conformational elaboration, the conformational elaboration is

done first. Then, the collection of final conformers is reflected through a plane such that exactly double the number of conformers are produced than otherwise. For racemic mixtures, there are two cases. For a molecule with a single tetrahedral chiral center, where that center is unspecified, and where no unspecified double bonds exist, that molecule will be automatically detected as being racemic. In that case, if `-enum_chiral` is greater than zero, the molecule will be racemized (otherwise not).

The second case deals with a molecule with *multiple* chiral centers that the user wishes to treat as being racemic. Neither SMILES nor SDF formats offer a means to indicate that the molecule may be racemic. In such cases, the user must provide additional information.

The `+racemize` option indicates that *all* molecules with any tetrahedral chiral centers to be processed by the current invocation of `forcegen` are to be treated as racemic. In this case, each molecule is checked to see whether or not it has any chiral centers. If it does, then no matter how many it has and no matter how many are specified, the particular 3D configuration for a ligand that is input will be treated as the specified isomeric configuration and the racemic conformational ensemble will be produced. This option can be useful if a particular series of molecules are racemic and their 2D representations have correctly specified one of the two enantiomers correctly. The `-racemic_list` option is very similar. The argument to the option is a pathname to a file containing a list of molecule names, one per line. If any molecule being processed by `forcegen` is on the list, and it has at least one chiral center, then it will be treated as with the `+racemize` option.

There is one other way in which a molecule will be treated as racemic, and this is specific to the SMILES format when a 3D structure is produced by the `smiles`, `smiles_list`, or `fgen3d` commands. The following string is a di-substituted cyclohexane with both chiral centers specified: C1[C@H](C)C[C@@H](CC)C=C1. The two substituents point in opposite directions on the ring. Being fully specified, it would be treated as a single isomer, but it could be racemized in `forcegen` by `+racemize` or by specifying its name in the list that is the argument to `-racemize_list`.

The alternative is to specify both enantiomers as a single SMILES string with the "." concatenation character separating them: C1[C@H](C)C[C@@H](CC)C=C1.C1[C@@H](C)C[C@H](CC)C=C1. In this case, the molecule will be marked as being racemic automatically by the 3D structure generation commands, and it will be enumerated as such if `-enum_chiral` is greater than zero. This method will only work if the SMILES string contains two isomers, each with fully specified chirality, written in the same atom order, and which have exactly the opposite chiral specifications for all tetrahedral centers.

2.5 MACROCYCLES AND NMR CONSTRAINTS

Macrocycles are an important focus of the ForceGen method, and they offer a use-case where multi-core hardware offers significant real-world speedups for interactive modeling. Detailed performance studies have been published, establishing ForceGen (as of mid-2019) as the fastest and most accurate method for generating conformer ensembles of macrocyclic compounds [1–4]. Examples of macrocycle exploration are given in the `macrocycles` sub-directory of the conformer generation examples. The following shows how to search a macrocycle and assess performance of the method when one has a crystal structure for reference.

```
# Directory: examples/tools/conformer_generation/macrocycles

# We can test conformer search by randomizing a molecule, then
# performing conformational search using the randomized starting
# point, and finally comparing the results with the bound
# configuration, as follows:

> sf-tools.exe randomize_archive vanip.mol2 vanip
> sf-tools.exe -pgeom forcegen vanip-random.mol2 pgvan
> sf-tools.exe rms_conflist pgvan.sfdb vanip.mol2 test-van

# Or, we can employ the testprep procedure, which combines these
# tasks and also provides useful timing information.
> sf-tools.exe -pgeom testprep vanip.mol2 test-van2

# KEY OUTPUT FILES:
```

```
#   test-van-log    Table reporting conformer pool quality
#   test-van2-log   Detailed table reporting conformer pool quality,
#   test-van2-*mol2 Specific conformers aligned to the reference mol
```

The information in the table `test-van2-log` is as follows:

1. *LigandName*: name of the tested ligand
2. *NAtoms*: number of non-hydrogen atoms
3. *NRot*: number of rotatable bonds outside of ring systems
4. *CMin*: energy of constrained minimized version of the randomized input ligand
5. *Min*: energy of fully minimized version of the randomized input ligand
6. *MRSize*: size of the largest macrocyclic ring (shortest ring path)
7. *MacSys*: number of atoms within the largest macrocyclic ring system including fused non-macrocyclic rings
8. *NMacroF*: number of rotatable bonds within the macrocyclic ring
9. *NConf*: number of conformers in final ensemble
10. *BestRMS*: best non-hydrogen atom RMS deviation (symmetry corrected) among the members of the ensemble compared to the reference
11. *Time3D*: wall-clock time in seconds to re-generate the 3D version of the molecule
12. *Time*: wall-clock time in seconds to perform conformational search
13. *BMin*: energy of the best matching conformer to the reference
14. *GMin*: energy of the conformer with the minimum energy
15. *R_RMS*: RMSD of the ring atoms (as opposed to the *BestRMS* value which is for all heavy atoms)
16. *I_RMS*: Initial RMSD of the randomized starting point from the reference
17. *R_RMS*: Initial ring RMSD of the randomized starting point from the reference

The following runs a test of the ForceGen search on nine macrocyclic examples:

```
# Directory: examples/tools/conformer_generation/macrocycles
> sf-tools.exe -pgeom testprep ForceGenMacroSmall.mol2 testsmall
# KEY OUTPUT FILES:
#   testsmall-log   Table reporting conformer pool quality
#   testsmall-log   Detailed table reporting conformer pool quality,
#   testsmall-*mol2 Specific conformers aligned to the reference mol

# testsmall-log: File Contents (omitting the last five columns)
# LigandName NAtoms  NRot CMin    Min     MRSize  MacSys  NMacroF NCnf BestRMS Time3D  Time
# 4YLA-ILV   22      3    76.62   76.44   9       15      6       240  0.28    0.27    7.20
# 4YZL-ILV   22      3    77.46   76.32   9       15      6       240  0.30    0.24    6.97
# 1XA5-KAR   59      9    213.77  211.39  9       19      6       246  0.52    2.80    64.92
# 4FFG-0U8   22      8    91.76   90.03   10      14      6       229  0.19    0.19    5.62
# 3EKS-CY9   37      6    103.24  101.21  11      18      8       237  0.23    1.57    26.07
# 5J2T-VLB   59      12   238.81  236.78  9       19      6       245  0.89    5.25    72.83
# 2HFK-E4H1  21      2    46.18   44.72   12      12      11      250  0.33    0.26    11.12
# 4B7S-QLE   28      6    64.18   61.93   12      12      11      239  0.49    0.58    25.10
# 4B7D-QLE   28      6    59.33   58.69   12      12      11      239  0.62    0.51    25.28

# To look at the superimpositions of the best generated conformers with
# the reference conformers
> pymol testsmall-orig-centered.mol2 testsmall-best-centered.mol2
```

Typical timings for cases where the total number of rotatable bonds (both inside and outside of a macrocycle) are a minute or less on a 36-core workstation. Further details about macrocycle performance and comparisons with other methods can be found in the 2019 ForceGen paper [2].

NMR Constraints: For large macrocycles (e.g. cyclic peptide compounds of with nine or more residues within a ring), ForceGen is able to effectively identify low-energy, biologically relevant conformers, particularly when hydrogen-bonding networks can be detected. However, the use of even sparse NMR constraints can both speed up the

search process and improve the sampling of biologically relevant conformational space. The following illustrates the use of ForceGen on Aureobasidin with NMR information: The following runs a test of the ForceGen search on nine macrocyclic examples:

```
# Directory: examples/tools/conformer_generation/macrocycles

# The trans-Aureobasidin example, with and without NMR constraints
> sf-tools.exe +findbeta -pgeom -molconstraint nmr-constraint-bounds forcegen
    aba_reference.mol2 testaba-bound-pg
> sf-tools.exe +findbeta -pgeom -molconstraint nmr-constraint-distances forcegen
    aba_reference.mol2 testaba-dist-pg
> sf-tools.exe +findbeta -pgeom -molconstraint trans-constraint forcegen aba_reference.mol2
    testaba-nonmr-pg
> sf-tools.exe +findbeta -pgeom -molconstraint nmr-constraint-bounds forcegen
    aba_reference.mol2 testaba-bound-pg
> sf-tools.exe +findbeta -pgeom -molconstraint nmr-constraint-distances forcegen
    aba_reference.mol2 testaba-dist-pg
> sf-tools.exe +findbeta -pgeom -molconstraint trans-constraint forcegen aba_reference.mol2
    testaba-nonmr-pg

# Calculation of an NMR profile for the ensembles using the NMR constraints expressed as
    bounds
> sf-tools.exe -molconstraint nmr-constraint-bounds profile testaba-bound-pg.sfdb
    aba_reference.mol2 profnmrbound
> sf-tools.exe -molconstraint nmr-constraint-bounds profile testaba-dist-pg.sfdb
    aba_reference.mol2 profnmrdist
> sf-tools.exe -molconstraint nmr-constraint-bounds profile testaba-nonmr-pg.sfdb
    aba_reference.mol2 profnonmr

# KEY OUTPUT FILES:
#   testaba-log         Detailed table reporting conformer pool quality
#   testaba-nonmr-log
#   testaba-*mol2       Specific conformers aligned to the reference mol
#   testaba-nonmr-*mol2

# nmr-constraint-bounds: File Contents
  # Type       force lb    ub      a1   a2
   nmr_bound    1.0   0.5   4.24    39   134   # 3   PHE   H   8   LEU   H   4.24   #pk 1
   nmr_bound    1.0   0.5   3.87    39   96    # 3   PHE   H   6   ILE   H   3.87   #pk 2
   nmr_bound    1.0   0.5   4.07    39   120   # 3   PHE   H   7   NMV   HA  4.07   #pk 4
   nmr_bound    1.0   0.5   2.50    25   39    # 2   NMV   HA  3   PHE   H   2.50   #pk 7
   ...
   nmr_bound    1.0   0.5   4.24    88   96    # 5   PRO   HG2 6   ILE   H   4.24   #pk 48
   nmr_bound    1.0   0.5   4.24    89   96    # 5   PRO   HG3 6   ILE   H   4.24   #pk 49
   nmr_bound    1.0   0.5   3.32    25   120   # 2   NMV   HA  7   NMV   HA  3.32   #pk 91
   nmr_bound    1.0   0.5   4.24    27   39    # 2   NMV   HB  3   PHE   H   4.24   #pk 176
   nmr_bound    1.0   0.5   2.70    98   101   # 6   ILE   HA  6   ILE   QG1 2.70   #pk 118
   #
   # Type       force lb    ub      na1 na2  a1...   a2...
   qnmr_bound   1.0   0.5   2.86    1   3    96      105 106 107                    #pk 50
   qnmr_bound   1.0   0.5   2.78    1   3    98      105 106 107                    #pk 119
   qnmr_bound   1.0   0.5   3.96    2   1    88 89   98                             #pk 120
   qnmr_bound   1.0   0.5   3.24    2   1    91 92   96                             #pk 44
   qnmr_bound   1.0   0.5   3.88    2   1   102 103  134                            #pk 38
   #
   # Type       force lb    ub      a1    a2    a3    a4
   torsion      0.3   -150  -90     36    38    40    56
   torsion      0.3   -160  -80     93    95    97    112
   torsion      0.3   -160  -80     131   133   135   150
   torsion      0.3    160   200    82    81    79    63
```

The information in the NMR constraints file is as follows:

1. *Comment*: Any line whose first character is "#"
2. *Type*: the type of bound: nmr_bound, qnmr_bound, or torsion

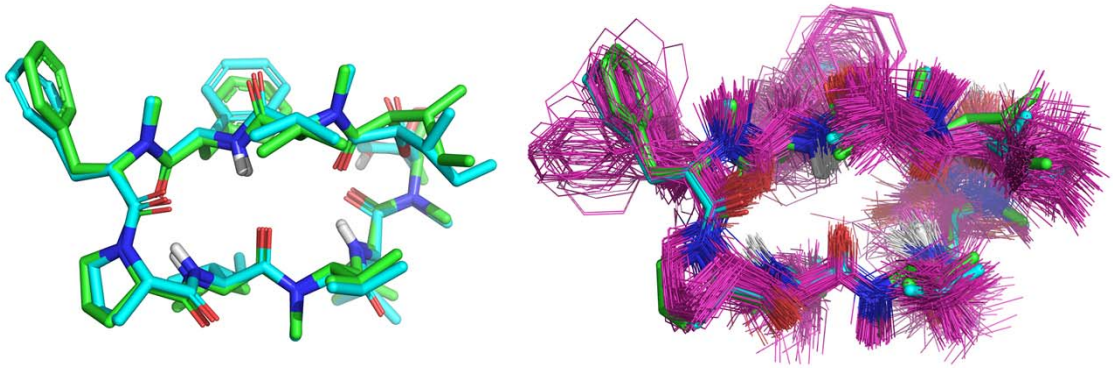


Figure 2.5 The result of NMR-constrained `-pgeom` search using bounds-style constraints on Aureobasidin. The CSD structure is shown in green, the best matching structure in cyan (0.7Å RMSD), and the full ensemble in magenta.

3. *Force*: force (kcal/mol/Å^2 for distance constraints and kcal/mol/degree^2 for torsional constraints
4. *lb*: the lower bound for the constraint (Angstroms or degrees), beyond which a quadratic penalty will be applied
5. *ub*: the upper bound for the constraint (Angstroms or degrees), beyond which a quadratic penalty will be applied
6. *a1*: atom 1 (1-based numbering) for the first atom of a distance or torsional constraint
7. *a2*: atom 2 (1-based numbering) for the second atom of a distance or torsional constraint
8. *a3*: atom 3 (1-based numbering) for the third of a torsional constraint
9. *a4*: atom 4 (1-based numbering) for the fourth of a torsional constraint
10. *na1*: number of atoms in an unresolvable group of a distance constraint (1–6 atoms)
11. *na2*: number of atoms in an unresolvable group of a distance constraint (1–6 atoms)
12. *a1...*: the list of exactly na1 atoms
13. *a2...*: the list of exactly na2 atoms

The example provided of Aureobasidin was discussed very extensively in the 2019 ForceGen paper [2]. Here, we cover the syntactic aspects of constraint application and profiling.

The NMR constraints file provides a flexible means to impose either distance or torsional constraints on molecules for which data are available. Constraints can be expressed as lower and upper bounds (as above), between which no penalty is incurred, and outside of which a quadratic penalty is providing in the energy calculated for the conformer configuration in question. Note that with very small lower bounds for distances (as in the above example), there is effectively only a single-sided upper bound because protons within a molecule cannot be so close together without incurring very large interpenetration penalties. Similarly, a large upper bound may be used along with a sensible lower bound to indicate that a pair of atoms must be at least some distance apart. Constraints can also be expressed as preferred distances with an allowable "wiggle" (see `nmr-constraint-distances`), depending on how one chooses to interpret the NMR data. In the case of distance-style constraints, the "`_bound`" string above is omitted, and the lower and upper bounds are instead specified as a preferred distance and wiggle, respectively.

The `profile` command produces multiple measurements of the quality and diversity of a conformer pool, measured against a set of NMR constraints. The information in the NMR profiling output is contained in two files. The first addresses conformers (e.g. `profnmrbound-confreport.txt`), as follows (on a per-conformer basis):

1. *Cnum*: The number of the conformer in the output sfdb (sorted from low E+EViol energy)
2. *RMS_MeanConf*: RMS deviation from the "average" conformer (see `profnmrbound-meanconf.mol2`)
3. *RMS_Pool_Min*: RMS deviation from the closest conformer within the pool
4. *RMS_Pool_mean*: Average RMS deviation from all other conformers in the pool
5. *Energy*: MMFF94sf energy
6. *E+EViol*: MMFF94sf energy *plus* constraint violation energies

7. *EViol*: Total constraint violation energy
8. *NDViol*: Number of distance constraint violations: a violation exists if the constrained atomic distance is 0.25 Angstroms or more *outside* of the given lower/upper bounds or the preferred distance ± wiggle
9. *NTViol*: Number of torsional constraint violations: a violation exists if the constrained torsion is 10 degrees or more *outside* of the given lower/upper bounds
10. *G_RMS*: Minimum RMSD from any of the given "gold" reference conformers
11. *G_Ring_RMS*: Minimum macrocyclic ring RMSD from any of the given "gold" reference conformers

The second addresses the individual NMR constraints, as follows:

1. *Cnum*: Constraint number
2. *NViol*: Number of violations within the conformer ensemble
3. *MeanViol*: The average magnitude of violations, when they occur
4. *Mean*: The average value of the constrained distance or torsion within the ensemble
5. *Constraint*: The text of the constraint specified in the file

In addition, the following molecular file output is produced:

1. *prefix-gold.mol2*: The best matching "correct" conformation for each conformer of the ensemble
2. *prefix-match.mol2*: The generated conformer aligned to its best matching reference conformation
3. *prefix-meanconf.mol2*: The "average" conformer from the ensemble

Consideration of the constraint report can help identify wrongly assigned peaks from the NMR data, for example, if a particular constraint is nearly always violated. In such cases, iteration without the offending constraint is recommended. Figure 2.5 shows the superimposition of all conformers produced using the bound-based NMR constraints above on top of the CSD structure of Aureobasidin.

2.5.1 Deeper Search: fgen_deep

For particularly complex macrocycles or situations where NMR restraints may produce a frustrated energy surface, the `fgen_deep` command offers an iterative approach to conformational search. It produces larger conformational ensembles than normal ForceGen search, with an unbounded possible number of conformers. It repeats a normal ForceGen search on new starting points, which are the result of conformational clustering (in macrocycles, the clusters are driven by ring variations). Several parameters control the depth and breadth of the search, clustering coarseness, final conformer pool energy window, final RMSD redundancy, and an upper bound on time. It is illustrated on the lariat peptide that was part of joint work with the Lokey Laboratory (UCSC), published in JACS [5], as follows.

```
# Directory: examples/tools/conformer_generation/fgen_deep

# Run the fgen_deep procedure using the final restraints from the JACS paper
# Then profile against the published conformer
sf-tools.exe -molconstraint nmr-bounds -pquant
             fgen_deep lariat-random.mol2 lariat-deep
sf-tools.exe -molconstraint nmr-bounds
             profile lariat-deep-final.sfdb JACS_Conformer.mol2 profdeep

# Run the fgen_deep procedure using the dihedral angle restraint to automatically determine
# the sign of the angle
sf-tools.exe -molconstraint nmr-bounds-dihedral -pquant
             fgen_deep lariat-random.mol2 lariat-deepdih
sf-tools.exe -molconstraint nmr-bounds-dihedral
             profile lariat-deepdih-final.sfdb JACS_Conformer.mol2 profdeepdih

# Run the fgen_deep procedure using dihedral angle restraints to automatically
# determine sign AND ALSO determine between pairs of alternate angles that
# typically result from NMR coupling constants
```

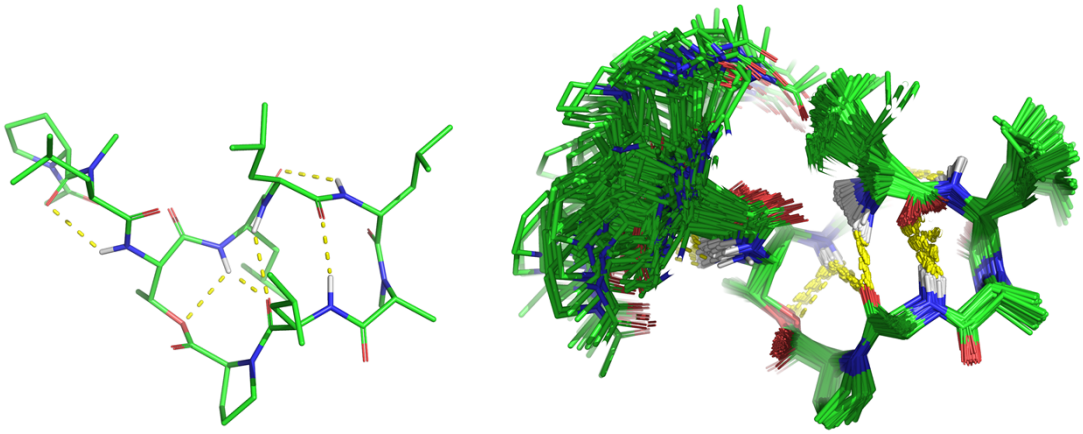


Figure 2.6 Lariat peptide [5] using `fgen_deep` with full disambiguation of dihedral angles from NMR data. The final deep conformer pool contains the same motif (see H-bond pattern above) as in the original paper, but the workflow did not require human-guided heuristic assignment of dihedral angle choice and angle signs.

```
20
   # Excerpt from the nmr-bounds-dihedral2 file:
22 # Dihedral constraints follow (FINAL revision, but with full degeneracy)
   # Type          force   Angle1 Angle2   Wiggle  a1    a2   a3    a4
24 dihedral2      0.01    94     146      10      2     45   51    52
   dihedral2      0.01    91     149      10      10     4    3     2
26 dihedral2      0.01    50     82       10      52    53   55    56
   dihedral2      0.01    101    139      10      18    12   11    10
28 dihedral2      0.01    88     152      10      32    26   25    24

30 # Here we are also specifying tighter ring clusters, a smaller final energy window,
   # and less redundancy in the final conformer pool
32 sf-tools.exe -deep_divrms 0.25 -en_window 5.0 -deep_clrms 0.5
                -molconstraint nmr-bounds-dihedral2 -pquant
34              fgen_deep lariat-random.mol2 lariat-deepdih2
   sf-tools.exe -molconstraint nmr-bounds-dihedral2
36              profile lariat-deepdih2-final.sfdb JACS_Conformer.mol2 profdeepdih2

38 # KEY OUTPUT FILES (last variation):
   #   lariat-deepdih2-final.sfdb           Final low-energy conformer pool
40 #   lariat-deepdih2-final-cluster*.mol2  Distinct clusters of conformers
   #   lariat-deepdih2-final-parent*.mol2   Parent conformer for each distinct cluster
42 #   profdeepdih2*                        Profile data (see above)
```

NMR data regarding torsional angles is derived from coupling constants, which often results in *two* possible dihedral angles in the range 0–180 degrees. The coupling constant is related to the angle *between* the atoms in question and cannot generally be automatically assigned a sign in order to fully specify and angle from -180–180 degrees.

Figure 2.6 shows the resulting lowest energy cluster (right) from the final variation of `fgen_deep` conformational search. The cluster is of the same energy as that reported in the original paper, with the same optimal macrocyclic ring geometry, but the procedure was fully automatic in terms of disambiguation of dihedral angle choices and signs of the angles. Note that the resulting cluster contains 169 individual conformers, all within 5 kcal/mol of the global minimum, redundancy filtered such that no pair of conformers is within 0.25Å RMSD from one another. It is strongly recommended that users characterize NMR-restrained conformer search results as ensembles, rather than picking a "favorite" solution (i.e. the right-hand depiction in Figure 2.6 rather than the left).

2.6 CONSTRAINING MOLECULAR CONFORMATION AND ALIGNMENT

In addition to specifying constraints on particular aspects of conformation through specific distance and torsional constraints, in some cases a user has knowledge about the likely conformation of a substructure within a molecule (e.g. from detailed energetic calculations) or about a molecule's likely bound configuration.

In such cases, it may be desirable to make use of constraints on molecular pose within the Surflex-Dock, Surflex-Sim, or Surflex-QuanSA modules. In order to do that most effectively, and to focus conformational exploration on parts of the molecule that are *unconstrained*, the `forcegen` command's behavior can be modified with the `-torcon` specification. This specifies molecular subfragments which, when present in a subject molecule, will constrain the matching portions. For the `-torcon` option, multiple fragments may be specified, and they will be applied in the order provided, where matched atoms within the subject molecule will be constrained by the first match. So, if multiple molecular fragments are provided that share substructures, the more specific one should be provided before the others.

The Tools module addresses torsional constraints, embedding the resulting constraints in the SFDB file that is produced. Positional constraints are imposed within the Docking, Similarity, and QuanSA modules. In order to provide a quick method to test the combination of torsional and positional constraints, the `apply_poscon` command is provided within the Tools module as well.

From the CDK2 example above, the ligand of 1H1S contains a substituted guanine that forms a common core among a large series of CDK2 inhibitors [6]. Figure 2.7 illustrates the use of the conformation and alignment constraints in ligand preparation, which are applied as follows:

```
# Directory: examples/tools/conformer_generation/cdk2

> sf-tools.exe -torcon cdk2-torcon.mol2 forcegen m66.mol2 pg-torcon
> sf-tools.exe apply_poscon cdk2-poscon.mol2 pg-torcon.sfdb test

# KEY OUTPUT FILES:
#   pg-torcon.sfdb                Torsion-constrained conformer ensemble
#   test-poscon.mol2              All confs with applied position constraint

# Look at the resulting search conformers
> pym disp.pml
```

In this example, two subfragments specify torsional constraints, with the second being a substructure of the first. The first specified fragment forces any matching di-substituted guanine (i.e. with both the phenyl and the cyclohexyl substituents) to follow the given conformation. The second fragment constrains molecules that lack the phenyl but match the remainder. The alignment fragment ensures that *all* guanine ligands will be aligned based on the central heterocycle. The degree of tightness of the torsional constraints is controlled by the following two parameters:

- `-torpen`: The penalty for deviating from the specified torsional pattern (kcal per squared-degree, default 0.1) outside of the wiggle allowance.
- `-twiggle`: The allowance for free wiggling of torsional angles below which no penalty is incurred (default 5 degrees).

Note that the constraints are flat-bottomed quadratics, which allow for a good deal of user flexibility. It is *seldom* advisable to use high penalties with zero wiggle-room, owing to the need to accommodate the effects of substituent variation on both the conformation of the ligand as well as its preferred alignment within the system being studied.

Note also that hydrogen atoms are matched explicitly within the fragment graph search. For example, the presence of hydrogen atoms on the cyclohexane substituent prevents matching to any *substituted* cyclohexane. Similarly, the lack of hydrogen atoms on the phenyl allows the fragment to match substituted phenyls.

2.7 MOLECULAR FORCEFIELD: MMFF94S/X/SF

Versions of the BioPharmics Platform prior to 4.0 relied upon a DREIDING-type forcefield. Now, a fully implemented MMFF94s forcefield is in place, with DREIDING only used in rare instances where parameters are not available within

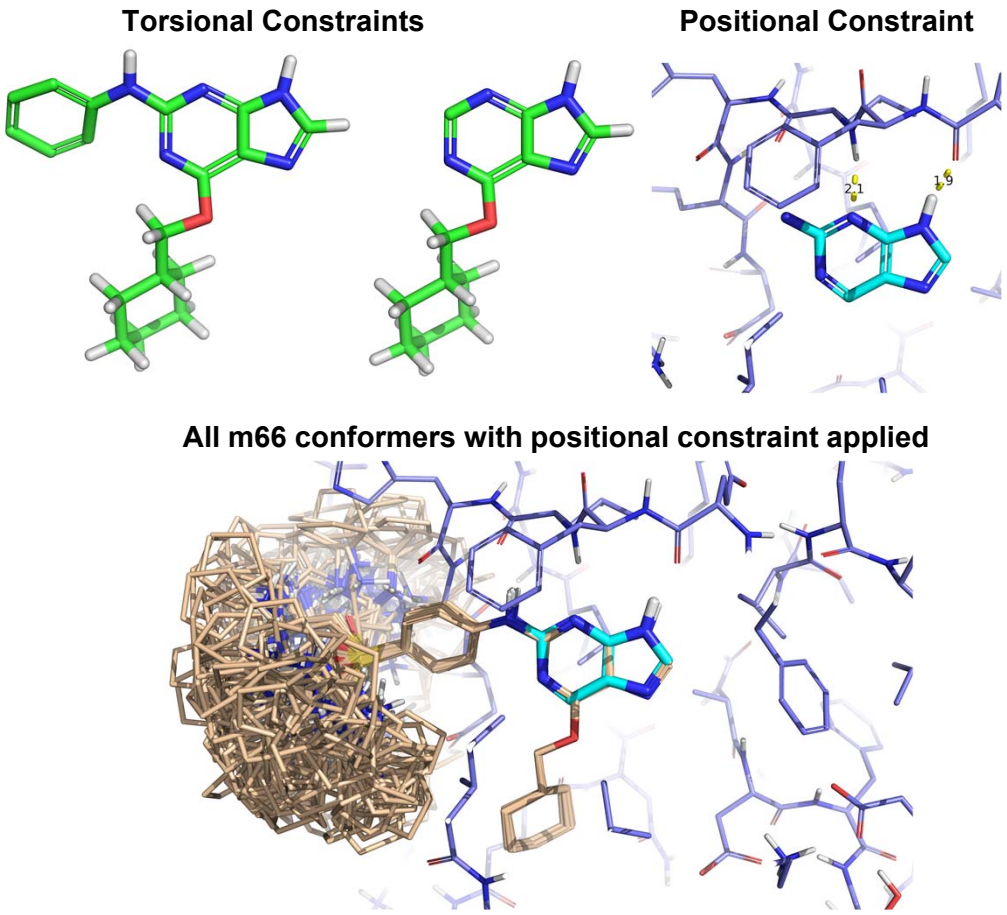


Figure 2.7 The torsional constraints ensure that ligands containing the two green substructures will match those conformations. The alignment constraint ensures that the guanine core will guide the physical alignment of the molecules. The flexible molecule (CDK2 ligand m66) is explored only with respect to the unconstrained degrees of freedom, which include the left-hand side of the molecule, extending into solvent in the CDK2 pocket. The `apply_poscon` command enforces the positional constraint shown in cyan.

MMFF94. Note that the principle difference between MMFF94 and MMFF94s is in regards to the treatment of nitrogen atoms and the degree to which they exhibit planar or near-planar geometry when they are attached, for example, to aromatic systems. The Surflex approach is similar to that introduced by CCG in the MOE platform, where the terms that force additional planarity are further stiffened (termed MMFF94x in MOE) to produce conformations that are more congruent with expectations. There are slight differences in the atom typing between the Surflex MMFF94 variant and one that is nominally fully compliant. These differences typically occur in the treatment of nitrogen atoms where there are multiple logical assignments for the atom types, generally in aromatic or conjugated systems that also include a formally charged nitrogen. We term our MMFF variant "MMFF94sf" to distinguish it from others, but it follows the MMFF94x variant closely.

By default, the dielectric is set at a constant value of 80.0, simulating molecular behavior in water. For ligand preparation, this provides excellent sampling of conformational space without introducing artificially constrained conformations that include strong Coulombic intramolecular interactions. Different treatments of the intramolecular dielectric value are being explored for use during ligand pose optimization in Surflex-Dock, Surflex-Sim, and Surflex-QuanSA. At present, we have found that using the default value produces consistent and favorable results across many

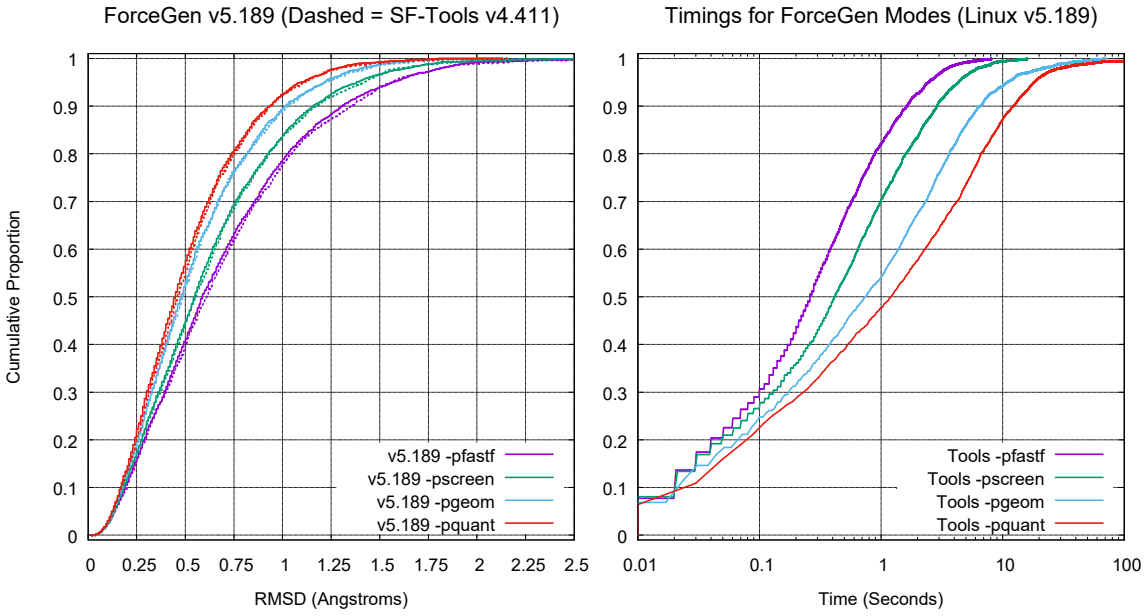


Figure 2.8 ForceGen performance: cumulative histogram of RMSD to crystallographic bound pose under different search protocols (left); wall-clock timings on a log scale (right).

different application scenarios. Note that the *inter*-molecular interactions between, for example, protein and ligand atoms are governed by explicitly derived empirical scoring functions.

Though not recommended, the dielectric value can be changed using the `-mmdielectric` option.

2.8 CONFORMER ENSEMBLE QUALITY AND LIBRARY PREPARATION

Details regarding ForceGen benchmarking with respect to the likelihood that a conformer ensemble will contain something close to the bioactive conformer can be found primarily in two papers [1, 2]. Figure 2.8 shows performance for the v5.189 version compared with the results shown in the original benchmarking paper [2]. The Platinum Diverse Benchmark is run as part of the regression testing of each new release (using the `testprep` procedure described above), as follows:

```
# Directory: examples/tools/conformer_generation/platinum-diverse-benchmark/

> sf-tools.exe -pfastf  testprep PlatinumDiverse.mol2 platf
> sf-tools.exe -pscreen testprep PlatinumDiverse.mol2 plats
> sf-tools.exe -pgeom   testprep PlatinumDiverse.mol2 platg
> sf-tools.exe -pquant  testprep PlatinumDiverse.mol2 platq

> grep -v Time platf-log | awk '{print $10}' > vals; sf-tools.exe hist vals hplatf
> grep -v Time plats-log | awk '{print $10}' > vals; sf-tools.exe hist vals hplats
> grep -v Time platg-log | awk '{print $10}' > vals; sf-tools.exe hist vals hplatg
> grep -v Time platq-log | awk '{print $10}' > vals; sf-tools.exe hist vals hplatq

# KEY OUTPUT FILES:
#   plat*-log              Log files with statistics on time and quality
#   hplat*-cdf             Cumulative histograms of performance
```

In some cases, one may want to used a particular depth of search to guarantee adequate exploration of conformational space but also control the maximum size of the output conformer pools. This can be done with the `-nfinal` parameter, as follows:

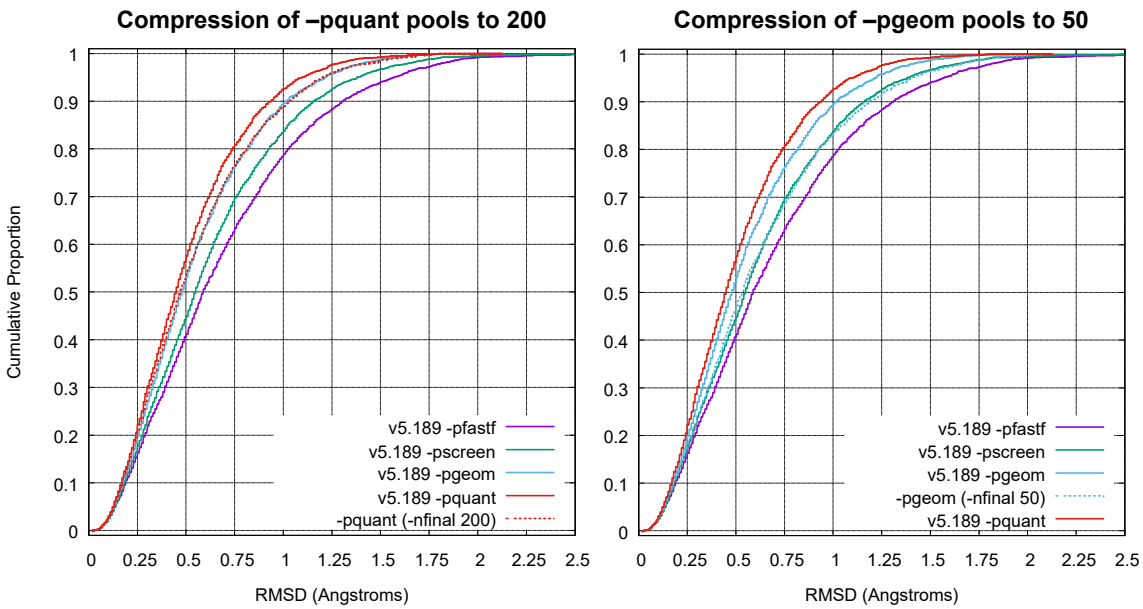


Figure 2.9 ForceGen performance: at left comparing standard `-pquant` performance (solid red) to `-nfinal 200 -pquant` performance (dotted red); at right comparing standard `-pgeom` performance (solid cyan) to `-nfinal 50 -pgeom` performance (dotted cyan).

```
# Directory: examples/tools/conformer_generation/platinum-diverse-benchmark/

# Use deep -pquant search but limit final conformer pools to 200. Also -pgeom -nfinal 50
> sf-tools.exe -nfinal 200 -pquant testprep PlatinumDiverse.mol2 platq200
> sf-tools.exe -nfinal 50  -pgeom  testprep PlatinumDiverse.mol2 platg50

> grep -v Time platq200-log | awk '{print $10}' > vals; sf-tools.exe hist vals hplatq200
> grep -v Time platg50-log | awk '{print $10}' > vals; sf-tools.exe hist vals hplatg50

# KEY OUTPUT FILES:
#   plat*-log              Log files with statistics on time and quality
#   hplat*-cdf             Cumulative histograms of performance
```

Figure 2.9 shows a comparison of the two pool-size-limited variations. At left, the standard up to 1000 conformer `-pquant` pool (solid red curve) is shown compared to the same search but limited with respect to pool size using `-nfinal 200` (dotted red curve). The limitation to a maximum 200 conformers per molecule reduces the accuracy compared with the standard `-pquant` performance, but just by a few percentage points, essentially matching the standard performance using the `-pgeom` setting, but doing so with a maximum pool size of 200 rather than 250 for the standard `-pgeom` setting.

Similarly, at right in Figure 2.9, the standard up to 250 conformer `-pgeom` pool (solid cyan curve) is shown compared to the same search but limited with respect to pool size using `-nfinal 50` (dotted cyan curve). The limitation to a maximum 50 conformers per molecule reduces the accuracy compared with the standard `-pgeom` performance, essentially matching the standard performance using the `-pscreen` setting. The standard `-pscreen` setting produces pools with maximal sizes of 50 or 120 depending on the conformational flexibility of each molecule, but the `-nfinal 50 -pgeom` protocol produces pools with a maximal size of 50. This is the same maximal size as is produced by default using the `-pfastf` setting, but the pools are of higher quality due to more thorough search prior to final conformer ensemble compression.

For preparing large databases that will be persistently used (e.g. an in-house compound collection), making use of a more thorough search setting (e.g. `-pquant` or `-pgeom`) along with a limitation on final conformer ensemble

size (using `-nfinal`) offers a way to increase conformer ensemble quality while reducing the size of the resulting conformer databases. This approach also offers a means to make effective comparisons among methods, considering all aspects of time efficiency, conformer pool size, and conformer pool accuracy.

2.9 MULTIPLE PROCESSOR LIGAND PREPARATION

When preparing large numbers of molecules, it may be wise to split the work across many processors. To achieve optimal core utilization on an N-core processing node, running N parallel single-threaded jobs will achieve maximal core utilization. It is possible to prepare a database of a few million molecules in a few days time on a 36-core linux workstation. The easiest way to implement this is by using a script that combines 2D to 3D generation and conformer search. An example script is provided, as follows:

```
# File contents: bin/util/RunFgen
#!/bin/bash
export OMP_THREAD_LIMIT=72

# Arg1 = filename.sdf or filename.smi
# Arg2 = outprefix
# Arg3 = nproc (total number of jobs)
# Arg4 = procnum (1-based indication of job number)
# Arg5 = nthreads (number of threads per job)

{ time sf-tools.exe +reprot -nthreads $5 -multiproc $3 $4 fgen3d $1 $2_$4 >& /dev/null ; }
    2> time_fg3d_$4

{ time sf-tools.exe -nthreads $5 -enum_chiral 1 -pscreen forcegen $2_$4.mol2 ps_$2_$4 >&
    /dev/null ; } 2> time_fgen_$4

# The following 36 commands should be run in parallel:

> RunFgen LargeDB.sdf largedb 36 01 1
> RunFgen LargeDB.sdf largedb 36 02 1
> RunFgen LargeDB.sdf largedb 36 03 1
> RunFgen LargeDB.sdf largedb 36 04 1

...

> RunFgen LargeDB.sdf largedb 36 33 1
> RunFgen LargeDB.sdf largedb 36 34 1
> RunFgen LargeDB.sdf largedb 36 35 1
> RunFgen LargeDB.sdf largedb 36 36 1

> cat ps_largedb_*.sfdb > ps_largedb.sfdb

# KEY OUTPUT FILES:
#   largedb_[01-36].mol2     (36 mol2 files with single 3D conformers,
#                            each containing 1/36 of the mols)
#   ps_largedb_[01-36].sfdb  (36 -pscreen level SFDB conformer files)
#   ps_largedb.sfdb          (All of the mols in a single SFDB)
```

Simple modifications to the script can be made to modify the level of conformer search, chirality enumeration, etc. The SFDB file format is binary, but concatenation of the files on most operating systems will result in a well-formed SFDB file. The SFDB format avoids overwhelming file systems with large numbers of files. Note, however, that file systems can have a limit on the size of a single file, and it might be sensible to keep multiple smaller SFDB files in cases of very large molecular databases. This can facilitate splitting large jobs across many nodes, as might be desired for a large virtual screening exercise.

Note that the last line of the log files for `fgen3d` and `forcegen` will indicate successful completion of the respective procedures. Occasionally, shared clusters will have time limits on jobs, and long jobs may be killed prior to completion.

Very infrequently, we have observed crashes due to a missed corner-case usually relating to molecular file format issues, and such cases of unexpected termination should be documented and reported as bugs.

2.10 MISCELLANEOUS SURFLEX-TOOLS COMMANDS AND ASSOCIATED OPTIONS

Supported commands not discussed in detail above are discussed briefly here.

The overall ligand preparation schemes have been described above. The default option `-pgeomf` will produce solid results across most use-cases. However, where speed is particularly important, for preparing and screening large sets of molecules, the `-pscreen` or `-pfast/f` options are preferred. For multiple alignment generation, high accuracy pose prediction, or for any application within Surflex-QuanSA, the `-pquant/f` option is preferred (selection of the faster mode may be sensible when macrocycles are not present).

The commands `smiles`, `smiles_list`, `fgen3d`, and `forcegen`, `prot` commands have been described above in detail. The `forcegen` options can be used to modify the behavior of the default parameter schemes. The `min` command offers constrained minimization of the input molecule, which can be useful to de-strain a crystallographic ligand pose. The `-pospen` parameter adjusts the constraint penalty, which should be greater than or equal to zero.

The options `-rms` and `-strict_rms` control the redundancy level in conformer pools produced by the `forcegen` command. The former does not correct for molecular symmetry but rather ensures diversity of sampling during the elaboration process. Note that toward the end of conformer elaboration, if a pool has room (e.g. has less than 250 conformers in a `-pgeom` protocol), then the pool will be "back-filled" with additional conformations that may be closer together than the specified `-rms`. The `strict_rms` value controls the degree to which conformers will be redundant *with* symmetry correction, and it is applied late in the ForceGen procedure. Note that a full set of pairwise conformer alignments is *not* carried out in order to find the actual minimum pairwise symmetry-corrected RMSD for each conformer pair in a pool. A more limited set of alignments is used in the interest of computational speed. However, one should not see truly redundant conformers with `-strict_rms` set to a value greater than zero.

The options for molecular constraint `-torcon` and `-mmdielectric` have been described above in detail. Note that imposition of torsional constraints occurs *only* within the ForceGen protocol, with the constraints being embedded within the resulting SFDB file. The penalty magnitude can be varied in subsequent calculations. Positional constraints can be tested within the Tools module (using `apply_poscon`), but they may be imposed only during subsequent alignment-driven calculations.

The `reorder` command will take an input mol2 archive and sort it according to flexibility from rigid to flexible, randomly outputting a proportion (from 0 to 1) of the molecules to OutputArchive. This is useful for sampling and running control sets of molecules or for ordering archives for fast run time on the majority of molecules.

The `adjust_dielectric` procedure is experimental. Given an SFDB for a single molecule, the pool will be minimized and re-sorted using the `end_diel` value. The procedure is intended to help with careful study of membrane permeability, especially for macrocycles. The recommended use is to employ the ForceGen procedure with the default dielectric (80.0) and then to employ the `adjust_diel` command, specifying the start and end values (e.g. 80.0 for water and 4.0 for a membrane-like lipid environment).

The `regen3d` command will randomize the pose of the input molecule and produce a single pose for each in *-random.mol2. This removes *all* memory of the previous pose, but it retains chirality and protonation state. The coordinates of the atoms are initially all set to 0.0 prior to regenerating the 3D molecular structure. The `ran_archive` command is similar.

The `rms` command computes rmsd between mol1 and mol2, correcting for internal symmetries.

The `get` command will retrieve the molecule named molname from the mol2archive and put it into the specified output molecule file name.

The `mget` command is analogous, but takes a list of names as input. The `mgetnum` command is similar, but takes a list of numbers as input (the numbers are zero-based: the first mol of the mol2 is number "0").

The `subset_sfdb` command is similar to mget and will retrieve the named molecules and create a new SFDB as a subset of the original one.

The `info` command provides information on the input molecule. It can be useful to understand how Surflex is parsing a molecular structure.

The `splitmols` and `mergemols` commands operate on SYBYL mol2 archives, either splitting the archive into individual molecule files or merging separate molecules within a single archive into a *single* molecule.

Bibliography

1. Ann E Cleves and Ajay N Jain. ForceGen 3D structure and conformer generation: From small lead-like molecules to macrocyclic drugs. *Journal of Computer-Aided Molecular Design*, 31(5):419–439, 2017.
2. Ajay N Jain, Ann E Cleves, Qi Gao, Xiao Wang, Yizhou Liu, Edward C Sherer, and Mikhail Y Reibarkh. Complex macrocycle exploration: Parallel, heuristic, and constraint-based conformer generation using forcegen. *Journal of Computer-Aided Molecular Design*, 33(6):531–558, 2019.
3. Ajay N Jain, Alexander C Brueckner, Christine Jorge, Ann E Cleves, Purnima Khandelwal, Janet Caceres Cortes, and Luciano Mueller. Complex peptide macrocycle optimization: Combining nmr restraints with conformational analysis to guide structure-based and ligand-based design. *Journal of Computer-Aided Molecular Design*, 37(11): 519–535, 2023.
4. Emily B Crull, Ajay N Jain, Paul CD Hawkins, Ann E Cleves, Edmund I Graziani, and R Thomas Williamson. Unmasking the true identity of rapamycin's minor conformer. *Journal of Natural Products*, 86(7):1862–1869, 2023.
5. Colin N Kelly, Chad E Townsend, Ajay N Jain, Matthew R Naylor, Cameron R Pye, Joshua Schwochert, and R Scott Lokey. Geometrically diverse lariat peptide scaffolds reveal an untapped chemical space of high membrane permeability. *Journal of the American Chemical Society*, 143(2):705–714, 2020.
6. R. Varela, A.E. Cleves, R. Spitzer, and Ajay N Jain. A structure-guided approach for protein pocket modeling and affinity prediction. *Journal of Computer-Aided Molecular Design*, 27(11):917–934, 2013.

CHAPTER 3

DOCKING MODULE TECHNICAL MANUAL

This chapter describes the use of the BioPharmics Platform Docking module (Surflex-Dock). The former combines a refined descendant of the Hammerhead scoring function coupled with the alignment/conformation optimization procedures implemented for morphological similarity [1–4]. The initial description of Surflex-Dock was published in 2003 [5], with many subsequent papers covering the detailed scientific and mathematical underpinnings, as well as improvements such as customization of scoring functions, full Cartesian pose optimization, use of multiple protein structures, protein pocket adaptation, and other aspects [6–15]. Examples referred to in what follows are included with the BioPharmics Platform distribution. Performance comparisons can be found in the references mentioned above.

Preparation of protein structures and ligands is critical to produce sensible and reliable results. The most crucial aspect for both is protonation, but issues of internal energetics can also be quite important. With respect to protonation, all BioPharmics Platform programs expect molecules to be protonated as expected in the relevant physiological condition, including non-polar hydrogen atoms. While there are utilities provided that aid in preparation (described below in the context of docking), aspects such as tautomer generation and special protonation states for metal chelation moieties are not currently addressed within the BioPharmics Platform platform and must be directly managed by the user.

Preparation of ligands is done using the Tools module, and the use of the ForceGen conformational elaboration method is now *required* [16, 17]. The Tools ligand preparation output file format is the SFDB (file extension: .sfdb). It is a compressed molecular format that pre-packages many required aspects of molecular parsing (e.g. atom typing and force field term generation) in order to make molecular input to downstream programs much faster. An important change for the Docking module is that specification of torsional restraints is handled by the Tools module, while positional constraints on molecular substructures remains within the Docking module.

The docking module provides protein-focused tools for obtaining and processing PDB files in bulk, with automated procedures for inferring ligand connectivity and for optimizing protein and ligand proton interaction networks. These tools are designed to be robust for use on large data sets. However, specific cases of keen interest, whether they be

ligands with unusual protonation/tautomer states or proteins that are the subject of focused study, should be handled with care and direct attention by the user.

Following preparation of proteins and ligands, there are three steps in performing docking:

1. Choosing how to identify the active site of the protein and constructing a docking target to which to match molecules (called a protomol).
2. Docking one or many molecules.
3. Analyzing the results.

Each of the basic tasks is controlled by a series of user-settable parameters, but the built-in defaults are reasonably robust to many different protein/ligand pairs. The user will generally specify either "-pgeom", or "-pscreen" options to select parameter set choices for geometric docking accuracy or for virtual screening.

3.1 SURFLEX-DOCK COMMAND LINE INTERFACE

Note that there may be minor variations between the figures shown in the manual and the precise results shown in the software distribution. There are no statistically significant differences, but, for example, the N^{th} ranked solution indicated in the manual may correspond more closely to the $(N\text{-}1)^{st}$ in the actual distribution. The variations are due to small algorithmic changes across minor version increments as well as cross-platform and compiler differences.

This is the command-line help listing of Surflex-Dock:

```
BioPharmics Platform Version 5.197
Usage: surflex-dock <options> <command> args

  PROTOMOL PREPARATION COMMANDS:
    mproto          SiteList prefix (SiteList: <protein.mol2 ligand.mol2> on each line)
    proto           sitemol protein protomol
      -proto_minvox  (500) Minimum number of voxels for protomol building
      -proto_corevox (100) Minimum number of voxels for core protomol building
      -proto_thresh  (*) Manual threshold for marking a voxel based on buriedness (default
          adaptive)
      -proto_bloat   (0)   Number of voxels to bloat protomol
      -proto_dist    (2.00)   Distance to ligand for marking

  [PARAMETER SELECTION CHOICES for DOCKING]
      -pgeom       Pose accuracy parameter set [DEFAULT]
      -pscreen     Screening parameter set
      -pfast       Screening parameter set, fastest search
      -pquant      High-accuracy pose prediction

  DOCKING and SCORING COMMANDS: (NOTE: MUST prepare ligands using sf-tools/forcegen)
    dock           ligand protomol core-proto protein
    dock_list      liglist protomol core-proto protein log
    gdock_list     liglist targpath log
      -ndock_final   (*)  Number of final docking poses (varies depending on parameter
          scheme)
      -lmatch        fmol   Turns on dynamic eSim-based alignment to given ligand poses
      -maxrot        (200)  Max number of rotatable bonds on which to operate
      -div_rms       (0.25) RMS minimum difference of final poses
      -multiproc npc pnum   Indicates NPC processor run, current processor is PNUM
      -min_output    (--)   Score below which to suppress output
    posefam        logfile
      -rmsbin        (1.50)  RMSD threshold for binning pose families
      -famdiff       (0.05)  Tanimoto thresh for pose family redundancy
      -poseprob      (0.001000)  Individual pose probability threshold
      -posehints   liglist  If provided, will use these poses to refine pose family scoring
    opt            ligand protein outprefix (single ligand local optimization of
```

```
                                          inter-molecular interactions)

MOLECULE SEARCH/ALIGNMENT CONSTRAINTS:
    -poscon      <frags>   Molecular fragments (multi-mol2) to constrain position
       -pospen     (5.00)  Penalty for deviating from specification (kcal per Angstrom^2)
       -pwiggle    (0.25)  Amount of free wiggle with zero penalty (Angstroms)
       -skipnonmatch       Turn OFF skipping non-matches to -poscon arg (DEFAULT ON)
    [-torcon must be applied using forcegen including penalty parameters]
    -mmsweight     (0.2)   Ligand strain weight above global minimum
    -mmwiggle      (0.0)   Free weighted strain in optimization protocols

PROTEIN PROCESSING COMMANDS
  getpdb          pdb_name_list outprefix
  grindpdb        protein.pdb outprefix
  grindpdblist    PDBList outprefix
    -maxprotein     (20000) Maximum number of ATOMs in PDB to process.
    -maxligand      (100) Max atoms in peptidic ligands to be pulled from protein.
    -verifypdb      (verifypdb.smi) SMILES file with PDB ligand structures and HET codes.
    -pdbcharge    Do not generate protein charges.
    -pdbalt         (A) Alt char to be included in protein and ligand.
    -pdbopt       Do not optimize protons (default OPT).
    +pdbquick     No charging, small max ligand... (default not quick)
    +mul_model      (+-) Process multiple models in PDB file.
  charge_protein  protein.mol2 outprefix


PROTEIN ALIGNMENT:
  psim_align_all  ProteinList outprefix       (e.g. PList trypsin)
                  (ProteinList: [protein.mol2 N lig1.mol2 ... ligN.mol2] on each line)
    -psim_overlap   (0.50) Overlap threshold for ligands in final alignments.
  psim_matrix     ProteinList outprefix
                  (ProteinList is [protein.mol2 ligand.mol2] on each line)
    -psim_keep    (0.60) Threshold to keep psim edges (psim_matrix and psim_align_all).
    -psim_tree    (0.65) Threshold to break psim trees (psim_matrix and psim_align_all).
    -psim_dump    (+-) Dump alignments in psim_list (if > -psim_tree param).
  psim_maketree   psim.datalog prefix
  psim_static     p1 l1 p2 l2  (Static psim computation with no alignment.)
  psim_one        p1 l1 p2 l2
  psim_list       ProteinList p2 l2 outprefix
  psim_two_list   ProteinList1 ProteinList2 outprefix
                  (ProteinList is [protein.mol2 ligand.mol2] on each line)
  psim_choose_k   cluster-prefix K outprefix  (e.g. trypsin-c0 5 trypchoose)
                  (Use K-means clustering to choose K representative sites)
  psim_findcav    protein outprefix
    -psim_cavthresh (0.60) Density thresh for cavities (lower --> larger + more).

MISC-PROCESSING COMMANDS and SPECIFIC ASSOCIATED OPTIONS:
  rms_list        multi_mol reference_mol <max_n> logfile
  rms_fam         reference_mol logfile outprefix <max_fam>
  +sdf            produce tagged sdf output from docking commands (OFF)
  dock_fp         protein.mol2 ligands.mol2 <distance> outprefix
  bbox            inmolpath <bloat> <mindim> <cube_p> outprefix
```

Each block of commands and options has a section heading indicating usage. The top block of commands, for example, relate to protomol preparation. The docking control options and the last block of commands share much in common with the Similarity module.

All commands should be typed lower-case. The preferred input file format is Sybyl mol2 from proteins, and ligands must be prepared using the Tools Module to yield SFDB files. Molecular output is generally in Sybyl mol2 format as well. MDL mol or sd file (for ligands) and PDB (for proteins) are also acceptable, although PDB files may generate errors or unexpected results, since the format is frequently variable. All input molecules must be protonated as expected at physiological pH including non-polar hydrogens. The protonation state may strongly affect docking.

3.2 PRIMARY CHANGES IN CURRENT VERSION

General notes about the current version can be found in the Release Notes in the Foreword to this manual. Detailed notes can be found here.

3.3 PROTEIN STRUCTURE PREPARATION

Preparing a clean, protonated, protein structure with its bound ligands and cofactors is well supported by a number of modeling packages. For small numbers of proteins, or for those where very particular attention must be paid to particular alternate conformations, rotamers, tautomers, protonation states, or other aspects, we recommend making use of an interactive modeling scheme. However, for large numbers of protein structures from the PDB, large-scale preparation, mutual alignment, and systematic selection of important and significant variants can be challenging. The Docking module provides three key commands to automate this process: `getpdb`, `pdbgrind`, and `pdbgrindlist` (the last being a multi-core parallel implementation). The module also provides for efficient cavity identification, surface-based alignment optimization, and variant selection (see `proto multicav`, `psim_align_all`, and `psim_choose_k` later in this chapter).

Within the examples provided with the distribution, an extensive case involving CDK2 is shown, based on our large-scale cross-docking benchmark compilation [15]. In that work, for ten protein targets, we identified the *earliest* 25% of co-crystal structures known and made use of that knowledge in order to predict the bound configurations of the ligands within the remaining future structures with an ensemble docking protocol. Here, we will see how the "early" CDK2 structures are processed (and aligned) automatically.

```
# Directory: examples/docking/protein
# Contents of CDK2-Early:
```

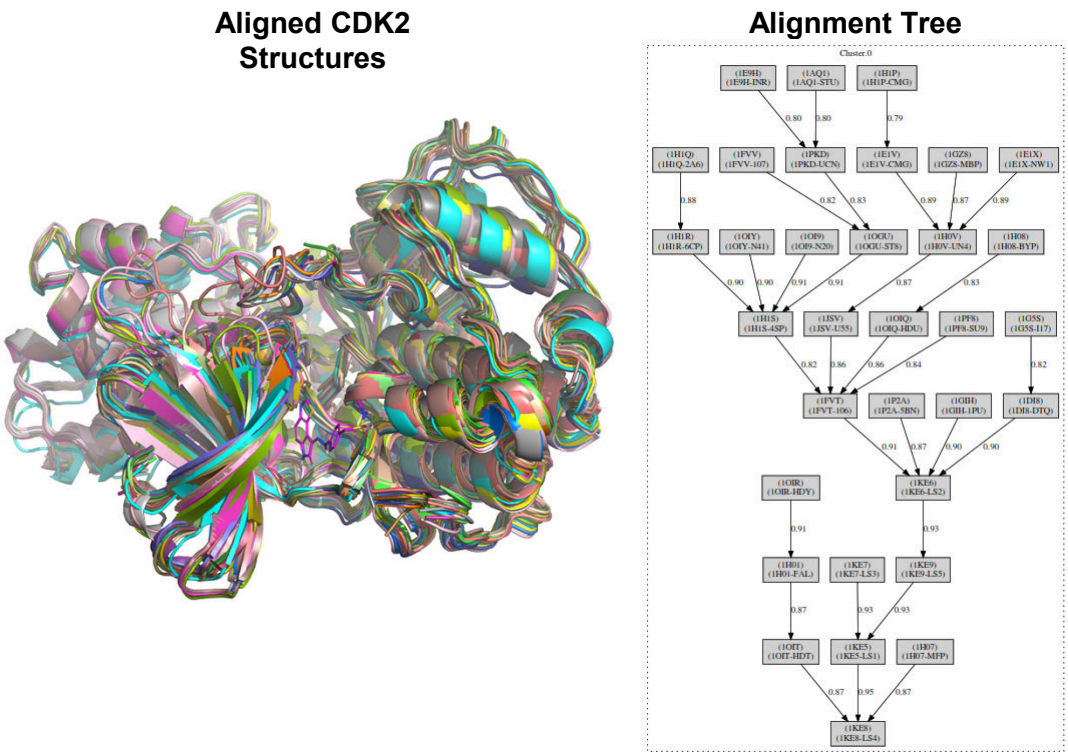


Figure 3.1 Automatically retrieved, prepared, and aligned protein/ligand complexes.

```
1AQ1
1CKP
...

> sf-dock.exe getpdb CDK2-Early cdk2

# KEY OUTPUT FILE:
#   cdk2-script
# Contents of cdk2-script:
wget https://files.rcsb.../1AQ1.pdb1.gz ; ... ; sf-dock.exe grindpdb 1AQ1.pdb cdk2
wget https://files.rcsb.../1CKP.pdb1.gz ; ... ; sf-dock.exe grindpdb 1CKP.pdb cdk2
...

# Run the script to grind the proteins and make a summary list of protein
# files with their ligands:
> source cdk2-script   # Now this is parallel. A serial script is cdk2-script-serial

> sf-dock.exe psim_align_all cdk2-plist cdk2_align ; rm *.grid
> dot -Tpdf -o cdk2-tree.pdf cdk2_align.AlignmentTree.dot

# KEY OPTIONS (applies to psim_align_all and psim_maketree):
#   -psim_overlap                 Minimal degree of ligand overlap between binding
#                                 sites (default 0.50).
#   -psim_keep                    Minimal value of similarity from a binding site to
#                                 any other in order to preserve as a non-outlier
#                                 (default 0.60).
#   -psim_tree                    Minimal value of similarity for an edge in a tree
#                                 any other in order to avoid breakage into
#                                 separate trees (default 0.65).

> sf-dock.exe psim_choose_k cdk2_align-c0 5 cdk2-choose

# KEY OUTPUT FILES:
#   cdk2-sum-*                    Summary for each structure with real ligands
#   cdk2-pro-<pdbcode>.mol2       Protein structure
#   cdk2-lig-<pdbcode>-<het>.mol2 Ligand structure
#   cdk2-water-<pdbcode>.mol2     Water molecules
#   cdk2_align-c0-*.mol2          Aligned proteins and transformed ligands
#   cdk2-choose                   Identifies the cluster center proteins

# Look at the aligned structures:
> pym disp.pml
```

One of the most challenging aspects of this automated process is the correct perception of ligand structures, particularly with respect to connectivity, bond order, and tautomeric state. The `grindpdb` command is successful over 95% of the time in producing a protonated set of protein and ligand structures that are suitable for direct use by Surflex-Dock. Cases where no interpretation can be made of ligand connectivity that are consistent with a low-energy conformer result in the ligand being skipped. In cases where, for example, the interpretation of alternate ligand conformations causes fatal connectivity problems, the entire complex is skipped. All structures for which at least one ligand is produced result in a summary file (here, `cdk2-sum-*`). In cases where a PDB file yields a protein structure with one or more ligands, the results are free from obvious defects over 98% of the time. The program automatically associates metal ions with the protein.

It also eliminates common co-solvents and organic anions (e.g. DMSO, ethylene glycol, tartaric acid). The full list of ignored common "ligands" is: 1PE (PEG400), ACN (Acetone), ACT (Acetate), AKG (2-Oxoglutaric acid), BET (Trimethyl glycine), BME (Beta-mercaptoethanol), BTB (Bis-tris buffer), CIT (Citric acid), DMS (Dimethyl sulfoxide), DTT (Dihydroxy-1,4-Dithiobutane), DTT (1,4-Dithiothreitol), EDO (Ethylene glycol), EPE (Hepes), FLC (Citrate anion), GOL (Glycerol), HED (2-Hydroxyethyl disulfide), IPA (Isopropyl alcholol), MES (2-N-morpholino-ethanesulfonic acid), NO3 (Nitrate ion), P6G (Hexaethylene glycol), PEG (Di-hydroxyethyl-ether), PG4 (Tetraethylene glycol), PGE (Triethylene glycol), PO4 (Phosphate ion), POP (Pyrophosphate 2), SGM (Monothioglycerol), SO4 (Sulfate ion), TLA (L(+)-Tartaric acid), TMO (Trimethylamine oxide), and TRS (Tris buffer).

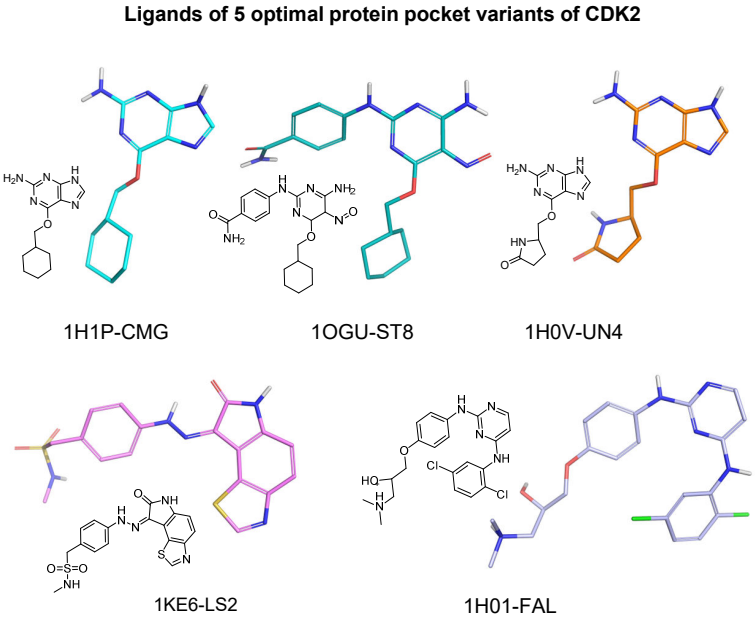


Figure 3.2 Examples of the ligand structures from those complexes chosen in the `psim_choose_k` procedure. Only polar protons are shown, but note that all protons are required for use in the docking protocols described here.

The ligand structures are built heuristically, first based on bond length inference, then geometric considerations (e.g. a planar six-membered carbon ring is probably a phenyl even if the bond lengths are too long), specific rules for common perception challenges, and finally some limited tautomeric exploration. The protein and all bound ligands are mutually optimized with respect to imidazole tautomers on the protein, tautomeric variants for the ligands, and for the orientations of hydroxyls and thiols. Note also that for three common metal chelation moieties (e.g. hydroxamic acids, sulfonamic acids, and thiolates), protons are removed as appropriate from the ligands when they are interacting with a metal ion.

Only those ligands that pass multiple quality tests will be listed in the summary file. All ligands will be listed in log files for each PDB structure processed. Within the software distribution binary folder, there is a file "verifypdb.smi" which contains roughly 25,000 SMILES structures for PDB ligands and their corresponding HET codes. If this file is present in the directory in which PDB structures are being processed, the inferred ligand structures will be checked against the corresponding SMILES structure (including some tautomeric variants). Alternatively, the `-verifypdb` argument can be used to specify a pathname to a SMILES file (each line having a SMILES string and a HET code). This argument can be specified to `getpdb` or to individual `grindpdb` commands. If this verification option is selected, all ligands must match a HET code in the given SMILES file and the ligand structures must match. If they do not match, an attempt is made to "coerce" the ligand into the hybridization state that is specified by the SMILES structure. Note, however, that we have found that when the automatic procedure for ligand preparation does not match the curated PDB SMILES, the ligand structure may be unreliable, often with neither the PDB structure nor the inferred structure matching that observed in the published report [16].

For the structures listed in `CDK2-Early`, all 38 were automatically retrieved and processed by the `grindpdb` protocol, and all 38 yielded a protein and ligand. However, only 33 of the 38 protein/ligand complexes passed the quality control checks as seen on the `cdk2-plist`. The complexes listed in `cdk2-plist` are the input to the `psim_align_all` protocol. Figure 3.1 shows the full set of the 33 aligned structures. Figure 3.2 shows the structures of the five ligands from the structures that were automatically selected by the the `psim_choose_k` procedure, with polar protons shown. Note that complete protonation of ligands is required for use in the protocols described here. All of the ligands have multiple reasonable tautomers, and the choice here was driven primarily by the interactions

between the CDK2 hinge-binding region and the ligands (the donor/acceptor/donor motif seen in four of the five ligands). Inference of the exocyclic double-bond from the ligand of 1KE6 was driven by the need to respect planarity of the respective five-membered ring. Manual inspection of the full set of results reveals no errors, but certainly there is room for some judgment with respect to some of the ligand tautomeric states. While we have tested the procedures quite extensively on a number of targets, guarantees of correctness are not possible given the variation in the syntax, semantic use, and quality of PDB files.

3.4 PROTOMOL GENERATION

Surflex-Dock uses a pseudo-molecule, a protomol, as the target to which to align putative ligands of a protein binding site. There are two methods to generate the protomol: ligand-based and cavity-search-based. A protomol can be generated as follows:

```
# Directory: examples/docking/streptavidin
> sf-dock.exe mproto plist p1

   Surflex Version 4.5
   Ligand: 31 atoms
   Original protein (ligdist 100000.00): 7044 atoms --> 7044

   Protein: 7044 atoms
   Marking TRP-B120
   Marking GLY-A48
...
   Marking ALA-A46
   Marking GLN-A24
   Marking PHE-A130
...

   475 marked grid voxels
   NS 0 score: 1.6 (bump 0.0)
   ND 0 score: 0.8 (bump 0.2)
   ND 1 score: 1.5 (bump 0.0)
...
   NA 230 score: 1.7 (bump 0.0)
   NS 300 score: 0.9 (bump 0.0)

# KEY OUTPUT FILE:
#   p1-protomol.mol2
#   p1-corevox.mol2
```

The result is two files, prefixed by ”p1”, that include the protomol (”p1-protomol.mol2”) and the core voxels (”p1-corevox.mol2”). The core voxels mark the most deeply buried regions of the ligand. Visualization can help in choosing protomol generation parameter modifications such as altering the minimum number of voxels in the protomol or core voxels using the `-proto_minvox` or `-proto_corevox` parameters, respectively. Figure 3.3 shows the protomol generated. The ligand (biotin) is shown in cyan sticks. Note that the core voxels cover the ligand features nicely, and the p1-protomol covers a larger volume of potential binding area. It is generally a good idea to visualize the generated protomol with either the ligand, or the protein, or both, to ensure that the proper site was probed and that no errors in file parsing occurred.

3.4.1 Protein Binding Pocket Detection

If the user has no ligand to mark where an active site may be, the user has the `multicav` option for the `proto` command. For this example, multi-cavity detection can be done as follows:

```
# Directory: examples/docking/streptavidin

# Find all cavities in the protein without any ligand information.
```

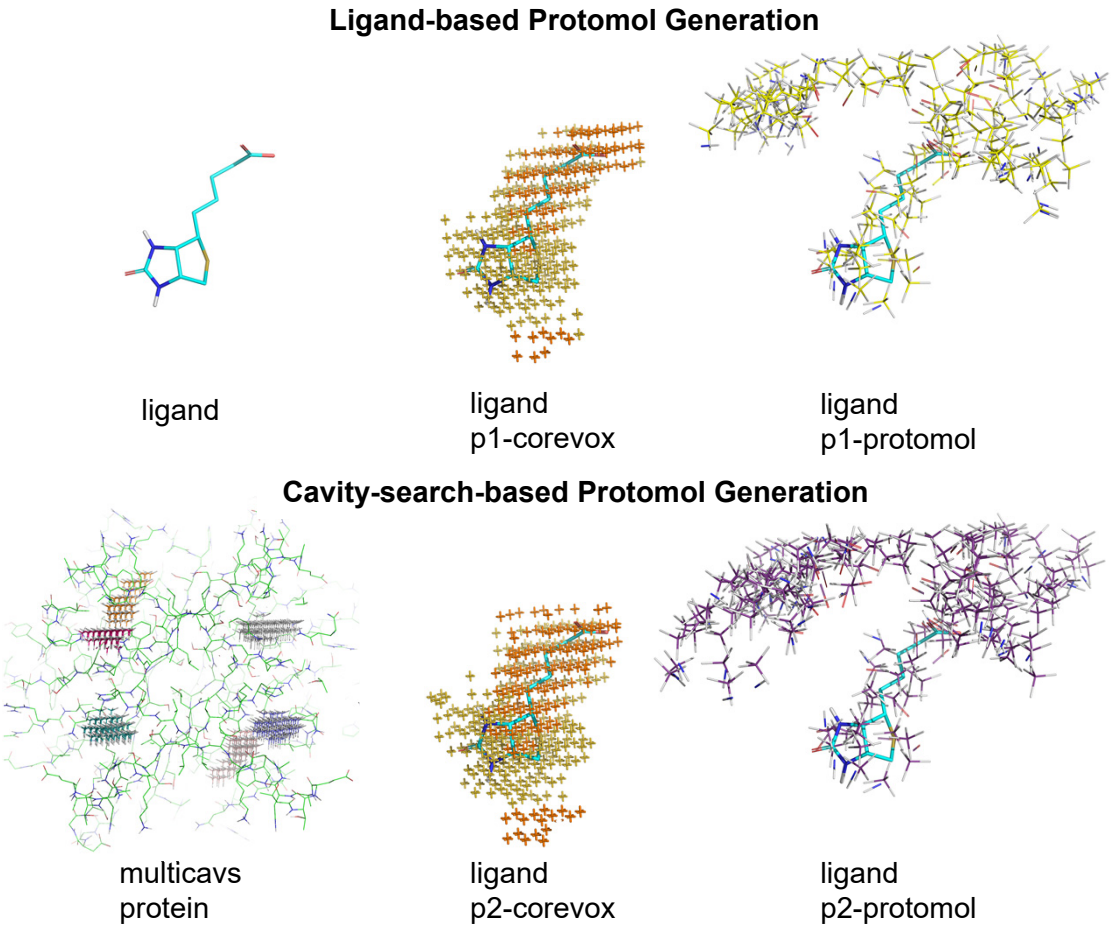


Figure 3.3 The core voxels mark the most deeply buried region of the pocket and the protomols cover a larger volume of potential binding area. The protomols are sets of polar and steric probes (p1, yellow carbons; p2, purple carbons) overlayed here on the biotin ligand (cyan). For clarity, non-polar hydrogens are not shown on the ligand and protein. The protomols generated by the ligand-based and cavity-search-based methods are highly similar.

```
# Make a protomol from one of those.
> sf-dock.exe proto multicav protein.mol2 mc
> echo protein.mol2 mc-comp-008.mol2 > plist2
> sf-dock.exe mproto plist2 p2

# KEY OUTPUT FILES:
#   mc-comp-*.mol2      All of the detected protein cavities marked with methane molecules
#   p2-protomol.mol2    Protomol generated using mc-comp-008.mol2 as a surrogate ligand
#   p2-corevox.mol2     Core voxels generated using mc-comp-008.mol2 as a surrogate ligand

# Look at the protomols and core voxels in the context of the protein:
> pym disp-multicav.pml
```

Specification of "multicav" will produce pseudo-molecule files corresponding to *all* of the protein cavities that were identified. The user must choose which of these to use to mark a binding site for protomol generation. In the case of the example above, many components are produced, one for each of the four biotin binding sites in the tetramer and one for each domain interface gap. To build a protomol for any of them, one can then specify the particular

component as the ligand. The success of the cavity-finding algorithm in identifying ligand binding sites within apo protein structures has been reported previously [18].

The default setting for `-proto_minvox` is 500 and the default for `-proto_corevox` is 100. The protomol and core voxels can be made smaller by lowering the values. However, if those values are increased, then `-proto_bloat` must be used with either the `-proto_minvox` or `-proto_corevox` parameters. Examples of modifying these parameters for the biotin/streptavidin example are shown here:

```
# Directory: examples/docking/streptavidin

> sf-dock.exe -proto_minvox 100 proto ligand.mol2 protein.mol2 sa-minvox-100
> sf-dock.exe -proto_corevox 50 proto ligand.mol2 protein.mol2 sa-corevox-50
> sf-dock.exe -proto_bloat 10 -proto_minvox 1000 proto ligand.mol2 protein.mol2
    sa-bloat-10-minvox-1000

# KEY OUTPUT FILES:
#   sa-minvox-100-protomol.mol2             Protomol is smaller
#   sa-minvox-100-corevox.mol2              Core voxels remain the same
#   sa-corevox-50-protomol.mol2             Protomol is the same
#   sa-corevox-50-corevox.mol2              Core voxels are smaller
#   sa-bloat-10-minvox-1000-protomol.mol2   Protomol is larger
#   sa-bloat-10-minvox-1000-corevox.mol2    Core voxels remain the same
```

3.4.2 Alternative Methods for Binding Site Definition

Here, we go through alternative methods for binding site definition. These include the use of the following: protein residues, a set of voxels, a centroid, or a bounding box. Rather than making use of either a ligand or something like a ligand molecule file, these other methods make use of a simple ".pts" file. This example is completed by a comparison to the traditional Surflex-Dock methods just covered above.

A .pts file can be any of 4 types: RESIDUES, VOXELS, CENTROID, or BOX. The type specification is the first line along with an integer number (N) of points. For example, the first line of `bsite_residues.pts` is "RESIDUES 20". Following the first line, there should be exactly N lines (20 in `bsite_residues.pts`), each with whitespace-delimited XYZ coordinates. In the case of RESIDUES, each of the specified points should correspond to the atom of a residue. These will be used to identify the full residues based on atomic connectivity and the residue name present in the file. This is more robust than using names explicitly, because residue naming conventions vary and in multimers, they can refer to multiple distant residues.

In the case of VOXELS, the set of points given will be used to identify a region in the binding site that is free of protein contact, essentially forming a ligand to mark the binding site, and a CENTROID is the XYZ coordinates of a single point as shown in `bsite_centroid.pts`. A BOX is defined by a set of points whose min/max XYZ defines a bounding box. There must be a minimum of two such points as shown in `bsite_box.pts`, but additional points (which may not be a literal box) will be used to defined the XYZ min/max box in the existing coordinate frame. The box will be used to identify a set of voxels that are free of protein contact, again forming a ligand to mark the binding site.

The following shows an example for each the RESIDUE, CENTROID, and BOX methods of defining a binding site. Also shown is an example for the standard ligand approach plus one example each for the two automatic cavity finding procedures, `multicav` (covered above) and `psim_findcav`.

```
# Directory: examples/docking/binding-site-definition
# Protein example: streptavidin/biotin on 3RY2
# protein/ligand files copied from ../streptavidin/ProcessPDB/
> cp ../streptavidin/ProcessPDB/stvn-lig-*3RY2*.mol2
> cp ../streptavidin/ProcessPDB/stvn-pro-*3RY2*.mol2

# Use the residues, marked by a couple of atoms each proximal to biotin
> sf-dock.exe proto bsite_residues.pts stvn-pro-3RY2.mol2 protores

# Use the approximate centroid of biotin
```

```
> sf-dock.exe proto bsite_centroid.pts stvn-pro-3RY2.mol2 protocent

# Use the bounding box of biotin defined by the two corners
> sf-dock.exe proto bsite_box.pts stvn-pro-3RY2.mol2 protobox

# The standard approach is to use a ligand:
> sf-dock.exe proto stvn-lig-3RY2-BTN.mol2 stvn-pro-3RY2.mol2 protolig

# We can also find cavities automatically.
# Here, the process finds all four of the biotin binding sites
# and also four interstitial sites that have no bound ligands:
> sf-dock.exe proto multicav stvn-pro-3RY2.mol2 protomc
# NOTE: protomc-comp-006 corresponds to stvn-lig-3RY2-BTN.mol2

# We can also find cavities and build protomols automatically
# using the psim_findcav command.
> sf-dock.exe psim_findcav stvn-pro-3RY2.mol2 protofind
# NOTE: protofind-p00[0,2,5,7] correspond to the different biotin binding sites.

# Visualize and compare the binding site definitions
> pym disp.pml
```

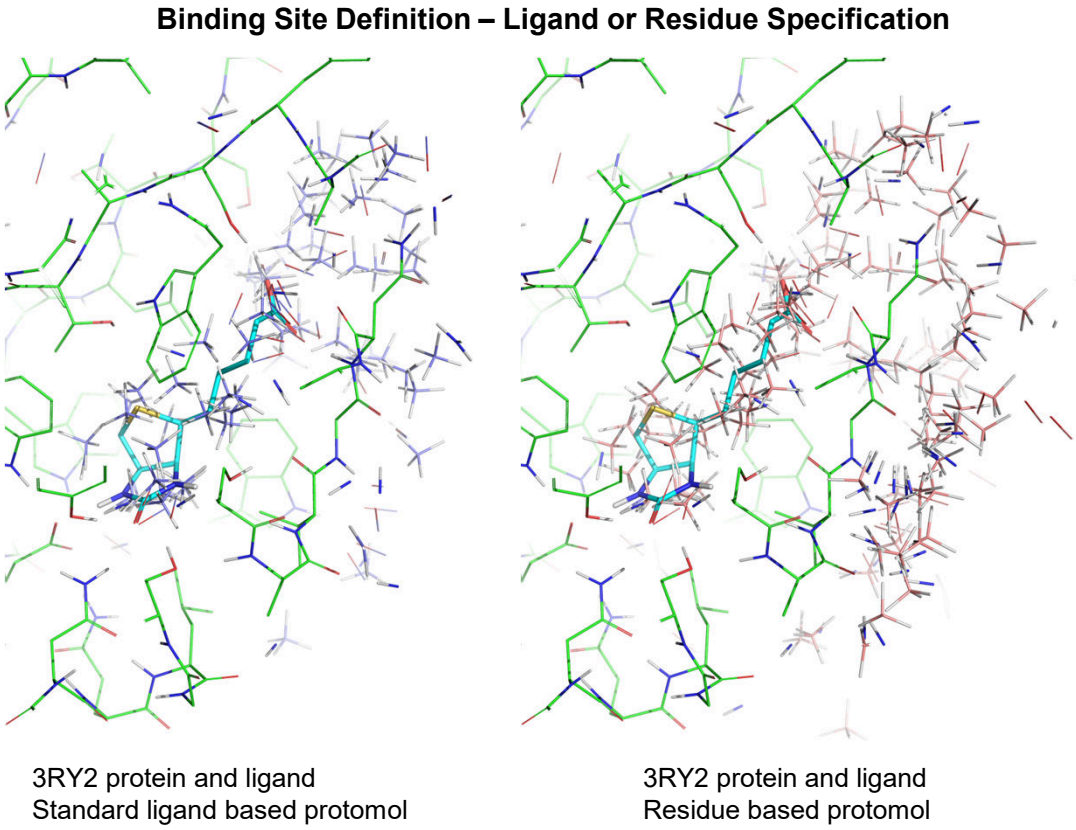


Figure 3.4 Shown are the crystallographic 3RY2 protein (green) and instantiation 1 of 4 for biotin (cyan). The thin sticks show the protomols derived from the standard ligand based protocol (left, purple sticks) and the residue based protocol (right, salmon sticks).

The six binding site definition protocols demonstrated here provide the user with extensive flexibility. Figure 3.4 shows the resulting protomols for the standard ligand procedure (left, purple sticks) and the RESIDUE procedure (right,

salmon sticks). Since the chosen residues were in the immediate vicinity of the native biotin ligand, it was expected that these two binding site definitions would yield these highly similar protomols. The protomols from the CENTROID and BOX specifications also yielded sensible and functional protomols and these can be visualized by opening the `disp.pml` file. Figure 3.5 shows the binding sites found by the two automatic cavity detection methods, `proto multicav` (left, tan sticks) and `psim_findcav` (right, pink sticks). The shown pseudo-ligand `protomc-comp-006` generated by `proto multicav` can be used in a subsequent step to generate a protomol. In contrast, `psim_findcav` performs both automatic cavity detection and protomol generation.

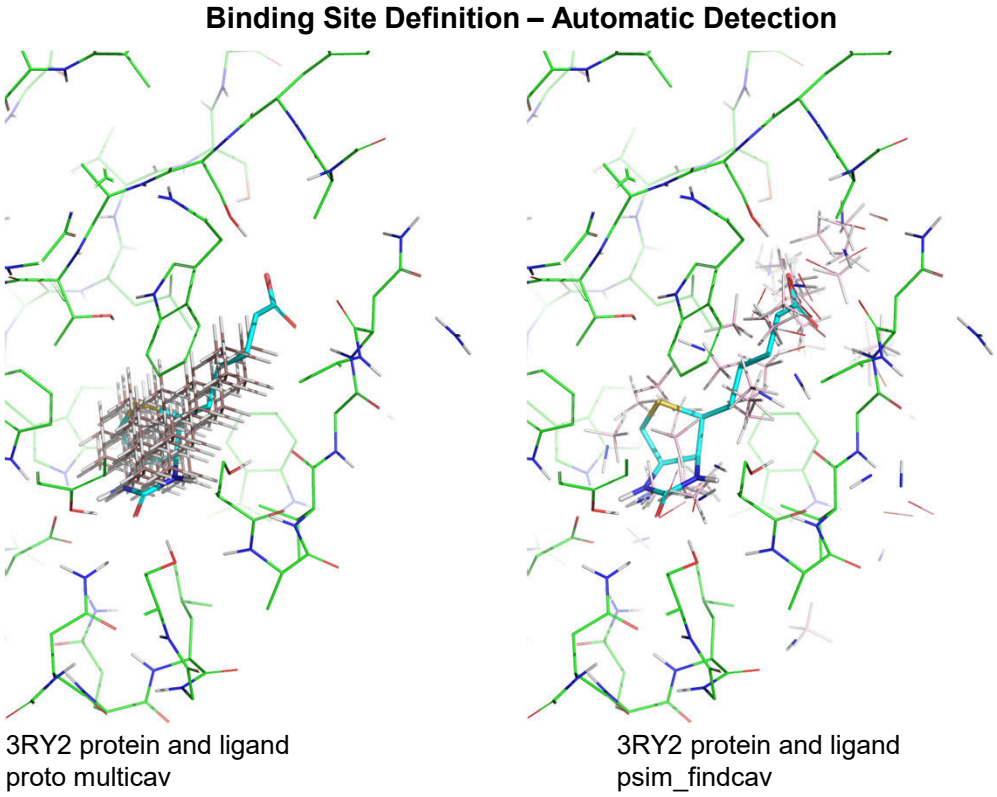


Figure 3.5 Shown are the crystallographic 3RY2 protein (green) and instantiation 1 of 4 for biotin (cyan). The thin sticks show the sites marked by two automatic cavity finding protocols: left in tan sticks, biotin site 1 found by `multicav`, and right in pink sticks, the protomol from `psim_findcav`.

3.5 DOCKING A SINGLE MOLECULE OR A LIST OF MOLECULES

Docking requires a *prepared* ligand, a protomol, and a protein. To dock a single molecule from a file containing a single conformation, using parameters for thorough search in order to produce reliable predictions of bound pose, run the following command:

```
# Directory: examples/docking/streptavidin

# Generate a randomized starting conformer
> sf-tools.exe ran_archive ligand.mol2 ligand

# Prepare the randomized conformer for docking
> sf-tools.exe -pgeom forcegen ligand-random.mol2 prep

```

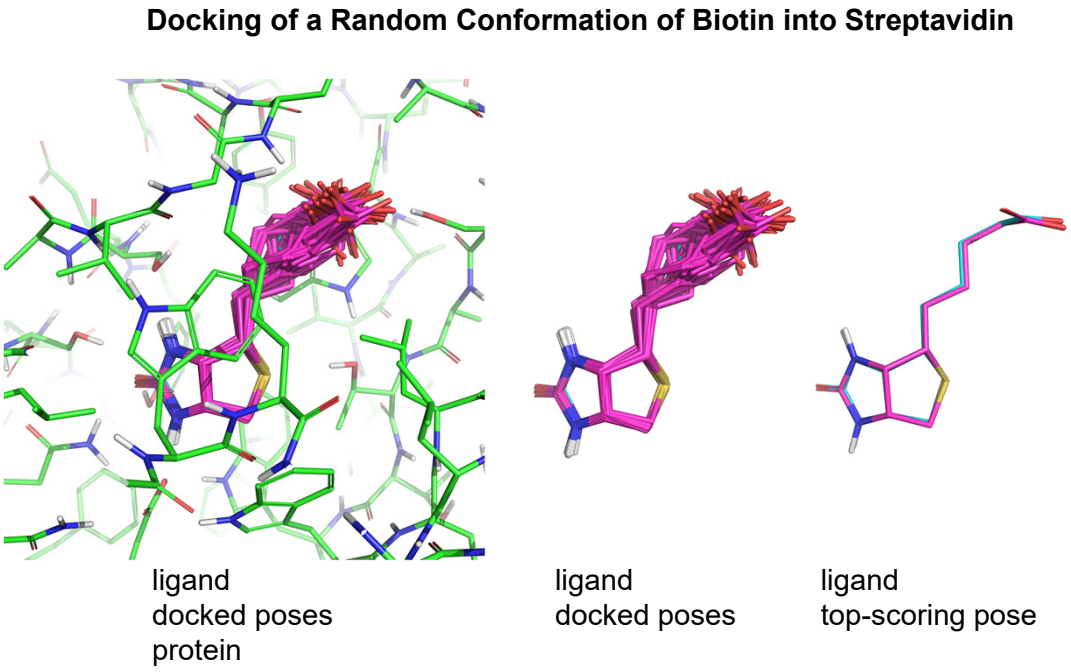


Figure 3.6 In magenta carbons are the docked conformations from a random starting pose of biotin relative to the reference ligand (cyan) shown with and without the streptavidin pocket (green). On the right is the top scoring pose overlayed on the native ligand (cyan).

```
# Then we dock the randomized conformation of biotin
> sf-dock.exe -pgeom dock prep.sfdb p1-protomol.mol2 p1-corevox.mol2 protein.mol2

# KEY OUTPUT FILES:
#   sfdock-log               The log file cataloging the scores for docked poses
#   sfdock-log-results.mol2  The docked poses themselves

# Look at the cognate docking results:
> pym disp-cognate.pml
```

Surflex-Dock will take a thorough approach to conformation and alignment optimization in order to optimize the value of its scoring function, subject to the energetic constraints of the molecule. The `dock` command is short-hand for the `dock_list` command, where the normal user-specified output prefix is simply set to "sfdock-log" instead of being specified by the user. It is much more common to actually dock a list of molecules, as follows:

```
# Directory: examples/docking/steptavidin
# FILE CONTENTS: prep.sfdb  (generated by the sf-tools/forcegen command)

# Dock the ligand as part of a list for geometric accuracy
> sf-dock.exe -pgeom dock_list prep.sfdb p1-protomol.mol2 p1-corevox.mol2 protein.mol2
    loggeom
...
Ligand biotin: 31 atoms, 5 bonds: About to align conformers (ligand 31 atoms, protomol 963
    atoms, nconfs 109)
...*********************************************Scoring_GDM: .(0).(200).(400)....
Refining: ......................
Polishing: ......................
...
Elab_Mol_in_PROTEINS: (NRot: 5)...biotin: 31 atoms, 5 rot (210.2 vol): time 6.000
    [biotin_000: 14.071 crash -0.43 polar 9.26 strain 2.01 ]
    [biotin_001: 14.020 crash -0.32 polar 9.14 strain 1.53 ]
```

```
    [biotin_002: 13.275 crash -1.29 polar 8.93 strain 5.75 ]

# KEY OUTPUT FILES:
#   loggeom
#   loggeom-results.mol2
#   loggeom-results_tab.log
```

The final output with the `-pgeom` parameter set is up to 100 conformations that are filtered to be diverse. The poses of the docked molecules are in the *-results.mol2 file. The specific log file contains information on each molecule that was docked. For the docking of biotin from above, the loggeom file was as follows:

```
biotin: 31 atoms, 5 rot (210.2 vol): time 6.000
    [biotin_000: 14.071 crash -0.43 polar 9.26 strain 2.01 ]
    [biotin_001: 14.020 crash -0.32 polar 9.14 strain 1.53 ]
    [biotin_002: 13.275 crash -1.29 polar 8.93 strain 5.75 ]
    [biotin_003: 13.228 crash -0.76 polar 8.98 strain 3.26 ]
    [biotin_004: 13.225 crash -0.77 polar 8.99 strain 3.29 ]
    [biotin_005: 12.673 crash -1.51 polar 8.64 strain 6.79 ]
...
```

The name of each molecule is followed by information from the docking, including the total time (in seconds). There are four scores for each docked conformation: a nominal affinity that is considered to be the total score (-log(K_d)), a crash score (also pK_d units), the portion of the total score that resulted from polar interactions, and the total strain energy of the conformer in kcal/mol. The polar contribution is the amount of the total affinity score that is due to polar interactions. This value can be useful in eliminating, for example, docking results that make no hydrogen bonds. The crash score is the degree of inappropriate penetration into the protein by the ligand as well as the degree of internal strain that the ligand is experiencing. Crash scores that are close to 0.0 are favorable. The total reported score includes the crash score reported. The final docked conformations are reported in descending order of total score. The name of the molecule in the original input molecule file is carried through into the output, with the pose number appended.

3.6 MOLECULAR STRAIN

With the implementation of MMFF94sf within Surflex, the treatment of molecular strain has become more detailed than in prior versions. Previously, using the DREIDING approach, molecular geometries regarding bond lengths and angles were tightly respected, but torsional variations were explored freely, with ligand self-clashing being used to prevent against excessively high energy poses. Now, a baseline energy is established using the Tools module `forcegen` command (and stored in the resulting SFDB files), and strain is calculated relative to this value. Our the more recent work involving explicit modeling of bound ligand strain is recommended [19–21].

There are two parameters that govern the calculation of strain: `-mmsweight` controls the weighting of the deviation from the baseline energy value (default 0.2), and `-mmwiggle` controls amount of weighted strain that a ligand gets "for free." The nominal value of the first parameter would be 0.74 if one were to exactly equate ligand strain (measured in kcal/mol in MMFF94) and docking scores (measured in pK_d in Surflex-Dock). However, by downweighting the nominal strain value, we achieve the effect of some binding pocket accommodation without needing to explicitly model the movement of protein atoms. Further, the `-mmwiggle` parameter allows for geometric excursions beyond the solution minimum energy value without affecting a ligand's score. The default pair of values has the effect that any nominal ligand strain above the global minimum is reported (as with previous BioPharmics Platform versions) in the "crash" value, which includes the inter-molecular clashing effect.

Note that because the nominal forcefield energy is downweighted, the strain values may appear to be nominally high. Better estimates of strain can be obtained by employing the Tools module `bound_energy` command on final pose families, as follows.

```
# Directory: examples/docking/streptavidin

# It is best to generate a pose family and consider the top one
# (or the best of the top few) for strain estimation.
```

```
> sf-dock.exe posefam loggeom

# KEY OUTPUT FILES:
#   loggeom-topfam.mol2    Top ranked pose family for biotin

# Now we can compute the bound conformational energy
# using restrained local minimization:
sf-tools.exe bound_energy loggeom-topfam.mol2 none bound

# And estimate the global minimum using the SFDB we made before:
sf-tools.exe unbound_energy prep.sfdb glob

# Grab the values and calculate the strain:
> grep Boltzmann bound-log | awk '{print $6}' > bound-en
> grep Boltzmann glob-log | awk '{print $6}' > glob-en
> paste bound-en glob-en | awk '{print ($1-$2)}' > strain

# The strain is negligible for biotin: 0.06 kcal/mol
```

The intramolecular strain of a ligand is estimated by subtracting the energy of the global minimum conformer from the energy of the bound ligand. First, docked poses are grouped into geometrically related ensembles that surround a given central pose, and a Boltzmann weighted probability yields a likelihood ranking of the families. To obtain the bound energy, the top pose family is subjected to restrained local minimization. Here, as seen in file `bound-en`, the bound energy is 29.62 kcal/mol. The conformer pool `prep.sfdb` was previously generated from an agnostic (unrestrained) `-pgeom` level search of a random conformer of biotin and therefore serves as a source of the global minimum conformer. As seen in file `glob-en`, the global minimum energy is 29.56 kcal/mol. Thus, the strain for biotin is 29.62 - 29.56 = 0.06 kcal/mol (a negligible strain). Additional examples of this type of strain calculation are shown in the Advanced Applications Chapter of the manual.

In cases where receptor structures are relatively flexible, increase in the `-mmwiggle` parameter can yield improvements in virtual screening enrichment. This is also true for decreases in the `-mmsweight` parameter. The converse is true for more rigid receptor structures. Systematic variation of these values under widely varying use cases is planned and will be the subject of a forthcoming paper.

3.7 DIFFERENT DOCKING SCHEMES FOR DIFFERENT APPLICATIONS

The earliest versions of both Hammerhead and Surflex-Dock exclusively explored ligand conformations in a tightly coupled fashion with the docking process. Because of definite benefits and efficiencies from systematic conformational exploration (including rings), all docking protocols now require the use of the Tools module ligand preparation protocols. For different types of applications, different levels of conformational elaboration are required.

3.7.1 Virtual Screening: -pscreen and -pfast

Virtual screening balances the need for fast calculations (seconds per molecule) against the need for adequate sampling of the pose space available to each ligand with respect to a particular target. Surflex-Dock offers two approaches, both of which make use of pre-elaborated conformers, `-pscreen` and `-pfast`. Both approaches are widely tested.

Two examples of the use of virtual screening are provided in the examples. Both come from some of the earliest public benchmarks for measuring virtual screening efficiency [5, 22]. Both the thymidine kinase and estrogen receptor examples run as follows:

```
# Directory: examples/docking/screen-tk
# Generate the protomol using the protein and ligand
> sf-dock.exe mproto plist mpro

# Prepare the ligands for docking
> sf-tools.exe -pscreen forcegen TestMols.mol2 psact

```

```
# Combine with previously (identically) prepared decoys
> cat psact.sfdb ../Zinc1000ps.sfdb > psmols.sfdb

# Run virtual screening using pre-searched conformers
> sf-dock.exe -pscreen gdock_list psmols.sfdb mpro-targets logpscreen

# KEY OUTPUT FILES:
#   logpscreen               Scores for all ligands
#   logpscreen-results.mol2  Corresponding poses

# Generate ROC analysis
# ROC curves and plot:
> echo ; echo ; echo PSCREEN
> grep -v ZINC logpscreen | grep _000 | awk '{print $2}' > pos ; grep ZINC logpscreen |
    grep _000 | awk '{print $2}' > neg
> sf-tools.exe roc -ci 95 1000 pos neg roclogpscreen

# Look at the virtual screening results
> pym disp.pml
```

Using the `-pscreen` option, the docking procedure investigates the pre-elaborated conformations most thoroughly, with additional local optimization within the binding pocket. Particularly for ligands with relatively limited flexibility (0–10 rotatable bonds), as is often the case in screening libraries, this approach is very effective. In the case of thymidine kinase, the resulting ROC area is 0.96, with very high early enrichment.

For the estrogen receptor case, an ensemble of protein structures is used along with a the set of cognate ligands for those protein variants. This is the strongly recommended protocol (a full study comparing single- to multi-structure screening enrichment provides more details [23].

```
# Directory: examples/docking/screen-er
# Generate the protomol using the protein and ligand
> sf-dock.exe mproto plist mpro

# Prepare the ligands for docking
> sf-tools.exe -pscreen forcegen TestMols.mol2 psact

# Combine with previously (identically) prepared decoys
> cat psact.sfdb ../Zinc1000ps.sfdb > psmols.sfdb

# Run virtual screening using pre-searched conformers
# This employs five variant proteins and the cognate poses of their ligands
> sf-dock.exe -lmatch KnownPoses.mol2 -pscreen gdock_list psmols.sfdb mpro-targets
    logpscreen

# KEY OUTPUT FILES:
#   logpscreen               Scores for all ligands
#   logpscreen-results.mol2  Corresponding poses

# Generate ROC analysis
# ROC curves and plot:
> echo ; echo ; echo PSCREEN
> grep -v ZINC logpscreen | grep _000 | awk '{print $2}' > pos ; grep ZINC logpscreen |
    grep _000 | awk '{print $2}' > neg
> sf-tools.exe roc -ci 95 1000 pos neg roclogpscreen

# Look at the virtual screening results
> pym disp.pml
```

In the case of estrogen receptor (see *examples/docking/screen-er*), where much more significant flexibility exists within the ligands, using the identical procedure, we see ROC areas of 0.98–0.99. Figures 3.7 and 3.8 show the ROC curves along with well-docked examples from both targets.

A common variation for screening makes use of the faster `-pfast` search setting: this is useful when screening very large libraries of compounds. This can be run as follows:

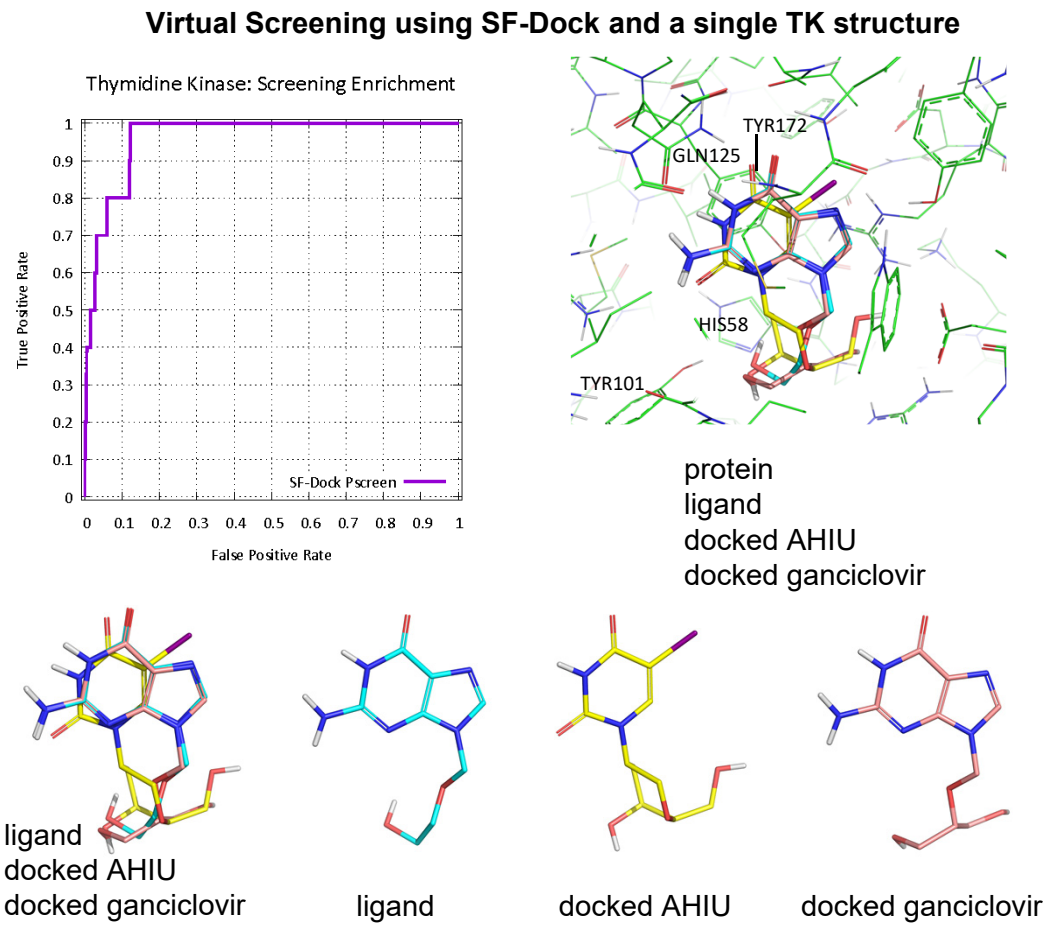


Figure 3.7 Docking results for the standard virtual screening protocol `pscreen` using thymidine kinase. The ROC curve show excellent performance for the method. The upper right shows the protein pocket and native ligand with the docked poses of positives AHIU and ganciclovir. The bottom row shows the 3D overlay of the native ligand and two docked positives as well as each molecule individually.

```
# Directory: examples/docking/screen-tk or .../screen-er
# Run virtual screening using the faster protocol (TK case)
> sf-dock.exe -pfast gdock_list psmols.sfdb mpro-targets logpfast

# Run ensemble virtual screening using the faster protocol with known poses (ER case)
> sf-dock.exe -lmatch KnownPoses.mol2 -pfast gdock_list psmols.sfdb mpro-targets logpfast

# KEY OUTPUT FILES:
#   logpfast               Scores for all ligands
#   logpfast-results.mol2  Corresponding poses
```

For the thymidine kinase and estrogen receptor cases, both the `pscreen` and `pfast` virtual screening protocols produce extremely good results.

Note that the detailed treatment of ligand strain can have an impact on screening enrichment. The default setting allows no ligand strain "for free" (`-mmwiggle 0.0`), though the nominal magnitude is downweighted by roughly a factor of four from idealized physics (`-mmsweight 0.2`). Depending on the particular target, loosening the strain weighting (increasing the wiggle or decreasing the strain weight) may produce better results, but the converse may

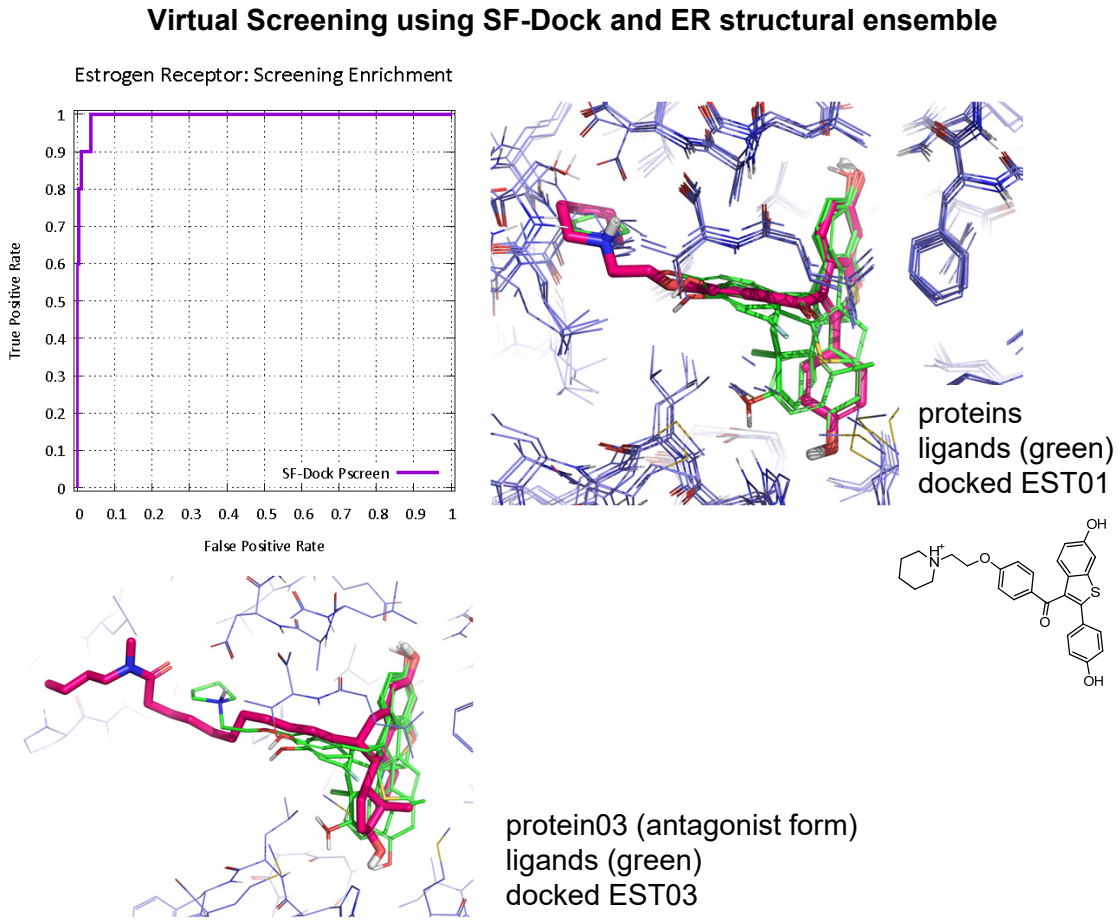


Figure 3.8 Docking results for the standard *ensemble protein* virtual screening protocol using estrogen receptor. The ROC curve shows excellent performance for the method. The upper right shows the five protein pocket variants and native ligands with the docked pose of positive EST01. The bottom left shows the docked pose of EST03, a very large antagonist, with the protein structure (PDB code 2R6Y) that is most compatible with it.

also be true. Performing some systematic experiments to understand the behavior of known positives and a pool of decoys (as used here) is recommended prior to embarking on a large screening campaign for a particular target.

3.7.2 Docking using Explicit Conformational and Positional Constraints

Surflex-Dock includes options to constrain the conformation and alignment of ligands under study, which can be useful in systematic analysis of series of molecules that share a common core. The relevant options are `-torcon` for torsional constraints during ligand prep (see the Tools chapter) and `-poscon` for positional constraints during docking. Both types of constraints are specified by providing a set of substructural fragments. An example of running both `-torcon` and `-poscon` is given using the streptavidin docking case:

```
# DIRECTORY: examples/docking/strepavidin
# KEY INPUT FILE: btn-frag.mol2

# Ensure that the underlying torsions conform to that constraint as well
> sf-tools.exe -torcon btn-frag.mol2 -pgeom forcegen ligand-random.mol2 prepcon

# KEY OUTPUT FILES:
```

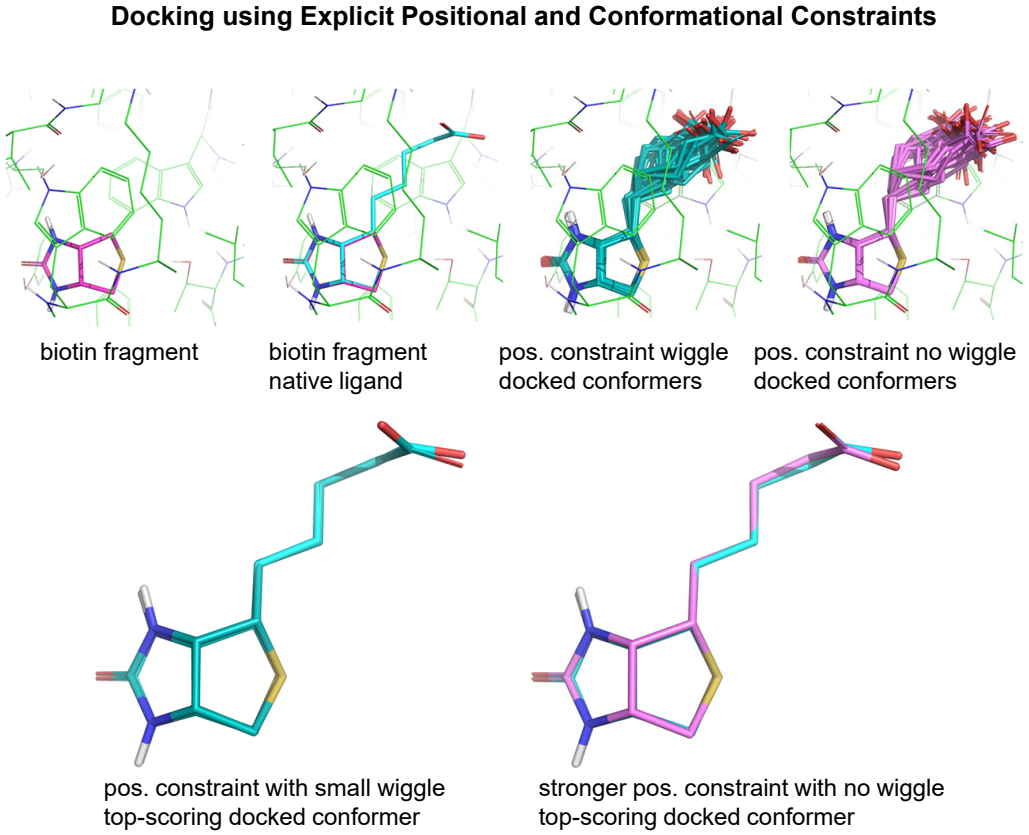


Figure 3.9 Examples of docking making use of a placed biotin fragment (magenta). Two of the four tryptophan residues in the streptavidin tetramer pocket (green lines) can be seen, one in front and one behind the ligand. Although both of the docking protocols generate top poses nearly identical to that of the native ligand (bottom row), the protocol with the stronger positional constraint yielded a pose family (violet) with tighter concordance to the native ligand head group. The user can specify positional constraints alone or include wiggle constraints.

```
# prepcon.sfdb

# Now, we will dock the ligand using the constraint:
> sf-dock.exe -pospen 5.0 -pwiggle 0.2 -poscon btn-frag.mol2 -pgeom gdock_list prepcon.sfdb
    p1-targets logcon1

# Now, we will dock the ligand using the constraint, but with no wiggle and a high penalty:
> sf-dock.exe -pospen 10.0 -pwiggle 0.0 -poscon btn-frag.mol2 -pgeom gdock_list
    prepcon.sfdb p1-targets logcon2

# Look at the positional constraint docking results:
> pym disp-poscon.pml
```

In this mode, the placed fragment tells Surflex-Dock where to place the ring system of biotin, and a flat-bottomed quadratic penalty is imposed upon deviations from the matched substructure (see the Tools chapter for more details). This guides the alignment aspect of docking, and also constrains the conformation of the subfragment. Conformational search beyond the fragment is still guided by the protomol and scoring function. This mode of docking is quite fast, and it allows for careful control on the part of the user. Using this approach with the `-pgeom` scheme may be useful in identifying optimal poses for a ligand with a particular core as well as visualizing the extent to which the ligand might move within the active site. Figure 3.9 shows the result of docking biotin with the geometric accuracy parameter `-pgeom` and positional constraint `-poscon`. The lower panel shows the top scoring docked conformer relative to

native biotin with and without a wiggle penalty `-pwiggle`. Note that both protocols result in a top pose with better placement of the constrained head fragment resulting in a favorable shifting of the biotin tail. This is due to the fixed geometry of the carbon-carbon bond that connects the biotin head and tail fragments. The protocol with the stronger positional constraint yielded a pose family (violet) with tighter concordance to the native ligand head group.

In a similar fashion, a torsional constraint can be placed on input ligands during ligand preparation. For ligands where maintaining a particular internal geometry is desirable based on either a design idea or based on a sophisticated calculation of ideal geometry, this option, `-torcon`, is used with the `forcegen` command. In a subsequent section, an example of combining a torsional constraint during conformer elaboration with ensemble docking with a positional constraint on a molecular fragment is given using thrombin. The molecular fragment is composed of a benzamidine moiety and a linker. During the `forcegen` ligand prep procedure, the `-torcon` parameter results in a torsional constraint on the linker.

NOTE: The default behavior is for ligands without matching substructures to the specified positional constraint to be skipped, but this behavior can be suppressed by specifying `-skipnonmatch`.

3.8 PROTEIN POCKET SIMILARITY AND ALIGNMENT

Frequently, when docking, several structures may be available. Surflex-Dock offers a surface-based similarity method (PSIM) for automatically and quantitatively aligning multiple protein binding sites. Surflex-Dock Version 2.6 and up offers a method to induce surface-based alignments of protein binding pockets (see [13]). There are four commands: psim_align_all, psim_matrix, psim_one, and psim_list.

The psim_align_all command works as follows:

```
# Directory: examples/docking/pde5

# FILE CONTENTS: PDBList
# 1T9S
# 1TBF
# 2H44
# 1XOZ

# Produce the pdbgrind script
> sf-dock.exe getpdb PDBList pde5

# Run it
> source pde5-script

# Align the proteins
> sf-dock.exe psim_align_all pde5-plist pde5_align ; rm *grid

# Create a pdf document of the alignment tree
> dot -Tpdf -o pde5_align-tree.pdf pde5_align.AlignmentTree.dot

# KEY OUTPUT FILES:
#   pde5-pro*mol2               Protein structures
#   pde5-lig*mol2               Ligand structures
#   pde5-plist                  List of complexes appropriate for psim_align_all
#   pde5_align-c0-lig*mol2      Aligned ligand structures
#   pde5_align-c0-pro*mol2      Aligned protein structures
#   pde5_align.AlignmentTree.dot Dot file to make a tree

# Look at the aligned PDE5 structures:
> pym disp-aligned.pml
```

The automatically generated protein site list file ("pde5-plist") contains on each line: a protein file path, a specified number of ligands that identify binding site locations, and ligand files corresponding to the sites. Common ligands such as acetate ion, ethylene glycol, and tris buffer are ignored. In the example, the proteins include PDE5 bound to icarisid II, GMP, sildenafil, and tadalafil. Tadalafil bound to PDE5 is also included, as we will be using that ligand as a test for the ensemble docking procedure. The alignment command performs an all-by-all comparison of the protein

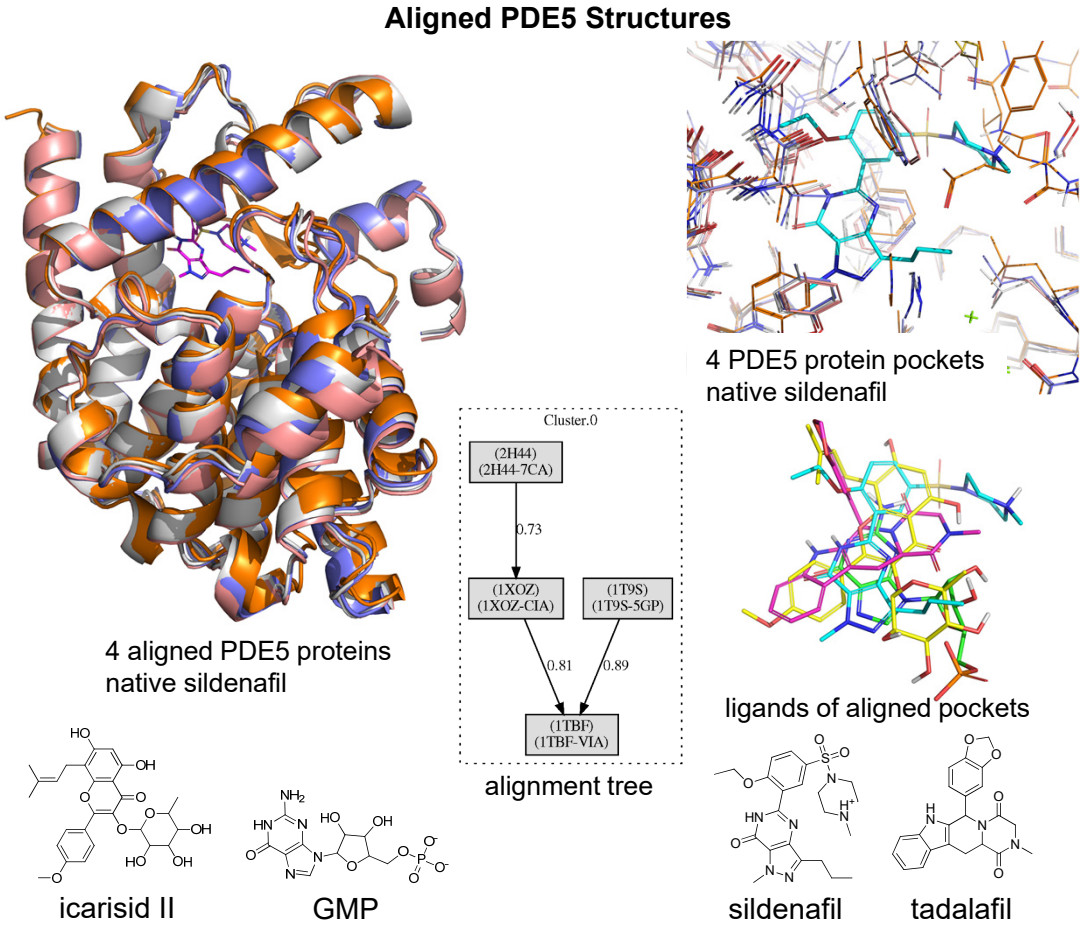


Figure 3.10 Four aligned PDE5 structures are shown in ribbons and in sticks with native sildenafil in the pocket of both depictions. The alignment of the structures (bound to icarisid II, GMP, sildenafil, and tadalafil) shows the movement possible within the pocket (upper right). The alignment tree of four PDE5 structures shows the degrees of similarity between the aligned protein binding sites.

binding sites, and it constructs a tree using single-linkage hierarchical clustering, where the edges are labeled with the binding site similarity values. PSIM will align all sites within a protein such that, for example, a binding site and an allosteric site may be assigned in different clusters. In contrast, the identical binding sites in dimers such as the estrogen receptor will likely occupy the same position in a cluster. The "dot" command (a widely available graphics package) will produce a PDF file that displays the resulting alignment tree (see Figure 3.10).

The clustering procedure may produce several separate clusters. Edges between protein binding sites are removed when they are lower than the value specified by the `-psim_tree` parameter (default 0.65). The aligned protein and ligand files and their relationship to the input files are listed in the respective cluster log file, as follows. Figure 3.10 shows the alignment of four PDE5 proteins using the psim_align_all command.

Note that the psim commands compute grids for proteins, which can be time consuming. These grids are cached to disk (*.grid), but because they are in binary format, they are not generally cross-platform compatible. So, if a problem is encountered when reading such a grid file (evident from the standard output on the console), the grids should be removed so that they will be recomputed (e.g. `rm *.grid`).

3.9 DOCKING TO MULTIPLE PROTEIN CONFORMATIONS (ENSEMBLE DOCKING)

Many proteins undergo substantial rearrangement on binding different ligands. Surflex-Dock supports docking to a set of protein targets with the `gdock_list` command being used in this new generalized protocol (beginning in Version 4.0). Multi-structure docking requires *aligned* protein structures in order to provide the full benefit of all available workflows. The "psim_align_all" command offers an ideal way to both make such alignments and to make sensible choices of which proteins should be used (it has been tested on 359 structures within a single tree). The "psim_matrix" command is identical, except that only one ligand per line is allowed in the site list file (and there is therefore no specification of the number of ligands, just the protein and ligand on each line). Often, there is a need to make a sensible choice from among many alternate structures. While it is tempting to make a choice of, say, the five most *different* structures, that can have the effect of allowing outlier variants to dominate the choice. Instead, one can choose the five central exemplars of the five most separate clusters. This is facilitated by the "psim_choose_k" command. Previously, Figure 3.2 showed the structures of five CDK2 ligands that were automatically selected by the the `psim_choose_k` procedure. These five structures will be used in an ensemble docking in a later example.

Here, we will simply choose the three structures *excluding* the cognate complex with tadalafil. The file "ProtList" contains each aligned protein and the corresponding ligand, one per line. This serves as input to the `mproto` command, which prepare the set of proteins for ensemble docking.

```
# Directory: examples/docking/pde5
# Create ProtoList for mproto:
# Contents is the aligned proteins/ligands for docking:
# pde5_align-c0-pro-1T9S-5GP.mol2 pde5_align-c0-lig-1T9S-5GP.mol2
# pde5_align-c0-pro-1TBF-VIA.mol2 pde5_align-c0-lig-1TBF-VIA.mol2
# pde5_align-c0-pro-2H44-7CA.mol2 pde5_align-c0-lig-2H44-7CA.mol2

# Rename the native pose of tadalafil
> mv pde5_align-c0-lig-1XOZ-CIA.mol2 tadalafil-bound.mol2

# Create a set of known poses from the other 3 native ligands:
> cat pde5_align-c0-lig*.mol2 > KnownPoses.mol2

# Generate the protomols and target list
> sf-dock.exe mproto ProtoList pde5

# KEY OUTPUT FILES:
#   pde5-targets          A target description file for docking
#   pde5*-protomol.mol2   Protomols for each binding site
#   pde5*-corevox.mol2    Core voxels for each binding site
```

Now we are ready to dock any number of new ligands using these three aligned protein structures as the target. In order to eliminate all bias regarding the known pose of tadalafil, we will begin from the structure generated by PubChem ("CID_110635.sdf"):

```
# Directory: examples/docking/pde5
# First, we need to prepare the ligand
> sf-tools.exe -pgeom forcegen CID_110635.sdf prep

# Dock tadalafil using the native poses of the other 3 ligands as hints
> sf-dock.exe -lmatch KnownPoses.mol2 -pgeom gdock_list prep.sfdb pde5-targets loggeom

# Build pose families using default parameters
> sf-dock.exe -posehints KnownPoses.mol2 posefam loggeom

# If desired, on can make the pose families larger by including lower scoring poses
# and by grouping slightly more different poses together
# sf-dock.exe -poseprob 1e-10 -rmsbin 2.0 -posehints KnownPoses.mol2 posefam loggeom

# KEY OUTPUT FILES:
#   loggeom                     Log file reporting the pose scores
#   loggeom-posefam             Log file reporting the pose family statistics
```

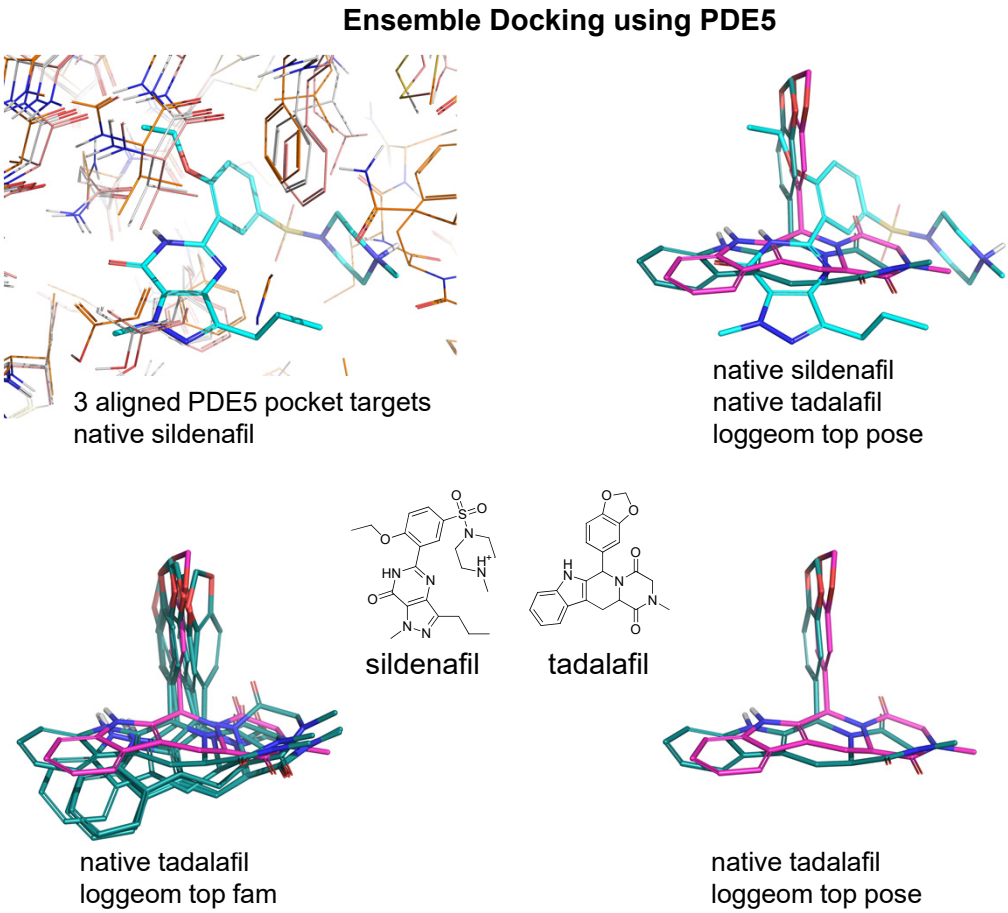


Figure 3.11 Results for docking tadalafil using gdock_list into three aligned protein structures. The cognate ligands of each of the three structures were very different from tadalafil (and from one another). These differences were reflected in the binding site configurations. Despite these challenges, the derived top-scoring pose of tadalafil was very close to that observed in the native structure (comparison shown at bottom right).

```
#   loggeom-fam.mol2              Pose families (family number and pose number
#                                                in the conformer names)
#   loggeom-topfampose.mol2       Top pose of top pose family

# Look at the multistructure cross docking results:
> pym disp-crossdock.pml
```

The overall result of the docking of tadalafil to this set of three structures (which were bound to very different scaffolds) is shown in Figure 3.11. Generation of the pose families for the tadalafil docking results made use of the `-poseprob` and `-rmsbin` parameters to illustrate the ability to re-group pose family members.

3.10 EXPLOITING ALL KNOWLEDGE IN DOCKING FOR BOUND POSE PREDICTION

The "gdock_list" approach to docking is the most thorough approach within Surflex-Dock for making pose predictions from protein structures. It can make use of pre-existing knowledge of many bound ligands, and it has been demonstrated to yield close-to-cognate performance on novel ligand docking using a very large benchmarking data set (the "PINC Benchmark") [15]. Extensive discussion of the method and its application are discussed in the paper. Use of the approach is very straightforward, with the key difference in docking being that ligands to be docked must be pre-searched. The typical use of the approach is illustrated as follows on an example involving CDK2. The structures here come from the Tools module example on protein preparation:

```
# Directory: examples/docking/cdk2
# Make a single multi-mol2 with *all* of the aligned ligands
> cat ../protein/cdk2_align-c0-lig*.mol2 > EarlyHints.mol2

# Generate a script to copy the cluster center proteins and ligands
# from examples/docking/protein:
> grep Center ../protein/cdk2-choose > cdk2-best
> cat cdk2-best | awk '{print "cp ../protein/" $2 " ../protein/" $3 " .  "}' > copy-cdk2
> source copy-cdk2

# Look at the aligned cluster center proteins
> pym disp-aligned.pml

# Construct protomols for the 5 CDK2 variants
> sf-dock.exe mproto cdk2-best cdk2dock

# KEY OUTPUT FILES:
#   EarlyHints.mol2          Aligned poses of bound ligands used for
#                            docking guidance *and* pose family prediction
#   cdk2dock*-protomol.mol2  Protomols for the 5 protein variants
#   cdk2dock*-corevox.mol2   Core voxels for the 5 protein variants
```

The approach makes use of multiple strategies for fitting the ligand into the binding pocket; consequently, it can require several minutes per ligand to complete the computation. Note that alignment and choice of protein variants using the *psim* methods yields improved performance over either random selection or selection of the N most different variants.

Given the preparation and selection, docking takes place as follows:

```
# Directory: examples/docking/cdk2
# KEY INPUT FILES:
#   cdk2dock-targets    The targets specification file (and what it points to)
#   EarlyHints.mol2     The aligned structures of known bound ligands

# We will grab an interesting future CDK2 inhibitor: the ligand of 2XNB
> echo 2XNB > getlist                    # Make this easy: no button clicks
> sf-dock.exe getpdb getlist xnb         # Make the script to get 2XNB
> source xnb-script                      # Grab and grind the protein structure
> cp xnb-lig-2XNB-Y8L.mol2 xnb.mol2      # Simpler file name
> sf-tools.exe ran_archive xnb.mol2 xnb  # Randomized ligand --> xnb-random.mol2

# Now prepare the ligand
> sf-tools.exe -pgeom forcegen xnb-random.mol2 prepxnb

# Dock the ligand using a thorough procedure with pre-existing knowledge
> sf-dock.exe -pgeom -lmatch EarlyHints.mol2 gdock_list prepxnb.sfdb cdk2dock-targets logxnb

# Generate pose families using similarity to the prior ligands to assist
> sf-dock.exe -posehints EarlyHints.mol2 posefam logxnb

# Find the true Xtal configuration in the right coordinate frame:
> sf-dock.exe psim_one xnb-pro-2XNB.mol2 xnb-lig-2XNB-Y8L.mol2
    cdk2_align-c0-pro-1KE6-LS2.mol2 cdk2_align-c0-lig-1KE6-LS2.mol2
> cp psim_ligand.mol2 gold-xnb.mol2

```

```
# Look at the top-scoring pose family and top-scoring pose relative to the native XNB pose:
> pym disp-xnb.pml

# KEY OUTPUT FILES:
#   logxnb-fam.mol2         All pose families
#   logxnb-topfampose.mol2  Top pose of top pose family
#   gold-xnb.mol2           Crystallographic pose
#   logxnb-posefam          Pose family statistics
```

The top-scoring pose family of the 2XNB ligand matches that observed from experiment.

The Surflex-Dock scoring function has been optimized to consider a large number of protein/ligand complex configurations while keeping computational costs low [12]. Increased sampling can lead to docked poses that appear to be energetically favorable but are unlikely to be biologically relevant. The pose family function was developed such that docked poses could be grouped into geometrically related ensembles that surround a given central pose. A Boltzmann weighted probability yields a likelihood ranking of the families and up to 10 pose families are generated. Figure 3.12 shows the top resulting pose family for the multi-structure docking of 2XNB to CDK2. As seen in the output file logxnb-posefam, pose family 1 has 21 poses and a 98% probability of containing the most relevant pose while pose family 2 (not shown) has 6 poses and only a 44% probability of having the most correct pose. The upper right panel of figure 3.12 shows the top scoring pose of pose family 1 relative to the pose of the 1KE6 ligand, an early hint molecule. The blue and red arrows indicate correspondence of H bond donors and acceptors, respectively. We have previously published a thorough discussion and illustration of the similarity of the surface shape and electrostatics between the predicted pose of 2XNB and an early hint molecule [15]. While the 2XNB and 1KE6 ligands are both aniline derivatives, it is the similarity of the more structurally-divergent parts that underlies the high similarity.

An example of combining ensemble docking with a positional constraint on a molecular fragment is given using thrombin, as follows:

```
# Directory: examples/docking/thrombin

# Grab some PDB structures, grind them, and align them together:
echo 1G32 > PDBList ; echo 1BHX >> PDBList ; echo 1DWD >> PDBList ; echo 1DWB >> PDBList

sf-dock.exe getpdb PDBList thr                  # Produce the pdbgrind script
source thr-script                               # Run the script
sf-dock.exe psim_align_all thr-plist thr_align  # Align the structures
cat thr_align-c0-lig-1*mol2 > PoseHints.mol2    # Produce a pose hints file

# Dock new thrombin ligands. First need to prepare them
sf-tools.exe -pgeom forcegen new-mols.mol2 pgmols

# Create ProtoList for mproto:
# Contents is the aligned proteins/ligands for docking:
# thr_align-c0-pro-1BHX-R56.mol2 thr_align-c0-lig-1BHX-R56.mol2
# thr_align-c0-pro-1DWB-BEN.mol2 thr_align-c0-lig-1DWB-BEN.mol2
# thr_align-c0-pro-1DWD-MID.mol2 thr_align-c0-lig-1DWD-MID.mol2
# thr_align-c0-pro-1G32-R11.mol2 thr_align-c0-lig-1G32-R11.mol2

# Look at the 4 aligned proteins:
> pym disp-aligned.pml

# Make protomols and then dock:
> sf-dock.exe mproto ProtoList thr
> sf-dock.exe -lmatch PoseHints.mol2 -pgeom gdock_list pgmols.sfdb thr-targets loggeom

# Make the pose families
> sf-dock.exe -posehints PoseHints.mol2 posefam loggeom

# Look at the top-scoring pose families for the 3 thrombin ligands:
> pym disp-posefam.pml

# The benzamidine fragment fits a bit differently in the pocket for
# the meta-substituted compounds. We can use a common core fragment to
```

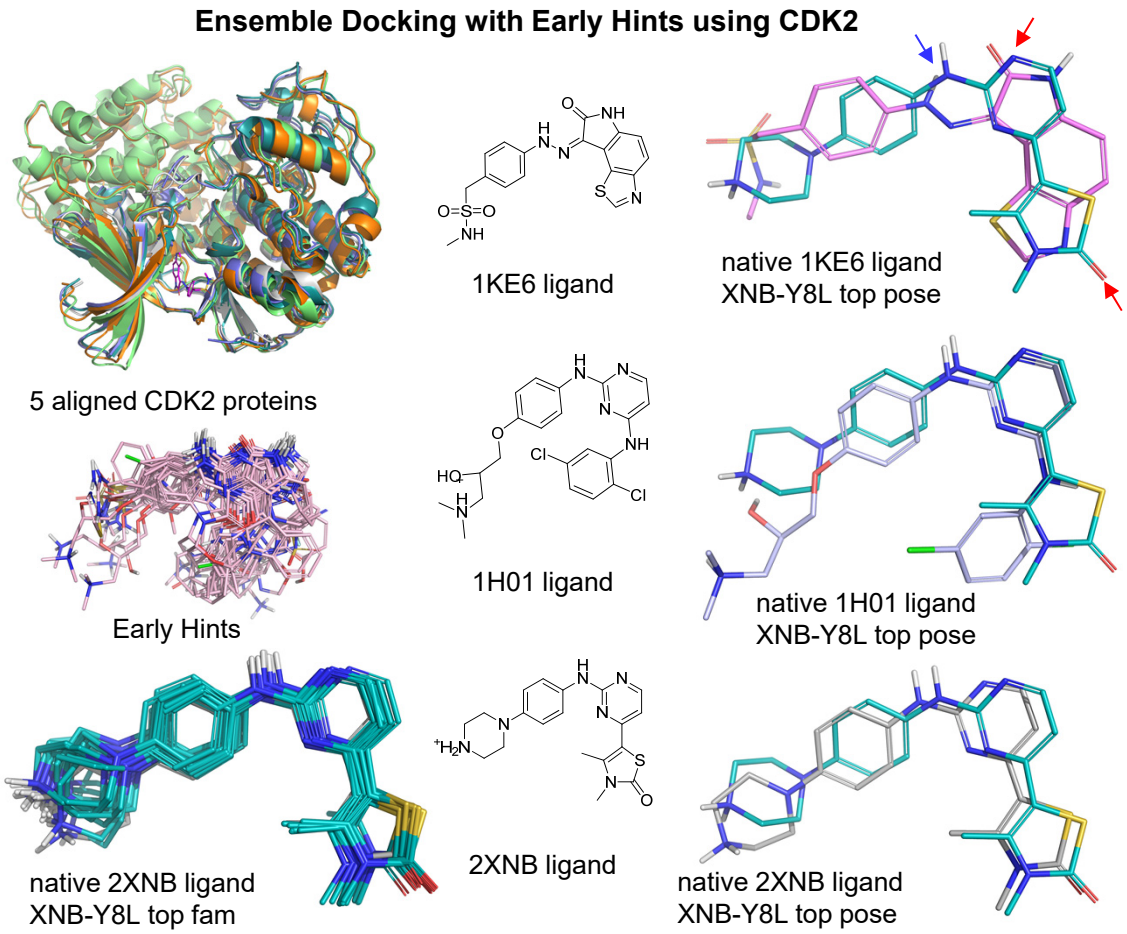


Figure 3.12 Results for docking the 2XNB ligand using `-lmatch` and Early Hints into five aligned CDK2 protein structures. The native ligands for two of the target proteins, 1KE6 and 1H01, are shown. The target proteins were the cluster centers that resulted from the psim choose k procedure. Despite the challenges, the derived top-scoring pose was very close to that observed in the native structure (comparison shown at bottom right).

```
# explore the poses that the ligand variants can adopt within thrombin

# We have made a fragment from the top-scoring pose of mol-03-fam000
# called pos-frag.mol2
# The fragment contains the benzamidine along with a part of
# the sulfonamide linker...

> sf-tools.exe -torcon pos-frag.mol2 -pquant forcegen new-mols.mol2 pcon
> sf-dock.exe -poscon pos-frag.mol2 -pquant gdock_list pcon.sfdb thr-targets logcon
> sf-dock.exe -posehints PoseHints.mol2 posefam logcon

# Now the top pose families have explored the poses where the
# subfragment has a constrained location.

# Look at the top-scoring pose families:
> pym disp-poscon.pml
```

The above series of commands first prepares a set of thrombin structures for docking, including PDB file retrieval, protonation of proteins and ligands (including inferences about bond order and tautomeric states), then a standard

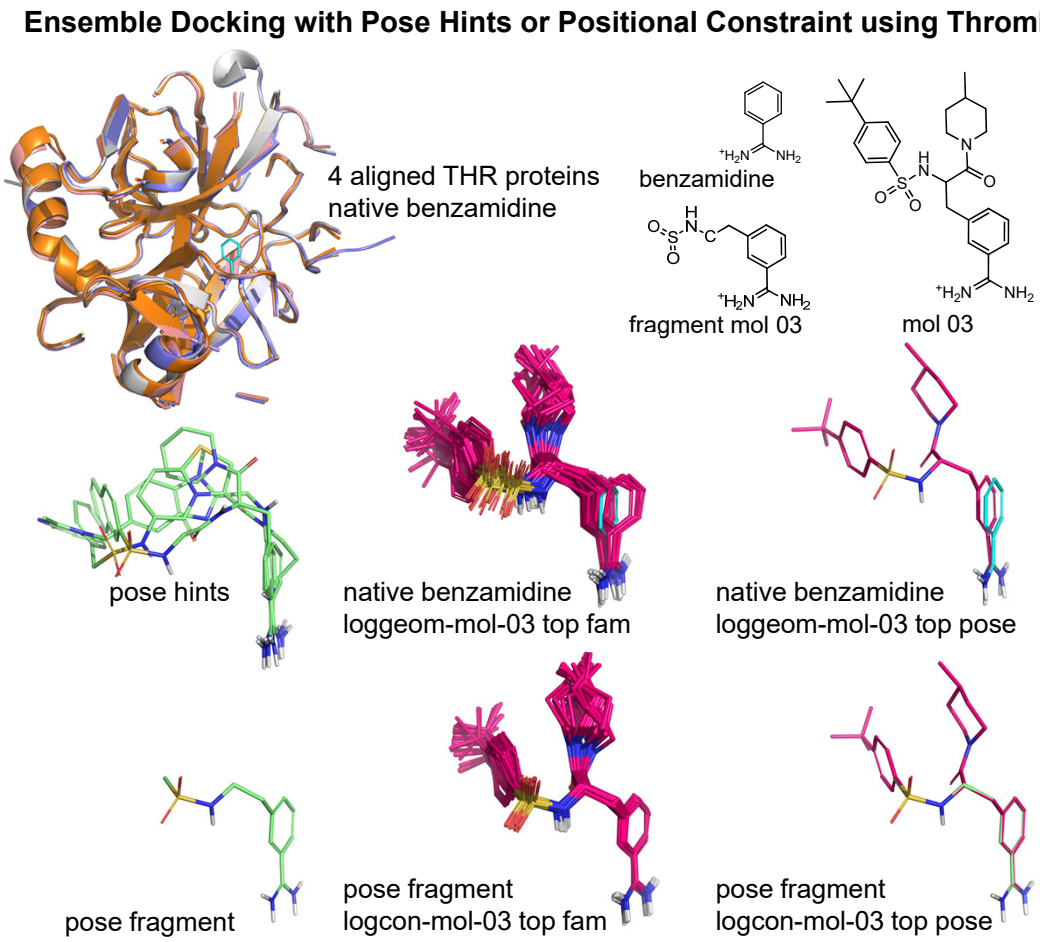


Figure 3.13 Four aligned thrombin structures and the native ligand benzamidine (cyan) are shown. The docking of thr-mol-03 was performed using either `-lmatch` with pose hints or `-poscon` with a positional fragment. For both protocols, the top scoring pose family (magenta) and the top pose are shown. The docking using pose hints resulted in a reasonable pose family and top pose. However, the pose family resulting from the use of a fragment-based torsional constraint during ligand preparation and positional constraint during docking shows remarkable coherence of poses and tight superimposition on the fragment.

geometric docking is performed. The top-scoring pose families show two different sets of positions for the benzamidine and its attached linker (see Figure 3.13).

Design of new ligands in a series may depend on ideas about precise placement of substituents within an active site. One may also want to derive a QuanSA model beginning from a well thought-out starting point. In such cases, a molecular fragment can be used to drive the docking process, focusing attention on other degrees of freedom within the ligands in question. Here, the benzamidine and its linker from the top scoring pose of the top pose family of mol-03 is used (see Figure 3.13) to re-dock the three ligands. During the `forcegen` procedure, the `-poscon` parameter results in both a positional constraint on the benzamidine *and* a torsional constraint on the linker. It would be a waste of conformational search to allow the linker to rotate since its geometry is established. Using the `-pquant` parameter scheme to allow for full exploration of the constrained poses, the final top-scoring pose families show coherent behavior (see Figure 3.13).

3.11 COVALENT DOCKING

Covalent inhibitors have been receiving increased interest, particularly with the development of reversible covalent ligands [24]. The idea is to use chemistry that allows for the covalent adduct to be reversible, especially with off-target proteins. In forming the covalent adduct the desired target protein, a specific non-covalent interaction is used to increase the propensity with which the electrophilic warhead will be subject to nucleophilic attack from the desired protein residue. This specific non-covalent interaction is lacking in off-target proteins, reducing the likelihood of forming undersirable adducts in the first place and increasing the chance that such undesirable adducts will fall apart.

A common nucleophile in protein targets is cysteine, exemplified by Bruton's tyrosine kinase (BTK). A prototype method for covalent docking has been implemented and tested in a limited fashion on the BTK target. To prepare for covalent docking, there are additional steps beyond what is required for the non-covalent docking protocols described above.

1. The protein sidechain residue (i.e. CYS-481 for BTK) must be trimmed back to the beta carbon such that the resulting residue is equivalent to alanine. This is most easily accomplished by manually editing a PDB file and simply deleting, in this example, the sulfur ATOM record. This will cause the PDB processing command `grindpdb` to parse the covalent ligand separately from the protein, *and* it will allow room for an unmodified covalent ligand to fit into the active site without clashing into the protein's nucleophile.

2. A covalent restraint specification must be made, which is comprised of four things:

   (a) A covalent fragment that will be a substructural match for the covalent warhead of putative inhibitors.

   (b) The atom ID number of the atom in the covalent fragment that will form the covalent adduct.

   (c) The 3D coordinates of the beta carbon of the protein's nucleophilic residue.

   (d) Distance bounds (lower and upper) on the required position of a ligand's warhead atom to those coordinates.

Figure 3.14 shows the covalent fragment, the trimmed protein active site of 5P9J (BTK bound to ibrutinib), and the results from docking two analogs of ibrutinib using a covalent restraint. Following is the procedure for docking, illustrated with a single protein structure as well as for an ensemble. Note that the formulation of distance bounds will be dependent on the protein target, with cysteine nucleophiles requiring shorter and narrower bounds than, for example, would be used for ligands designed as lysine modifiers.

```
# Directory: examples/docking/covalent/btk-single

# Grab the PDB structure and the ligand file manually from the PDB:
#   5p9j_B_8E8.mol2
#   5p9j.pdb --> 5p9j-orig.pdb

# Edit 5p9j-orig.pdb to delete the sulfur atom from residue 481, leaving it as an alanine.
# Delete the following two lines:
     ATOM    684  SG  CYS A 481      23.184  12.843  -1.467  1.00    21.18          S
     ANISOU  684  SG  CYS A 481     3140   2366   2542     56     -5    -32          S
# Save as 5p9j.pdb

# Grind the PDB file
> sf-dock.exe grindpdb 5p9j.pdb trg

# Make the protomol using the PDBGrind version of the ligand to locate the binding site:
> echo trg-pro-5p9j.mol2 trg-lig-5p9j-8E8.mol2 > targ
> sf-dock.exe mproto targ p1

# Create the covalent restraint specification file (called covalent-restraints):
     # Frag-Spec       Path            atom
     covalent_fragment cov-frag.mol2   2
     # Restraint values for distance to atom 674 in trg-pro-5p9j.mol2
     #              force             lb     ub      x          y          z
     covalent_bound 10.0              2.5    3.5     21.738     12.531     -0.604
```

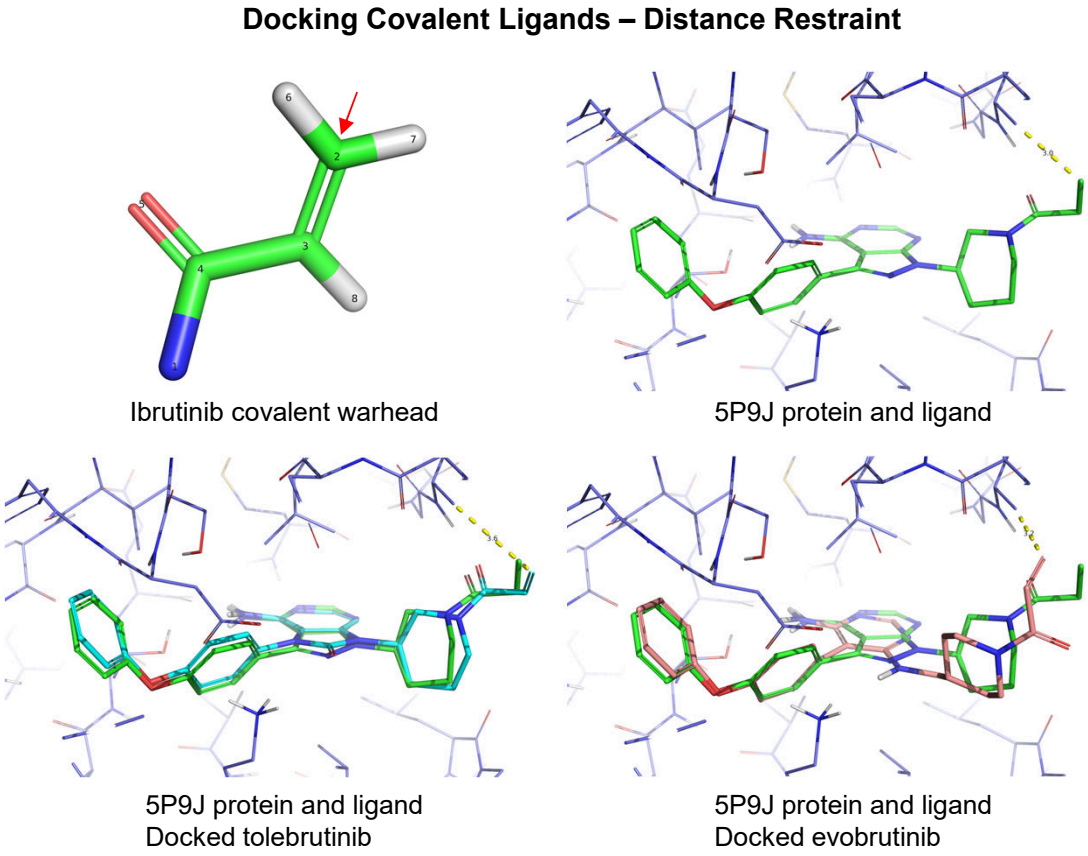


Figure 3.14 Example of covalent ligand docking using BTK. Top left: warhead subfragment to match with indication that atom 2 is the electrophile to be attacked. Top-right: trimmed cysteine residue (sidechain is now an alanine) along with the distance to the cognate covalent inhibitor ibrutinib. Bottom-left: docking of tolebrutinib compared to ibrutinib. Bottom-right: docking of evobrutinib compared to ibrutinib.

```
# Generate 3D of intact ligands (with warheads before covalent modification)
# and perform conformational search:
> sf-tools.exe fgen3d newligs.smi new
> sf-tools.exe -pgeom forcegen new.mol2 pg-new

# Dock using the native ligand as a prior known and specify the covalent restraint:
> sf-dock.exe -ndock_final 10 -lmatch trg-lig-5p9j-8E8.mol2 -covcon covalent-restraint
              -pgeom gdock_list pg-new.sfdb p1-targets pgnew
# Build pose families
> sf-dock.exe -posehints trg-lig-5p9j-8E8.mol2 posefam pgnew

# KEY FILES:
#   covalent-restraint       Specification of covalent restraint
#   cov-frag.mol2            Warhead match fragment
#   trg-pro-5p9j.mol2        Prepared protein with modified CYS-481 residue
#   pgnew-topfam.mol2        Top pose families for two new ligands
```

The process of covalent docking is algorithmically similar to that of non-covalent docking, with the addition of a distance-bounds based penalty term. That term is defined by the match of the covalent restraint fragment to the ligand to be docked, which identifies the atom on the ligand that will be subject to attack by the protein. The lower and upper bound in the `covalent-restraint` file, along with the specified force, creates a square-welled quadratic penalty that

will ensure that the covalent attachment will occur, if it is geometrically possible in the context of the non-covalent binding event that is helping to drive the initial reaction.

It is important to note that such reactions may allow for a good deal of variation in the geometry of binding between the ligand warhead and the target protein, including conformational variation in the warhead itself. In Figure 3.14, we see that the predicte bound poses of tolebrutinib and evobrutinib follow what would be expected from ibrutinib. Their respective top scoring poses were roughly 10.8 and 6.9, compared with 10.0 of the re-docking of ibrutinib. Their respective warhead atoms land well with respect to the covalent positional/distance restraint specified by the covalent restraint file.

The foregoing example made use of a single protein structure variant and docked close analogs of the cognate ligand. The covalent docking protocol can also be applied to protein ensembles and sucessfully predict poses for more novel ligands. Figure 3.15 shows the docking of two novel BTK ligands compared with their experimentally determined poses using an ensemble of five protein structures. The ligand of 4YHF (left) is similar to ibrutinib, but the warhead is a t-butyl α-cyanoacrylamide. The ligand of 6TFP (remibrutinib, right) has a very different non-covalent binding substructure from ibrutinib, but shares the same warhead.

```
# Directory: examples/docking/covalent/btk-ensemble

# NOTE: This cross-docking example was hand-curated due to the complexities of covalent
#       ligands having an impact on automatic PDB processing. All CYS-481 residues
#       were hand-trimmed. This produced protein0[1-5].mol2 and ligand0[1-5].mol2
#       The ligands were combined into KnownPoses.mol2

# Build protomols
> sf-dock.exe mproto trg-list mpro

# Test ligands required hand-editing of bond orders, so they need reprotonation
> sf-tools.exe prot TestLigs-needprot.mol2 TestLigs

# Regenerate the coordinates to randomize the poses
> sf-tools.exe regen3d TestLigs.mol2 TestLigs

# Search the ligands at the -pquant level
> sf-tools.exe -pquant forcegen TestLigs-random.mol2 pqtest

# Run covalent docking with fragment that allows for matches to various acylamides
> sf-dock.exe -lmatch KnownPoses.mol2 -covcon covalent-restraint -pquant
             gdock_list pqtest.sfdb mpro-targets pqdock

# Build pose families
> sf-dock.exe -posehints KnownPoses.mol2 posefam pqdock

# KEY FILES:
#   covalent-restraint       Specification of covalent restraint
#   cov-frag.mol2            Warhead match fragment
#   protein-*.mol2           Prepared proteins with modified CYS-481 residues
#   pqdock-topfam.mol2       Top pose families novel ligands
```

The procedure follows the same pattern as for the single-structure docking protocol above. Correctly editing the PDB files to delete the covalent attachment to the cognate ligands for the protein variants is critical. Also, the warhead fragment used in the ensemble docking was modified so that it could match a greater variety of functionalized acrylamides. In particular, the protons with atom IDs 7 and 8 (from Figure 3.14) were deleted. This allowed the fragment to match, for example, the t-butyl α-cyanoacrylamide of the 4YHF ligand. In all, eight novel ligands were docked in this ensemble docking example, with two shown in Figure 3.15. Both predicted binding poses were very close to the experimentally determined poses, despite the significant changes of the warhead in one case and the non-covalent binding component of the other.

This example made use of BTK, which has a nucleophilic cysteine, but other protein residues can be used in the design of reversible covalent inhibitors. The strategy for docking is the same: the protein residue to participate in the covalent adduct should be trimmed back to the beta carbon, resulting in an alanine. Distance bounds to the warhead

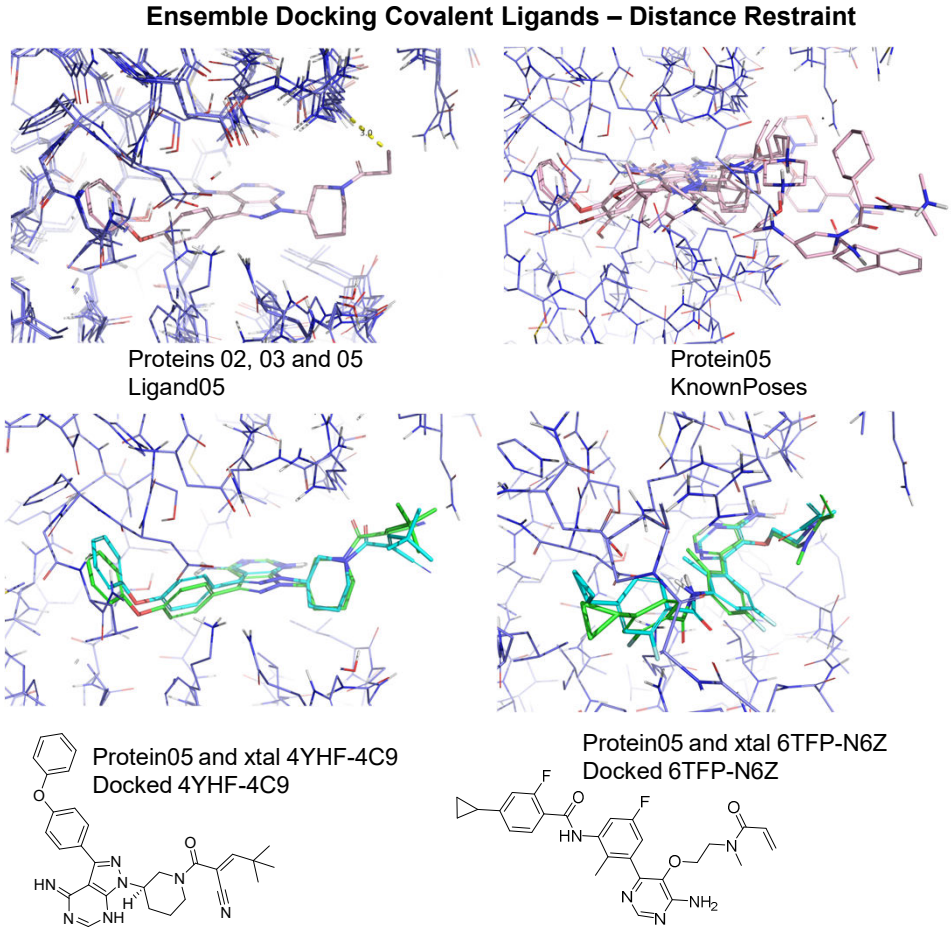


Figure 3.15 Example of covalent ligand docking using an ensemble of BTK structures. Top left: protein pocket variants shown with ibrutinib (Ligand05). Top-right: Protein05 (5P9J, slate blue) shown with the five cognate ligands (salmon). Bottom-left: docking of a t-butyl α-cyanoacrylamide compared to its experimental pose. Bottom-right: docking of remibrutinib compared to its experimental pose.

atom will vary. For example, a lysine modifier will have a larger preferred distance and a wider gap between lower and upper bounds compared with cysteine. It is recommended to make use of multiple cognate ligands with experimentally determined binding modes in order to help define appropriate bounds for the covalent restraint.

3.12 POST-PROCESSING RESULTS

The recommended procedures for post-processing involve simple use of Unix-style utilities for identification of high-scoring ligands based on any combination of criteria a user desires. For example, the following will retrieve all molecules from docking log and results files that achieved scores greater than 7.0.

```
> grep _000 docking-log | awk '{if ($2 > 7.0) print $1}' | sed s/:// > hitlist
> sf-tools.exe mget docking-log-results.mol2 hitlist poses.mol2
```

This uses grep, awk, and sed in combination with the Tools module helper command `mget` to retrieve the top scoring poses for each molecule in the hit list. It is strongly recommended that users of the command-line BioPharmics Platform tools obtain a working knowledge of Unix commands and scripting.

3.13 MISCELLANEOUS DOCKING COMMANDS AND ASSOCIATED OPTIONS

The following covers Surflex-Dock commands and options not discussed in detail above. Note that options that are discussed very thinly are not recommended for user experimentation.

The protomol generation command `mproto` was used in the streptavidin, CDK2, PDE5, and thrombin docking examples as well as the thymidine kinase and estrogen receptor virtual screening examples.

The `-pgeom` protocol was used in the streptavidin, CDK2, PDE5, and thrombin docking examples. The `-pscreen` and `-pfast` protocols were used in the thymidine kinase and estrogen receptor virtual screening examples. The `-pquant` protocol was used in the docking with a positional constraint example using thrombin.

All of the docking commands serve as wrappers for a single generalized docking protocol that can incorporate multiple strategies for pose optimization. The exposed options listed above control some details of the search process as well as the number of final conformers produces and the diversity of the final conformer set. It is *strongly recommended* to make use of the `-lmatch` parameter to specify the poses of cognate ligands (these should be prepared as if they are a target for eSim screening).

The `-multiproc` option makes it easy to split a large virtual screening run across multiple computing nodes. Note that the Docking Module (by default) makes use of multiple computing cores. For large screening runs, better core utilization can be obtained by running N parallel single-threaded jobs. Given 8 single-core processing nodes, one would initiate runs as follows (each on a different node of a cluster or in parallel on a single 8-core computing node):

```
> sf-dock.exe -nthreads 1 -pscreen -multiproc 8 1 gdock_list Ligs.sfdb Targets logproc1
> sf-dock.exe -nthreads 1 -pscreen -multiproc 8 2 gdock_list Ligs.sfdb Targets logproc2
> sf-dock.exe -nthreads 1 -pscreen -multiproc 8 3 gdock_list Ligs.sfdb Targets logproc3
> sf-dock.exe -nthreads 1 -pscreen -multiproc 8 4 gdock_list Ligs.sfdb Targets logproc4
> sf-dock.exe -nthreads 1 -pscreen -multiproc 8 5 gdock_list Ligs.sfdb Targets logproc5
> sf-dock.exe -nthreads 1 -pscreen -multiproc 8 5 gdock_list Ligs.sfdb Targets logproc6
> sf-dock.exe -nthreads 1 -pscreen -multiproc 8 7 gdock_list Ligs.sfdb Targets logproc7
> sf-dock.exe -nthreads 1 -pscreen -multiproc 8 8 gdock_list Ligs.sfdb Targets logproc8
```

The final results will be contained in the union of the log and results files, which can be combined together for final analyses.

Pose families should be produced in all cases when geometric docking predictions are desired. It is *strongly recommended* to make use of pose hints from known bound ligands (making sure that they are in the coordinate frame of the current docking run!). One can vary the size of the bins that contain different pose families (`-rmsbin`), the pose family membership similarity level (`-famdiff`), and the probability threshold below which a pose family will not be reported (`-poseprob`). The PDE5 example made use of the `-poseprob` and `-rmsbin` parameters.

The `rms_list` command will take a collection of conformers of a single molecule and will compare it (correcting for internal symmetries) to the "correct" pose of the ligand.

A command for producing protein fingerprints has been added (`dock_fp`), whose syntax is as follows: `sf-dock.exe dock_fp protein.mol2 ligands.mol2 <distance> outprefix`. The specified single protein is used to statically score the ligand poses provided, with the protein atoms being included in the output fingerprints being within the specified distance of the first ligand pose. The output file is suitable for use in the similarity module's `iscreen_list` command.

Bibliography

1. A. N. Jain. Scoring noncovalent protein-ligand interactions: A continuous differentiable function tuned to compute binding affinities. *J Comput Aided Mol Des*, 10(5):427–40, 1996.
2. W. Welch, J. Ruppert, and A. N. Jain. Hammerhead: Fast, fully automated docking of flexible ligands to protein binding sites. *Chem Biol*, 3(6):449–62, 1996.
3. J. Ruppert, W. Welch, and A. N. Jain. Automatic identification and representation of protein binding sites for molecular docking. *Protein Sci*, 6(3):524–33, 1997.

4. A. N. Jain. Morphological similarity: A 3D molecular similarity method correlated with protein-ligand recognition. *J Comput Aided Mol Des*, 14(2):199–213, 2000.
5. A. N. Jain. Surflex: Fully automatic flexible molecular docking using a molecular similarity-based search engine. *J Med Chem*, 46(4):499–511, 2003.
6. A. N. Jain. Virtual screening in lead discovery and optimization. *Curr Opin Drug Discov Devel*, 7(4):396–403, 2004.
7. A. N. Jain. Scoring functions for protein-ligand docking. *Curr Protein Pept Sci*, 7(5):407–20, 2006.
8. T. A. Pham and A. N. Jain. Parameter estimation for scoring protein-ligand interactions using negative training data. *J Med Chem*, 49(20):5856–68, 2006.
9. T. A. Pham and A. N. Jain. Customizing scoring functions for docking. *J Comput Aided Mol Des*, 22(5):269–86, 2008.
10. A. N. Jain. Surflex-dock 2.1: Robust performance from ligand energetic modeling, ring flexibility, and knowledge-based search. *J Comput Aided Mol Des*, 21(5):281–306, 2007.
11. A. N. Jain. Bias, reporting, and sharing: Computational evaluations of docking methods. *J Comput Aided Mol Des*, 22(3-4):201–12, 2008.
12. A. N. Jain. Effects of protein conformation in docking: Improved pose prediction through protein pocket adaptation. *J Comput Aided Mol Des*, 23(6):355–74, 2009.
13. R. Spitzer, A. E. Cleves, and A. N. Jain. Surface-based protein binding pocket similarity. *Proteins*, 79(9):2746–63, 2011.
14. R. Spitzer and A.N. Jain. Surflex-dock: Docking benchmarks and real-world application. *J Comput Aided Mol Des*, pages 1–13, 2012.
15. A.E. Cleves and A.N. Jain. Knowledge-guided docking: Accurate prospective prediction of bound configurations of novel ligands using Surflex-Dock. *Journal of Computer-Aided Molecular Design*, pages 1–25, 2015.
16. Ann E Cleves and Ajay N Jain. ForceGen 3D structure and conformer generation: From small lead-like molecules to macrocyclic drugs. *Journal of Computer-Aided Molecular Design*, 31(5):419–439, 2017.
17. Ajay N Jain, Ann E Cleves, Qi Gao, Xiao Wang, Yizhou Liu, Edward C Sherer, and Mikhail Y Reibarkh. Complex macrocycle exploration: Parallel, heuristic, and constraint-based conformer generation using forcegen. *Journal of Computer-Aided Molecular Design*, 33(6):531–558, 2019.
18. R. Spitzer, A.E. Cleves, R. Varela, and A.N. Jain. Protein function annotation by local binding site surface similarity. *Proteins: Structure, Function, and Bioinformatics*, 82(4):679–694, 2014.
19. Alexander C Brueckner, Qiaolin Deng, Ann E Cleves, Charles A Lesburg, Juan C Alvarez, Mikhail Y Reibarkh, Edward C Sherer, and Ajay N Jain. Conformational strain of macrocyclic peptides in ligand–receptor complexes based on advanced refinement of bound-state conformers. *Journal of Medicinal Chemistry*, 64(6):3282–3298, 2021.
20. Ajay N Jain, Alexander C Brueckner, Ann E Cleves, Mikhail Reibarkh, and Edward C Sherer. A distributional model of bound ligand conformational strain: From small molecules up to large peptidic macrocycles. *Journal of Medicinal Chemistry*, 66(3):1955–1971, 2023.
21. Ajay N Jain, Alexander C Brueckner, Christine Jorge, Ann E Cleves, Purnima Khandelwal, Janet Caceres Cortes, and Luciano Mueller. Complex peptide macrocycle optimization: Combining nmr restraints with conformational analysis to guide structure-based and ligand-based design. *Journal of Computer-Aided Molecular Design*, 37(11): 519–535, 2023.
22. C. Bissantz, G. Folkers, and D. Rognan. Protein-based virtual screening of chemical databases. 1. Evaluation of different docking/scoring combinations. *Journal of Medicinal Chemistry*, 43(25):4759–4767, 2000.
23. Ann E Cleves and Ajay N Jain. Structure-based and ligand-based virtual screening on DUD-E$^+$: Performance dependence on approximations to the binding-pocket. *Journal of Chemical Information and Modeling*, 2020.

24. Faridoon, Raymond Ng, Guiping Zhang, and Jie Jack Li. An update on the discovery and development of reversible covalent inhibitors. *Medicinal Chemistry Research*, 32(6):1039–1062, 2023.

CHAPTER 4

SIMILARITY MODULE TECHNICAL MANUAL

As with docking, molecular similarity functions are implemented within a single program module. Beginning with version 4.5, the Similarity Module has implemented the eSim approach to molecular similarity. The eSim 3D similarity approach was introduced in mid-2019, with an extensive set of benchmarks for both virtual screening and pose prediction, demonstrating superior performance to numerous alternative approaches [1, 2]. The eSim method itself is an augmentation and refinement of the predecessor method Surflex-Sim, which has been extensively documented [3–11].

The eSim method adds Coulombic electrostatic field comparison to the antecedent morphological similarity method that prior versions implemented. One particular difference of notable importance is that pose optimization for molecular similarity can be accomplished extremely quickly. Search speeds of roughly between 200–300 molecules per second on a single computing core (nearly 25,000,000 molecules per day per core). The relevant screening options are `-pfast/f` and `-pscreen`, described in detail below. A paper detailing performance of the eSim method has been published concurrently with this release [1].

Preparation of ligand structures is critical to produce sensible and reliable results. Please refer to the Surflex-Tools chapter for details on ligand preparation. All molecules must be prepared using the `forcegen` command prior to processing with Surflex-Sim. Typical use cases for Surflex-Sim include elucidation of ligand binding modes and virtual screening. Many aspects of the underlying technologies are incorporated into the Surflex-Dock and Surflex-QuanSA modules. The specific use cases involving pure ligand-based molecular similarity will be addressed here. Surflex-sim shares many command line parameters (both syntactically and semantically) with Surflex-Dock, and it is recommended that users also make themselves familiar with that section of the manual. The discussion that follows will address the features that are specific to Surflex-Sim.

4.1 SURFLEX-SIM COMMAND LINE INTERFACE

Note that there may be minor variations between the figures shown in the manual and the precise results shown in the software distribution. There are no statistically significant differences, but, for example, the N^{th} ranked solution indicated in the manual may correspond more closely to the $(N-1)^{st}$ in the actual distribution. The variations are due to small algorithmic changes across minor version increments as well as cross-platform and compiler differences.

```
BioPharmics Platform Version 5.197
Usage: surflex-sim <options> <command> args

  OVERALL PARAMETER SELECTION CHOICES
      -pfastf        Extremely fast screening parameter set
      -pfast         Fast screening
  --> -pscreen       Fast/accurate screening [DEFAULT]
      -pgeom         Pose accuracy
      -pquant        Higher pose accuracy

      -nthreads     (16)   Maximum number of threads to use
      -nbuffer      (200)  Number of mols to buffer from SFDB

  MUTUAL ALIGNMENT (HYPO) COMMANDS AND SPECIFIC OPTIONS:
    mult_esim     namelist input.sfdb outprefix (use forcegen -pgeom/f or -pquant/f)
      -me_nmake     (10)   Max number of cliques to make
      -me_rms       (0.10) RMS for grouping final cliques
      -me_known     (none) Set of given poses to guide alignment
      -me_kthresh   (4.00) eSim threshold against known poses
      -me_compress  (var)  Alignment compression level (varies based on option)
      -me_norm             Turn OFF normalization of clique scores (default ON)

  SIMILARITY COMMANDS:
    esim          input.sfdb targetlist_or_archive (= esim_list input.sfdb targ "esim")
    esim_list     input.sfdb targetlist_or_archive log
      -min_output    (--)   Score below which to suppress output
      -min_2way      (--)   Two-way score below which to suppress output
      -vrange        (--)   Volume range relative to query (e.g. -vrange 0.8 1.2)
      -maxrot        (200)  Max number of rotatable bonds on which to operate
      -nfinal        (*)    Number of final poses (varies depending on parameter scheme)
      +two_way              Also report two-way eSim score (default OFF)
      -joint                Turn OFF joint treatment of multi-ligand target (default ON)
      -div_rms       (0.25) RMS minimum difference of final poses
      -win_prop    (0.0000) If non-zero, return this proportion of best hits (e.g. 0.001)
      -multiproc npc pnum   Indicates NPC processor run, current processor is PNUM
    esim_display   mol1 mol2 outprefix
    esim_static    mol1 mol2 outprefix

    diverse2d      liglist div2dout nmols
      -gsim_thresh   (1.00) GSIM threshold to terminate getting new diverse mols.
    choose_ref     mol-poses.mol2 mols.sfdb outprefix Nchoose
    imprint        input.sfdb basis-set.mol2 outprefix
      +im_one_way  Use single-sided eSim scores for imprinting (default two-way)
    iscreen_list   imprint-subj imprint-targ outprefix
    gsim           mol1 mol2
    gsim_list      liglist targetlist log
    targprep       InMols.{mol2/sdf} outprefix   --> outprefix-targ.mol2 outprefix-log
                   Protonates if needed, charges, and validates eSim query

  MOLECULE SEARCH/ALIGNMENT OPTIONS:
      -epolar        (1.00)  Weighting of the Coulombic *and* donor/acceptor eSim terms
      -esteric       (1.00)  Weighting of the steric eSim term
      -ecoul         (1.00)  Weighting of the Coulombic eSim term
      -edonacc       (1.00)  Weighting of the donor/acceptor eSim term
      -estrain       (0.05)  Weighting of ligand strain
      +exclude               Turn ON out-of-target exclusion force (DEFAULT OFF)
      -poscon      <frags>   Molecular fragments (multi-mol2 file) to constrain position
```

```
        -pospen     (5.00)  Penalty for deviating from specification (kcal per Angstrom^2)
        -pwiggle    (0.25)  Amount of free wiggle with zero penalty (Angstroms)
        -skipnonmatch       Turn OFF skipping non-matches to -poscon arg (DEFAULT ON)
    [-torcon must be applied using forcegen including penalty parameters]

MISC OPTIONS:
    +sdf                    Produce tagged sdf output from esim_list (OFF)
```

All commands should be typed lower-case. Molecular output is generally in Sybyl mol2 format, though a transition will occur to SDF. The required input file format in most cases is the Surflex-Tools SFDB compressed molecular database format (suffix is ".sfdb").

4.2 PRIMARY CHANGES IN CURRENT VERSION

General notes about the current version can be found in the Release Notes in the Foreword to this manual. Detailed notes can be found here.

4.3 VIRTUAL SCREENING WITH ESIM

Virtual screening using molecular similarity can be very effective in identifying novel ligands that share specific binding with the ligand or ligands used as the target of the screen. The eSim method may be used with a single molecule as the "query" (or "target"), or it can be used with a joint overlay of multiple ligands. The syntax of the command is the same.

4.3.1 Example 1: Carbonic Anhydrase

The primary command for virtual screening with Surflex-Sim is `esim_list`, illustrated as follows, with three variations on the protocol, the first two employing a single molecular target, and the last involving a joint target comprised of five molecules.

```
# Directory: examples/similarity/virtual_screening/cah2

# Preparation of the actives and decoys
# Build 3D structures
> sf-tools.exe fgen3d actives.smi actives
> sf-tools.exe fgen3d decoys_sampled.smi decoys

# Conformer search in the fastest screening mode
> sf-tools.exe -pfastf forcegen actives.mol2 pffact
> sf-tools.exe -pfastf forcegen decoys.mol2 pffdec

# Combine actives and decoys into a single file:
> cat pffact.sfdb pffdec.sfdb > pffall.sfdb
# Key output file: pffall.sfdb

# Fastest 3D screening: -pfastf with ForceGen/pfastf conformers
> sf-sim.exe -pfastf -nfinal 1 esim_list pffall.sfdb xtal-orig-charged.mol2 esimpf

# Generate ROC curve and statistics (for the -pfastf run):
> cat esimpf-log | grep act_000 | awk '{print $2}' > pos
> cat esimpf-log | grep dec_000 | awk '{print $2}' > neg
> sf-tools.exe roc pos neg rocpf

# Look at the virtual screening results
> pym disp-pfastf.pml

# More thorough 3D screening: -pscreen with ForceGen/pfastf conformers
> sf-sim.exe -pscreen -nfinal 1 esim_list pffall.sfdb xtal-orig-charged.mol2 esimps
```

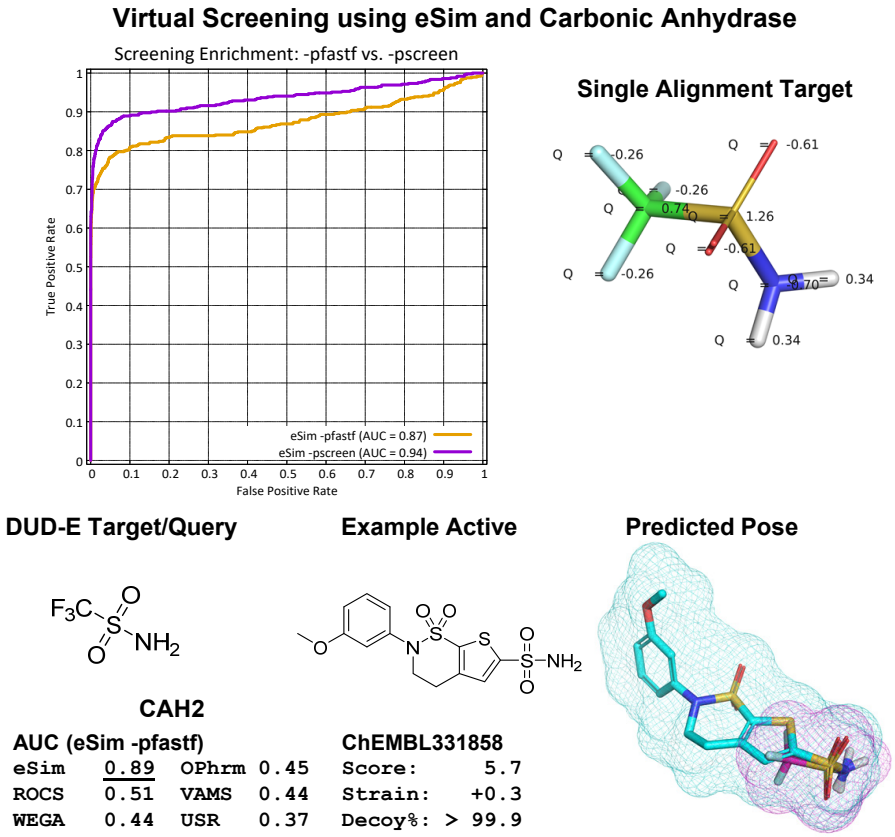


Figure 4.1 Virtual screening with eSim, using a particularly small carbonic anhydrase ligand as a target. Note that the target molecule has partial charges assigned in order to allow for eSim Coulombic field comparison. The more thorough screening mode `-pscreen` performed better than the `-pfastf` approach, though the latter yielded comparable early enrichment.

```
# Generate ROC curve and statistics (for the -pscreen run):
> cat esimps-log | grep act_000 | awk '{print $2}' > pos
> cat esimps-log | grep dec_000 | awk '{print $2}' > neg
> sf-tools.exe roc pos neg rocps

# Look at the virtual screening results
> pym disp-pscreen.pml

# KEY OUTPUT FILES:
#   esimpf-log esimpf-results.mol2
#   esimps-log esimps-results.mol2
```

The fastest screening mode, using a single molecule as a target, yields single-core processing times of hundreds of molecules per second. Figure 4.1 shows the performance of the `-pfastf` screening run and includes comparative results for alternative approaches. Performance for eSim is characterized by very high early enrichment, with an overall ROC area of 0.87 for the fastest screening mode. The eSim approach is *not* overly sensitive to size differences between a query molecule and actives to be retrieved. However, most other ligand-based approaches have difficulty in such cases, stemming from from inadequate alignment search and from the definition of their similarity functions. Here, despite the size difference, the example active is shown in its predicted alignment, which correctly corresponds the “warheads” of the carbonic anhydrase inhibitors.

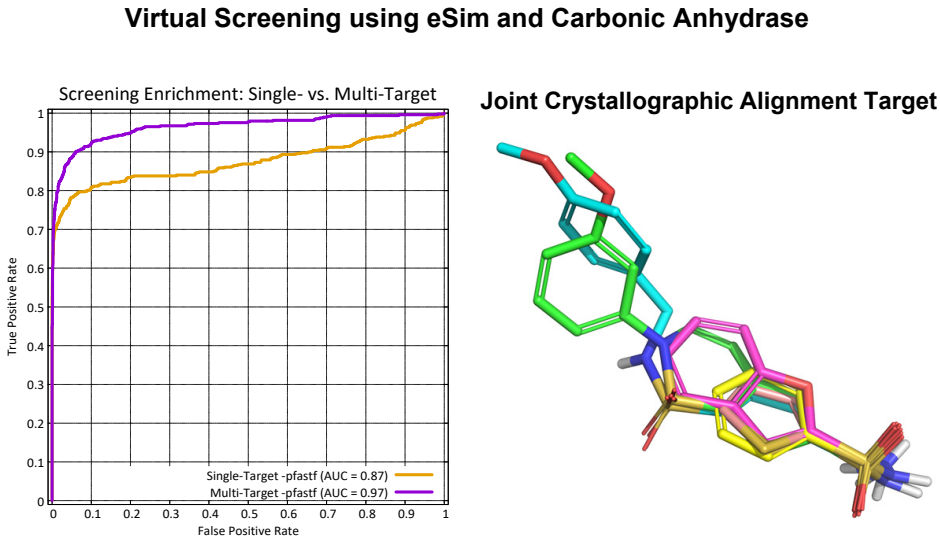


Figure 4.2 An alternative to more thorough search using a single ligand as a molecular target, which is also usually more effective, is to make use of a *joint* molecular target. Here, five typical carbonic anhydrase ligands were used, in their crystallographic poses. Performance using the `-pfastf` setting separated active and inactive ligands very effectively.

There are three screening modes for eSim: `-pfastf`, `-pfast`, and `-pscreen`. Roughly speaking, using single-core calculations, each step in thoroughness produces a five-fold decrease in speed. However, both the quality of predicted poses as well as the quantitative performance in terms of screening enrichment improves. Figure 4.1 shows the ROC curves for both the `-pfastf` and `-pscreen` modes, with the latter showing an improvement of 0.07 in ROC AUC.

Another approach to improve screening performance is to provide *more* information rather than to optimize more carefully. This can be done by making use of a *joint* target of multiple ligands. Figure 4.2 shows the crystallographic poses of five carbonic anhydrase ligands, which are much more typical than that see in Figure 4.1. The eSim approach allows ligands being aligned to "pick and choose" parts of each ligand of a joint target. The syntax is the same as when using a single target, and the processing time is generally sub-linear (i.e. using a 5 molecule joint target may only double or triple processing time).

The output of the `esim_list` command is as follows:

```
# Directory: examples/similarity/virtual_screening/cah2
# File Contents: esimpf-log (with nfinal set at the default 3)
CHEMBL331858-act: 37 atoms, 3 rot (284.2 vol): time 0.002
    [CHEMBL331858-act_000: 5.740 pen -0.02 strain 0.33 ]
    [CHEMBL331858-act_001: 5.685 pen -0.06 strain 1.30 ]
    [CHEMBL331858-act_002: 5.681 pen -0.02 strain 0.36 ]
CHEMBL273599-act: 42 atoms, 5 rot (290.2 vol): time 0.002
    [CHEMBL273599-act_000: 5.141 pen -0.03 strain 0.55 ]
    [CHEMBL273599-act_001: 5.141 pen -0.03 strain 0.55 ]
    [CHEMBL273599-act_002: 5.122 pen 0.00 strain 0.00 ]
CHEMBL282157-act: 19 atoms, 1 rot (151.4 vol): time 0.000
    [CHEMBL282157-act_000: 6.179 pen -0.00 strain 0.05 ]
    [CHEMBL282157-act_001: 6.179 pen -0.00 strain 0.05 ]
    [CHEMBL282157-act_002: 6.155 pen 0.00 strain 0.00 ]
CHEMBL77402-act: 53 atoms, 6 rot (400.8 vol): time 0.002
    [CHEMBL77402-act_000: 5.148 pen -0.06 strain 1.16 ]
    [CHEMBL77402-act_001: 3.219 pen -0.06 strain 1.16 ]
    [CHEMBL77402-act_002: 2.943 pen -0.02 strain 0.33 ]
CHEMBL304757-act: 29 atoms, 4 rot (260.6 vol): time 0.004
    [CHEMBL304757-act_000: 6.994 pen -0.02 strain 0.37 ]
    [CHEMBL304757-act_001: 6.866 pen -0.03 strain 0.54 ]
    [CHEMBL304757-act_002: 6.819 pen -0.08 strain 1.51 ]
```

```
CHEMBL140110-act: 34 atoms, 4 rot (298.0 vol): time 0.005
    [CHEMBL140110-act_000: 6.721 pen -0.29 strain 5.89 ]
    [CHEMBL140110-act_001: 6.684 pen -0.02 strain 0.40 ]
    [CHEMBL140110-act_002: 3.998 pen -0.02 strain 0.40 ]
CHEMBL487938-act: 59 atoms, 6 rot (386.6 vol): time 0.003
    [CHEMBL487938-act_000: 6.420 pen -0.07 strain 1.41 ]
    [CHEMBL487938-act_001: 6.165 pen -0.18 strain 3.53 ]
    [CHEMBL487938-act_002: 6.114 pen -0.08 strain 1.64 ]
```

For each molecule, the first line includes information about number of atoms, rotatable bonds, Van der Waals volume, and run-time. For each pose, the similarity score is provided, the calculated penalty value, and the ligand strain, measured in kcal/mol2 relative to the lowest energy conformer from the sf-tools/forcegen search. The molecule CHEMBL331858, shown in Figure 4.1, has a score of 5.7, with a strain value of 0.33 kcal/mol2. The strain yields a nominal penalty of -0.02 given that the `-estrain` parameter is set to 0.05 by default. In practice, the default value of `-estrain` yields low energy conformers; however, if yet lower energy solutions are desired, the parameter value can be increased.

One other aspect is important to understand, having to do with cases where molecules being aligned may be of significantly different size than the molecular target. Surflex-Sim uses "observer" points around a molecule's surface to compute similarity. In a screen, the target molecules have a single observer set that is used for all molecules to be screened. So comparisons of molecule A to B will, in general, yield different numerical results than comparisons of molecule B to A. For molecules of similar size, scores will tend to be close to one another, but as sizes begin to differ, they will diverge. Our experience in extensive benchmarking has been that this asymmetrical metric, which gives the degree to which a subject molecule is capable of mimicking the target but which allows for excursions, gives the best results in most systems.

4.3.2 Example 2: PARP

The case of PARP presents a more typical case of enzyme inhibition, where highly specific features such as metal chelation are not present. Here, using either single or multiple-ligand targets for virtual screening parallels the results seen with carbonic anhydrase.

```
# Directory: examples/similarity/virtual_screening/parp1

# Preparation of the actives and decoys is done in RunPrep
> source RunPrep
# Key output file: pffall.sfdb

# Fastest 3D screening: -pfastf with ForceGen/pfastf conformers
> sf-sim.exe -pfastf -nfinal 1 esim_list pffall.sfdb xtal-orig-charged.mol2 esimpf

# Generate ROC curve and statistics (for the -pfastf run):
> cat esimpf-log | grep act_000 | awk '{print $2}' > pos
> cat esimpf-log | grep dec_000 | awk '{print $2}' > neg
> sf-tools.exe roc pos neg rocpf

# Look at the virtual screening results
> pym disp-pfastf.pml

# Use a joint molecular target (five PARP inhibitors)
> sf-sim.exe -pfastf -nfinal 1 esim_list pffall.sfdb xtal-alt.mol2 jesimpf

# Generate ROC curve and statistics (for the joint target run):
> cat jesimpf-log | grep act_000 | awk '{print $2}' > pos
> cat jesimpf-log | grep dec_000 | awk '{print $2}' > neg
> sf-tools.exe roc pos neg rocjpf

# Look at the virtual screening results
> pym disp-pfastf-joint.pml

# JOINT_TARGET -PSCREEN PARAMETER
```

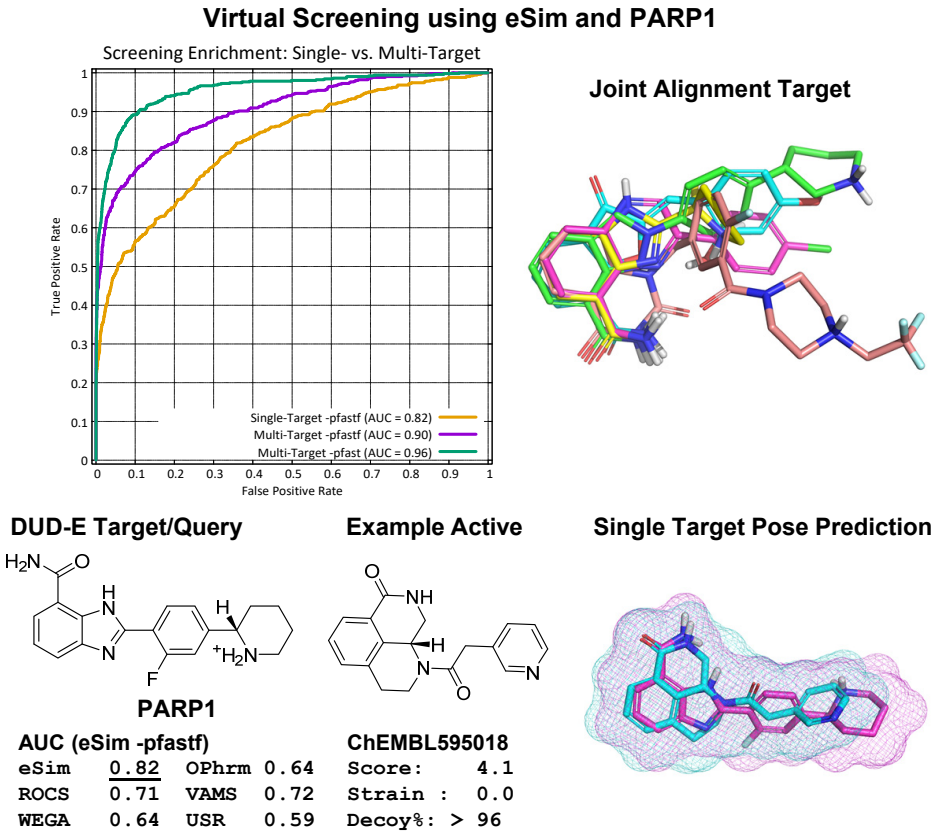


Figure 4.3 An alternative to more thorough search using a single ligand as a molecular target, which is also usually more effective, is to make use of a *joint* molecular target. Here, five typical carbonic anhydrase ligands were used (top right), in their crystallographic poses. Performance using the `-pfastf` setting separated active and inactive ligands more effectively than using a single target (bottom).

```
> sf-sim.exe -pscreen -nfinal 1 esim_list pffall.sfdb xtal-alt.mol2 jesimps
> cat jesimps-log | grep act_000 | awk '{print $2}' > pos
> cat jesimps-log | grep dec_000 | awk '{print $2}' > neg
> sf-tools.exe roc pos neg rocjps

# KEY OUTPUT FILES:
#   esimpf-log esimpf-results.mol2
#   jesimpf-log jesimpf-results.mol2
#   jesimps-log jesimps-results.mol2
```

Figure 4.3 shows the comparison between the screening performance under the three different conditions. Moving from a single molecular target to a joint one, using the fastest screening setting, increased ROC area from 0.82 to 0.92. For a slightly increased computational cost, using the second-fastest screening mode setting increased performance to an AUC of 0.96, with a maximal early enrichment of over 300-fold.

4.4 POSE PREDICTION

The PARP example offers a good opportunity to demonstrate aspects of pose prediction. Over a very large number of molecular targets, including nearly 400,000 predictions, eSim generally is able to produce a close-to-correct pose within the top 20 returned under the `-pgeom` parameter regime. That performance level was established using a

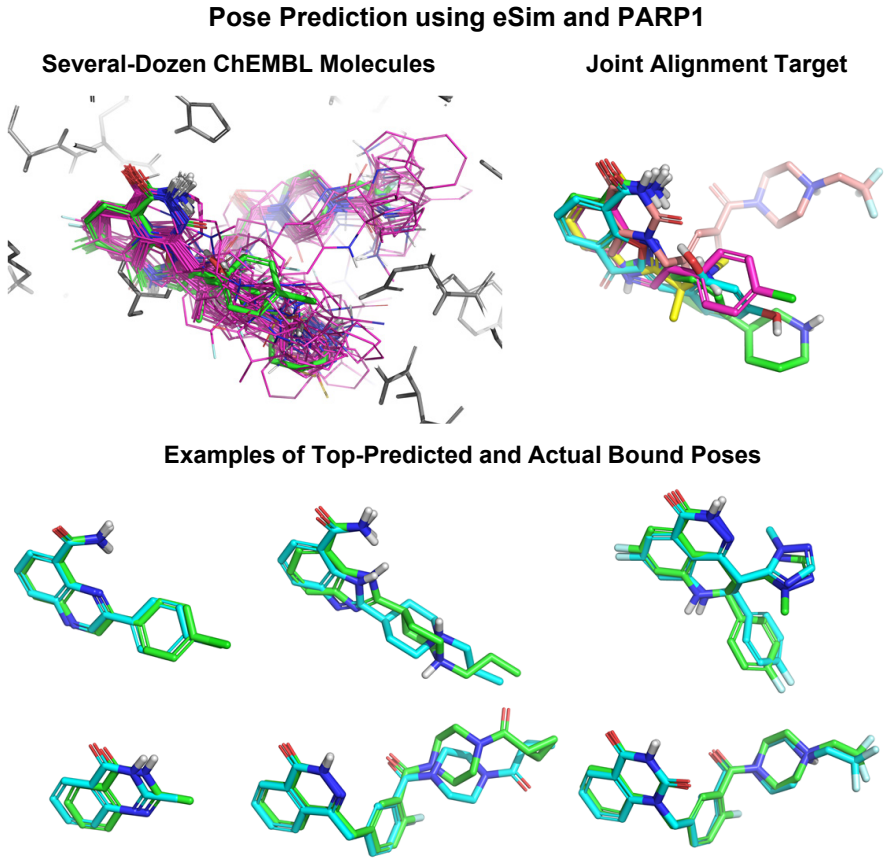


Figure 4.4 Using the `-pgeom +exclude` protocol, accurate predictions of molecular pose can be obtained using the same joint target that was employed for virtual screening. At top, the first few dozen top-scoring pose predictions for actives are shown (magenta) with the crystallographic poses of the joint target molecules (green). At bottom, six examples of accurate top-scoring pose predictions for PARP inhibitors whose bound pose was known are shown (green for the experimental poses and cyan for the prediction ones).

crystallographic pose of a target molecule and aligning a randomized version of subject molecule to it. Performance to the very top scoring pose was close to 60% correct at the 2.0 Angstrom threshold.

Generally, the `-pscreen` parameter setting produces poses that comport with intuition and also frequently agree with experiment. However, for accurate pose prediction, deeper conformational sampling is beneficial, using the ForceGen `-pgeom` or `-pquant` settings. Also, using the correspondingly more thorough eSim settings (`-pgeom` or `-pquant`) will produce more accurate and more detailed results. Note that, as above, one can mix and match different levels of conformational sampling and alignment search depth. In addition, the `+exclude` option enforces a small penalty in order to prefer poses that protrude as little as possible from the provided target (whether a single molecule or a joint target).

```
# Directory: examples/similarity/virtual_screening/parp1

# Large-scale pose prediction: Keeping inside the target
# Prepared ligands using -pscreen, align: -pgeom +exclude
> sf-sim.exe +exclude -pgeom -nfinal 1 esim_list psact.sfdb xtal-alt.mol2 jesimpgact

# Look at the pose prediction results
> pym disp-posepred.pml
```

```
# Pose prediction for ligands with known bound configurations
# Prepared ligands using -pgeom, align: -pgeom +exclude
> sf-tools.exe -pgeom forcegen xtal-random.mol2 pg-xtal
> sf-sim.exe +exclude -pgeom -nfinal 1 esim_list pg-xtal.sfdb xtal-alt.mol2 xtalpred

# Look at the pose prediction results
> pym disp-posepred-xtal.pml

# KEY OUTPUT FILES:
#   jesimpgact-log jesimpgact-results.mol2
#   xtalpred-log xtalpred-results.mol2
```

In the example above, a joint screening target is used to try to make reasonable predictions about the poses a large number of known actives. The first run uses a thorough alignment search on molecules that were searched using a ForceGen screening-level protocol, and the second uses a proper geometric search protocol for both conformer search and eSim alignment. Figure 4.4 shows results of the screening-like protocol (top) and the thorough pose prediction protocol (bottom). The screening poses clearly put the warhead variants in the correct place across the dozens of ligands shown, and the "tails" fall into the near and far positions that are defined by the joint alignment target. The six specific examples showing top-scoring poses compared with crystallographic ones are typical examples in this case, where relatively rigid ligands are to be aligned and where a clear common binding motif is present. A formal benchmark and analysis across a very large variety of targets and small molecules is presented in the eSim benchmark paper [1].

4.5 MULTIPLE LIGAND ALIGNMENT

Multiple ligand alignment is a difficult problem, but the accuracy of eSim pose predictions combined with the speed of the method provides an effective means. It may be the case that one has biophysical information that gives confidence about the bound conformation of a ligand that could serve as a target for a similarity-based screen, for example from X-ray crystallography. However, it often happens that no such information is available, in which case identifying a reasonable notion of the bioactive pose is useful. Surflex-Sim's hypothesis generation procedure offers a means to produce a mutual alignment of two or more molecules that simultaneously maximizes similarity while minimize the relative volume of the joint superimposition relative to the largest volume of a single ligand (see [4] for more information).

4.5.1 Serotonin Example

The procedure works as follows, beginning with use of the Surflex-Tools `forcegen` command.

```
# Directory: examples/similarity/multiple_alignment/serotonin
# Contents of hypo-names:
# m4a
# m8b

# Prep the ligands for the two-molecule hypothesis
> sf-tools.exe -pquant forcegen HypoMolList pqser

# Generate an alignment hypothesis.
> sf-sim.exe -pgeom mult_esim hypo-names pqser.sfdb malign

# Split the best multiple-alignment into separate files:
> sf-tools.exe splitmols malign-08.mol2 hmol

# Look at the different alignments
> pym disp.pml

# Generate a sim-stick display
> sf-sim.exe esim_display hmol-m4a.mol2 hmol-m8b.mol2 ser
```

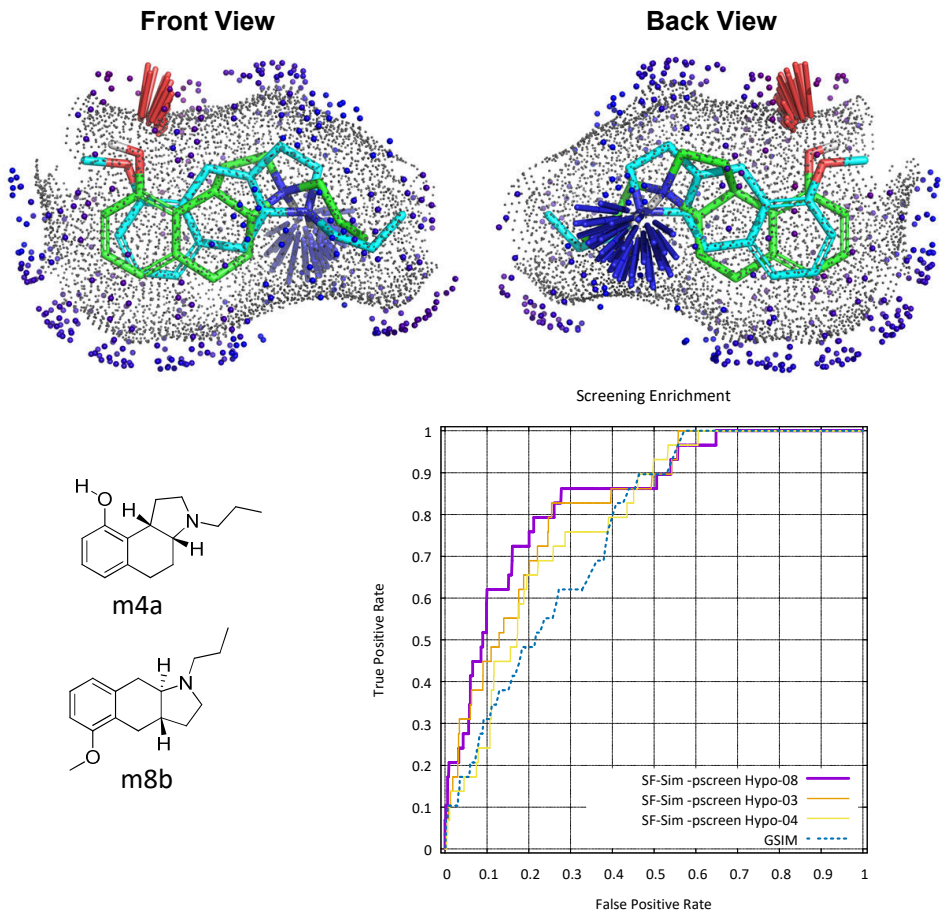


Figure 4.5 Bottom left: 2D chemical structures of m4a and m8b. Top: one multiple alignment of m4a (cyan carbons) and m8b (magenta carbons) surrounded by 4 types of similarity sticks, steric (green), Coulombic (orange), H-bond donors (red), and H-bond acceptors (blue). Bottom right: Plot of different ROC curves for different eSim 3D multiple alignments and a 2D GSIM screen.

```
20
   # Look at the similarity display
22 > pym ser-disp.pml

24 # KEY OUTPUT FILES:
   #   malign-*.mol2        All multiple ligand alignment hypotheses
26 #   malign-log           Log file with scores and strain values
   #   ser-donacc.mol2      Display of donor/acceptor similarity sticks
28 #   ser-steric.mol2      Display of steric surface overlap
   #   ser-coul.mol2        Display of Coulombic similarity
```

The `mult_esim` command produces multiple ligand alignments based on the eSim method (mesim-*.mol2). The last command above computes the static similarity of the final poses for the second hypothesis (after the hypothesis file was manually split into the respective ligands).

The `esim_display` command produces a similarity display, as shown in Figure 4.5. These two competitive serotonin ligands, despite the underlying difference in scaffolds, show a remarkably concordant surface, both with respect to pure shape (small gray surface spheres), electrostatic field (larger blue-to-red spheres), and directional hydrogen bonding preferences (blue and red sticks). The breaks in the surface concordance are minor, with "holes" where the methoxy protrudes relative to the hydroxyl and where the nitrogen substituents do not come into perfect

concordance. The protonated amines are both capable of making a salt-bridge with the same part of space, and the oxygen atoms (one from a hydroxyl and one from a methoxy) similarly can accept hydrogen-bonding from the potential donor region. A purely topological superimposition would favor exact concordance of the two matching pairs of amino-tetralin substructures. However, by flipping molecule m8b and rotating it, the resulting alignment exhibits a much higher degree of surface shape and polarity concordance.

It is important to understand that multiple ligand alignment is a challenging problem, both from a purely computational aspect owing to a large search space, and also because it is not possible to know which parts of a set of molecules are "important" and which are not. In a typical binding event, a part of a molecule will be making specific and complementary contact with the protein, often with other parts exposed to solvent. Also, protein conformations can vary significantly, so ligands may not be "seeing" the same target.

One way to assess the quality of multiple alignments is by using virtual screening. In this example, the top multiple alignments were tested for screening effectiveness, as follows:

```
# Directory: examples/similarity/multiple_alignment/serotonin

# Prepare the ligands for screening
> sf-tools.exe -pscreen forcegen TestMols.mol2 psser
> cat psser.sfdb ../../../docking/Zinc1000ps.sfdb > screen.sfdb

# eSim standard screening: -pscreen
> sf-sim.exe -pscreen esim_list screen.sfdb malign-00.mol2 p00
> sf-sim.exe -pscreen esim_list screen.sfdb malign-01.mol2 p01
...
> sf-sim.exe -pscreen esim_list screen.sfdb malign-08.mol2 p08

# GSIM screen
> sf-sim.exe gsim_list screen.sfdb malign-00.mol2 gsimlog

# Generate ROC analysis
# ROC curves and plot:
> echo ; echo ; echo HYPO-00
> grep -v ZINC p00-log | grep _000 | awk '{print $2}' > pos ; grep ZINC p00-log | grep _000
    | awk '{print $2}' > neg
> sf-tools.exe roc -ci 95 1000 pos neg rocp00

> echo ; echo ; echo HYPO-01
> grep -v ZINC p01-log | grep _000 | awk '{print $2}' > pos ; grep ZINC p01-log | grep _000
    | awk '{print $2}' > neg
> sf-tools.exe roc -ci 95 1000 pos neg rocp01

> echo ; echo ; echo HYPO-02
> grep -v ZINC p02-log | grep _000 | awk '{print $2}' > pos ; grep ZINC p02-log | grep _000
    | awk '{print $2}' > neg
> sf-tools.exe roc -ci 95 1000 pos neg rocp02

...

> echo ; echo ; echo HYPO-08
> grep -v ZINC p08-log | grep _000 | awk '{print $2}' > pos ; grep ZINC p08-log | grep _000
    | awk '{print $2}' > neg
> sf-tools.exe roc -ci 95 1000 pos neg rocp08

> grep -v ZINC gsimlog | awk '{print $6}' > pos ; grep ZINC gsimlog | awk '{print $6}' > neg
> sf-tools.exe roc -ci 95 1000 pos neg rocgsim

# KEY OUTPUT FILES:
#   p0*-log             Screening scores for different alignments
#   p0*-results.mol2    Predicted poses
#   gsimlog             GSIM 2D similarity scores
```

Here, the `malign-08` is the best (violet curve in Figure 4.5), though all performed better than 2D similarity (blue curve), especially with respect to early enrichment.

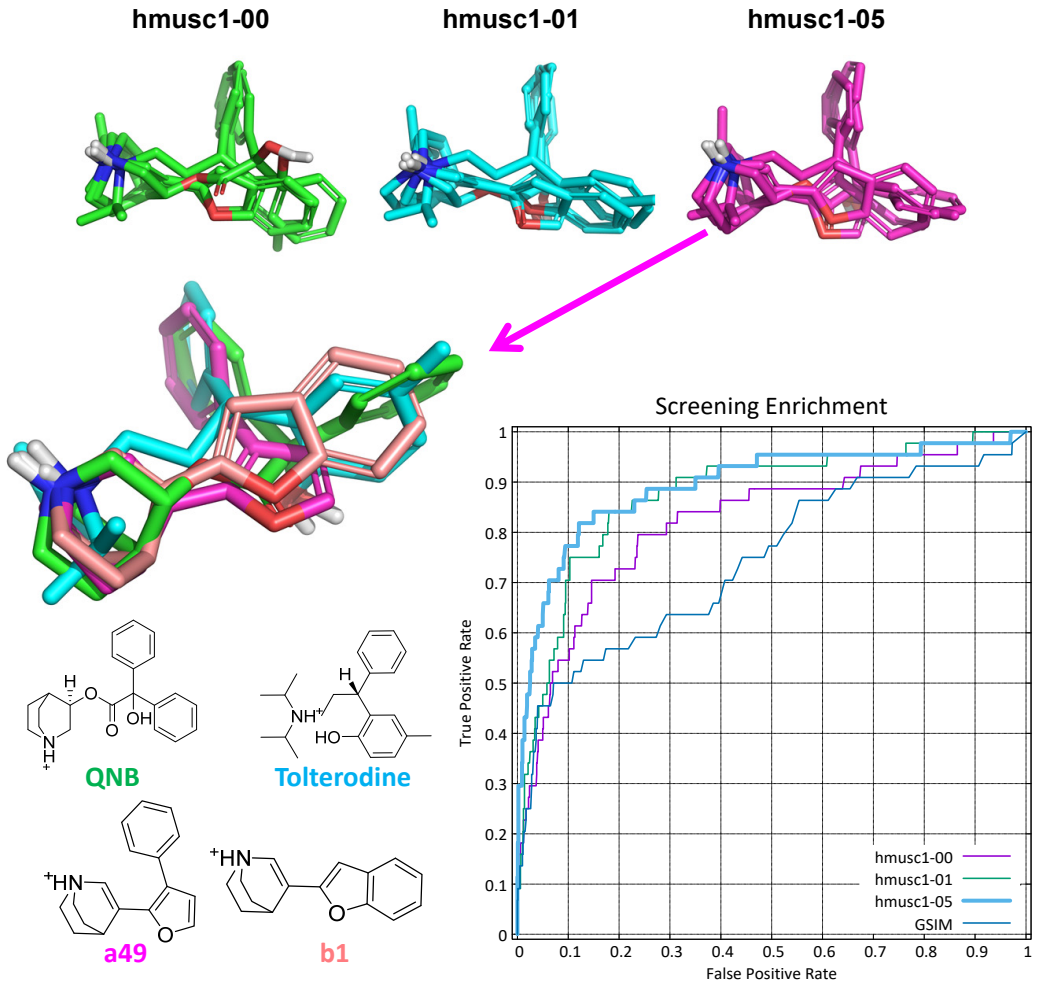


Figure 4.6 Three alternative hypotheses for the relative alignment of four muscarinic antagonists.

4.5.2 Muscarinic Example

A more complicated example of multiple ligand alignment is given using the muscarinic receptor, where more substantial ligand flexibility is present in the subject molecules [4]. Here, we will see that even producing mutual alignments of four muscarinic antagonists produces multiple reasonable alternatives.

```
# Directory: examples/similarity/multiple_alignment/musc

# First prepare the ligands thoroughly
sf-tools.exe -pquant forcegen HypoList pqmusc

# Now build a set of multiple hypotheses
sf-sim.exe -pgeom mult_esim hypo-names1 pqmusc.sfdb hmusc1

# Contents of hypo-names1:
# tolterodine
# a49
# b1
# qnb

```

```
# Look at some of the multiple ligand alignments
> pym disp-hypos.pml

# KEY OUTPUT FILES:
#   hmusc1-*.mol2  Alternative alignments
#   hmusc1-log     Information about the alignment quality and ligand strain
```

The output log file provides information about each of the final ligand superimpositions (or "cliques"):

```
# Directory: examples/similarity/multiple_alignment/musc
# Contents of hmusc1-log:
Input mols in sorted order (high to low eSim from others):
  Mol a49 mean score: 6.85
  ...
  Mol tolterodine mean score: 6.13

Final hypo 00 prob: 0.4749
           tolterodine strain: 4.2 kcal/mol
                   a49 strain: 3.0 kcal/mol
                    b1 strain: 1.8 kcal/mol
                   qnb strain: 4.3 kcal/mol

Final hypo 01 prob: 0.462652 (min_rms = 1.70)
           tolterodine strain: 5.2 kcal/mol
                   a49 strain: 2.5 kcal/mol
                    b1 strain: 1.6 kcal/mol
                   qnb strain: 3.3 kcal/mol
...
Final hypo 05 prob: 0.395539 (min_rms = 0.83)
           tolterodine strain: 5.8 kcal/mol
                   a49 strain: 1.7 kcal/mol
                    b1 strain: 1.4 kcal/mol
                   qnb strain: 2.3 kcal/mol
...
```

Figure 4.6 shows three solutions with high probability and with relatively low strain (numbers 00, 01, and 05). None has substantially different probability values, but the ligand energetics vary. As shown with the serotonin example, using a virtual screening approach can help distinguish between different multiple alignments. Here, alignment 05 shows superior performance (see the script in the examples for details), and the eSim-based 3D screens perform better than the GSIM 2D approach, as expected.

It is possible to use a selected multiple ligand alignment to guide the construction of a larger one, as follows.

```
# Directory: examples/similarity/multiple_alignment/musc

# Let's build a larger hypothesis de novo and also using a smaller
# clique to seed it. We will use the one that worked best above.  That
# was hypo-05 (Version 4.511) based on early enrichment and overall
# ROC area from the ROC plots --> defhypo.mol2

> cp hmusc1-05.mol2 defhypo.mol2

> sf-sim.exe -pgeom mult_esim hypo-names2 pqmusc.sfdb hmusc2denovo

# Here, in making use of the prior clique using the -me_known option,
# we also choose to reduce the threshold for eSim similarity using the -me_kthresh
# parameter so that all of the molecules will survive the cutoff.
> sf-sim.exe -me_kthresh 1.0 -me_known defhypo.mol2 -pgeom
              mult_esim hypo-names2 pqmusc.sfdb hmusc2given

# Contents of hypo-names2:
# tolterodine
# a49
# b1
# atropine
```

```
# qnb
# oxybutynin

# Look at some of the seeded multiple ligand alignments
> pym disp-hypos-seeded.pml

# KEY OUTPUT FILES:
#   hmusc2denovo-*.mol2  Alternative alignments using no prior knowledge
#   hmusc2given-*mol2    Alternative alignments using hmusc1-05 from above
```

Note that in many cases (as in the examples shown here), the overall multiple alignment probability score, which is used in sorting the output, will not differ much. The scores reflect a combination of ligand mutual similarity, volumetric concordance, and strain. In cases where there is no clear reason to prefer one of the several alternatives, as can occur, we recommend empirical testing of the highest scoring cliques using new ligands. In many cases, other data involving structure-activity variations will be useful in adjudicating between alternative mutual alignment hypotheses. There will be situations where multiple equally plausible alternatives exist, and when that occurs, the recommended approach is to make use of each alternative independently until it is possible to identify the best one using experimental data.

The `-me_kthresh` parameter controls the level at which any of the molecules will be dropped from a multiple alignment when the `-me_known` parameter is specified.

Fundamentally, the problem of multiple ligand alignment is more difficult than that of non-cognate ligand docking, which itself can be very challenging. We recommend considering the top several multiple alignments for any serious application, whether it be for virtual screening or for quantitative modeling.

4.5.3 BZR Example: Large-Scale Multiple Ligand Alignment

In the QuanSA module, facilities exist for fully automated large-scale multiple ligand alignment in the context of SAR data. Within the Similarity module, such large-scale alignments can be explored, and they may be utilized for downstream calculations such as re-scoring within a binding site or for guiding the building of a 3D-QSAR model with QuanSA or another technique.

Figure 4.7 shows two multiple ligand alignments of $GABA_AR$ ligands that are competitive at the benzodiazepine binding site. The left-hand alignment made use of the fastest screening-mode conformational search along with the fastest alignment settings. The right-hand alignment made using of deeper conformational search and slightly more thorough alignment. Conformational search depth and alignment search depth may be mixed, depending on speed and accuracy considerations. The alignments were generated as follows.

```
# Directory: examples/similarity/multiple_alignment/bzr

# First, we will prep the BZR hypo mols for exploring a mutual alignment: -PSCREEN
# Then, will generate a multiple alignment of 4 molecules

# Randomize the input conformers (which will also charge the mols)
# bzr3d.mol2 is a multi-mol2 of 147 bzr ligands
# Get the small set for the hypothesis generation
> sf-tools.exe regen3d bzr3d.mol2 bzrall
> sf-tools.exe mget bzrall-random.mol2 BZRNames bzrhypo3d.mol2
> sf-tools.exe -pscreen forcegen bzrhypo3d.mol2 ps-bzr
> sf-sim.exe -pscreen mult_esim BZRNames ps-bzr.sfdb mesim-ps

# Look at the alignments
> pym disp-mesim-ps.pml

# Now, we will prep the BZR hypo mols for a very thorough mutual alignment: -PQUANT
# Then, will generate a thorough (-PQUANT) multiple alignment of 4 molecules

> sf-tools.exe -pquant forcegen bzrhypo3d.mol2 pq-bzr
> sf-sim.exe -pquant mult_esim BZRNames pq-bzr.sfdb mesim-pq
```

```
# Look at the alignments
> pym disp-mesim-pq.pml

# The thorough mult_esim run produced a number of plausible alternatives.
# mesim-pq-02.mol2 copied to good-align.mol2.
# This alignment is very similar to that which seeded a very predictive
# BZR QuanSA model in the 2018 QuanSA paper. There are multiple such solutions.
> cp mesim-pq-02.mol2 good-align.mol2

# KEY OUTPUT FILES:
#   mesim-ps-*.mol2  Alignment with fast conformation and alignment search
#   mesim-pq-*.mol2  Alignment with thorough conformation and alignment search
```

Both alignments agree in terms of the correspondence of parts between the different scaffolds. However, the tightness of the alignments is improved by the additional sampling in the second case. Mutual alignments such as those shown in Figure 4.7 may be used to generate multiple ligand alignments of large numbers of molecules though to share a binding site, as follows.

```
# Directory: examples/similarity/multiple_alignment/bzr
# For maximal alignment accuracy, we will prep the full set of BZR
# ligands using the deepest level of conformer search
> sf-tools.exe -pquant forcegen bzrall-random.mol2 pq-bzrall

# We can run 147 molecules quickly to consider the full SAR.
# Align the full 147 molecule set and filter out
# a few outliers (controlled by -min_output) for visualization.
> sf-sim.exe -min_output 7.0 -nfinal 1 -pgeom esim_list pq-bzrall.sfdb good-align.mol2 quick

# Look at the alignments
```

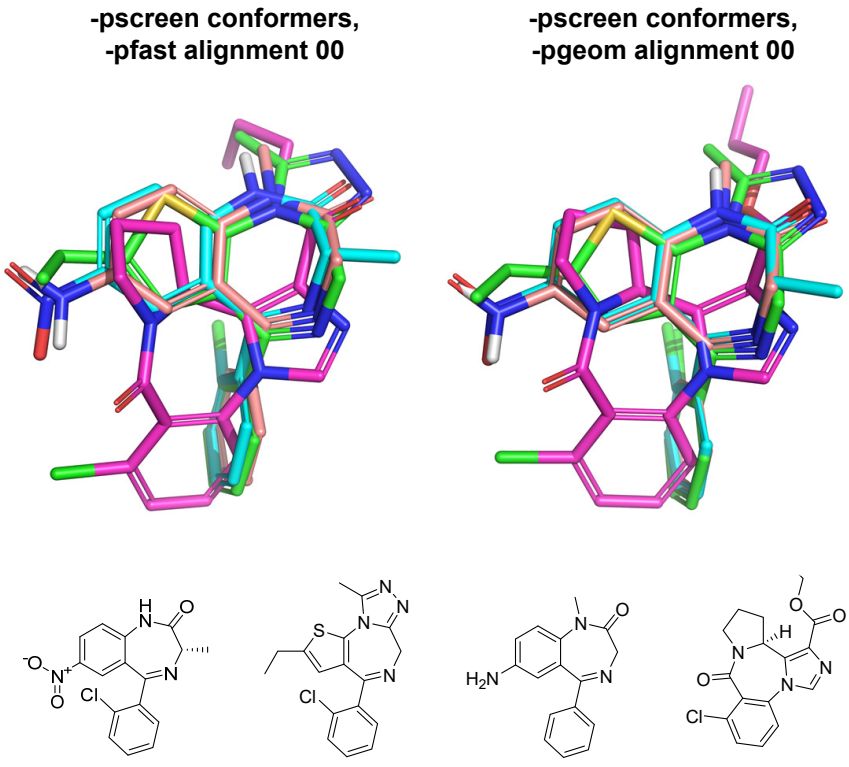


Figure 4.7 Two slightly different alternative hypotheses for the relative alignment of four BZR ligands.

```
> pym disp-mesim-all.pml

# KEY OUTPUT FILES:
#   quick-results.mol2  Top scoring alignment for each of 147 ligands
```

Figure 4.8 shows the results of the alignment procedure. At left, 137 of the total 147 ligands are shown. Note that the underlying scaffolds are *not* right on top of one another; rather, they shift depending on the pendant functionality present, considering each molecule on the basis of its overall surface and electrostatic characteristics. At right, 10 of the ligands with greater side-chain variability are shown. In these cases, the scaffold alignments shift somewhat more than at left, due to the size and diversity of substitutions, though none of these ligands display completely different predicted binding modes.

4.6 MOLECULAR POSITIONAL AND CONFORMATIONAL CONSTRAINTS: -POSCON AND -TORCON

As with Surflex-Dock, the Surflex-Sim module offers the ability to constrain either conformation or absolute alignment position of ligand subfragments. In the case of Surflex-Sim, because it is used frequently to identify relative ligand alignments, the conformational constraint is more often relevant. The syntax and behavior of these two options is the same as with Surflex-Dock. Torsional constraints must be applied during conformational elaboration with ForceGen, where the are embedded within the resulting SFDB (the strength of the constraints can be varied within Surflex-Sim). Positional constraints can be imposed within Surflex-Sim. The default behavior is for ligands without matching substructures to the specified positional constraint to be skipped, but this behavior can be suppressed by specifying `-skipnonmatch`.

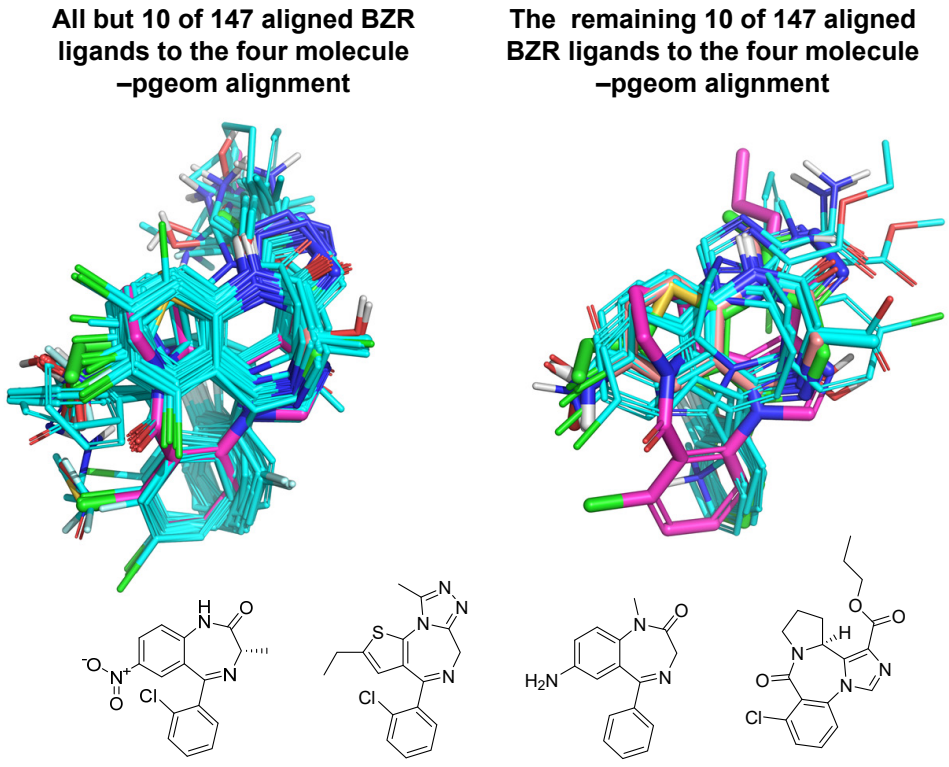


Figure 4.8 Complete alignments of 147 molecules to the right-hand alignment from Figure 4.7.

In cases where very flexible molecules are to be aligned, especially in large numbers and when a good deal of substituent diversity exists, use of torsional constraints is beneficial. Figure 4.9 shows such an example, generated as follows.

```
# Directory: examples/similarity/multiple_alignment/bzr
# Produce 3D from SMILES, -pquant conformers
# Then produce -pquant mult_esim alignment for the two truncated NK ligands
> sf-tools.exe fgen3d nk-scaff.smi nkscaff3d
> sf-tools.exe -pquant +racemize forcegen nkscaff3d.mol2 pq-nkscaff

> sf-sim.exe -pquant mult_esim ScaffNames pq-nkscaff.sfdb mesim-scaff

# The alignment mesim-scaff-02.mol2 is shown in the Figure.
# To make this into a torsional constraint file,
# hydrogen atoms are deleted along with some pendant functionality
# --> nkscaff-torcon-linkers.mol2

# Then, the conformer ensembles for four NK2 ligands (NKHypo.smi)
# are produced, using the torsional constraint from above.
> sf-tools.exe +reprot -torcon nkscaff-torcon-linkers.mol2 fgen3d NKHypo.smi nkhypotc
> sf-tools.exe -torcon nkscaff-torcon-linkers.mol2 -pgeom forcegen nkhypotc.mol2 pgnktc

# KEY FILES:
#   nk-scaff.smi                Two SMILES structures for NK ligand scaffolds
#   mesim-scaff-*.mol2          Alternative mutual alignments of the scaffolds
#   nkscaff-torcon-linkers.mol2 Linkers for the amide and ether scaffolds
#   NKHypo.smi                  Four SMILES structures for active NK ligands
```

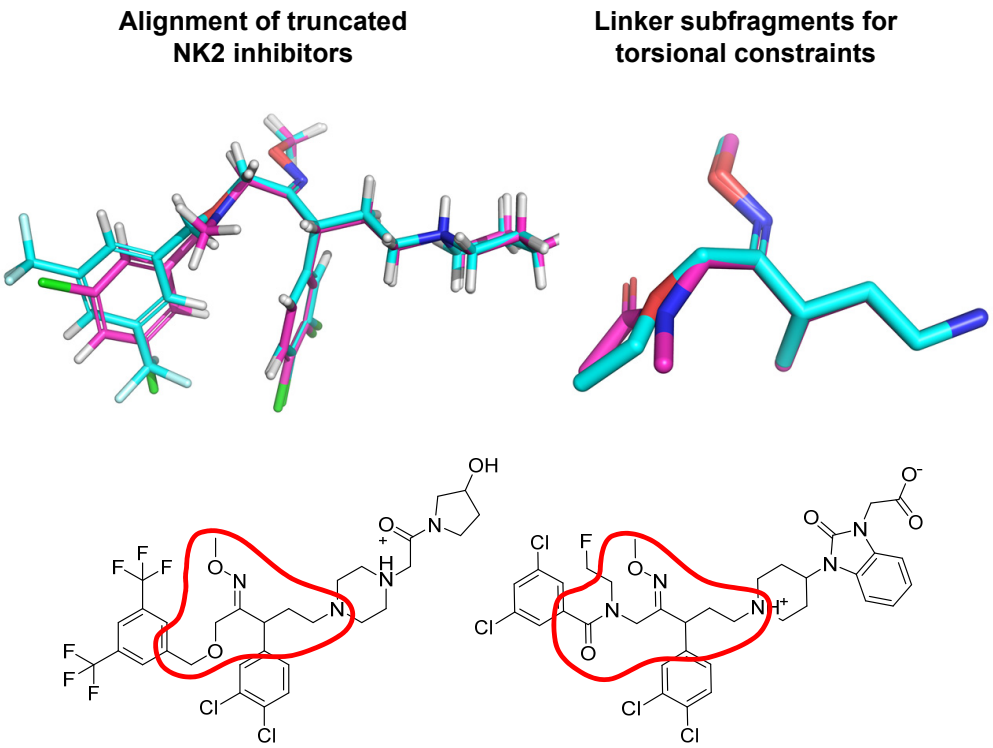


Figure 4.9 Two potent NK2 ligands are shown at bottom. The right-hand substituent of each was trimmed and made into a piperidine in order to produce a mutual alignment that addressed the question of how the ether and amide linkers might be related. At right, the linkers from the mutual alignment are shown and can be used directly as the argument of the `-torcon` switch within the SF-Tools/ForceGen command.

```
#   pgnktc.sfdb                 SFDB of the torsionally constrained NK ligands
```

Here, a maximally thorough multiple ligand alignment was carried out, using `-pquant` level search for both the conformations and alignments of the truncated NK2 ligands. The top left alignment shows edge-to-face intramolecular phenyl arrangements for both the ether and amide scaffolds along with tight correspondence of the right-hand substituent in common. The linkers were then made by manual deletion of extraneous atoms from the resulting alignment. The resulting mol2 file containing two molecular subfragments was used to conformationally constrain four, larger active NK2 ligands, resulting in a detailed exploration of the remaining functionality on their scaffolds.

A multiple ligand alignment was carried out using the torsional constraints. One of the resulting alignments was used as the target for a full series of 176 NK2 ligands, as follows.

```
# Directory: examples/similarity/multiple_alignment/bzr
# Now, we will generate the multiple alignment with the torsional constraint
> sf-sim.exe -pgeom mult_esim NKNames pgnktc.sfdb mesim-tc

# The previous step produces several good alignments, copied mesim-tc-00.mol2 to
    good-align.mol2
# Now, we will generate 3D for all of the NK2 mols with torsional constraints
> sf-tools.exe +reprot -torcon nkscaff-torcon-linkers.mol2 fgen3d Neurokinin2model.smi
    nkalltc
# time sf-tools.exe -torcon nkscaff-torcon-linkers.mol2 -pgeom forcegen nkalltc.mol2
    pgnkalltc

# The following full alignment takes about 10 seconds
> sf-sim.exe -nfinal 1 -pgeom +exclude esim_list pgnkalltc.sfdb good-align.mol2 quick

# KEY FILES:
#   mesim-tc-*.mol2    Constrained alignments for four NK2 actives
#   quick-results.mol2 One pose each for 176 NK2 ligands
```

Figure 4.10 shows the the multiple ligand alignment (left) and the superimposition of all but a few of the 176 molecules in the series (right). As with the BZR example, the NK2 scaffolds are seen to shift slightly in order to accommodate their substituents. In this case, nearly all of the activity variation among the inhibitors is caused by variations at the right-hand side, where a good deal of variability was explored with chemical variation.

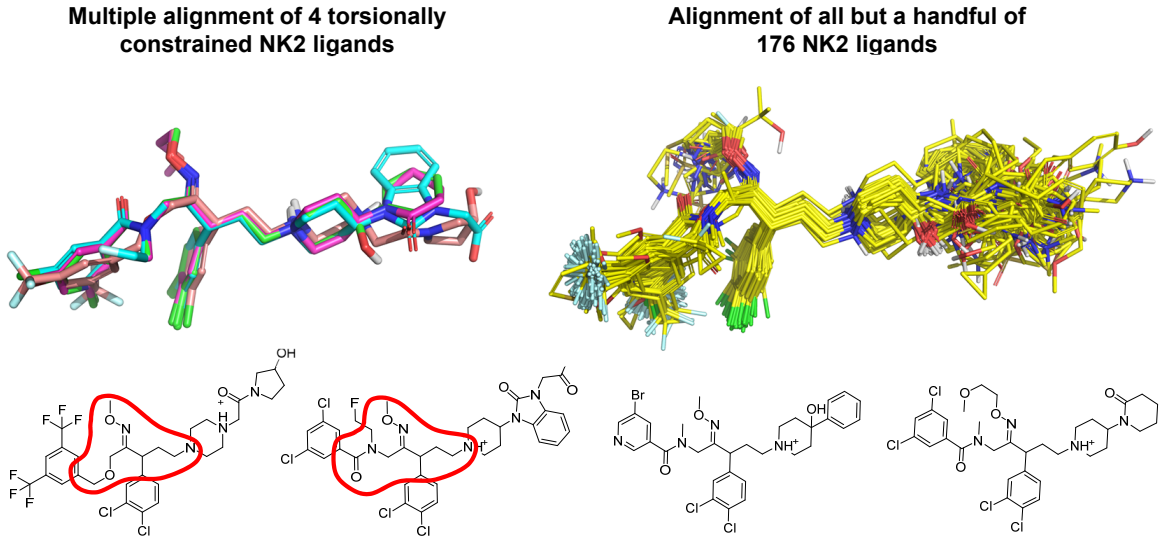


Figure 4.10 Complete alignments of 147 molecules to the `-pgeom` quality alignment from Figure 4.7.

4.7 OFF-TARGET PREDICTION: LOG-ODDS COMPUTATIONS

The prior set-based calculations of similarity in Version 4.411 are not currently available. With the replacement of morphological similarity by the eSim approach, a re-calibration of the probability normalization method is required. Replacement of the functionality with a much faster eSim-based approach will be done in the version 6.0 release. Users that require the logodds methods may use Version 4.411, though it is no longer supported.

4.8 SCREENING LARGE COMPOUND DATABASES

Similarity-based virtual screening of large databases has become a routine practice in computational chemistry workflows. Here, we show a quick example using a 1/100th subsampled Enamine Stock collection containing roughly 37,000 molecules. The `esim_list` command was used with both the `-win_prop` parameter (limits the number of hits to a proportion, here the top 1/1000) and the `-vrange` parameter (limit the hits to a volume range, here between 0.8 and 1.2 times the volume of the target molecule). It is recommended to use the fastest screening setting (`-pfastf`) for exploration of parameters on subsets of large databases. Screening enrichment will be very high with the `-pfastf` setting, and that can be used when time is a very significant factor. However, the `-pfast` or `-pscreen` settings will produce higher enrichment of actives.

```
# Directory: examples/similarity/virtual_screening/large_sfdb

# We make use of the PARP example here to show how to efficiently screen large databases
# We have downloaded a 1/100th subsample of the Enamine Stock Collection SFDB:
# https://optibrium.com/community-downloads/?active_filter=virtual-screening

# When screening large databases, one typically prefers
# to avoid storing final poses and scores for poor-scoring
# molecules. The Similarity module provides a means to
# automatically identify and set a threshold such that
# only the top-scoring hits are retained, as follows:

time sf-sim.exe -vrange 0.8 1.2 -nfinal 1 -win_prop 0.001 -pfastf esim_list
    Enamine-03-2023-Hundredth.sfdb xtal-lig.mol2 enam100fast

# NOTE: Here, we also specified a volume range
# for matching compounds of 0.8 to 1.2 in proportion to the target
# along with a single final pose.

# Screening as above, but with the -pscreen setting
# required about 3 minutes on a 36-core workstation:
time sf-sim.exe -vrange 0.8 1.2 -nfinal 1 -win_prop 0.001 -pscreen esim_list
    Enamine-03-2023-Hundredth.sfdb xtal-lig.mol2 enam100ps
```

In this example, using both the `-win_prop` and `-vrange` parameters, the fastest search mode `-pfastf` yielded 36 hits in about 0.5 minutes on a Dell XPS15 laptop, which means that the full Enamine stock collection could be screened in under an hour using `-pfastf`. The same search using the `-pscreen` search mode required about 3 minutes on a 36-core workstation. So, running on the more thorough `-pscreen` search mode would take just a few hours for the entire Enamine stock collection. Given the cost of obtaining, assaying, and QC'ing experimental results, the higher-quality results from the `-pscreen` approach will generally be worth the relatively minor additional computational cost.

4.9 REFERENCE SETS AND MOLECULAR IMPRINTS

With version 5.142, the use of molecular similarity to produce vector-based indexes of molecular structure (called molecular imprints) has been recalibrated using the new eSim methodology. Much faster and more effective vector-based methods are now available. The commands are illustrated with the PARP example seen earlier, as follows.

```
# Directory: examples/similarity/virtual_screening/parp1
# See RunImprint script

# Copy the provided 20-molecule basis set
cp ../../../../bin/data/basis20.mol2 .

# Conformer search in screening mode for decoys (actives done before)
sf-tools.exe -pscreen forcegen decoys.mol2 psdec ; cat psact.sfdb psdec.sfdb > psall.sfdb

# Calculate molecular imprints for the decoys and actives
sf-sim.exe -pscreen imprint psall.sfdb basis20.mol2 all-im
# Calculate molecular imprints for the multi-ligand target
sf-tools.exe -pscreen forcegen xtal-alt.mol2 ps-xtal-alt
sf-sim.exe -pscreen imprint ps-xtal-alt.sfdb basis20.mol2 xtal-alt-im

# Use iscreen_list to screen the active/decoy imprints against the target imprints
sf-sim.exe iscreen_list all-im xtal-alt-im imprint_screen

# We get just an OK ROC area (0.62) using the 20-molecule generic basis set
cat imprint_screen | grep act: | awk '{print $4}' > pos
cat imprint_screen | grep -v act: | awk '{print $4}' > neg
sf-tools.exe roc pos neg rocim20

# Use crystal-structure ligands to choose a PARP-specific basis set
# Prefer -pgeom prep and eSim for choosing reference sets
sf-tools.exe -pgeom forcegen xtal-all-ligs.mol2 pgxtal
sf-sim.exe -pgeom choose_ref xtal-all-ligs.mol2 pgxtal.sfdb imparp 10

# KEY OUTPUT FILES (from choose_ref command):
#   imparp-mols.mol2            The 10 chosen diverse PARP ligands

sf-sim.exe -pscreen imprint psall.sfdb imparp-mols.mol2 all-imparp
sf-sim.exe -pscreen imprint ps-xtal-alt.sfdb imparp-mols.mol2 xtal-alt-imparp
sf-sim.exe iscreen_list all-imparp xtal-alt-imparp imprint_screen_parp

# We get a much better ROC area (0.83) by using more directly relevant molecules
cat imprint_screen_parp | grep act: | awk '{print $4}' > pos
cat imprint_screen_parp | grep -v act: | awk '{print $4}' > neg
sf-tools.exe roc pos neg rocim10custom

# Copy the provided 200-molecule basis set
cp ../../../../bin/data/pdbbind200.mol2 .
sf-sim.exe -pscreen imprint psall.sfdb pdbbind200.mol2 all-im200
sf-sim.exe -pscreen imprint ps-xtal-alt.sfdb pdbbind200.mol2 xtal-alt-im200
sf-sim.exe iscreen_list all-im200 xtal-alt-im200 imprint_screen200
# We get an equally good ROC area (0.83) by using a larger agnostic set (200 PDBBind
    molecules)
cat imprint_screen200 | grep act: | awk '{print $4}' > pos
cat imprint_screen200 | grep -v act: | awk '{print $4}' > neg
sf-tools.exe roc pos neg rocim200
```

For additional discussion of the theory and application of molecular imprints, please refer to these studies [5, 7, 9, 12, 13].

4.10 MISCELLANEOUS SIMILARITY COMMANDS AND ASSOCIATED OPTIONS

The following covers Surflex-Sim commands and options not discussed in detail above (or in the previous chapter). Note that options that are discussed very thinly are not recommended for user experimentation.

Building hypotheses for multiple ligand alignments was covered in detail in the foregoing. Control of the process is governed by the listed parameters: `-me_nmake` controls the maximum total number of separate full alignment cliques to generate; `-me_rms` controls the RMS threshold on identity in grouping cliques; and `-me_known` offers a means to provide poses for a subset of the molecules in the desired clique (the closeness to which is governed by `-me_kthresh`).

The `-names` option offers a means to restrict molecules to be aligned to a subset within an SFDB; `-min_output` is a threshold, below which a molecule's results in a virtual screen will be suppressed; `-vrange` is a relative Van der Waals volume compared to the given query of subject molecules to consider, which can be useful in skipping large numbers of database ligands with very different size than desired; `-maxrot` sets a threshold above which database ligands will be skipped; and `-div_rms` is a threshold on the degree of difference between final poses.

The `diverse2d` allows selection of a diverse subset of molecules from the given SFDB or ligand pathname list. The `gsim` and `gsim_list` command use 2D graph-based similarity to provide analogous functionality to `esim_list` except using a 2D molecular similarity method.

Though not recommended, the weighting of the different aspects of the eSim similarity function can be altered using the `-epolar`, `-esteric`, `-ecoul`, `-edonacc` parameters. The weighting of the ligand strain term can be modified using `-estrain`. Also, imposition of a penalty for extending a subject ligand outside of a query is done using `+exclude`.

Bibliography

1. Ann E Cleves, Stephen R Johnson, and Ajay N Jain. Electrostatic-field and surface-shape similarity for virtual screening and pose prediction. *Journal of Computer-Aided Molecular Design*, 33(10):865–886, 2019.
2. Ann E Cleves and Ajay N Jain. Structure-and ligand-based virtual screening on DUD-E$^+$: Performance dependence on approximations to the binding pocket. *Journal of Chemical Information and Modeling*, 60(9): 4296–4310, 2020.
3. A. N. Jain. Morphological similarity: A 3D molecular similarity method correlated with protein-ligand recognition. *J Comput Aided Mol Des*, 14(2):199–213, 2000.
4. A. N. Jain. Ligand-based structural hypotheses for virtual screening. *J Med Chem*, 47(4):947–61, 2004.
5. A. E. Cleves and A. N. Jain. Robust ligand-based modeling of the biological targets of known drugs. *J Med Chem*, 49(10):2921–38, 2006.
6. A. E. Cleves and A. N. Jain. Effects of inductive bias on computational evaluations of ligand-based modeling and on drug discovery. *J Comput Aided Mol Des*, 22(3-4):147–59, 2008.
7. E. R. Yera, A. E. Cleves, and A. N. Jain. Chemical structural novelty: On-targets and off-targets. *J Med Chem*, 54(19):6771–85, 2011.
8. A.N. Jain and A.E. Cleves. Does your model weigh the same as a duck? *J Comput Aided Mol Des*, 26:57–67, 2012.
9. E. R. Yera, A. E. Cleves, and A. N. Jain. Prediction of off-target drug effects through data fusion. In *Pacific Symposium on Biocomputing*, volume 19, pages 160–171. World Scientific, 2014.
10. Ann E Cleves and Ajay N Jain. Chemical and protein structural basis for biological crosstalk between PPARa and COX enzymes. *Journal of Computer-Aided Molecular Design*, 29(2):101–112, 2015.
11. Ann E Cleves and Ajay N Jain. Quantitative surface field analysis: Learning causal models to predict ligand binding affinity and pose. *Journal of Computer-Aided Molecular Design*, 32(7):731–757, 2018.
12. J. Mount, J. Ruppert, W. Welch, and A. N. Jain. Icepick: A flexible surface-based system for molecular diversity. *J Med Chem*, 42(1):60–6, 1999.
13. A. M. Ghuloum, C. R. Sage, and A. N. Jain. Molecular hashkeys: A novel method for molecular characterization and its application for predicting important pharmaceutical properties of molecules. *J Med Chem*, 42(10):1739–48, 1999.

CHAPTER 5

REAL-SPACE LIGAND REFINEMENT (XGEN) MODULE TECHNICAL MANUAL

Real-space refinement of ligands within the electron density of protein-ligand complexes offers a means to quantitatively explore the conformational ensembles that give rise to the *snapshots* that are captured in diffraction experiments. The semantics of the xGen module mirror some aspects of the Tools module and the Docking module. In general, it is expected that users of xGen will have access to both of those modules, though many aspects of real-space refinement can be carried out within the xGen module independently.

NOTE: The current implementation of the xGen tool set requires a means to extract real-space electron density from diffraction data. In order to make the tools accessible to non-crystallographers, the current means to do this makes use of PyMOL, which is widely available to researchers. A future revision of xGen may include direct reading of MTZ files, which will obviate the need for either PyMOL or special-purpose crystallographic software to obtain real-space density. Note also that the Advanced Applications chapter shows how to use a prototype PyMOL GUI rather than the command-line interface described here.

It is strongly recommended that users read the extensive paper [1] that introduced the xGen method prior to reading this chapter or making use of the xGen tools. The prior work introducing ForceGen is also recommended [2, 3], as is the more recent work involving explicit modeling of bound ligand strain [4, 5].

5.1 SURFLEX-XGEN COMMAND LINE INTERFACE

Note that there may be minor variations between the figures shown in the manual and the precise results shown in the software distribution. There are no statistically significant differences, but, for example, the Nth ranked solution indicated in the manual may correspond more closely to the (N-1)st in the actual distribution. The variations are due to small algorithmic changes across minor version increments as well as cross-platform and compiler differences.

This is the command-line help listing of Surflex-xGen:

```
BioPharmics Platform Version 5.197
```

```
Usage: surflex-xgen <options> <command> args

  [xGEN PARAMETER SELECTION CHOICES]
      -pstrict     Strict xGen ensembles [DEFAULT]
      -pdiverse    Diverse xGen ensembles
      -pvstrict    Very strict xGen ensembles
      -pvdiverse   Very diverse xGen ensembles

  [LIGAND SEARCH DEPTH SELECTION CHOICES]
      -pgeom       Deeper ring search + macrocycles, max 250
      -pscreen     Screening parameter set, variable max 50/120
      -pfast       Screening parameter set, max 50
      -pquant      Deep ring+conformer search including macrocycles, max 1000 [DEFAULT]

  [XRAY DENSITY COMMANDS]
    dbox           raw-dbox.pdb scope-lig.mol2 X-ray-res-file outprefix
      -xg_dcarve     (5.00)   Carve from scope-lig
    build          scope-lig.mol2 protein.mol2 density-prefix outprefix
     +-xg_denovo             Pure de novo Gaussian sphere generation [default: OFF]
     +-xg_scope              Use ligand to scope the binding site [default: ON]
      -xg_carve    (0.50)    Carve density further than this from scope-lig
      -xg_res      (---)     X-ray res.: [default: read from <density-prefix>-resolution]
     +-xg_contour            Turn on/off generation of isosurface contours (default: OFF)
    refine         xgen_density approx_lig.mol2 outprefix
    fit            xgen_density inmol.sfdb outprefix
      -xg_win      (40.0)    Energy window for output conformers
      -xg_rms      (0.25)    Final RMSD difference for output conformers
      -xg_nfinal   (100)     Number of final conformers
    regen          xgen_density_constrained approx_lig.mol2 outprefix
    expand         xgen_density refined.sfdb approx_lig.mol2 outprefix
      -xg_pen       (1.0)    Force to remain close to refined conf
      -xg_wiggle    (0.2)    Wiggle for -xg_pen
      -xg_en        (3.00)   Energy above refined pool min for final pools
      -xg_pct       (0.100)  Percentage of best density fit for final pools
      -xg_srms      (0.650)  sRMSD threshold to use for neighborhood building
    ensemble       xgen_density density-prefix protein.mol2 scope.mol2 pool.mol2 outpfx
      -xg_rsr_win   (0.500)  Single-conf RSR window
      -xg_nopt      (100)    Max number of confs for ensemble optimization
      -xg_clip      (0.10)   Occupancy clipping threshold
      -xg_pcrash    (-0.20)  Threshold of protein interpenetration to allow
     +-xg_contour            Turn on/off generation of isosurface contours (default: OFF)
    eval  occ-list ensemble.mol2 protein.mol2 scope.mol2 prtn.pdb density-pfx outpfx
     +-xg_contour            Turn on/off generation of isosurface contours (default: OFF)

  [MISCELLANEOUS COMMANDS]
    extract_sfdb   input.sfdb outprefix
    rms_conflist   inmol2archive.mol2 goldmol2archive.mol2 out-prefix
    get            mol2archive molname outmolname
    mget           mol2archive molnamelist outmolarchive
    mgetnum        mol2archive molnumberlist outmolarchive
    mergemols      mol2archive outprefix [merges all mols in archive into a single mol]
    rms            mol1 mol2
```

All commands should be typed lower-case. The xGen module is intended for use as part of the overall integrated BioPharmics Platform, particularly with the Tools and Docking modules. However, ligand refinement beginning from a pre-existing approximate placement can be performed within the xGen standalone module.

5.2 PRIMARY CHANGES IN CURRENT VERSION

General notes about the current version can be found in the Release Notes in the Foreword to this manual. Detailed notes can be found here.

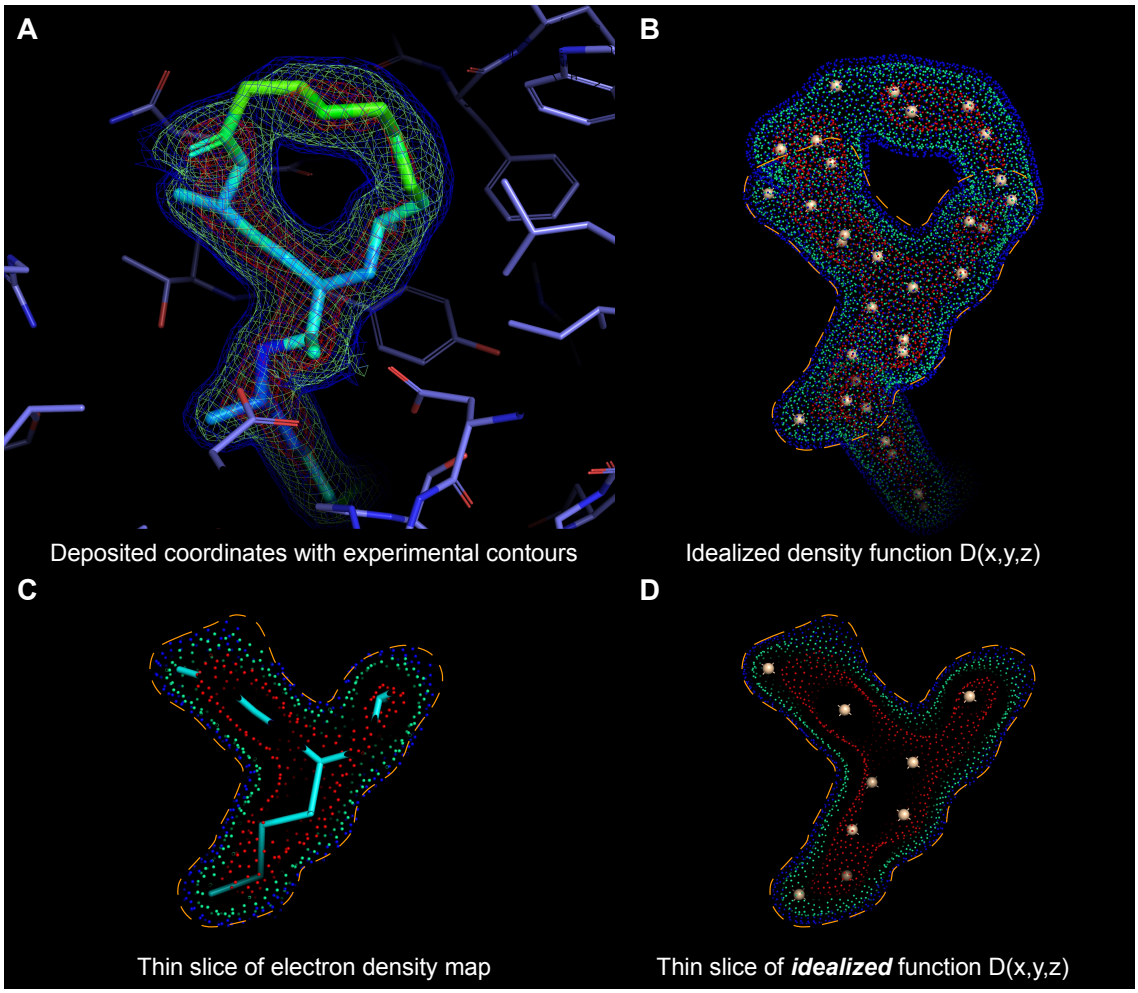


Figure 5.1 Transformation of crystallographic experimental density into a spherical Gaussian approximation. (A) 3DV1 protein, with bound ligand colored by B-factors, showing three experimental electron density contours from the $2|F_o| - |F_c|$ map (shown at 2.0σ (red), 1.5σ (green) and 1.0σ (blue)). (B) The derived idealized density function $D(x, y, z)$ has roughly the same number of Gaussian centers as ligand atoms (small tan spheres), and the parameters of the function are set to produce density values close to experimental (isosurface contours are shown for $D(x, y, z)$). (C) A thin section in the X-Y plane of the ligand coordinates and the three experimental density contours. (D) The corresponding thin X-Y section of the idealized function $D(x, y, z)$, the positions of the spherical Gaussian centers, and the isosurface contour points for the idealized function.

5.3 BRIEF SUMMARY OF REFINEMENT STEPS

The algorithms are detailed in the above-referenced paper. The steps for refinement will be briefly summarized here:

1. Idealized density: Experimental density is converted into a spherical Gaussian approximation for the region of space that includes the reference ligand's Van der Waals volume plus a buffer zone of 0.5Å.

2. Restrained conformational search: The normal ForceGen conformational search procedure is carried out beginning with the ligand's reference coordinates. The MMFF94sf force field is augmented with a reward for overlap between the ligand's density function $L(x, y, z)$ and the idealized experimental density $D(x, y, z)$ (see Figures 5.1 and 5.2).

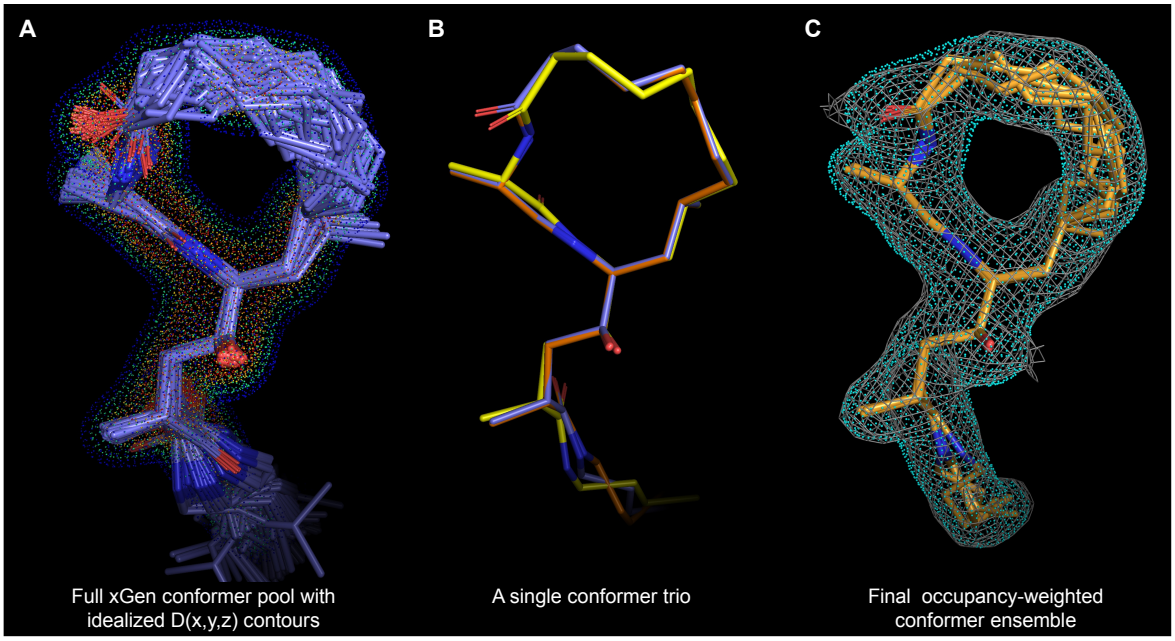


Figure 5.2 Conformational search and ensemble derivation. (A) All conformers resulting from a restrained search of the 3DV1 ligand blending MMFF94sf energetics with a spherical overlap integral reward for matching the idealized density. (B) A single high-quality conformer trio, representing both good density fit (orange) and low energy (yellow) along with a bridging central conformer (slate). (C) An occupancy-weighted conformer ensemble with the 1.0σ experimental density contour (gray mesh) and corresponding calculated real-space density (ρ_{calc}) contour (cyan dots).

3. High-quality trio generation: Each conformer resulting from the search is re-minimized: a) under a condition in which the density overlap is strongly weighted; and b) with no density overlap reward but with a square-welled quadratic positional restraint. The three pools of conformers (high density weighting, blended weighting from the thorough conformer search, and minimized under positional constraint) are used to find high-quality trios that are characterized by high congruence to electron density and by low energy.

4. Ensemble generation: Conformers from the trios are used to construct occupancy-weighted ensembles that minimize real-space R. Calculations are done using $L(x, y, z)$ for real-space electron density (the fast approximation to ρ_{calc}) and using the full experimental density sampled on a 0.25Å grid.

5. Fit evaluation: Final statistics for real-space correlation coefficient and real-space R (RSCC and RSR, see Experimental Section) are made by comparing the 0.25Å grid-sampled experimental density to the density derived from the ligand (or ligand ensemble). For PDB reference ligand coordinates, this is done using exact, as-deposited, atom-specific B-factors, the resolution of the diffraction data, and the standard truncated Fourier approach with fitted scattering factor functions for ρ_{calc}. For xGen ensembles, the same procedure is followed, except that a constant, grossly estimated, B-factor is used for all atoms of all conformers within an ensemble (B-factor optimization does not enter into any aspect of deriving an xGen ensemble).

The entire procedure is fully automatic and required just a few minutes per example, even for macrocycles, beginning with extracting electron density information from the PDB structure and reflection data files and ending with final conformer ensemble evaluation.

5.4 REAL-SPACE LIGAND REFINEMENT

5.4.1 Extracting the Real-Space X-Ray Density

The steps for ligand refinement begin with building a density approximation from real-space density that has been extracted from a crystallographic MTZ map file. The current method (soon to be replaced) for extracting density makes use of PyMol, as follows:

```
# Directory: examples/xgen/refine/

# PyMol Script File: BuildAllMesh.pml
    load protein.mol2
    load lig-orig.mol2
    remove (hydro)
    load pdbmap.mtz
    map_double phases.2fofc, -1
    select ligand, lig-orig
    isomesh map, phases.2fofc, 5.0, ligand, carve=5.0
    dump mesh50.txt, map
    isomesh map, phases.2fofc, 4.9, ligand, carve=5.0
    dump mesh49.txt, map
    ...
    isomesh map, phases.2fofc, -0.1, ligand, carve=5.0
    dump mesh-01.txt, map

    quit

# Directory: examples/xgen/refine/test3dv1

> pymol ../BuildAllMesh.pml

# KEY OUTPUT FILES:
#   mesh*.txt        Set of mesh contour files from 5.0 to -5.0 sigma
#                    in steps of 0.1
```

This procedure takes just seconds on typical desktop workstations and takes slightly longer on a laptop. The resulting very fine-grained sets of 3D points at the different contour levels are used to construct a full 0.25Å grid of real-space density, as follows:

```
# Directory: examples/xgen/refine/test3dv1/
# See examples/xgen/refine/RunXgen-Build

> source ../MakeDensity
> gunzip -f raw_density_box.pdb.gz
> sf-xgen.exe dbox raw_density_box.pdb lig-orig.mol2 pdb-res density
> gzip -f raw_density_box.pdb

# KEY OUTPUT FILES:
#   density-resolution        The specified resolution of the X-ray data
#   density-clist             The file spec for input to xgen_build
#   density-mesh-[012345]     Data for building the idealized density
#   density_dpts.mol2         Visualizable density grid (color by partial charge)
#   density_contour.mol2      Visualizable density contours (color by partial charge)
```

A number of files are produced, all prefixed as specified (here “density”). These are used in all subsequent refinement steps.

5.4.2 Building the Real-Space Idealized Density Approximation

Part of the computational complexity of prior real-space refinement methods stemmed from using a grid-based representation of the X-ray density. The xGen approach is to construct an idealized approximation, composed of spherical Gaussian functions, as follows:

```
# Directory: examples/xgen/refine/test3dv1

# The build command builds the Gaussian spherical density approximation
> sf-xgen.exe build lig-orig.mol2 protein.mol2 density xg

# KEY OUTPUT FILES:
#   xg-ligfit       Spherical Gaussian specification
#   xg-constraint   Spherical Gaussian specification + positional constraints
#   xg-exptl*.mol2  Experimental density contour dots (color by partial charge)
#   xg-fit*.mol2    Idealized density contour dots (color by partial charge)
```

The contour files may be visualized in order to see the concordance of the experimental and idealized density functions. These contours are also useful in visualizing the output ligand ensembles from xGen ligand refinement and fitting procedures.

5.4.3 Ligand Refinement

Refinement of an approximate ligand pose combines the standard ForceGen conformer search procedure with the restraints that were constructed in the foregoing. The xGen module implements the algorithm, along with the conformer expansion described above, as follows:

```
# Directory: examples/xgen/refine/test3dv1

# Use the density constraint from xgen_build to produce better xtal ligand possibilities
> sf-xgen.exe -pquant refine xg-ligfit lig.mol2 xgref

# Expand the initial balanced conformer pool, then ensembles can be built
> sf-xgen.exe expand xg-ligfit xgref.sfdb lig-orig.mol2 xgref-prof

# KEY OUTPUT FILES:
#   xgref.sfdb                    The full conformer pool from refinement
#   xgref-prof-best-density.mol2  Density-weighted conformers from high-quality trios
#   xgref-prof-best-center.mol2   Balanced conformers from high-quality trios
#   xgref-prof-best-min.mol2      Constrained minimized conformers from high-quality trios
#   xgref-prof-confreport         Detailed per-conformer report
#   xgref-prof-log                Log file from conformer expansion
```

The `refine` command produces an SFDB conformer database file, which is then used as input to the `expand` command. The `expand` command is related to the `profile` command from the Tools module, and the conformer report file contains the following columns:

1. *Cnum*: The number of the conformer in the output sfdb (sorted based on blended energy)
2. *Energy*: MMFF94sf energy (without density overlap reward)
3. *E+EViol*: MMFF94sf energy *plus* density overlap reward
4. *EViol*: Total density overlap reward
5. *BlendScore*: Normalized value of the density overlap integral (maximal values will be close to 1.0).
6. *HighRMS*: Density-weighted conformer RMSD from input `lig-orig.mol2`
7. *HighScore*: Overlap score for density-weighted conformer
8. *CminRMS*: RMSD from original ligand for the constrained minimized conformer
9. *CminScore*: Overlap score for the constrained minimized conformer
10. *CminEn*: MMFF94sf energy for the constrained minimized conformer
11. *NeighRMS*: Scaled RMSD for the neighborhood that the trio for this row represents

Consideration of the conformer report is not generally necessary for a typical case. However, users that want to visualize each of the expanded conformer trios sequentially may wish to see the detailed scores and deviations.

Options for the `expand` command affect the degree to which the high-quality conformer trios that are produce have particularly high density overlap scores or particularly low energy. The parameters are as follows:

1. `-xg_pen`: In the constrained minimization, this is the force to remain close to the blended conformer (in kcal/mol/Angstrom^2)
2. `-xg_wiggle`: In the constrained minimization, this is the amount of free wiggle to allow each atom (Angstroms)
3. `-xg_en`: Initial energy window for high-quality trio generation, which will be increased if no trios are found
4. `-xg_pct`: Percentage below the maximum density overlap for high-quality trios. Increasing this value will allow for less perfect fits to X-ray density to become part of the output high-quality trios.
5. `-xg_srms`: Threshold on the neighborhood size for the output trios. The default value of 0.65Å has been found to keep the conformers within the trios close enough to one-another that the set can be treated as part of a closely-related manifold of poses.

Extensive testing has not been done with these parameters.

5.4.4 Generating Conformer Ensembles and Evaluating Them

The central goal of the xGen refinement procedure is to replace a single conformer with optimized atom-specific B-factors (the input approximate ligand pose) with a conformer *ensemble* that makes use of a constant B-factor for the entire ensemble. The final step takes a pool of conformers and identifies the ensemble that minimizes a fast calculation of real-space R:

```
# Directory: examples/xgen/refine/test3dv1

> sf-xgen.exe +xg_contour -pstrict
              ensemble xg-ligfit density protein.mol2 lig-orig.mol2
                       xgref-prof-best-density.mol2 xgensemble

# Required command-line parameters:
#   density restraints: xg-ligfit
#   density prefix:     density
#   protein:            protein.mol2
#   scope-lig:          lig-orig.mol2
#   input pool:         xgref-prof-best-density.mol2
#   output prefix:      xgensemble

# KEY OUTPUT FILES:
#   xgensemble-confs.mol2      xGen strict ensemble
#   xgensemble-occ             Occupancies for the conformers (one per line)
#   xgensemble-iso10.mol2      Equivalent 1.0 sigma contour (if +xg_contour was specified)
#   xgensemble-iso01.mol2      Equivalent 0.1 sigma contour (if +xg_contour was specified)
```

The density prefix is required in order to access the original real-space density grid values, which are used in ensemble optimization. The protein structure is used to subtract the density attributable to protein atoms and to identify conformers within the input pool that clash with protein atoms.

The scope ligand is used to identify the spatial volume of density to be considered during the ensemble generation process. This may be, for example (as above), the original approximate ligand pose. However, it may also be a multi-mol2 file that identifies a (typically larger) volume. For example, one can specify the same argument as the scope ligand and input pool, which will have the effect of the xGen pool generation and expansion self-defining the scope of the final ensemble. Overall parameter set choices that affect ensemble generation are as follows:

- `-pstrict`: Selects a strict set of parameters for ensemble generation.
- `-pdiverse`: Selects parameters for more diverse ensemble generation.
- `-pvstrict`: Selects very strict parameters.
- `-pvdiverse`: Selects parameters for very diverse ensemble generation.

Specific individual parameters:

- `-xg_rsr_win`: Proportion above the single lowest-RSR conformer from the input pool to allow ensemble members.
- `-xg_nopt`: Maximum number of conformers to begin ensemble optimization (reduction made through progressive RMSD compression).
- `-xg_clip`: Minimum fraction of occupancy that the least-occupied confomer may have relative to the most occupied.
- `-xg_scope`: Distance in Angstroms beyond the multi-mol2 scope ligand to carve the real-space density for ensemble optimization.
- `-xg_pcrash`: Amount of ligand-protein heavy atom surface interpenetration beyond which a conformer will be dropped (more negative indicates more penetration allowed).

Evaluation of an ensemble for real-space correlation-coefficient and real-space R is done as follows:

```
# Directory: examples/xgen/refine/test3dv1
# Evaluation for RSCC and RSR

> cat xgensemble-confs.mol2 lig-orig.mol2 > scope-lig.mol2
> sf-xgen.exe +xg_contour
              eval xgensemble-occ xgensemble-confs.mol2 protein.mol2 scope-lig.mol2
                   protein.pdb density xgeval

# Required command-line parameters:
#   occupancies:        xgensemble-occ
#   ensemble:           xgensemble-confs.mol2
#   protein:            protein.mol2
#   PDB protein:        protein.pdb
#   density prefix:     density
#   output prefix:      xgeval

# Evaluate the original ligand coordinates using the same density scope:
> echo 1.0 > occ-single
> sf-xgen.exe +xg_contour
              eval occ-single lig-orig.mol2 protein.mol2 scope-lig.mol2
                   protein.pdb density pdbeval

# KEY OUTPUT FILES:
#   xgeval-log                Log file with RSCC and RSR
#   xgeval-density-diff.mol2  Difference dots from RSR calculation
#   xgeval-iso10.mol2         Equivalent 1.0 sigma contour (if +xg_contour was specified)
#   xgeval-iso01.mol2         Equivalent 0.1 sigma contour (if +xg_contour was specified)
#   pdbeval-*                 [Analogous files for pdbeval]
```

In many ways, the statistical evaluation is the most complex of the xGen procedures, as it must make use of atom-specific and resolution-specific truncated Fourier expansions of atomic scattering factor functions in order to accurately calculate real-space density for xGen-style ensembles and traditional PDB ligand representations. For the xGen ensembles, an approximate B-factor is identified during the evaluation process (from 10–100 in steps of 5). For specified ligand ensembles whose atomic coordinates exactly match ATOM and HETATM records within the given PDB file, those B-factors will be used.

RSCC, for a number of reasons, is more reliable than RSR in meaningfully assessing xGen ensembles compared with traditionally optimized ligand models: 1) RSCC is not sensitive to electron density scaling; 2) RSR is often overfit in a highly atom-centric manner (e.g. a single aryl halogen atom often has a very high B-factor when the arene does not); and 3) neither xGen nor traditional refinement protocols directly optimize RSCC, but both (to some extent) optimize functions related to RSR.

The PyMol .pml files in `examples/xgen/refine/test3dv1` will load either the strict or diverse ensembles and the most relevant associated files. Figure 5.3 shows such a PyMol session. The loaded files are as follows:

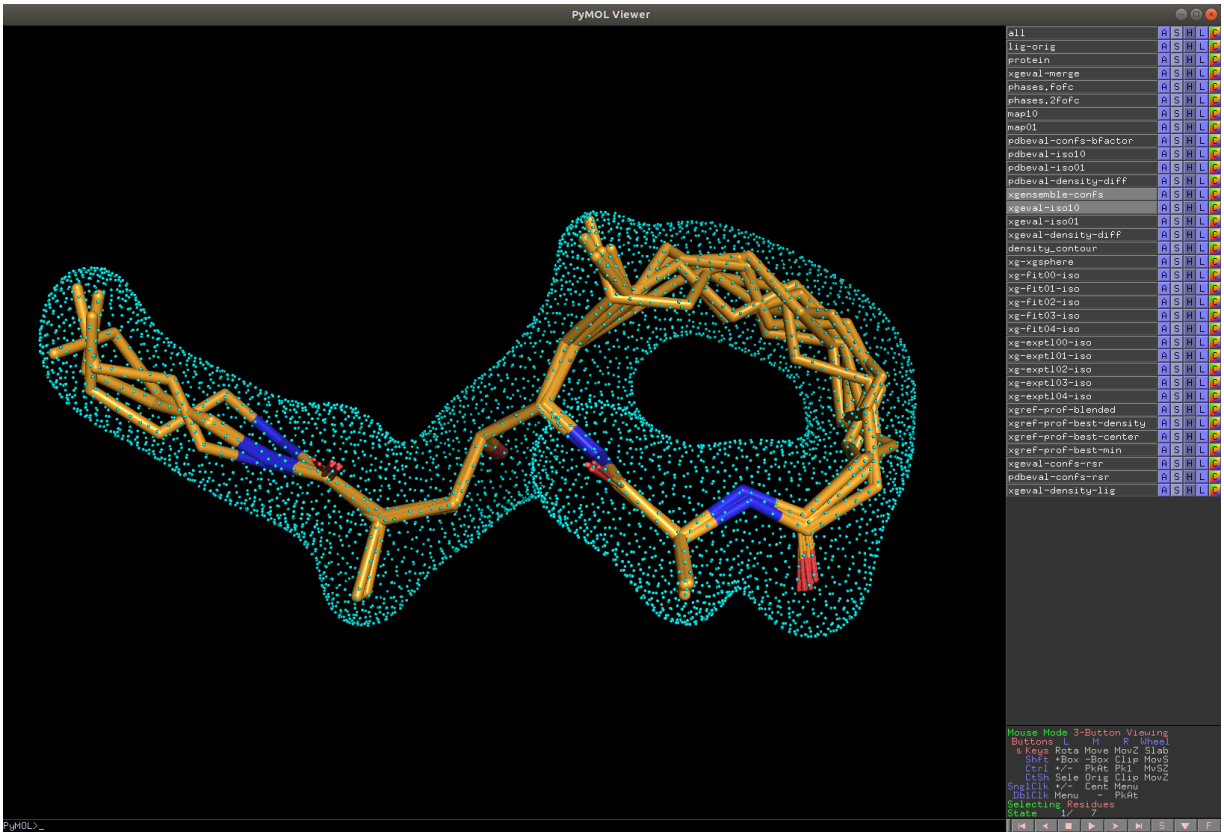


Figure 5.3 PyMol session with results from strict ensemble generation for the ligand of 3DV1. The two visible items are the conformer ensemble (`xgensemble-confs`) and the corresponding 1.0σ contour (`xgeval-iso10`).

```
lig-orig.mol2                # Original ligand coordinates
protein.mol2                 # Protein with no ligand
xgeval-merge.mol2            # Scope ligand used for evaluation
phases.mtz                   # Map coefficients for 3DV1 (PDB download)

pdbeval-confs-bfactor.mol2   # PDB ligand colored by B-factor
pdbeval-iso10.mol2           # PDB ligand 1.0 sigma contour
pdbeval-iso01.mol2           #            0.1
pdbeval-density-diff.mol2    # Difference dots between observed/calculated

xgensemble-confs.mol2        # xGen ensemble
xgeval-iso10.mol2            # xGen ensemble 1.0 sigma contour
xgeval-iso01.mol2            #               0.1
xgeval-density-diff.mol2     # Difference dots between observed/calculated

density_contour.mol2         # Full extracted density shown as contours
xg-xgsphere.mol2             # xGen Gaussian sphere centers
xg-fit00-iso.mol2            # xGen ligand-density contours
xg-fit01-iso.mol2
xg-fit02-iso.mol2
xg-fit03-iso.mol2
xg-fit04-iso.mol2
xg-exptl00-iso.mol2          # Corresponding experimental contours
xg-exptl01-iso.mol2
xg-exptl02-iso.mol2
xg-exptl03-iso.mol2
xg-exptl04-iso.mol2
```

```
xgref-prof-blended.mol2       # All blended conformers
xgref-prof-best-density.mol2  # High-quality density-weighted conformers
xgref-prof-best-center.mol2   #              blended
xgref-prof-best-min.mol2      #              minimized (constrained)

xgeval-confs-rsr.mol2         # xGen ensemble, colored by attributed RSR
pdbeval-confs-rsr.mol2        # PDB ligand, colored by attributed RSR

xgeval-density-lig.mol2       # Real-space density on 0.25 Angstrom grid,
                              # near the ligand for evaluation of density fit
```

Many of these visual depictions are not standard in crystallography workflows, but they have proven useful in understanding why a particular conformer or ensemble is either nominally better or worse than another.

5.5 FITTING A LIGAND *DE NOVO*

The xGen module is also capable of fully *de novo* fitting of a ligand into a large volume of X-ray density. The process is similar to the refinement process, with three exceptions: 1) the initial density approximation is very approximately scoped for a binding site; 2) derivation of the density approximation makes no use of pre-identified atomic positions; and 3) the xGen command `fit` is used to fit an agnostically pre-searched conformer pool into X-ray density. The following example demonstrates the initial rough fitting process:

```
# Directory: examples/xgen/denovo/1exx
# Fully automatic density fitting

# We are using the Docking module's multicav method to identify the
# binding site (see site-lig.mol2, used in ../BuildSiteMesh.pml)

# Randomize the input xtal conformer and -pquant forcegen
> sf-tools.exe regen3d ligand.mol2 lig
> sf-tools.exe -pquant forcegen lig-random.mol2 pqlig

# Build the density from the density_box.pdb that came from PyMol meshes
> sf-xgen.exe dbox raw_density_box.pdb site-lig.mol2 pdb-res density

# Make constraints without the ligand *at all*
> sf-xgen.exe -xg_scope +xg_denovo build NONE protein.mol2 density xgden

# Align randomized ligand conformer SFDB to the xgden target and make ensemble.
# NOTE: expand uses the "ligand" only to produce RMSD
# information, so we can use the randomized one
> sf-xgen.exe fit xgden-ligfit pqlig.sfdb xgfitblob
> sf-xgen.exe expand xgden-ligfit xgfitblob-results.mol2 lig-random.mol2 xgfitblob-prof

> sf-xgen.exe -pvdiverse ensemble
                         xgden-ligfit density protein.mol2 xgfitblob-prof-best-density.mol2
                         xgfitblob-prof-best-density.mol2 xgautoblobensemble

# KEY OUTPUT FILES:
#   xgfitblob-results.mol2        Raw fit into the initial density blob
#   xgautoblobensemble-confs.mol2 Ensemble that will serve as new density scope
```

At the end of this process, we have a ligand ensemble that was derived from the large initial density volume. In many cases, this ensemble may be adequate for downstream modeling purposes. However, it may also be used to define a more local volumetric scope for further refinement, as follows.

```
# Directory: examples/xgen/denovo/1exx
# Re-build the density using the automatically derived blob ensemble

> sf-xgen.exe +xg_contour +xg_denovo build xgautoblobensemble-confs.mol2
                                         protein.mol2 density xgauto
```

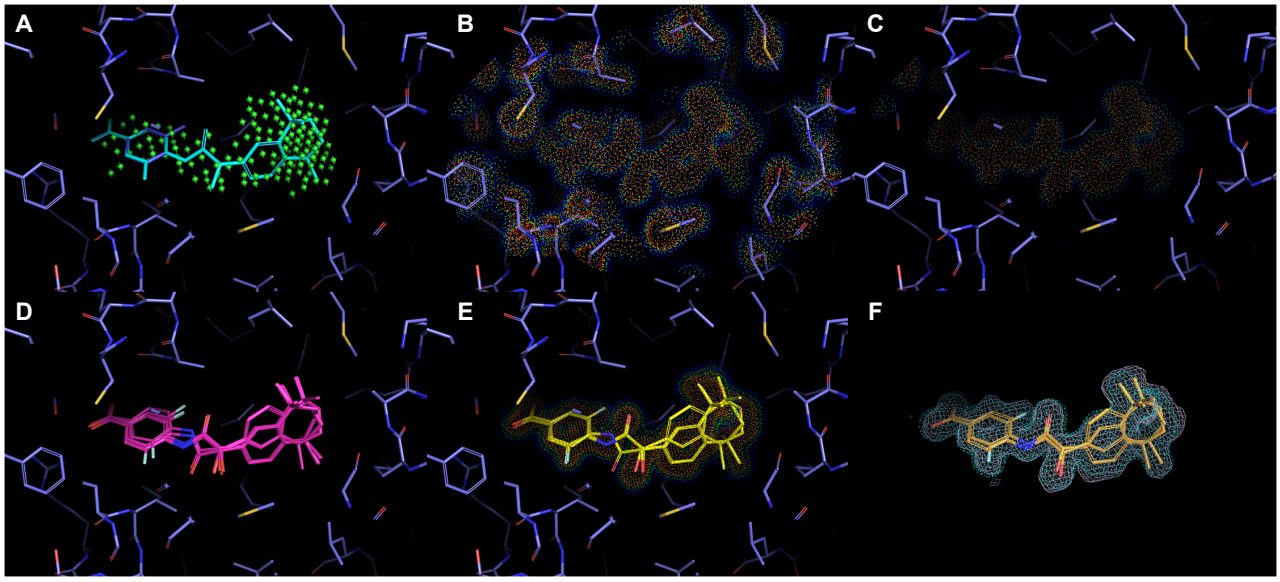


Figure 5.4 Fitting a ligand into a volume of density with no prior ligand placement (PDB: 1EXX): (A) Green spheres show the multi-cav protomol produced by Surflex-Dock (PDB ligand and alternate shown for reference only); (B) Experimental density contour points extracted from the full MTZ grid; (C) Experimental density contour points with protein atoms subtracted; (D) Initial rough fit into the *de novo* density approximation; (E) Ensemble from which the final density approximation is built and ligand conformers are produced for the final ensemble; (F) The final xGen ensemble.

```
# Align randomized ligand to the xg AUTO target
> sf-xgen.exe fit xgauto-ligfit pqlig.sfdb xgautofit

# Now let's make an ensemble from the xgautofit results using the refined density
# This will be used as the seed for a final refinement

> sf-xgen.exe expand xgauto-ligfit xgautofit-results.mol2 lig-random.mol2 xgautofit-prof
> sf-xgen.exe -pvdiverse ensemble
                        xgauto-ligfit density protein.mol2 xgautoblobensemble-confs.mol2
                        xgautofit-prof-best-density.mol2 xgautofitensemble

# KEY OUTPUT FILES:
#   xgauto-ligfit                 Idealized density, smaller scope than initial blob
#   xgautofitensemble-confs.mol2  Ensemble to serve as input for a final refinement
```

Here, we have created xGen idealized density from a more well-defined X-ray density scope, and we have a new ensemble. Again, as with the previous step, the ensemble here may be sufficient for downstream use. However, a final full refinement will produce a more deeply searched and very likely lower-energy ensemble.

Now, we have a well-defined scope, and we have some (possibly very good) conformers in terms of density fit. A final full refinement will produce a low-energy ensemble (see below). Figure 5.4 shows the entire process for PDB code 1EXX (human retinoic acid receptor).

```
# Directory: examples/xgen/denovo/1exx
# Now let's thoroughly refine the ensemble we got
> sf-xgen.exe -pquant refine xgauto-ligfit xgautofitensemble-confs.mol2 xgautoref
> sf-xgen.exe expand xgauto-ligfit xgautoref.sfdb lig-random.mol2 xgautoref-prof

# Now let's make an ensemble from the refine results
> sf-xgen.exe -pdiverse ensemble xgauto-ligfit density protein.mol2
                                 xgautofitensemble-confs.mol2
                                     xgautoref-prof-best-density.mol2 xgautoensemble
```

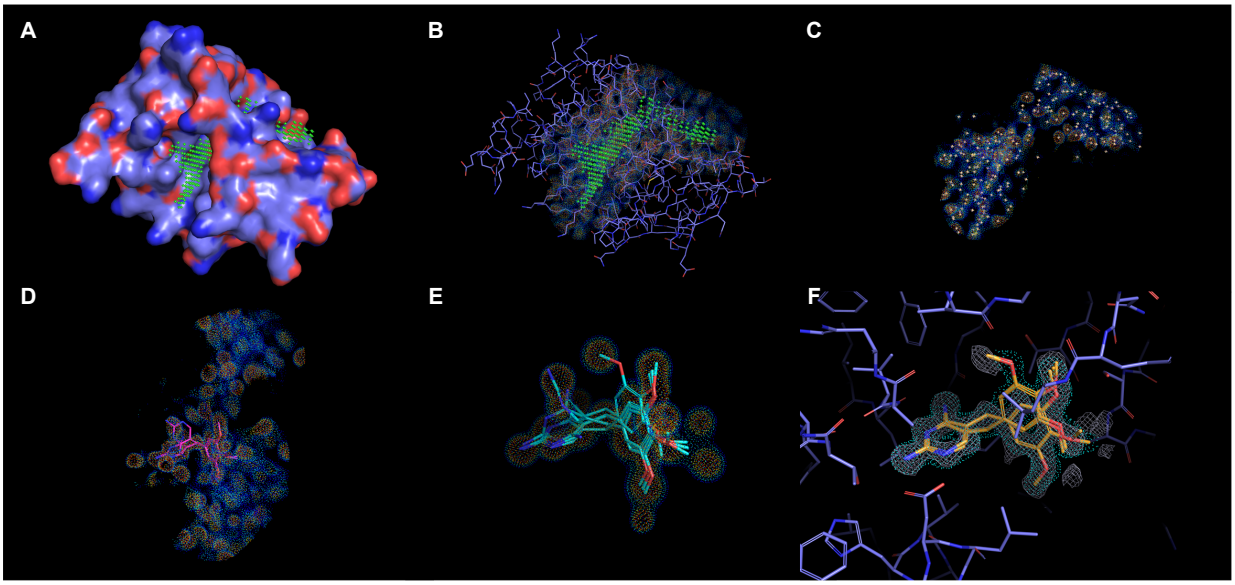


Figure 5.5 Fitting a ligand into a *very* large volume of density with no prior ligand placement (PDB 2W9H): (A) Green spheres show the multi-cav protomol produced by Surflex-Dock (full surface of Staph. aureus shown); (B) Experimental density contour points extracted from the full MTZ grid; (C) Experimental density contour points with protein atoms subtracted along with the xGen spherical Gaussian centers (tan spheres); (D) Initial rough fit into the *de novo* density approximation; (E) Re-fit into the idealized density calculated based on the scope defined in (D); (F) The final fully refined xGen ensemble.

```
# Identify the scope to compare the PDB ligand and ligand-alt to the xGen ensemble
> cat xgautoensemble-confs.mol2 ligand.mol2 ligand-alt.mol2 > scope-lig.mol2

# Make an "ensemble" from the PDB ligand and alternate
# We have set the file occ-double with occupancies of the primary and alternate conformer
> cat ligand.mol2 ligand-alt.mol2 > ligand-both.mol2

# Now we will rigorously evaluate the xgen ensemble and the PDB ensemble
> sf-xgen.exe +xg_contour eval xgautoensemble-occ xgautoensemble-confs.mol2
                               protein.mol2 scope-lig.mol2 protein.pdb density xgautoeval

> sf-xgen.exe +xg_contour -xg_ptile xgautoeval-ptile eval occ-double ligand-both.mol2
                               protein.mol2 scope-lig.mol2 protein.pdb density pdbautoeval

# KEY OUTPUT FILES:
#   xgautoensemble-confs.mol2  Final ensemble
#   xgautoensemble-occ         Corresponding occupancies
#   [Analogous files as seen with eval in section on refinement]
```

The 1EXX case involved a relatively small change in volume from the initial scope identified by the Surflex-Dock multicav protomol procedure (Panel C) and the final scope for fitting and refinement (Panel E). Figure 5.5 shows a contrasting case (PDB code 2W9H), where the original volume identified for ligand fitting is several times larger than the ligand to be fit. The procedure is identical, but the process of creating the idealized density approximation is substantially longer, due to the large volume. However, the example demonstrates the specificity with which ligands can be fit into very large X-ray density volumes.

5.6 MISCELLANEOUS ADDITIONAL SURFLEX-XGEN COMMANDS AND OPTIONS

Some of the xGen functionality is present to allow for standalone ligand refinement without making use of other Surflex Modules, and the Tools chapter should be consulted for those commands and options (`extract_sfdb`, `rms_conflist`, `get`, `mget`, `mgetnum`, `mergmols`). The remaining command (`regen`) is present to allow for fresh 3D structure regeneration in cases where the initial ligand approximation is an extremely poor initial starting point. Rather than using the `<prefix>-ligfit` idealized density, it uses the `<prefix>-constrained` version, which provides very rough restraints to guide the new ligand conformation close to the given one.

Bibliography

1. Ajay N Jain, Ann E Cleves, Alexander C Brueckner, Charles A Lesburg, Qiaolin Deng, Edward C Sherer, and Mikhail Y Reibarkh. xgen: Real-space fitting of complex ligand conformational ensembles to x-ray electron density maps. *Journal of Medicinal Chemistry*, 63(18):10509–10528, 2020.
2. Ann E Cleves and Ajay N Jain. ForceGen 3D structure and conformer generation: From small lead-like molecules to macrocyclic drugs. *Journal of Computer-Aided Molecular Design*, 31(5):419–439, 2017.
3. Ajay N Jain, Ann E Cleves, Qi Gao, Xiao Wang, Yizhou Liu, Edward C Sherer, and Mikhail Y Reibarkh. Complex macrocycle exploration: Parallel, heuristic, and constraint-based conformer generation using forcegen. *Journal of Computer-Aided Molecular Design*, 33(6):531–558, 2019.
4. Alexander C Brueckner, Qiaolin Deng, Ann E Cleves, Charles A Lesburg, Juan C Alvarez, Mikhail Y Reibarkh, Edward C Sherer, and Ajay N Jain. Conformational strain of macrocyclic peptides in ligand–receptor complexes based on advanced refinement of bound-state conformers. *Journal of Medicinal Chemistry*, 64(6):3282–3298, 2021.
5. Ajay N Jain, Alexander C Brueckner, Ann E Cleves, Mikhail Reibarkh, and Edward C Sherer. A distributional model of bound ligand conformational strain: From small molecules up to large peptidic macrocycles. *Journal of Medicinal Chemistry*, 66(3):1955–1971, 2023.

CHAPTER 6

AFFINITY MODULE TECHNICAL MANUAL

The QuanSA method is the successor to the QMOD approach (versions 4.2 of the BioPharmics Platform and earlier) [1, 2]. QuanSA builds a pocket *field* rather than constructing a physical set of probes. QuanSA brings together ideas and algorithms from molecular similarity [3–10], and multiple-instance learning [11–17] together with many lessons learned and features developed with QMOD [18–22]. The basic process of constructing a QuanSA model and testing it on a new molecule parallels that of QMOD. An initial hypothesis of ligand alignment is generated, usually by using molecular similarity, possibly augmented using knowledge of some ligand's bound configurations from docking or from direct experiment. Full cliques of molecular poses (one for each training molecule) are constructed, along with alternative poses. A model is then constructed that is a physical field around the training molecules that behaves much in the same way as a protein binding site but without the literalistic aspect of QMOD. The final QuanSA model requires iterative refinement of both the pocket-field parameters and the ligand poses, but the process is relatively rapid. New molecules are flexibly fit into the model, and this results in a prediction of both binding affinity and bound pose families. In addition, computations are made to yield quantitative measurements of model parsimony, confidence for each test molecule prediction, and estimates of structural novelty for test molecules.

Our most recent work has demonstrated a striking synergy between QuanSA model predictions and those from FEP$^+$ [2] on a wide range of protein targets.

The basic construction process consists of a sequence of commands, all implemented within the QuanSA module. The procedure requires straightforward simple user input (e.g. molecule structure files and activity values). Together with the Tools module, the equivalent of SMILES strings and molecular activities are all that is needed to begin building models.

6.1 QUICK START

It is easy to build physically-based activity models with QuanSA. As with all other Surflex modules, calculations *must* be preceded by ligand preparation with the Tools module `forcegen` command. The recommended option for ligand preparation is **-pquant**. Shared with the other modules are the `-torcon` and `-poscon` options, which allow a user to specify preferred geometries and alignments for particular molecular subfragments. QuanSA models are portable as single files, referenced by *model name* (e.g. model *qm00* refers to the structured QMP multi-mol2 file

qm00.qmp.mol2). The corresponding learning state, used for future refinement, is also portable (e.g. model *qm00* refers to the QML file `qm00.qml.mol2`).

The command structure is streamlined, with a typical pocket-field building and testing sequence going as follows:

```
> sf-quansa.exe init TrainData mols.sfdb qm                    # Initialize-->alignments
> sf-quansa.exe build qm-init-00 qm00                          # Model building
> sf-quansa.exe -namelist TestNames score qm00 test.sfdb qtest00 # Scoring new molecules
> sf-quansa.exe eval qtest00-report.txt TestData qtest00stats  # Statistical evaluation
```

Owing to the central importance of an initial alignment hypothesis in producing predictive QuanSA models, hypothesis generation is incorporated as a user-controllable process within model-building. Prior to training a QuanSA model, a user may derive a partial alignment hypothesis and provide a given collection of training molecule poses to QuanSA. Alternatively, the process can be run in a fully automatic fashion. Procedures within the Similarity and Docking modules can both be effectively used in guiding ligand alignments. Both user-guided and fully automatic approaches are illustrated in what follows.

Rather than producing a large pile of ranked individual poses, QuanSA produces a set of pose families, which are influenced not just by the score. As with docking, the process of fitting molecules into a QuanSA model can produce odd poses that score nominally high, but which are clearly incorrect. We use a statistical-physics approach to adjust the scores used for ranking the pose families that takes into account the similarity of the poses within a family to the optimal training molecule poses. Using the pose-families makes the interpretation of models more intuitive. One sees coherent and sensible predicted poses nearly always when they should exist. Also, the pose families give a nice idea of which parts of the ligand are tightly held and which can move around a bit. The pose families can be viewed in the `build` command output file `qm00-traintest-topfam-results.mol2`.

QuanSA constructs a soft envelope around the full set of training molecules and encourages test ligands to find poses that lie within that envelope (a more permissive envelope is used during training to help prevent wandering). The envelope also helps to produce predicted poses that are only infrequently odd-looking. The degree of penetration into this exclusion surface also provides an idea of when a new ligand really has exceeded the physical limits of where training ligands have explored. This is an especially helpful value to consider if one uses a QuanSA model to screen a very large number of new ligands. Those that stay within the envelope, even when they look pretty novel, are predicted much better than the ones that cannot stay inside.

In order to help quantify the molecular space upon which a QuanSA model will be predictive, QuanSA produces probabilistically normalized confidence and novelty scores. So, for different QuanSA models and targets, a confidence score of 0.7 means the same thing relative to the variation seen in the training set. So, for a highly congeneric training set, with little variation, raw similarity scores that are very high will produce adjusted confidence values that are comparable to those seen for a different training set with lower raw similarity values but where the training ligands are more diverse. It gives a more intuitive feeling for how far one can push a model. Generally speaking, a normalized novelty score ("pNov") of ≤ 0.85 will reject clear outliers. Similarly a normalized confidence score ("pConf") of ≥ 0.35 will identify molecules that have good support from the structures present in the training set. Last, a normalized exclusion envelope score ("pExcl") of ≤ 0.95 indicates that a molecule has not obviously extended beyond the physically well-defined space within a model.

We term "in-model" predictions to be those that meet *both* the novelty and exclusion criteria. It is recommended to be judicious with out-of-model predictions, typically using such molecules to further refine one's understanding of a binding pocket or for ambitious scaffold jumps.

6.2 QUANSA COMMAND LINE INTERFACE

Note that there may be minor variations between the figures shown in the manual and the precise results shown in the software distribution. There are no statistically significant differences, but, for example, the N^{th} ranked solution indicated in the manual may correspond more closely to the $(N-1)^{st}$ in the actual distribution. The variations are due to small algorithmic changes across minor version increments as well as cross-platform and compiler differences.

This is the command-line help listing of QuanSA:

```
BioPharmics Platform Version 5.197

sf-quansa (v5.197) <options> <command> <req command args>
  CORE COMMANDS:
    init          train-act train.sfdb initname
      -clnmols      (10)     Number of molecules to select for core multiple-alignment
      -clselwin     (2.50)   Activity window:
                                  0.0 --> select first <clnmols> mols
                               > 0.0 --> select from top activity window
      -clrms        (0.10)   RMS for grouping final cliques
      -clknown      (none)   Set of known poses for competitive ligands
      -clkthresh    (6.50)   Threshold for winners against known poses
      -clkmaxn      (30)     Max number of winners against known poses
      -clnorm                Turn OFF normalization of clique scores (default ON)
      -addthresh    (6.50)   Threshold for eSim alignment to winners

      -clnmake      (5)      Max number of initialization QMLs to make
      -compress     (50)     Number of poses for multi-align NxN targets
      -assay_delta  (0.10)   Absolute assay deviation below which error is defined to be 0.0
    build         initname modelname
      -act_win      (4.0)    Activity window for alignment target choice
      -badpose      (0.65)   Sim threshold for decoy pose detection
    score         modelname testmols.sfdb testname
      -fastalign     (100)    Number of train mols to use for fast alignment
      -hardalign     ( 15)    Number of train mols to use for thorough alignment
      -multiproc npc pnum   Indicates NPC processor run, current processor is PNUM
    add           modelname new-act newmols.sfdb newname
      -addthresh    (6.50)   Threshold for eSim align to existing trainmols to add

    xval          initname modelname n_xval_rounds
    eval          test-report.txt exptl-act outprefix
    select        ModelList outprefix
    disp          modelname outprefix
    paint         modelname inposes.mol2 outprefix testmolname
    paintall      modelname inposes.mol2 outprefix

  MOLECULE SEARCH/ALIGNMENT CONSTRAINTS:
      -poscon       <frags>   Molecular fragments (multi-mol2 file) to constrain position
         -pospen      (5.00)  Penalty for deviating from specification (kcal per Angstrom^2)
         -pwiggle     (0.25)  Amount of free wiggle with zero penalty (Angstroms)
      [-torcon must be applied using forcegen, with penalty modifiable here:]
         -torpen      (0.05)  Penalty for torsional deviation (kcal per deg^2)
         -twiggle     (5.00)  Amount of free wiggle with zero penalty (degrees)
      -mmsweight      (0.5)   Strain weight for energy above global minimum
      -mmwiggle       (0.0)   Among of strain free weighted strain

  CORE OPTIONS:
      -namelist               List of molecule names for SCORE (default: full sfdb)
      -workdir       (work)   Specifies a particular work directory
      -nthreads        (16)   Maximum number of threads to use
```

All commands should be typed lower-case. Molecular output is generally in Sybyl mol2 and SFDB formats, and these are the preferred input file formats as well. In some cases, SFDB format (produced by ForceGen) is *required*. Note that the output of the QuanSA procedures (provided in the `examples/quansa` directory) have been run using the current primary development and production platform, which is Ubuntu Linux running Intel processors. There may be slight differences with different architectures, but the results will not be meaningfully different.

6.3 PRIMARY CHANGES IN CURRENT VERSION

General notes about the current version can be found in the Release Notes in the Foreword to this manual. Detailed notes can be found here.

6.4 LIGAND PREPARATION, MOLECULE NAMES, AND ACTIVITY VALUES

As with docking and similarity computations, ligand preparation, especially regarding protonation and tautomer choice can be critical. Detailed discussion of the use of the Tools module for ligand preparation has been discussed in the preceding chapters. The recommended option for the Tools module `forcegen` command is `-pquant`, which performs the most rigorous and extensive conformational elaboration of the different parameter schemes. Typical preparation is done as follows.

```
# Directory: examples/quansa/serotonin
> sf-tools.exe -pquant forcegen serotonin.mol2 pqser

# KEY OUTPUT FILE:
#   pqser.sfdb       SFDB file containing enumerated conformations
```

Please note that molecule *names* are critically important in the way that QuanSA tracks molecular identity from procedure to procedure. So, for example, the pathname to a molecule may be `foo.mol2`, but the name *inside* the file might be unrelated (for example, "bar"). Within QuanSA, the molecule name specified within the files prepared through the Tools module `forcegen` command will be used as the sole means to identify the molecule and assign its activity value. The best practice, to avoid confusion, is to make sure that molecule names are simple, begin with a letter, and are purely alphanumeric, possibly with hyphens. The use of underscores, whitespace characters, slashes, or backslashes will cause serious problems, as will accidental duplication of molecule names. In the event that customized procedures are used to produce, for example, poses to guide molecular alignment, care must be taken that the molecule names are manually checked and matched in order to ensure proper behavior of the procedures.

Note also that, as with other modules, if the intention is to make use of either the `-torcon` or `-poscon` constraint parameters, it is strongly recommended to include the `-torcon` constraint during ligand preparation as well. This will help to ensure that the conformational constraints are adhered to as well as possible and that conformational exploration is focused on unconstrained degrees of freedom within the molecules.

QuanSA scores the interactions between the induced pocket-field and ligands in units of $\mathbf{pK}_d$. Consequently, the units in which activity should be expressed are $\mathbf{pK}_d$, illustrated as follows:

```
# Directory: examples/quansa/serotonin
# FILE CONTENTS: TrainData
   m4a  =  10
   m1a  = 9.7
   m4b  = 8.5
   m8b  = 7.77
   m10b = 7.3
   m11b = 7.3
   m5a  = 7.3
   m2b  = 6.7
   m3a  = 6.6
   m3b  = 6.5
```

Here, molecule m4a has a specified activity of 10.0, which corresponds to a dissociation constant of 0.1nM. The formula for conversion is $pK_d = -log_{10}(K_d)$, where the K_d must be specified in *molar* units. The provided `TrainData` file relates training activity values to *molecule names*, which, as mentioned earlier, are critical to many QuanSA procedures. The `TrainData` file has a formal syntax. It is possible to specify either "=" or "<" or ">" as activity modifiers in the `TrainData` file. The ">" constraint is helpful to indicate when molecules certainly have some level of reasonable activity, but where the precise value may not be known in the preferred assay (as when the data come from literature). The "<" constraint is helpful to indicate molecules that lack activity at some threshold concentration.

Generally speaking, it is recommended to be thoughtful about sorting the `TrainData` file. The simplest approach is to sort from high activity to low activity, which will ensure that, for example, automatic selection of molecular alignment (see `init` below) seeds will proceed beginning with the highest activity molecule. Alternatively, as was done above, the first molecules were selected to cover the main scaffolds present in the data set. Here there are just two scaffolds, with molecules *m4a* and *m8b* being the most active for each class.

It is important to understand the limitations of activity values derived from different types of experiments. Typical enzyme assays can distinguish activity levels within about a factor of 3 (so 1nM and 3nM are not significantly different). In pK_d space, $-log_{10}(1.0 \times 10^{-9})$ is 9.0 and $-log_{10}(3.0 \times 10^{-9})$ is 8.5. So, if a completed model has roughly 0.5 log units of mean error in the computed versus experimental activities, that approaches the resolution of many assays. Very high quality radioligand displacement assays can distinguish activities within a factor of 2, which translates to a pK_d difference of 0.3. For more complex cell-based or functional assays, the reliability of the assay might be only about a factor of 5, which yields 0.7 in pK_d space. Note that the fundamental limitation on experimental data accuracy may preclude nominally high predictive correlation values depending on the overall activity range within the data.

Non-Standard Assay Values: Typical values from receptor binding or enzyme assays are easily treated by QuanSA, because the fundamental interaction that is being measured is a non-covalent binding interaction whose magnitude is quantified by a dissociation constant. Similarly, assays values that are reported as effective concentrations (e.g. EC_{50} or IC_{50}) can be converted to QuanSA activity values as shown above. The assumptions made by the QuanSA method are that the activity being modeled among a set of ligands is governed by a shared binding event: i.e. that the ligands are competitively binding the same active site. Further, QuanSA assumes that the activity being modeled is properly related to a thermodynamic event: the association of a ligand and macromolecule. There are cases in ligand design where the assay may be either qualitative (inactive, somewhat active, very active), quantifiably binary (inactive or active), or quantitative but expressed as a lethality percentage. In such cases, the assays may be done at a single concentration, in which case the preferred QuanSA activity values cannot be directly produced.

In such cases, the recommended approach is to map the assay values into nominal equivalent pK_d values and make use of the $<$ and $>$ constraints. So, for a binary assay or one where a lethality percentage of some critical value is treated as a dividing line between "active" and "inactive" compounds, activities can be specified as follows:

```
# Binary assay specification. The first six compounds are
# active. The last four are inactive.
# FILE CONTENTS: TrainData
   m0   >  7
   m1   >  7
   m2   >  7
   m3   >  7
   m4   >  7
   m5   <  5
   m6   <  5
   m7   <  5
   m8   <  5
   m9   <  5
```

This specification requires QuanSA to build a model that puts roughly 2.0 log units of activity separation between the active and inactive ligands. Further, it equates activity with a nominal pK_d of at least 7.0, which requires QuanSA to place a significant amount of field strength against which the ligands will interact. Note that this is simply a guideline that has been tested on a small number of cases where real-valued activity values were binarized. The high activity value specified may be larger than what is proposed here, and the gap between the high and low values can be different. However, the values should make sense in an interpretation where they actually are thought of as pK_d. So, for example, activity values of greater than 15.0 (corresponding to a femtomolar effective concentration) make little sense, and activity values much lower than 3.0 (corresponding to a millimolar effective concentration) also make little sense.

In a similar fashion, cases where multiple levels of activity are desired can also be approximated, and these are facilitated by the `-assay_delta` parameter, which specifies an amount of "slop" around a equality-specified activity value (the default for this value is 0.1). The following would be a reasonable approach for four activity levels:

```
# Binary assay specification. The first six compounds are
# active. The last four are inactive.
# FILE CONTENTS: TrainData
   m0   >  8.5
   m1   >  8.5
   m2   =  7
   m3   =  7
   m4   =  7
   m5   =  5
```

```
   m6   =  5
   m7   =  5
   m8   <  3.5
   m9   <  3.5
# The -assay_delta parameter would be specified as 0.5
```

For this case, the result of the combination of inequality constraints, equality constraints, and a specified `-assay_delta` value of 0.5 is that each activity group (high, medium high, medium low, and low) has a buffer of 1.0 log unit between them. Again, this represents a suggested guideline, but systematic experiments using qualitative assay values have not been undertaken.

6.5 NAMING CONVENTIONS FOR MODEL BUILDING

For a particular model-building exercise, the recommended setup is to arrange a QuanSA run folder as follows:

```
# Directory:  Runs from multiple alignment hypotheses may reside here
    quansa-target-run/

# Prepared training mols
> sf-tools.exe -pquant forcegen targetname.mol2 pq

# Training data:                  Activity values and molecule structures
    quansa-target-run/TrainData   # Contains (molname [= < >] activity) on each line
    quansa-target-run/pq*.sfdb    # Prepared molecules

# QuanSA working subdirectory:    This is required
    quansa-target-run/work/       # The path can be changed with -workdir but it should be
                                  # subordinate to the run folder to avoid file clobbering

> sf-quansa.exe init TrainData mols.sfdb qm  # Initialize runs automatically
                                             # Produces qm-smallclique*.mol2
                                             # and      qm-init*qml.sfdb

# For each chosen initial alignment, build three variant models
> sf-quansa.exe build qm-init-00 qm00    # build model from qm-init-00.qml.sfdb
> sf-quansa.exe build qm-init-01 qm01    # build model from qm-init-01.qml.sfdb
> sf-quansa.exe build qm-init-02 qm02    # build model from qm-init-02.qml.sfdb
> sf-quansa.exe build qm-init-03 qm03    # build model from qm-init-03.qml.sfdb
> sf-quansa.exe build qm-init-04 qm04    # build model from qm-init-04.qml.sfdb

# Run a statistical analysis of training performance
> sf-quansa.exe select ModelList qm     # ModelList contains the
                                        # names of all models, one per line
```

For a particular target corresponding to a set of training ligands and activities, a single run folder containing models derived from multiple alignment hypotheses can be maintained. For purposes of inter-model comparison, this arrangement can make visualization simpler than, for example, putting models based on different hypotheses in different places. Also, this arrangement allows for shared use of prepared training ligands and new ligands (which might reside in a `NewMols` subdirectory).

```
# Directory: examples/quansa/serotonin
> sf-quansa.exe disp qm00 disp00

# KEY OUTPUT FILES:
#   disp00-pm.mol2              shows the structure of the pocket-field
#   disp00-<trainmolname>.mol2  individual mol2 files for training ligands
#                               shows interactions with the QuanSA pocket-field
```

Visualization of a particular pocket-field requires running the `disp` command on the corresponding QMP mol2 file (e.g. `qm00.qmp.mol2`). Note that the QMP file contains the final optimal poses of the training ligands and the pocket-field in a non-visualizable form.

Generation of multiple models is recommended for initial model building when there is uncertainty about the molecular alignment. The training process includes a complete re-fit of the training molecules into the derived pocket-field, with the rank correlation and the mean error being the key descriptive statistics, and this information is reported within the "*-trainreport.txt" files for each `build` command. In addition, model *parsimony* is an important criterion. This value measures the degree to which molecules that have similar levels of activity also are predicted to optimally bind in similar poses, both in terms of surface shape and polarity. Higher parsimony models will appear more "coherent" when visualized. The `select` command performs a statistical analysis of the model performance and parsimony in order to offer guidance as to which of several models are likely to be the most predictive on new ligands.

6.6 SIMPLE QUANSA MODEL BUILDING

In the simplest and most automated manner of constructing a QuanSA model, once ligands have been prepared, the model-building process requires two steps (using the `init` and `build` commands), illustrated below on the simple serotonin example.

```
# Directory: examples/quansa/serotonin

# Prepare deeper ring searched conformational ensembles
> sf-tools.exe -pquant forcegen train.mol2 pqtrain
# Key Output File: pqtrain.sfdb
  Compressed format of deeper ring-searched conformational ensembles

# Build multiple initial alignments. Default behavior is to use 10
# training molecules as core alignment seeds (controlled by
# -clnmols). These are automatically selected from a window of
# activity from the most active molecule (default 2.5, controlled
# by -clselwin)
> sf-quansa.exe init TrainData pqtrain.sfdb qm

# File Contents: TrainData (abbreviated and annotated)
   m4a = 10          # Molecules may be listed in a specific order
   m1a = 9.7         # so that the user can choose which molecules
   m4b = 8.5         # to use as alignment seeds. Here, QuanSA chooses.
   ...
   m2b = 6.7
   m3a = 6.6
   m3b = 6.5

# Key Output Files:
#   qm-log                The log file from init
#   qm-*.mol2             An initial coarse alignment of a few mols
#   qm-smallclique*.mol2  Alignment of a few more mols chosen by 2D dissimilarity
#   qm-init*.qml.sfdb     Full initialization information for build
```

It is important to understand the parameters of the `init` command. The default is to build cliques within 0.1 RMSD of each other, however `-clrms` allows more coarsely grained choices. Further, the number of molecules in the initial alignment should reflect the diversity of scaffolds in the training molecules or the major substituents of variants of the same scaffold. As the initial alignment number becomes larger, the process becomes slower so users should not just arbitrarily set a high number. For multiple-core workstations, this is less of an issue. The ten training molecules are automatically selected from within an activity window (2.5 by default) of the most active molecule. (The small serotonin example has only 4 molecules within 2.5 logs of the most active molecule.)

If the activity window is set to zero, then the first N listed in the TrainData file (5 by default) will be used for the initial alignment, which gives the user complete control over which molecules to use, if desired. By default, up to five full alignments of the training molecules (`qm-init*.qml.[mol2/sfdb]`) are generated. If necessary for larger or more complicated data sets, the number of cliques produced can be increased using the `-clnmake` parameter. In some cases, it may be necessary to decrease the `-clrms`. In the serotonin example, the `init` command first

generates a 4-molecule coarse alignment (qm-0*.mol2). Next, 2D dissimilarity is used to choose molecules to add in order to generate a refined alignment (qm-smallclique*.mol2). The refined small clique is then used to generate the learning state with multiple poses per training ligand and initial observers (`qm-init*.qml.sfdb`). The output file `qm-init*.qml.mol2` is a full clique of the top poses of all training ligands. In summary, the coarse alignments are used to seed the small clique alignments which then seed the final alignments. This step-wise building results in more coherent and accurate alignments.

The `init` command has two primary effects. It produces training molecule alignments fully automatically, but it can be influenced by user knowledge and guidance. It also generates pose variants for each molecule and the possible pool of initial observers. The information is embedded along with the training data and user-specified options into the resulting "QML" file, which will be used in the subsequent model building steps. The provided training data file relates training activity values to *molecule names*, which, as mentioned earlier, are critical to many QuanSA procedures. The training data file has a formal syntax, which is described above. The `pq*.sfdb` file provides a deeper ring-searched set of conformational ensembles in a compressed format. The names specified within these molecule files must match the names specified in the training data file one-to-one, although the order in the list is unimportant. These molecule files are expected to have been prepared using the Tools module `-pquant forcegen` procedure.

QuanSA expects that a "work" directory exists, in which temporary files can be placed (these are cleaned up automatically). By default, this directory is assumed to exist (named "work") subordinate to the directory in which the QuanSA run is contained. Multiple QuanSA runs can be made in the same place, the only requirement being that the different names are given for each initialization and for each model.

Figure 6.1 highlights several aspects of the QuanSA initialization information produced from the `init` command. As can be seen from inspection of the log file (qm-log) produced by `init`, four molecules were chosen as being diverse structural representatives first from within 2.5 log units of the most active molecule: m4a, m8b, m1a, and m4b. The coarse alignment of these four molecules, `qm-*.mol2` (green), is then used to seed a small clique alignment `qm-smallclique*.mol2` (cyan) which is used to generate the learning state `qm-init*.qml.sfdb`. Two chemotypes (a more potent angular tricyclic scaffold, e.g. m4a, and a linear tricyclic one, e.g. m8b) are both represented within the chosen set, with tight superimposition of the protonated amines and of the common acceptor oxygen atoms. The initialization procedure also generates variants for each molecule.

The user is strongly encouraged to examine and consider which alignment cliques appear to make the most sense with knowledge of structure-activity information as well as considerations about conformational preferences that may have been designed into the chemical series. As with other modules, guidance can be provided to the `init` procedure in many ways, which will be discussed in more detail later in this chapter.

Following the initial setup, the recommended procedure is to produce one model for each clique (using the `build` command), for *each* alignment that appears sensible. In this example, five slightly different clique alignments were produced, all of which appeared to be reasonable. This completes the model construction steps, including re-fitting each of the training molecules as a nominal test at the end of model construction. Resulting models are typically assessed based on convergence and parsimony criteria using the `select` command.

```
# Directory: examples/quansa/serotonin

# One model each for qm-smallclique-00, qm-smallclique-01, ... qm-smallclique-05
> sf-quansa.exe build qm-init-00 qm00
> sf-quansa.exe build qm-init-01 qm01
> sf-quansa.exe build qm-init-02 qm02
> sf-quansa.exe build qm-init-03 qm03
> sf-quansa.exe build qm-init-04 qm04

# Key Output Files:
#   qm0[01234].qmp.mol2                      models with training ligand best poses
#   qm0[01234].qml.mol2                      model learning state for refinement
#   qm0[01234]-trainreport.txt               Summary of convergence and parsimony
#   qm0[01234]-traintest-topfam-results.mol2 Top re-fit pose families for training mols
```

Each complete model building, refinement, and re-fitting of training molecules takes just a few minutes minutes for this example. The critical files include the report file and the QMP and QML files. The training report contains the

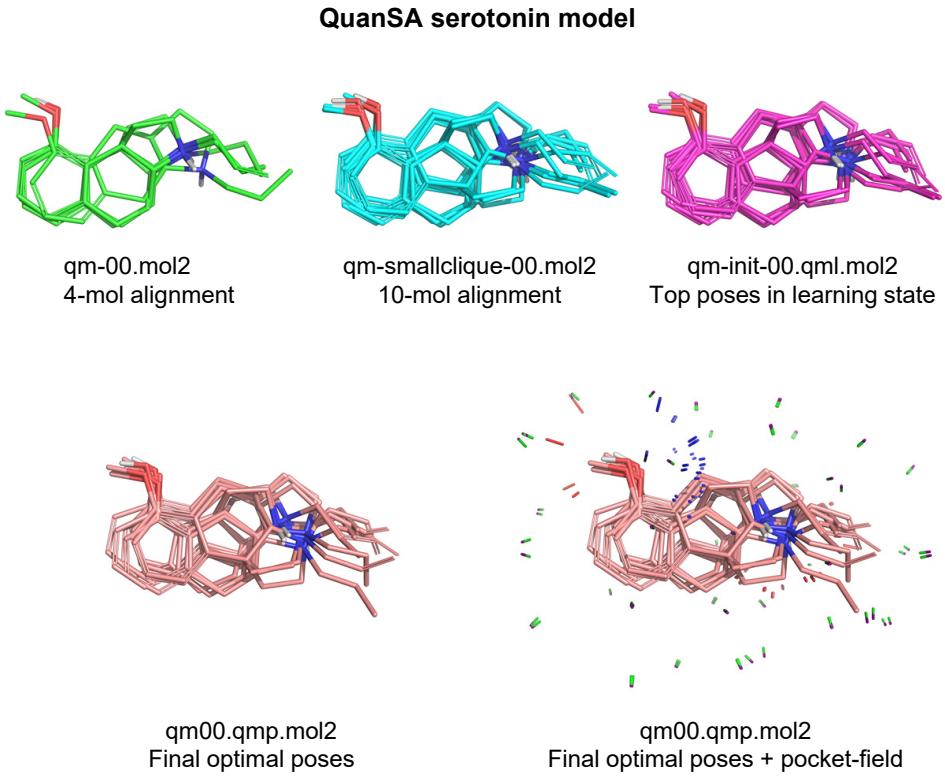


Figure 6.1 Top row: Shown is an initial coarse alignment for 4 serotonin training molecules `qm-00.mol2` (green) used to seed a 10-molecule small clique, `qm-smallclique-00.mol2` (cyan), which is used to generate the learning state `qm-init*.qml.sfdb`. Shown here in magenta is the top pose from the learning state for each training ligand, `qm-init-00.qml.mol2`. Bottom row: Shown in salmon are the final optimal poses of the training molecules, `qm00.qmp.mol2` without and with the derived pocket-field. Although the top poses in the learning state are in a high quality mutual alignment (magenta), the final optimal poses (salmon) are even more compactly aligned.

final computed activities for the training ligands from the model refinement procedure, then the parsimony calculation, and finally the statistics corresponding to a full re-fit of the training ligands into the model. It is advisable for the user to check for adequate model convergence carefully. MSE values of less than 0.1–0.2 (beginning of the training report file) generally will mean that a model has converged to within the accuracy of typical biochemical assays. Outliers in terms of deviations between actual and computed scores may indicate an assay issue. For example, a molecule that is persistently overpredicted may be very similar in all respects to multiple other highly active molecules. In such a case, a solubility or other issue may be responsible for an inaccurate computed score. If outliers are persistent, the user should consider making use of an initial hypothesis alignment that differs in significant respects from the current one. The user may also need to contemplate the potential for multiple binding sites or of partially overlapping binding modes that will be poorly modeled by the normal QuanSA procedure, which seeks to find a *single* parsimonious model for the activity of all training molecules.

It is also important to consider the convergence using the full re-fit of training molecules to the model. Occasionally, the procedure is able to uncover new poses that were not explored in model-building (usually resulting in an overprediction). Also, underpredictions can occur if a ligand in its optimal final pose from training is in some kind of saddle-point configuration that is difficult to recapitulate through the normal fitting procedure. Given models that have converged equally well (considering the re-fit statistics), we have found that the parsimony of the different models is the most useful feature in determining which model variant is likely to be the most predictive. The parsimony measures the degree to which molecules that have similar activity levels also have optimal poses that are similar to one

another. Mean error and Tau are the preferred measurements of model convergence (measures such as r^2 or RMSE tend to overweight the effects of single outliers).

6.6.1 Automatic QuanSA Model Selection using just Training Data

The `select` command, illustrated below, automates the process of considering model convergence and parsimony across multiple different build conditions.

```
# Directory: examples/quansa/serotonin
# File contents: ModelList
  qm00
  qm01
  qm02
  qm03
  qm04

# The select command combines information on model quality to help
# adjudicate which model or models are most likely to be predictive
> sf-quansa.exe select ModelList qm

# Key Output Files
# qm-selectreport.txt    Human-readable file summarizing model quality
# qm-selectreport.tab    Tab-delimited summary table

# Excerpted qm-selectreport.txt:

Model 000: qm00 NO_HOLDOUTS    # Indicates reading of model qm00, with
Model 001: qm01 NO_HOLDOUTS    # no holdout set of molecules specified.
...                            # Holdout sets are described later.

# Statistics for model quality across the different conditions
Model statistics:  mean_parsim = 0.698   mean_traintau = 0.901    mean_trainerr = 0.357
Model statistics: stdev_parsim = 0.037  stdev_traintau = 0.075   stdev_trainerr = 0.079

# Probabilistically normalized statistics for each model and a summary score (prod_score)
Model    N.    prod_Score       Pars     pPars    TrTau    pTrTau   TrErr    pTrErr
qm00     0     0.045954         0.714    0.665    0.900    0.494    0.443    0.140
qm01     1     0.019393         0.628    0.028    0.951    0.746    0.231    0.945
qm02     2     0.175949         0.724    0.757    0.950    0.742    0.396    0.313
qm03     3     0.004841         0.728    0.790    0.756    0.027    0.416    0.230
qm04     4     0.278472         0.698    0.496    0.949    0.738    0.301    0.762
```

Here, model number 4 (qm04) had the best overall combination of training statistics. In this case, given the very small training set, model convergence likely reflects some degree of overfitting as well. Figure 6.1 shows an example of a final derived pocket-field and optimal training molecule poses for this small serotonin example. As described above, the `disp` command will be used to display the pocketfield using qm00 as an example (`disp-pm.mol2`) and the interactions with the final poses of the training molecules (`disp-<trainmolname>.mol2`).

```
# Directory: examples/quansa/serotonin
> sf-quansa.exe disp qm00 disp

# KEY OUTPUT FILES:
#   disp-pm.mol2              shows the structure of the pocket-field
#   disp-<trainmolname>.mol2  individual mol2 files for training ligands
#                             shows interactions with the QuanSA pocket-field
```

In this case, the initial training produced the following training report for qm04:

```
# Directory: examples/quansa/serotonin
# File Contents: qm04-trainreport.txt
    Completed pocketfield qm04 training report:
      Trainmol                  m4a: act    10.00 ...   PRED: 9.82 ... DIFF  0.078
```

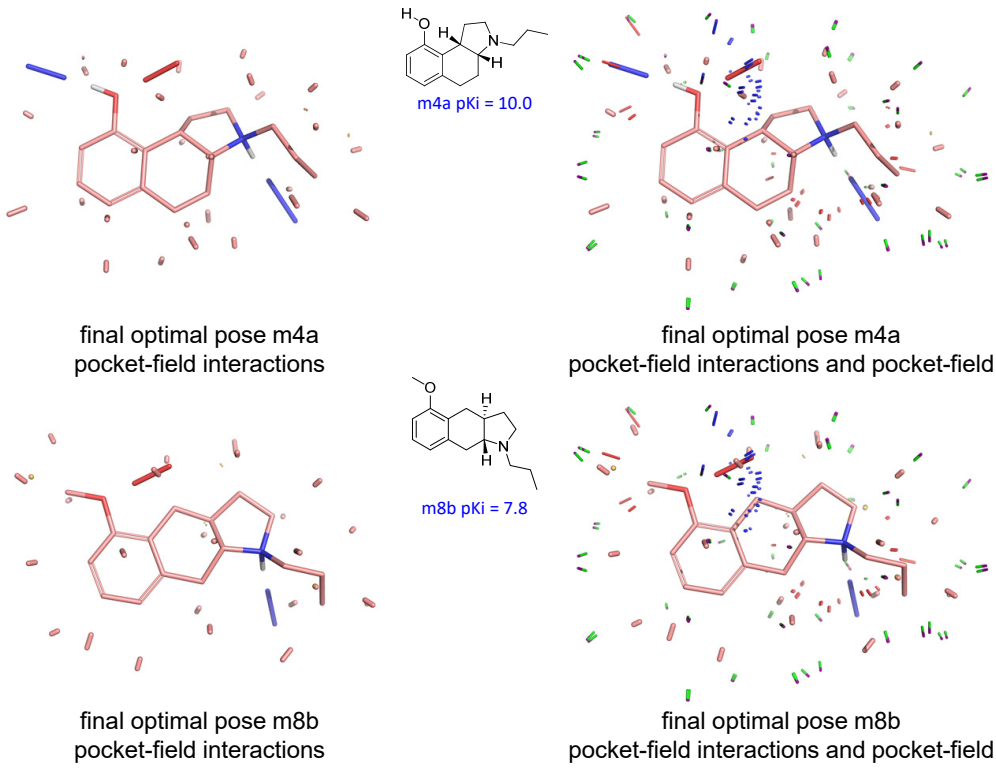


Figure 6.2 Shown without and with the pocket-field are the optimal final training poses and pocket-field interactions for the training molecules m4a and m8b.

```
  Trainmol                m8b: act     7.77 ...   PRED: 7.30 ... DIFF  0.373
  Trainmol                m1a: act     9.70 ...   PRED: 9.10 ... DIFF  0.501
  Trainmol               m10b: act     7.30 ...   PRED: 7.16 ... DIFF  0.037
  Trainmol               m11b: act     7.30 ...   PRED: 6.99 ... DIFF  0.207
  Trainmol                m4b: act     8.50 ...   PRED: 8.16 ... DIFF  0.241
  Trainmol                m5a: act     7.30 ...   PRED: 7.10 ... DIFF  0.095
  Trainmol                m3a: act     6.60 ...   PRED: 6.37 ... DIFF  0.127
  Trainmol                m3b: act     6.50 ...   PRED: 6.68 ... DIFF  0.081
  Trainmol                m2b: act     6.70 ...   PRED: 7.07 ... DIFF  0.271
Overall MSE:   0.0603
-------------------------------------------------------------------------

Parsimony calculation:
  Mol  m4a   m1a: (Act1 10.0 Act2  9.7) weight 0.96  sim 0.81
  Mol  m8b  m10b: (Act1  7.8 Act2  7.3) weight 0.90  sim 0.76
  ...
Total parsimony: 0.698


Number of Mols 10
95.00% Confidence Interval define by CI_Low and CI_High, with 1000 resamples.
================================
Stat    Value   CI_Low  CI_High
KTau    0.949 0.739 1.000
R       0.972 0.813 0.994
R2      0.944 0.662 0.987
```

```
AvgErr  0.301 0.222 0.381
RMSE    0.328 0.235 0.408
---------------------------------
Linear Fit: slope 0.831022 (xint -1.460190  yint 1.213450)
K_Tau pval = 0.00008, estimated using 100000 iterations.
=================================

Summary sorted by novelty.
================================================================================
Mol     Exp      Pred    Err     PNov    PConf   PExcl   TMol    MaxSim   JointSim
m4a     10.000   9.770   0.230   0.005   1.000   0.294   m4a     0.996     9.995
m1a      9.700   9.110   0.590   0.296   0.998   0.334   m1a     0.999    10.000
m5a      7.300   7.120   0.180   0.505   0.992   0.271   m5a     0.999    10.000
m4b      8.500   8.170   0.330   0.508   0.991   0.318   m4b     0.998     9.989
m11b     7.300   7.170   0.130   0.524   0.962   0.383   m11b    0.958     9.890
m10b     7.300   7.710   0.410   0.565   0.994   0.334   m10b    0.957     9.973
m2b      6.700   7.090   0.390   0.587   0.992   0.348   m2b     0.994     9.934
m8b      7.770   7.450   0.320   0.603   1.000   0.314   m8b     0.994     9.953
m3a      6.600   6.380   0.220   0.737   1.000   0.295   m3a     0.999     9.985
m3b      6.500   6.710   0.210   0.971   0.999   0.336   m3b     0.992     9.925
--------------------------------------------------------------------------
Min      6.500   6.380   0.130   0.005   0.962   0.271   TMol    0.957
Avg      7.767   7.668   0.301   0.530   0.993   0.323   TMol    0.989
Max     10.000   9.770   0.590   0.971   1.000   0.383   TMol    0.999
================================================================================
...
```

The first part of the QuanSA trainreport contains the final scores for the optimal poses of the training ligands (see `qm00-trainreport.txt` for this example). Those scores reflect the highest-scoring poses for each molecule during the entire evolution of the initial alignments that were derived from the alignment hypothesis. The second part summarizes the computation of model parsimony (higher values are better). The `build` procedure performs a full re-fit of the training molecules (as if they were "new" molecules to be scored) in order to provide a firm quantification of convergence, which is summarized after the parsimony calculation report. In this example, the re-fit of the molecules produced a good fit to the training data (see above). The first table of information contains summary statistics and confidence intervals. The final mean error of 0.3 log units is at or below the resolution of typical biochemical assays, and the correlation and rank statistics show a very high degree of fit. The next table shows specific data for each molecule. Here, molecule m2b found a slightly higher-scoring pose than was uncovered during initial model refinement. Figure 6.1 shows the resulting model. The concordance of the different scaffold types is high, though it differs somewhat from the initial hypothesis alignment. Much of the activity variation depends on the ability of ligands within this series to simultaneously make the favorable interaction with their amine protons while also being able to fit into the complicated shape of a hydrophobic pocket.

The model building process computes statistics of variation for the training ligands in order to provide scaled, probabilistic values for novelty, confidence, and exclusion penetration for new molecules. The novelty value (PNov in the trainreport table) for a particular molecule represents the degree to which the *union* of the training molecules covers the predicted pose for that molecule, both in terms of pure shape as well as the type and placement of electrostatic functionality. High values of novelty indicate that a molecule may lie beyond what can be predicted from the model (novelty values larger than 0.85 mark outlier territory). The confidence value (PConf) indicates the degree to which a *single* training molecule (in its optimal pose) looks like the molecule in question (the particular training molecule along with the raw similarity score are provided). High confidence generally is predictive of lower prediction error (a PConf value of 0.35 or greater is a sensible cutoff). The exclusion penetration value (PExcl) measures the degree to which the final predicted pose of a molecule protrudes beyond the joint envelope surrounding the final training poses. Molecules that push past this envelope are much less likely to be predicted well than those that fall within it (PExcl values greater than 0.95 generally indicate structural envelope outliers). Note that it is possible for a training molecule itself to present as an outlier with any of these measures, because the statistics of the measurements come from the full population of training ligands.

6.7 NEW MOLECULE PREDICTION AND MODEL REFINEMENT

The process of fitting a new molecule into the model for scoring usually takes 10–20 seconds for relatively rigid molecules of small size and up to a 1–2 minutes for larger and more flexible cases. The alignment is done in an analogous fashion to the process used for training molecules, and it requires quite thorough optimization in order to produce reliable scores. The method makes use of the final positions of the training molecules, and the particular training molecules that were chosen in the initialization phase serve as alignment targets. User-selectable options for modifying this procedure will be discussed later. The output is similar to that from docking (see previous chapters), with scores in pK_d units and with poses and pose families in multi-mol2 files. Reported novelty and confidence measurements make use of statistics from model construction and are normalized using a probabilistic framework such that threshold values can be sensibly used across different cases. The following summarizes the procedure and results from scoring 7 new molecules based on the model just described.

```
# Directory: examples/quansa/serotonin

# Prepare deeper ring searched conformational ensembles
> sf-tools.exe -pquant forcegen test.mol2 pqtest
# Key Output File: pqtest.sfdb
  Compressed format of deeper ring-searched conformational ensembles

> sf-quansa.exe score qm04 pqtest.sfdb qtest04

# Key Output Files:
#   qtest04-report.txt          (Test molecule scores and related values)
#   qtest04-topfam-results.mol2 (The single top pose family for each molecule)
#   qtest04-fam-results.mol2    (All pose families for each molecule)
```

This procedure generates multiple poses per molecule prior to building and ranking pose families. Pose family ranking is primarily dependent on the model score for each pose. However, outlier poses that appear clearly "crazy" based on quantitative assessment of their relative novelty, confidence, or exclusion envelope penetration will cause the probability associated with their respective pose family to decrease. Generally, the user only need examine the top-scoring pose family for each molecule.

```
# Directory: examples/quansa/serotonin
# File Contents: qtest04-report.txt

   mol  pred    pconf  pexcl  pnov   tmol  maxsim bexcl    jointsim
   m1b  6.8900  0.9940 0.3750 0.7280 m2b   0.9610 -0.0100  9.8680
   m2a  8.9700  0.9910 0.4170 0.2410 m1a   0.9620 -0.0000  9.9930
   m6a  7.5400  0.5240 1.0000 0.9700 m1a   0.8300 -0.1000  8.7010
   m7b  6.0100  0.2470 1.0000 1.0000 m2b   0.7890 -0.1500  7.8200
   m9b  7.3800  0.9440 0.4280 0.6660 m8b   0.9610 -0.0000  9.8570
   m10a 6.0400  0.8640 0.5130 0.8370 m10b  0.9370 -0.0000  9.7310
   m11a 6.4400  0.8230 0.3130 0.6980 m11b  0.9410 -0.0000  9.7320
```

In this case, five molecules had novelty values that indicated they were non-outliers. Of these, those predicted to be most potent were m2a, m9b, and m6a. Each of these was predicted with high confidence as well. The scores for confidence, novelty, exclusion, etc. are analogous to those seen above in the train report file. Raw values are also provided for similarity to the most similar training molecule as well as the least amount of exclusion envelope penetration (bexcl). Typically, exclusion values that correspond to normalized *pexcl* values of less than 0.70 represent molecules that fall cleanly within the envelope of the training set. Those with values greater than 0.95 fall outside of the envelope (none here). The report also provides a K-nearest-neighbor score based on similarity calculations using the final optimal predicted poses, which can be useful for out-of-model prediction estimates.

The experimental activities for these three molecules were close to those predicted: m2a = 9.7, m6a = 6.4, and m9b = 7.4 (nominal errors of -0.3, +0.7, and +0.4 respectively). Figure 6.3 shows the poses comprising the top predicted pose families relative to the optimal pose of the most similar training molecule. The variation in the poses reflects the different ways in which the ligand can fit into the model. Here, there is room for movement for the ring system as well as for variation in the placement of the nitrogen substituent. Two molecules protruded beyond the

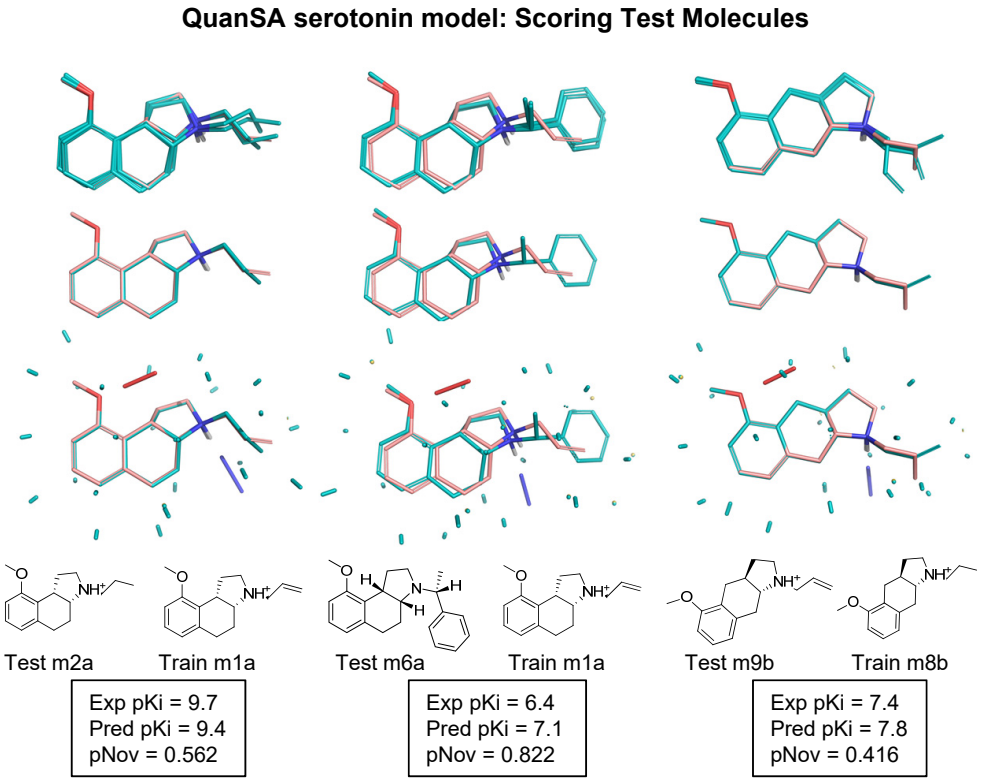


Figure 6.3 Shown for each of three serotonin test molecules (teal) are the alignments of the top pose family or the top-scoring pose of the pose family with the optimal pose of the most similar training molecule (salmon). The alignments are also shown with the interactions with the pocket-field. Test molecules m2a, m6a, and m9b were non-outliers with pNov < 0.85 and the activities were accurately predicted despite the very small training set.

physical envelope of training ligands (m6a and m7b). The quantitative assessment of prediction confidence, novelty, and exclusion envelope penetration can all be useful in making decisions during lead optimization. In general, it is wise to select molecules that are predicted to be potent and which have high confidence and low novelty, as expected. But active selection of novel molecules or of those that extend beyond the explored training envelope, even in the case of predicted low activity, can drive model refinement toward a better characterization of a binding site [20].

Given assay data on newly obtained molecules, statistics may be automatically calculated using the previously generated prediction report. Assay data should be formatted in a 3-column file containing the molecule names, activity modifiers, and activity measurements (exactly analogous to the TrainData file used in the `init` procedure). The `eval` procedure takes as input a prediction report and newly gathered activity data, and produces statistics such as Kendall's Tau ranking, Pearson's R, and root-mean-squared-error (SDEP) to name a few. Below shows an example of the activity data file format, command for generating the statistics, and a truncated example of the output. Note this example makes use of an extremely sparse test set, so performance metrics shown in the evaluation report are not expected to be statistically significant. However, it illustrates the automated process of evaluating prediction performance requiring minimal user preparation.

```
# Directory: examples/quansa/serotonin
# File Contents: TestData
   m1b  = 7.3
   m2a  = 9.7
   ...
   m10a = 6.3
   m11a = 6.3
```

```
> sf-quansa.exe eval qtest04-report.txt TestData qtest04stats

# Key Output File: qtest04stats-eval.txt

# File contents: qtest04stats-eval.txt (excerpted)

   ...
   Stats for all the mols.

   Number of Mols 7
   95.00% Confidence Interval define by CI_Low and CI_High, with 1000 resamples.
   ================================
   Stat    Value CI_Low CI_High
   KTau    0.667  0.000  1.000
   R       0.865  0.061  0.996
   R2      0.748  0.017  0.992
   AvgErr 0.456  0.211  0.720
   RMSE    0.577  0.271  0.817
   --------------------------------
   Linear Fit: slope 0.736418 (xint -2.429272  yint 1.788960)
   K_Tau pval = 0.03845, estimated using 100000 iterations.
   ================================
   ...
```

The report provides several different presentations of the predicted versus experimental activity values. In general, the non-parametric rank statistic Kendall's Tau is quite reliable. Given that QuanSA builds a physical representation of a binding site based on predicted bound poses, it is important to make use of the prediction quality metrics in assessing the meaning of predictions on new molecules. Here, we see that, considering all 7 molecules, we have a mean error of prediction of 0.7 log units. This includes ligands identified as being outliers, based on either structural novelty or volume exclusion penetration criteria. For ease of visualization of the test results from the `score` command, the `paint` and `mget` commands parse the `qtest-topfam-results.mol2` into the top pose or top pose family as follows:

```
# Directory: examples/quansa/serotonin
> sf-quansa.exe paint qm00 qtest00-topfam-results.mol2 paint00 m2a

# KEY OUTPUT FILES:
#   paint00-m2a_fam00_00.mol2   shows top pose and interaction with pocketfield

> sf-tools.exe mget qtest00-topfam-results.mol2 m2a-molnamelist m2a_fam00.mol2

# KEY OUTPUT FILES:
#   m2a_fam00.mol2   top pose for m2a
```

6.7.1 Automatic QuanSA Model Selection using Holdout Data

This simple serotonin example will be used to further discuss the QuanSA model selection protocol. The `select` procedure makes use of model training quality statistics (reported in the three `qm*-trainreport.txt` files) *and* performance on a holdout set of molecules (reported in the three `qtest*stats-eval.txt` files). The holdout molecules were not used directly in the initial model building. In the serotonin example, three alignment cliques were used to generate three models. Each of the three models were then used to score a set of holdout molecules. Two text files, `ModelList` and `ModelHoldoutList`, were manually generated. These files contained the names of the three resulting models, `qm0[012]`, with `ModelHoldoutList` also containing the corresponding `qtest*stats-eval.txt` file names. The `select` command uses either `ModelList` or `ModelHoldoutList` as input and generates a model quality score that is a product of p values for each model listed. The score is labeled Tr_Score for information based only on the training data and Tr+H_Score for information that combines that with performance on a holdout set.

The former score is the multiplicative product of the normalized values for the parsimony, KTau, and average error, and provides a relative ranking of model performance. As shown below in `qmholdout-selectreport.txt`, the

qm04 model had the highest score from the training statistics (Tr_Score 0.278) and from the holdout set (Tr+H_Score 0.219). Superior predictive performance of models with the highest model selection scores has been observed for many targets with much larger datasets (not shown). While these scores can be generated purely from the training data alone, use of a holdout set for model selection is still advised. Overfitting is possible in constructing QuanSA models, as is the case with any machine-learning procedure.

The `select` command, illustrated below, can make use of both training statistics and performance on a holdout set.

```
# Directory: examples/quansa/serotonin
# File contents: ModelHoldoutList
  qm00 qtest00stats-eval.txt
  qm01 qtest01stats-eval.txt
  qm02 qtest02stats-eval.txt
  qm03 qtest03stats-eval.txt
  qm04 qtest04stats-eval.txt

# The select command combines information on model quality to help
# adjudicate which model or models are most likely to be predictive
> sf-quansa.exe select ModelHoldoutList qmholdout

# Key Output Files
# qmholdout-selectreport.txt    Human-readable file summarizing model quality
# qmholdout-selectreport.tab    Tab-delimited summary table

# Excerpted qmholdout-selectreport.txt:

Model 000: qm00 qtest00stats-eval.txt    # Indicates reading of model qm00, with
Model 001: qm01 qtest01stats-eval.txt    # holdout set of molecules specified.
...

# Probabilistically normalized statistics for each model and a summary score (Tr+H_Score)
Model N. Tr_Score Parsim pPars TrTau pTrTau TrErr pTrErr Tr+H_Sco TeTau pTeTau TeErr pTeErr
qm00  0  0.045954 0.714  0.665 0.900 0.494  0.443 0.140  0.013328 0.882 0.593  0.516 0.489
qm01  1  0.019393 0.628  0.028 0.951 0.746  0.231 0.945  0.008861 1.000 0.828  0.497 0.552
qm02  2  0.175949 0.724  0.757 0.950 0.742  0.396 0.313  0.023355 0.778 0.347  0.549 0.382
qm03  3  0.004841 0.728  0.790 0.756 0.027  0.416 0.230  0.000014 0.556 0.042  0.690 0.072
qm04  4  0.278472 0.698  0.496 0.949 0.738  0.301 0.762  0.219337 1.000 0.828  0.312 0.951
```

In constructing models for application in prospective design, it is recommended to segregate data temporally, with training being done on an early set and some testing on an adjacent but later time window. For example, one might train on the earliest N molecules to produce some set of models. Those models with good convergence and parsimony might then be tested on the next M molecules, with those having the best performance used to make predictions on as-yet-unsynthesized ligands.

Given assay data on new experimentally tested molecules, refinement of a model may proceed in two ways. Clearly, *de novo* construction of a model is possible, but incremental refinement of an existing model is usually a more attractive option, especially if the new data reflect relatively minor adjustments to prior predictions. Here, the procedure for refinement using true activity values for m2a, m6a, and m9a will be shown (a more complex example involving multiple rounds of iterative refinement is shown in on the CDK2 example).

```
# Directory: examples/quansa/serotonin
# File Contents: Add3Data
   m2a   =   9.7
   m6a   =   6.4
   m9b   =   7.4

> sf-quansa.exe add qm04 Add3Data pqtest.sfdb qm04new

# Key Output Files
# qm04new-trainreport.txt (Report of refinement results)
# qm04new.qmp.mol2        (Refined model and poses)
```

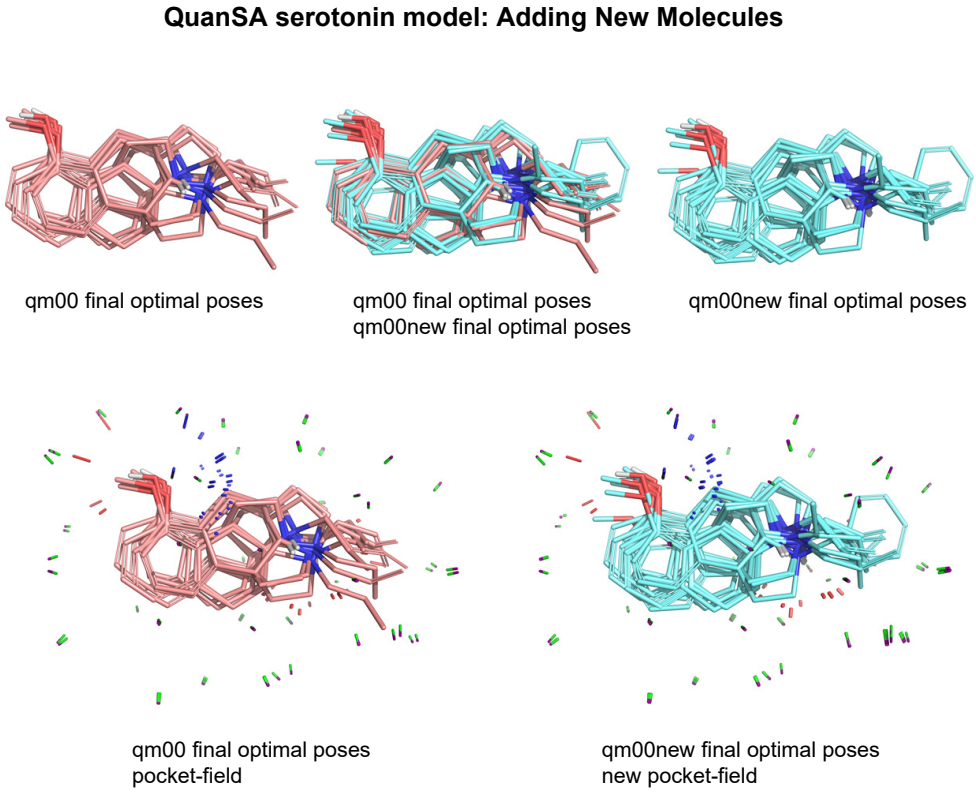


Figure 6.4 Shown are the optimal final training poses for the original model (salmon) and the optimal final poses after refining the model with 3 new serotonin ligands (light blue). Note that the newly refined pocket-field is different from the original pocket-field (shown, from Figure 6.1).

```
# qm04new.qml.mol2        (New learning state)
```

Scoring new molecules with the refined model can be done as before (specifying "qm04new" as the model name), and the refinement process can be iterated. The refinement process generally takes close to the same amount of time as the initial model building in order to allow for new data to significantly influence the model. Figure 6.4 shows the subtle changes in model refinement relative to a starting original model.

```
# Look at the results from the serotonin QuanSA qm00 model as an example:
pym disp.pml
```

A more complex example involving the $GABA_A$ Receptor benzodiazepine binding site is also included in the examples:

```
# Directory: examples/quansa/bzr
# File Contents: RunFull
    # BZR Example from first QuanSA paper \citep{cleves2018}.

    ...

    # Use ForceGen to generate conformer pools of the train/test sets
    sf-tools.exe -pquant forcegen train.mol2 pqtrain
    sf-tools.exe -pquant forcegen test.mol2 pqtest

    # copy the bzr multiple ligand alignment
  # Version 5.173 --> mesim-pq-12.mol2 was copied as good-align.mol2
```

```
  cp ../../similarity/multiple_alignment/bzr/good-align.mol2 .

    # We can drive the alignment toward a particular core alignment
    sf-quansa.exe -clknown good-align.mol2 init TrainData pqtrain.sfdb qm

    ...

    sf-quansa.exe build qm-init-00 qm00
    sf-quansa.exe score qm00 pqtest.sfdb test00
    sf-quansa.exe eval test00-report.txt TestData qm00

    ...

    sf-quansa.exe build qm-init-04 qm04
    sf-quansa.exe score qm04 pqtest.sfdb test04
    sf-quansa.exe eval test04-report.txt TestData qm04

    sf-quansa.exe select ModelList sel
  # Version 5.173 --> qm04 is the best model from the run using good-align as known poses.

# Key Output Files
# sel-selectreport.txt    (Report of variant model performance)
```

The results parallel those presented in the introductory QuanSA paper [1].

6.8 MORE SOPHISTICATED ALIGNMENT GENERATION

Ligand Diversity for Alignment Constraint: The mutual ligand alignments form the core of the learning process for QuanSA, and the quality of the alignments, and the degree to which they mimic the relative molecular alignments seen biologically, are the biggest factor influencing the predictive power of QuanSA models, especially on new scaffolds.

It is frequently the case that a number of ligands are known to be competitive with a series that is the subject of a lead-optimization exercise. Often, these compounds may have different underlying scaffolds, but activity data may not necessarily be available. Such compounds can be productively utilized within QuanSA to help derive more accurate hypotheses for ligand alignments. One way to do this is as follows (using CDK2 as an example target):

```
# Directory: examples/quansa/cdk2/Hypo-Denovo

# RunMakeHypo is the script to generate the multiple ligand alignment

# Prepare the CDK2 ligands using sf-tools/forcegen
# The file "hypo.smi" contains the SMILES strings for 3 CDK2 ligands
  O=S(C1=CC=C(NC2=NC(OCC3CCCCC3)=C(N=CN4)C4=N2)C=C1)(N([H])C)=O m42
  O=S(C1=CC=C(NC2=NC(OCC3CCCCC3)=C(N=CN4)C4=N2)C=C1)(CC[NH2+]C5CCCC5)=O m66
  O=C1NC2=C(C(SC=N3)=C3C=C2)/C1=N/NC4=CC=C(CS(=O)(NC)=O)C=C4 lig-1ke6

> sf-tools.exe fgen3d hypo.smi hypo3d
> sf-tools.exe -pquant forcegen hypo3d.mol2 hprep

# Use the Similarity module to build a de novo hypothesis with no structural information
> sf-sim.exe mult_esim hypo-mol-names hprep.sfdb hypo

# Excerpted hypo-log:

   ...

   Final hypo 00 prob: 0.7229
                      m42 strain: 4.1 kcal/mol
                      m66 strain: 2.9 kcal/mol
                  lig-1ke6 strain: 2.6 kcal/mol

   Final hypo 01 prob: 0.704865 (min_rms = 3.24)
                      m42 strain: 2.0 kcal/mol
```

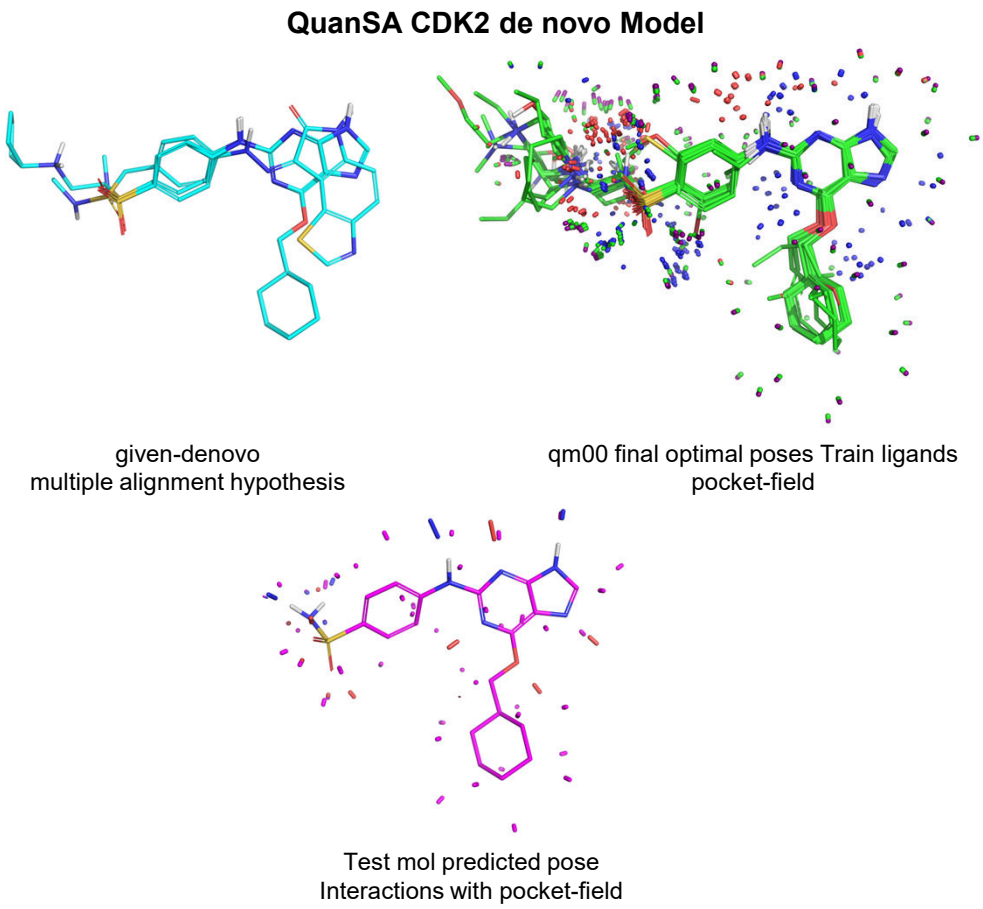


Figure 6.5 The selected *de novo* mutual alignment for 3 CDK2 ligands (`hypo-01.mol2`, cyan), the final optimal poses of the training molecules (green), and the predicted pose of a test molecule with the interactions with the pocket-field (magenta).

```
                    m66 strain:  1.4 kcal/mol
                lig-1ke6 strain:  2.3 kcal/mol

   Final hypo 02 prob: 0.681283 (min_rms = 1.48)
                    m42 strain:  0.7 kcal/mol
                    m66 strain:  1.1 kcal/mol
                lig-1ke6 strain:  3.4 kcal/mol

   Final hypo 03 prob: 0.637590 (min_rms = 2.32)
                    m42 strain:  3.1 kcal/mol
                    m66 strain:  0.5 kcal/mol
                lig-1ke6 strain:  3.3 kcal/mol

   Final hypo 04 prob: 0.629990 (min_rms = 1.80)
                    m42 strain:  2.4 kcal/mol
                    m66 strain:  2.1 kcal/mol
                lig-1ke6 strain:  3.1 kcal/mol
   ...
> cp hypo-01.mol2 ../QuanSA-Denovo/given-denovo.mol2

# Look at the chosen and other de novo hypotheses:
pym disp.pml
```

The Similarity module `mult_esim` procedure builds pose cliques without requiring activity data for the ligands under consideration, and it tries to simultaneously maximize mutual similarity, minimize strain energy, and minimize the ratio of the volume of the union of the ligands compared with the volume of the single largest ligand (see the Similarity module chapter for additional details). Here, ten alignment cliques were produced, all of which are quite similar to one-another. Given that we are building a potency prediction model, consideration of ligand energetics within the context of the other values should receive special attention. Here, hypo-00 had the highest probability and the lowest MMFF strain energy values for the three ligands. This mutual alignment is shown in Figure 6.5. Enough is known about the binding of CDK2 inhibitors from the substituted guanine series and the other ligands in Figure 6.5 to say that the correspondence of parts between the different ligands is correct in this pure similarity-based alignment. The conformation of the left-hand substituent is slightly different than the bound conformations. Nonetheless, hypo-01 could be used to guide model building by using it as the given-clique with the `-clknown` parameter and the `init` command. Many of the cliques generated here from the `hypo` procedure have similar superimpositions and strain energies. In a real-world exercise, the top few hypos should be used to build QuanSA models that could then be compared using the `select` or `xval` procedures.

```
# Directory: examples/quansa/cdk2
# Prepare the ligands for QuanSA
# The file cdk2.mol2 is a multi-mol.2 of 80 cdk2 ligands
# RunPrep script:
> sf-tools.exe -pquant forcegen cdk2.mol2 pqcdk2

# Directory: examples/quansa/cdk2/QuanSA-Denovo
# Generate full alignment cliques
# using similarity-derived poses to guide the alignment
# RunInit script:
> sf-quansa.exe -clknown given-sguided.mol2 init TrainData pqcdk2.sfdb qm

# Build 5 alternative models
# RunBuild script:
> sf-quansa.exe build qm-init-00 qm00
> sf-quansa.exe build qm-init-01 qm01
> sf-quansa.exe build qm-init-02 qm02
> sf-quansa.exe build qm-init-03 qm03
> sf-quansa.exe build qm-init-04 qm04

# Select a model
# RunSelect script:
> sf-quansa.exe select ModelList qm

# Score the test molecules (using all models here/could use only winner model)
# RunTest script:
> sf-quansa.exe -namelist TestNames score qm00 pqcdk2.sfdb qtest00
> sf-quansa.exe eval qtest00-report.txt TestData qtest00stats

> sf-quansa.exe -namelist TestNames score qm01 pqcdk2.sfdb qtest01
> sf-quansa.exe eval qtest01-report.txt TestData qtest01stats

...

> sf-quansa.exe -namelist TestNames score qm04 pqcdk2.sfdb qtest04
> sf-quansa.exe eval qtest04-report.txt TestData qtest04stats

# Display the pocket-field
sf-quansa.exe disp qm00 disp00

# Display the interactions with the pocket-field for a Test molecule
sf-quansa.exe paint qm00 qtest00-topfam-toppose.mol2 paint m29

# Visualize the model, the given-denovo poses, the pocket-field, and the poses of the test
    molecules.
```

```
# pym disp.pml
```

Direct Structural Guidance for Alignment Constraint: We have shown that using direct knowledge of protein structures to guide model building leads to *both* improved quantitative prediction of activity as well as improvements in pose prediction from the QuanSA precursor algorithm, QMOD, *over* docking alone [23] . QuanSA should exhibit similar behavior. This can be done using the docking protocols described in the Docking module chapter, as follows.

```
# Directory: examples/quansa/cdk2/Hypo-SG

# Prepare conformer pools for the CDK2 ligands to be docked:
> sf-tools.exe fgen3d ligs.smi ligs
> sf-tools.exe -pquant forcegen ligs.mol2 pq

# Use Surflex-Dock to generate protomols
# Use Surflex-Dock to perform ensemble docking against five aligned proteins.
# See examples/docking/protein to see the mutual alignment procedure.
# PDB codes 1H0V, 1OGU, 1H01, 1KE6, and 1H1P are used here (automatically selected
# using psim_choose_k).
# copy and rename the prepared cluster center protein and ligands
> source copy-cdk2-centers

# combine the crystallographic ligands into a multi-mol2
> cat ligand-0*.mol2 > PoseHints.mol2

# Build the set of protomols
> sf-dock.exe mproto ProtoList mpro

# Dock the two high-activity ligands
> sf-dock.exe -pgeom -lmatch PoseHints.mol2 gdock_list pq.sfdb mpro-targets log
> sf-dock.exe -posehints PoseHints.mol2 posefam log

# Now take the top pose of the top predicted family for each of m42
# and m66 and provide them to the QuanSA intialization procedure for
# guidance
# Copy the top poses of the top families
> cp log-topfampose.mol2 given-sguided.mol2
> cp given-sguided.mol2 ../QuanSA-SG

# Look at the docking results
> pym disp.mol2
```

When making use of the `-clknown` option, it is possible to provide multiple poses for each of the given molecules. Where multiple poses are all sensible choices, is recommended to use all of them. The QuanSA initialization procedure balances the needs of overall congruence among the ligands to be modeled with the other impinging constraints. Here, only the top pose for each of the two docked molecules will be used as structural guidance for the QuanSA model building.

Figure 6.6 shows two of the five CDK2 protein structures that were aligned and used as multi-structure docking targets. Three amino acids (Lys33, Gln85, and Lys89) are labeled in the structures to demonstrate the significant structural diversity exhibited by the CDK2 pocket. Figure 6.7 shows the combined top poses from the top-scoring pose families for molecules m42 and m66 (yellow) resulting from a standard ensemble docking procedure in the context of CDK2 structure 1KE6 (protein and ligand in green and teal, respectively). As mentioned earlier, the correct pose of the guanine ligands orients the sulfonamide substituent downward where the kinase hinge is across the top of the ligands and solvent is to the left. Here, there is slight variation in the precise poses of the pendant substituents (molecule m42 had 5 total poses in the top-scoring pose family, and molecule m66 had 14). We will use of the well-placed top pose ligand alignment within the QuanSA `init` procedure directly as "given" poses for these two training ligands, as follows:

```
# The ligands were prepared for QuanSA above
# Directory: examples/quansa/cdk2
# Prepare the ligands for QuanSA
# The file cdk2.mol2 is a multi-mol.2 of 80 cdk2 ligands
```

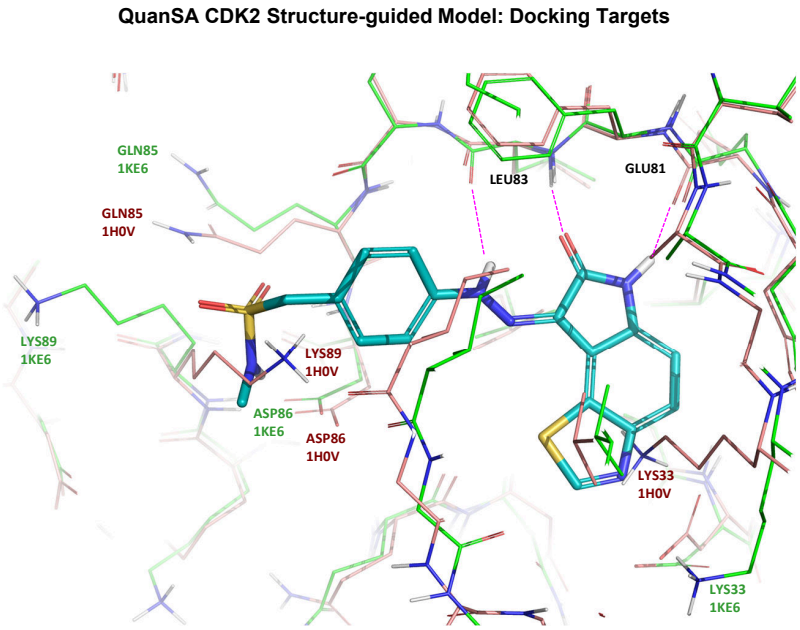


Figure 6.6 Shown in thin sticks are two of the five CDK2 structures that were aligned and used for multi-structure docking of two CDK2 training ligands. 1KE6 protein and native ligand are in green and teal, respectively. 1H0V protein is in salmon. Leu83 and Glu81 are the hinge region residues of which the backbone carbonyl and amine groups make hydrogen bonds with the ligand (purple dashed lines). Note that the hinge region is tightly aligned between the two protein structures. In contrast, residues Lys33, Gln85, and Lys89 show the significant mobility that has been observed in the CDK2 pocket.

```
# RunPrep script:
> sf-tools.exe -pquant forcegen cdk2.mol2 pqcdk2

# Directory: examples/quansa/cdk2/QuanSA-SG
> cp ../pqcdk2.sfdb .

# We have docked molecules m42 and m66 and combined their
# top-scoring pose families into given-sguided.mol2

# We will run a QuanSA initialization using the two molecules listed first
# in the TrainData file (m42 and m66) using the poses derived from docking
# RunInit script
> sf-quansa.exe -clknown given-sguided.mol2 init TrainData pqcdk2.sfdb qm

# The RunBuild script constructs five alternative models
> sf-quansa.exe build qm-init-00 qm00
...

# The RunSelect script runs the recommended model selection procedure
> sf-quansa.exe select ModelList qm

# Contents of qm-selectreport.txt:
    Model statistics:  mean_parsim = 0.717   mean_traintau = 0.893    mean_trainerr = 0.253
    Model statistics: stdev_parsim = 0.005  stdev_traintau = 0.011   stdev_trainerr = 0.014

    Model   Number  prod_Score      Parsim   pPars   TrTau   pTrTau  TrErr   pTrErr
    qm00    0       0.817942       0.724    0.929   0.913   0.965   0.234   0.912
    qm01    1       0.027568       0.716    0.398   0.891   0.420   0.267   0.165
    qm02    2       0.088883       0.721    0.795   0.881   0.133   0.239   0.842
    qm03    3       0.010897       0.712    0.130   0.895   0.565   0.268   0.148
```

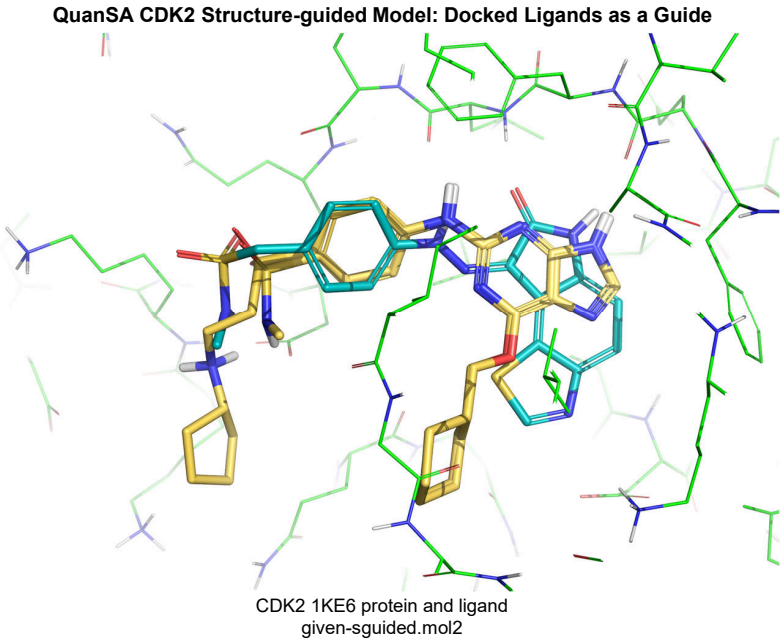


Figure 6.7 Shown is the docking result for two CDK2 training ligands, m42 and m66 (yellow) in the context of the CDK2 pocket (1KE6 protein and ligand in green and teal, respectively). Shown are the top poses from the top-scoring pose families for m42 and m66 combined to generate `given-sguided.mol2` (yellow).

```
35      qm04     4         0.017066          0.713    0.182    0.886    0.256    0.258    0.367
```

The selection procedure shows low training error and high Tau values for all models. Because model qm00 shows consisitently (if just slightly) better statistics across the board, it is judged to be the best model. Note that there is a cross-validation procedure that can be used instead (with the `xval` procedure). However, the `select` procedure, which was initially described above in the serotonin example, has been shown to be reliable across many targets. This is particularly true when a holdout set can be used to provide an additional measurement of model quality.

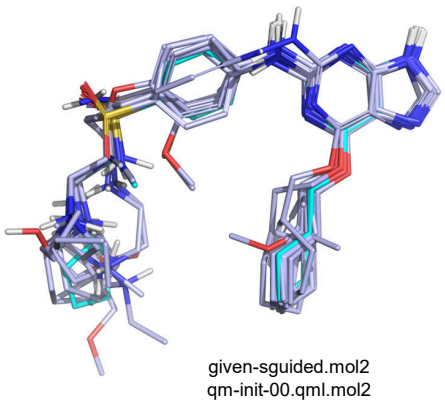


Figure 6.8 Shown in cyan are the m42 and m66 top poses derived from docking that were used as given poses (`given-sguided.mol2`) to guide the generation of the full initial pose clique, `qm-init-00.qml.mol2`, for 30 CDK2 training ligands (light blue).

Figure 6.8 shows the m42 and m66 top poses derived from docking (cyan) that were part of the given poses to guide the generation of the initial full pose clique for 30 CDK2 training ligands, `qm-init-00.qml.mol2` (light blue). The variation of the full set of training ligands falls nicely within the envelope that was explored by the docked ligands. Proceeding to build models from this particular alignment is as follows.

```
# Directory: examples/quansa/cdk2/QuanSA-SG
# File Contents: TrainData (abbreviated and annotated)
   m42   =   8.2
   m66   =   7.6
   ...

# Preferred method is to build 5 alternative models
# RunBuild script:
> sf-quansa.exe build qm-init-00 qm00
> sf-quansa.exe build qm-init-01 qm01
> sf-quansa.exe build qm-init-02 qm02
> sf-quansa.exe build qm-init-03 qm03
> sf-quansa.exe build qm-init-04 qm04

# Select a model
# RunSelect script:
> sf-quansa.exe select ModelList qm
# Display the pocket-field (using qm00 as the winner model)
sf-quansa.exe disp qm00 disp00
```

Visualization of a particular pocket-field requires running the `disp` command on the corresponding QMP mol2 file (e.g. `qm00.qmp.mol2`). Note that the QMP file contains the final optimal poses of the training ligands and the pocket-field in a non-visualizable form.

The `qm00-trainreport.txt` shows that model qm00 has high parsimony, a high Kendall's Tau, and a low average error. Figure 6.9 shows the final model along with the optimal poses for the 30 training ligands. The specific field interactions that drive the activity predictions correspond in many cases directly with protein atoms that are know to be critical for CDK2 inhibition [23]. Figure 6.9 also helps to illustrate the parsimony concept. Here, all of the active molecules share similar dispositions of not just their central scaffolds, but also of their pendant substituents. Where there are, for example, amine-containing substituents of the sulfonamide at left, they tend to find optimal positions for interaction with the model in congruent positions. Figures 6.10 and 6.11 show all of the training molecules and training molecule m42 (gray) in the context of the CDK2 pocket. Figure 6.11 includes the interaction sticks which mimic the ligand interactions with CDK2 residues Leu83 and Glu81 in the hinge region, Lys33 of the catalytic triad, and Asp86 and Lys89.

6.9 PREDICTIONS ON NEW MOLECULES: CONSIDERING NOVELTY AND CONFIDENCE

The following illustrates testing the 50 molecules present in the file *cdk2/TestList*:

```
# Directory: examples/quansa/cdk2
# File Contents: TestNames
   m02
   m03
   m04
   m07
   ...

# Score all 50 test molecules (using qm00 as the winner model)
# RunTestRefine script:
> cp TestNames qm00-round0-TestNames
# ROUND 0: TEST model against qm00-round0-TestNames
> sf-quansa.exe -namelist qm00-round0-TestNames score qm00 pqcdk2.sfdb qm00-test

# Key Output Files
# qm00-test-report.txt          Report of scoring results
```

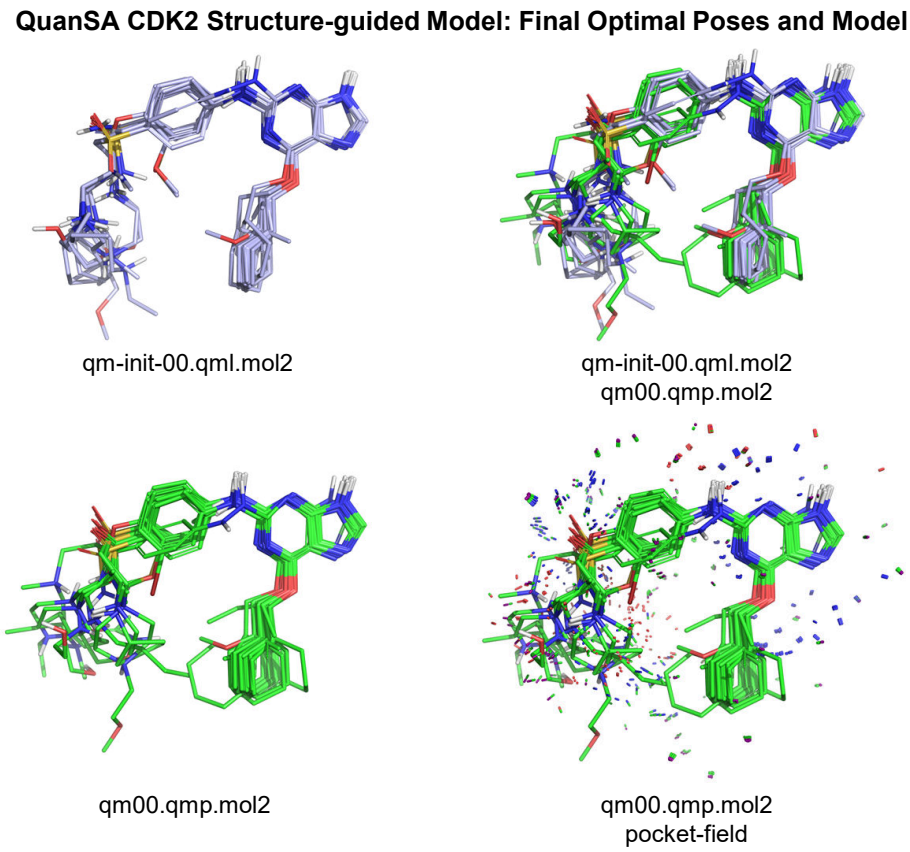


Figure 6.9 Shown are training molecule alignments and the final derived CDK2 pocket-field from the structure-guided QuanSA protocol for model qm00. The `init` procedure produces a full clique of the top poses of all training ligands, `qm-init-00.qml.mol2`. The `build` procedure generates the optimal final training poses of all 30 CDK2 ligands, `qm00.qmp.mol2`, as well as the pocket-field.

```
# qm00-test-topfam-results.mol2 Top pose families for test mols

# Excerpt of qm00-test-report.txt: (top scoring mols)
   mol   pred      pconf     pexcl     pnov      tmol   maxsim     bexcl     jointsim
   m29   8.8900    0.7310    0.4290    0.2020    m42    0.9120    -0.0000    9.4380
   m64   7.1800    0.6200    0.6180    0.3800    m76    0.8760    -0.0000    9.4500
   m46   8.5300    0.3560    1.0000    0.5470    m42    0.8600    -0.0000    9.1060
   m44   7.8100    0.8170    0.4170    0.0630    m42    0.9230    -0.0000    9.9650
   m45   6.7500    0.9200    0.3610    0.0960    m51    0.9500    -0.0000    9.9130
  ...
```

The report provides a prediction of activity along with multiple pieces of information that compare each new ligand to the training set. The first three columns after the activity prediction yield probabilistically normalized measures of confidence, exclusion envelope penetration, and novelty. pConf: This is a normalized score reflecting the degree to which a molecule that has been scored has a final optimal top pose that is highly similar to a single training molecule. Molecules with high confidence have at least one close near-neighbor in the training set. The subset of molecules with relatively high confidence (the default threshold is set at ≥ 0.35, but this is not a hard and fast value), usually have lower prediction error values. pExcl: This is a normalized score that measures whether a molecule in its final optimal pose protrudes beyond the full envelope of the training molecules. If the protrusion is small, a high pExcl value (≥ 0.95) might indicate that the molecule is sticking into space that has not been explored. pNov: This is the most complicated of the metrics. It is looking at what types of functionality have been explored by the union of the training molecules. So, it is possible to see a molecule with a low pConf value because there is not a single training

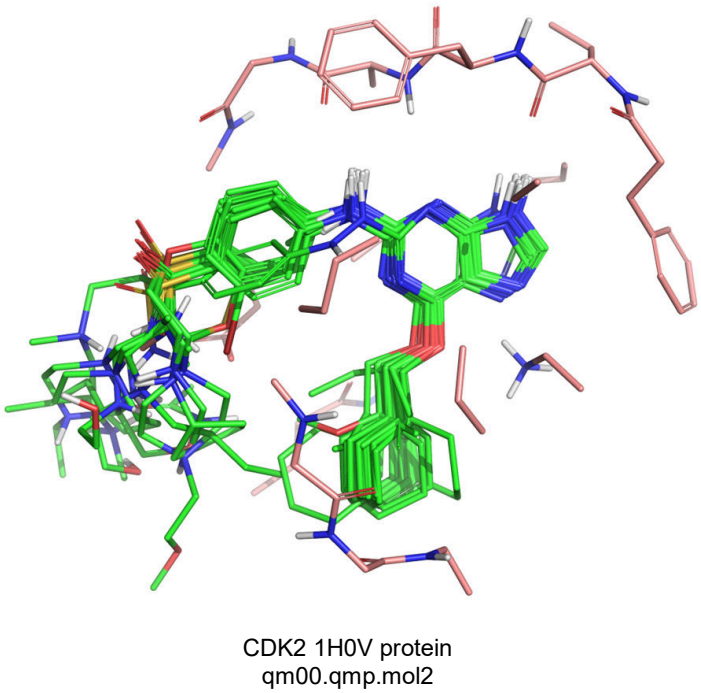


Figure 6.10 The final optimal poses of the training molecules (green) shown in the context of the CDK2 protein.

mol that looks like it. But the same molecule might score as having low novelty if, for example, the union of two training molecules covers it. The new molecule might have an acid on the left and a large hydrophobic chunk on the right. You might have training molecules that align well with it but where one has the acid and one has the grease (but not a single mol with both). Low novelty is considered to be ≤ 0.85. The most similar train molecule is in column 6 with the raw similarity score (scale 0–1) in column 7 (maxsim).

Figure 6.12 shows the top single pose of test molecule m29 (magenta) in the context of the final optimal pose of the nearest neighbor train molecule m42 (green). The core of the molecule is similarly situated to that of the training ligand m42. Of the five top-scoring test molecules, m45 was predicted with the highest confidence, which stemmed from high similarity to the optimal pose of training molecule m51. The molecules for which pNov < 0.85, pConf $>$ 0.35, and pExcl < 0.95 will tend to have the highest level of predictive accuracy. For the CDK2 example, as can be seen in qm00-test-report.txt, 12 of the 50 test molecules penetrated the space beyond the training ligand envelope and 7 molecules had high novelty. Nonetheless, as seen below, the test activity predictions were good with an average error less than 0.5 log.

```
# Directory: examples/quansa/cdk2
# File Contents: TestData
   m02 =   4.3
   m03 =   4.3
   ...
   m78 =   6.7
   m80 =   4.9

> sf-quansa.exe eval qm00-test-report.txt TestData qm00stats

# Key Output File: qm00stats-eval.txt

# File contents: qm00stats-eval.txt
```

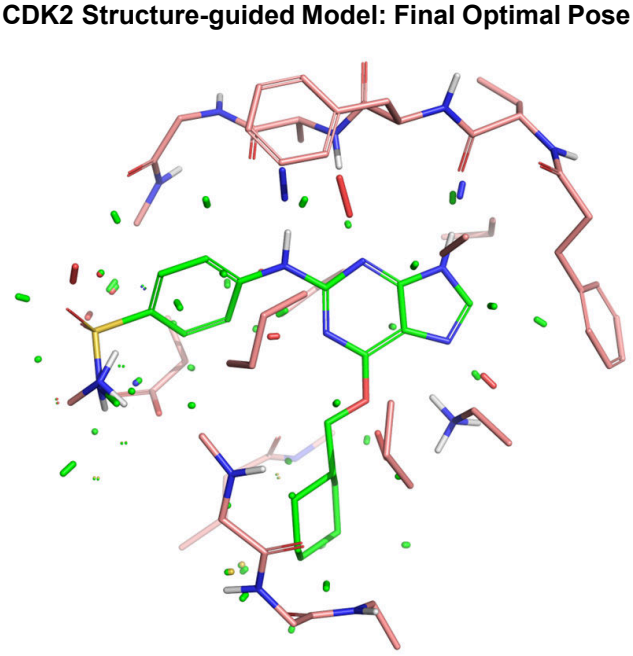


Figure 6.11 The final optimal pose of training molecule m42 (green) is shown in the context of CDK2 protein 1H0V (salmon). The m42 interaction sticks mimic the ligand interactions with CDK2 residues Leu83 and Glu81 in the hinge region, Lys33 of the catalytic triad, and Asp86 and Lys89.

```
...
Stats for all the mols.

Number of Mols 50
95.00% Confidence Interval define by CI_Low and CI_High, with 1000 resamples.
=================================
Stat     Value    CI_Low   CI_High
KTau     0.742    0.609    0.866
R        0.862    0.759    0.944
R2       0.743    0.577    0.892
AvgErr   0.445    0.315    0.592
RMSE     0.664    0.444    0.858
---------------------------------
Linear Fit: slope 1.039718 (xint 0.136753  yint -0.142184)
K_Tau pval = 0.00000, estimated using 100000 iterations.
=================================
...
```

```
# Look at the results from the cdk2 QuanSA example:
# Display the interactions with the pocket-field for a Test molecule
> sf-quansa.exe paint qm00 qm00-test-topfam-toppose.mol2 paint m29

# Visualize the model, the given-sguided poses, the pocket-field, and the poses of the test
    molecules.
> pym disp.pml
```

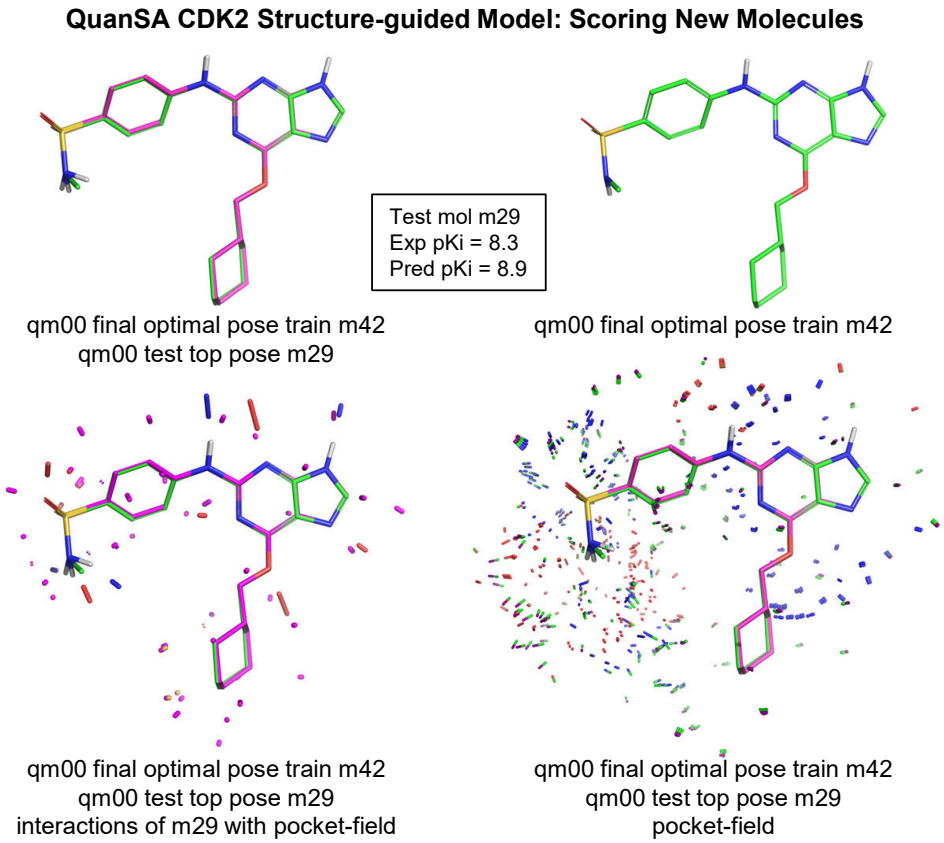


Figure 6.12 Test molecule m29 (magenta) was correctly identified as having relatively high activity. Here the top pose of m29 is shown with the most similar training molecule, m42 (green), with the m29 interactions with the pocket-field and in the context of the pocket-field.

6.10 ITERATIVE MODEL BUILDING AND REFINEMENT

Generally speaking, with molecules exceeding a probabilistic novelty of 0.85, inaccurate predictions should be expected. However, explicit inclusion of such molecules for model refinement has been shown to be helpful, especially with respect to increasing the structural diversity of highly active molecules over the course of iterative model-guided lead optimization [20]. Beginning with the structure-guided CDK2 model just constructed, we will see how to iteratively select new molecules and refine models over time, as new data are exposed. In this example, as seen from the information produced by the `select` procedure and through visualization of the chosen model optimal training ligand poses, we see that both model convergence and parsimony are good.

However, the model construction process is complex, and the endpoint sought involves a convergence criterion that requires that the scores for molecules in their *optimal* poses be close to the experimental values. Especially in cases where molecules are highly flexible, convergence may be difficult. In this example, the mean error of fit for the training molecules approached a level comparable to expected levels of assay noise, with extremely high Kendall's Tau. In cases where the combination of convergence and parsimony are not acceptable, models may be re-refined from the point at which they were evaluated using the `add` command. In cases where convergence with a particular alignment hypothesis remains elusive, the user is encouraged to consider alternate alignment hypotheses. Consideration may also be given to the possibility that some of the underlying assumptions for model building are not being met, particularly that the molecules are all binding in the same target pocket and that the activity values can be interpreted as being related to the free energy of binding.

Many factors generally influence the selection of molecules to synthesize and assay: quantitative estimates of activity, confidence, and novelty can all be brought into consideration. Of course, visual analysis of predicted binding modes, especially when structural variations begin to be substantial, also can be very helpful in making candidate selections. For the example being discussed, summary information is in `qm00-test-report.txt`, with the corresponding model and the corresponding optimal training poses in `qm00.qmp.mol2`. The top scoring pose families of the test molecules are in `qm00-test-topresults.mol2`. Iterative refinement of model 1 is discussed here, using the top 5 molecules (based on potency predictions alone) being selected for "synthesis" and "assay."

The following summarizes the iterative refinement process for one round, beginning with scoring all 50 molecules with the initial model:

```
# Directory: examples/quansa/cdk2

> cp TestNames qm00-round0-TestNames

# ROUND 0: TEST model against qm00-round0-TestNames (all 50 holdouts)
> sf-quansa.exe -namelist qm00-round0-TestNames score qm00 pqcdk2.sfdb qm00-test
> sf-quansa.exe eval qm00-test-report.txt TestData qm00stats

# The next commands grab the 5 best predicted mols, and make a new train and new test set
> tail -n +2 qm00-test-report.txt | sort -r -k 2 | awk '{print $1}' | head -5 >
    qm00-round0-winners
> grep -w -f qm00-round0-winners TestData > qm00-round0-NewTrainData
> grep -w -f qm00-round0-winners TestNames > qm00-round0-NewTrainNames
> grep -w -v -f qm00-round0-winners qm00-round0-TestNames > qm00-round1-TestNames

# Now REFINE qm00 using the 5 predicted winners --> qm00r1
> sf-quansa.exe add qm00 qm00-round0-NewTrainData pqcdk2.sfdb qm00r1

# ROUND 1: TEST model against qm00-round1-TestNames
> sf-quansa.exe -namelist qm00-round1-TestNames score qm00r1 pqcdk2.sfdb qm00r1-test
> sf-quansa.exe eval qm00r1-test-report.txt TestData qm00r1stats

# The next commands grab the 5 best predicted mols, and make a new train and new test set
> tail -n +2 qm00r1-test-report.txt | sort -r -k 2 | awk '{print $1}' | head -5 >
    qm00-round1-winners
> grep -w -f qm00-round1-winners TestData > qm00-round1-NewTrainData
> grep -w -f qm00-round1-winners TestNames > qm00-round1-NewTrainNames
> grep -w -v -f qm00-round1-winners qm00-round1-TestNames > qm00-round2-TestNames

# Now REFINE qm00r1 using the 5 predicted winners --> qm00r2
> sf-quansa.exe add qm00r1 qm00-round1-NewTrainData pqcdk2.sfdb qm00r2

# ROUND 2: TEST model against qm00-round2-TestNames
> sf-quansa.exe -namelist qm00-round2-TestNames score qm00r2 pqcdk2.sfdb qm00r2-test
> sf-quansa.exe eval qm00r2-test-report.txt TestData qm00r2stats

# The next commands grab the 5 best predicted mols, and make a new train and new test set
> tail -n +2 qm00r2-test-report.txt | sort -r -k 2 | awk '{print $1}' | head -5 >
    qm00-round2-winners
> grep -w -f qm00-round2-winners TestData > qm00-round2-NewTrainData
> grep -w -f qm00-round2-winners TestNames > qm00-round2-NewTrainNames
> grep -w -v -f qm00-round2-winners qm00-round2-TestNames > qm00-round3-TestNames

# Now REFINE qm00r2 using the 5 predicted winners --> qm00r3
> sf-quansa.exe add qm00r2 qm00-round2-NewTrainData pqcdk2.sfdb qm00r3

# ROUND 3: TEST model against qm00-round2-TestNames
> sf-quansa.exe -namelist qm00-round3-TestNames score qm00r3 pqcdk2.sfdb qm00r3-test
> sf-quansa.exe eval qm00r3-test-report.txt TestData qm00r3stats

# The next command grabs the 5 best predicted mols
> tail -n +2 qm00r3-test-report.txt | sort -r -k 2 | awk '{print $1}' | head -5 >
    qm00-round3-winners
```

```
> cat qm00-round[0123]-winners > qm00-all-winners
> grep -w -f qm00-all-winners TestData | awk '{print $3}' > qm00-all-selected-vals
> grep -w -v -f qm00-all-winners TestData | awk '{print $3}' > qm00-all-nonselected-vals
> cat TestData | awk '{print $3}' > qm00-all-vals

# This will generate performance plots:
> source RunPlot0
```

Scripts to run the refinement process automatically for three initial alignments are in *RunBuildTestRefine[012]*. The process of selection and refinement shown above was repeated, producing progressively refined models, with the final model containing a total of 20 original test molecules being selected, five from each of four successive models. Figure 6.13 shows the behavior of the original QuanSA model on the 50 test ligands (Round 0) and Rounds 1, 2, and 3 of refinement testing on the remaining 45, 40, and 35 test molecules, respectively. As seen in the `qm00*stats-eval` output files, all four models produced high rank correlation performance and had absolute prediction error of roughly 0.4 pk_i units. Note that all inhibitors from the 50 initial holdouts with activity > 7.0 were correctly selected by the modeling process.

Figure 6.13 shows plots of the performance of the initial QuanSA model (top left) along with the performance of three iteratively refined models (procedure discussed below). We see that the predicted top-scoring molecules are part of a cluster of nominally overpredicted molecules that are all relatively high in activity. The following discussion shows repeated selection of the top five predicted molecules for incorporation into newly refined models.

Figure 6.14 shows the distributions of the 20 *selected* (top five for each model) versus the 30 *unselected* molecules from the set of 50 for iterative refinement. The distribution of experimental activities are shown in green for selected inhibitors and purple for unselected ones. The distributions reflect selection of the top five predicted molecules for each of the four models. Selection of an additional five molecules from the final model would identify all but one of the inhibitors with sub-micromolar inhibition constants.

6.11 USING ADDITIONAL INFORMATION TO INFLUENCE A MODEL

There are cases where more might be known about either the ligands to be modeled or the target itself than illustrated above. As with the Docking and Similarity modules, torsional and positional constraints can be made explicitly using the `-torcon` and `-poscon` options, along with parameters that modify the strength of the constraints and the amount of wiggle for each type of constraint. Here, as with the other modules, the Tools module `forcegen` procedure should be used *prior* to making use of these options within QuanSA.

Using such constraints can help overcome limitations in purely agnostic similarity-based alignment procedures and help to test specific hypothesis about the relationship between molecular structural variation and activity. It is important, however, for users to bear in mind that, even though one's intuition might be that something binds "this" way, it may actually bind in a different orientation depending on its substituents.

6.12 LARGE DATA SETS AND SKEWED ACTIVITY DATA

The foregoing simple serotonin example used a completely automated method for alignment generation, and models were built from all alignments that were produced. In real-world applications where no information exists to help define desirable alignment hypotheses, one must take a brute-force approach that relies on the molecular similarity engine to identify alternative alignments. In such cases, it is strongly recommended that the user split any initial set of molecules into separate training ligands and *holdout* ligands to aid in model selection (as described above).

Take for example, a lead optimization experiment that resulted in a set of 2000 molecules with an overall activity range of pK_i 4.0–9.0, but where the activity distributions is over-represented in the 6.0–7.0 range. There are three issues. First, constructing a QuanSA model with 2000 training ligands is not currently feasible, so a model must be built incrementally. Second, one needs to identify an informative subset from which to begin the training process. Third, one needs to refine the model in order to cover the larger set of 2000 molecules.

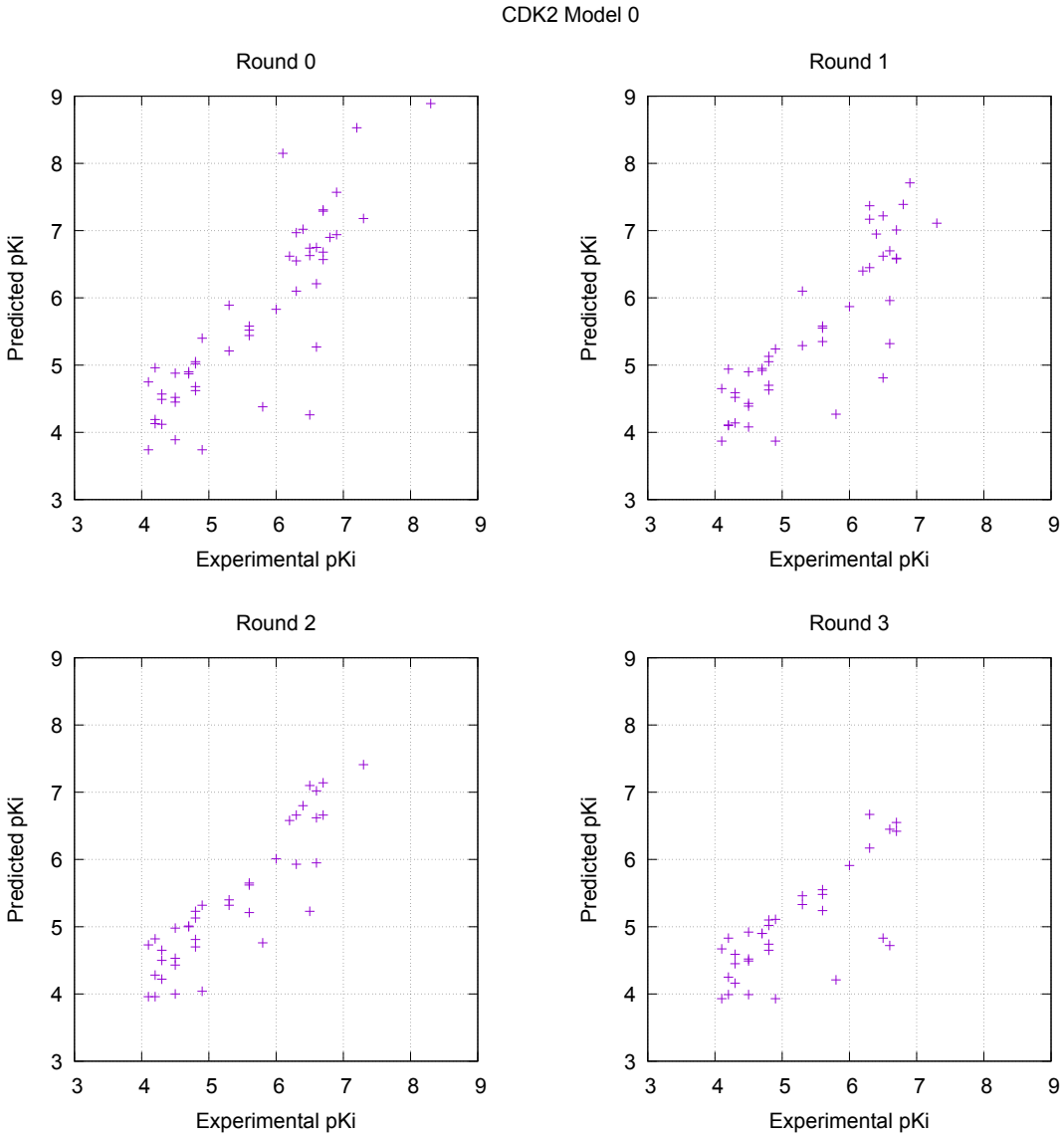


Figure 6.13 The original QuanSA CDK2 model and rounds 1–3 of the iterative QuanSA procedure produced solid correlations between predicted and experimental activities and identified the potent molecules.

For the initial model, 200 molecules is a reasonable number for constructing the training and holdout sets. Random selection of the set of 200 risks having many molecules with similar activity values, which provides little discriminatory power for QuanSA. A better selection procedure involves binning. For this example, given the 5 logs of activity range, there would be 5 bins of molecules based on activity (4-5, 5-6, 6-7, 7-8 and 8-9). *Within* each bin, random selection of 40 molecules would yield an overall subset of 200.

This 200 molecule set would be split 2/3 to 1/3 into 134 for training and 66 for a holdout model selection set. The set of 200 should be sorted by activity highest to lowest and every 3rd molecule moved to the holdout set. The resulting 134 molecule training set and 66 molecule holdout set would therefore have similar activity distributions. Multiple QuanSA models would be built and then `select` would be used to select the best models. The remaining 1800 molecules would be scored using the best models and perhaps 50 of those molecules with the largest errors could be used to refine the models. The premise of this refinement protocol is that the largest errors likely reveal structures

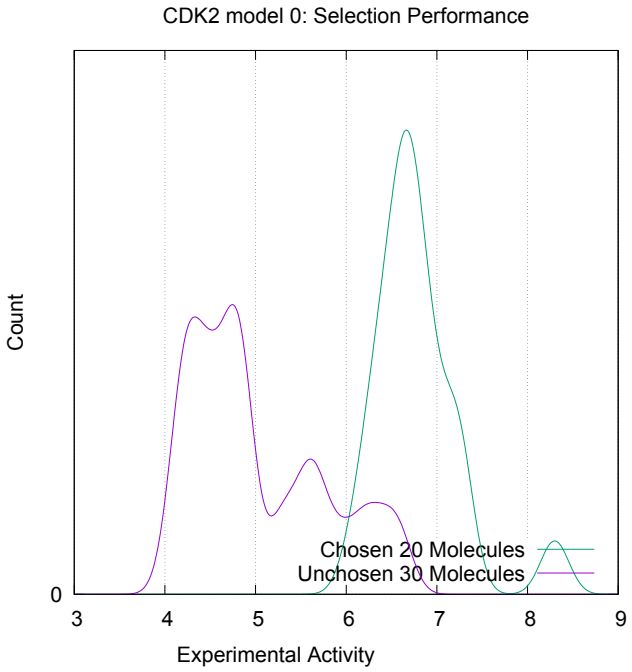


Figure 6.14 The distributions of selected (green, 20 molecules) versus non-selected (purple, 30 molecules) molecules were very different.

not well represented in the original model, but when added during model building result in highly predictive models. Iteration of this procedure would produce a model that covers the data within the 2000 molecule series, but which relies on only a few hundred to do so.

6.13 GABA$_A$R BENZODIAZEPINE BINDING SITE EXAMPLE

The example presented here utilizes a benzodiazepine (BZR) dataset from the classic QSAR benchmark of Sutherland et al. [24]. In the previous chapter on similarity, the entire set of 147 BZR ligands was aligned to a 4-molecule mutual alignment using the `esim_list` command as shown in Figure 4.8. Here, the dataset is used as the original 98 training ligands and 49 test ligands, specifically for the goal of purely ligand-based affinity prediction in a multi-scaffold case. BZR was one of many target sets in the introductory QuanSA paper used to show the accuracy and broad applicability of the method [1]. BZR ligands are agonists of the GABA$_A$ receptor, a complex, membrane-bound, hetero-multimeric ligand-gated ion channel. No crystal structures of this target were used in these purely ligand-based QuanSA models.

```
# Directory: examples/quansa/bzr/
# See file:  RunAll

# Protonate the molecules sensibly
# sf-tools.exe prot bzr-3dqsar-mndo-train.mol2 train
# sf-tools.exe prot bzr-3dqsar-mndo-test.mol2 test

# Use ForceGen to search the train/test sets
# sf-tools.exe -pquant forcegen train.mol2 pqtrain
# sf-tools.exe -pquant forcegen test.mol2 pqtest

# copy the bzr multiple ligand alignment
# cp ../../similarity/multiple_alignment/bzr/good-align.mol2 .

# We can drive the alignment toward a particular core alignment
```

```
> sf-quansa.exe -clknown good-align.mol2 init TrainData pqtrain.sfdb qm

# OR, we can just let things run de novo
# sf-quansa.exe init TrainData pqtrain.sfdb qm

# Build the models and then score the blind test molecules
> sf-quansa.exe build qm-init-00 qm00
> sf-quansa.exe score qm00 pqtest.sfdb test00
> sf-quansa.exe eval test00-report.txt TestData qm00

# Repeat for other initial alignments 01 to 03 ...

> sf-quansa.exe build qm-init-04 qm04
> sf-quansa.exe score qm04 pqtest.sfdb test04
> sf-quansa.exe eval test04-report.txt TestData qm04

> sf-quansa.exe select ModelList sel

# Display the pocket-field
> sf-quansa.exe disp qm04 disp04

# Display the interactions with the pocket-field for a Test molecule
> sf-quansa.exe paint qm04 test04-topfam-toppose.mol2 paint Ro11-1465
```

A multiple ligand alignment of 4 topologically-diverse training ligands (`good-align.mol2`) was performed to seed the initial alignments of the full set of training ligands. Five QuanSA models were built and then used to score the test ligands. The selection process showed model `qm04` with the highest normalized probability score (see `sel-selectreport.tab`).

Figure 6.15 shows the initial multiple ligand alignment (cyan), and the final poses of the full set of training molecules (green) alone and with the pocket-field. QuanSA learns from activity data, and during the learning process, the alignment of the training molecules is optimized according to the evolving model. So, the final optimal ligand alignments are different than when using similarity alone.

Also shown in Figure 6.15 is the predicted pose of test molecule Ro11-1465 (magenta) relative to the pose of a training molecule with similar activity, Ro16-4019 (green). The sticks represent the interactions of Ro11-1465 with the pocket-field both in magnitude and in type of interaction. QuanSA yielded statistically significant predictions for both the in-model test subset and the full blind test. As seen in `qm04-eval.txt`, for the in-model and full test set predictions, respectively, Kendall's Tau was 0.69 and 0.52, and MAE was 0.41 and 0.56.

Another BZR test set was curated from a large number of BZR ligands (n=1158) in ChEMBL. Although the BZR QuanSA model was constructed from canonical BZR ligands such as diazepam and alprazolam, the model was predictive on for example, pyrazolo-pyridine esters, a significant scaffold leap [1].

6.14 CONCLUSION

The key considerations in employing QuanSA predictive modeling revolve around the relationship between the physical models that are constructed and the degree to which those models represent a reasonable approximation to reality. When choosing from among different initial molecular alignment hypotheses, one should make use of quantitative measurements like the hypothesis scores or the convergence/parsimony of the derived models. But, to the extent that a user may have additional information, it can be very helpful to use it. For example, the known bound configuration of a particular inhibitor may provide an excellent hint as to which of multiple hypotheses is sensible (rigid alignment and similarity scoring to the different alignment hypotheses can be used to assess this). If a modeled structure of the protein site is available, multiple variants can be used as the target of docking in order to establish whether an ensemble of poses of one's training ligands looks like a particular alignment hypothesis. In the case of congeneric molecular series, where extrapolation beyond the series is not required, these choices become somewhat less important. However, when generalization and SAR transfer to new chemical series matters, the closer the QuanSA model approaches the correct physical configuration, the better the performance will be.

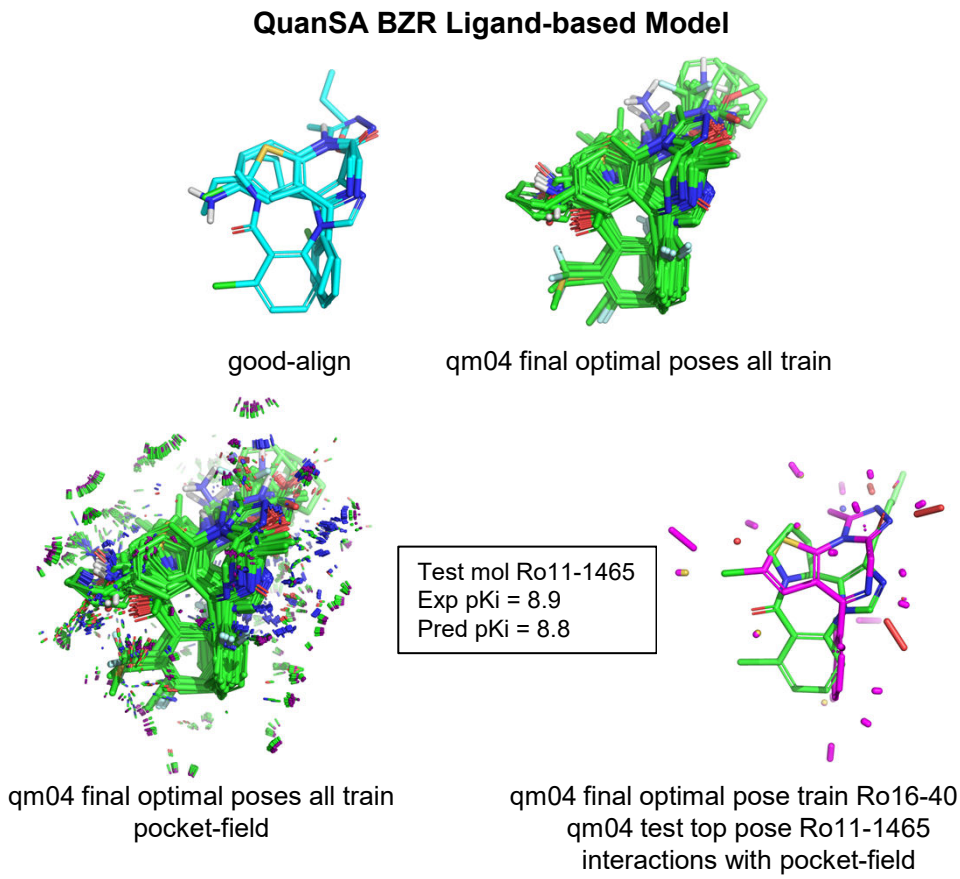


Figure 6.15 Upper left, the multiple ligand alignment `good-align.mol2` (cyan) that was used to seed the alignment of the training molecules. Upper right and lower left, the final poses of the full set of training molecules (green) alone and in the context of the pocket-field, respectively. Lower right, the predicted pose of test molecule Ro11-1465 (magenta) relative to the final pose of a train molecule with similar activity, Ro16-4019 (green). The sticks show the interactions with the pocket-field.

The process of model construction, testing, and refinement requires very simple commands from the user to be run in a default mode, without requiring much in the way of choice from a user. However, it is possible for a user to make use of more complex workflows that combine aspects of known or modeled protein structures as well. The goal with the QuanSA approach is to make use of as much information as is known in order to produce models that approach the causal underpinning of the binding events that lead to molecular activity. The extent to which a user is able to meet this goal will influence the performance of the resulting models.

6.15 RECOMMENDED QUANSA OPTIONS

The following covers the QuanSA options that are recommended for user variation. Each of the options is accessed by specifying the option to one of the core QuanSA commands.

For the `init` command, the following options are useful in controlling the initialization process:

-clnmols <value>: Sets the number of molecules to select for core multiple-alignment (default = 10).

-clselwin <value>: Activity window from most active mol for which to select first N mols (default = 2.5).

-clrms <value>: During initial alignment clique generation, this value controls how close alternative pose groups must be in order to be considered separate. Increasing this value will produce a smaller number of cliques with coarser structural variation (default = 0.1).

-clknown <*poses.mol2*>: Used with a multi-mol2 file that contains molecular poses to guide the construction of the resulting full cliques. This can be used to specify, for example, a known bound pose of a ligand or many plausible docked alternatives. Also, it may be used to specify a carefully worked out small set of poses from detailed multiple alignment work within the Similarity module (default = none).

-clkthresh <value>: Sets the eSim threshold for similarity to the known poses (default = 6.5).

-clkmaxn <value>: Sets the maximum number of core poses in the first round of alignment against known poses (default 20).

-clnmake <value>: Sets the maximum number of final initialized alignment variations (default 5).

-compress <value>: Sets the number of poses for align targets (default = 50).

For the following options, the parameter options are *inherited* from the `init` command onward. So, specifying something like "-torcon common_frag.mol2 -poscon warhead.mol2" to the `init` command will carry the parameter specification through subsequent commands such as `build` and `score`.

-torcon <frags.mol2>: This has been described in the previous chapters along with the `-torpen` and `-twiggle` parameters.

-poscon <frags.mol2>: This has been described in the previous chapters along with the `-pospen` and `-pwiggle` parameters.

-namelist <value>: List of molecule names for the `score` procedure (default: full sfdb).

-assay_delta <value>: Generally, all biological assays have some level of noise, and attempting to fit a model to precise values may hamper model convergence. Values of 0.1–0.3 are typical for this parameter (the default is 0.1).

-workdir directory-name: Directory in which all QuanSA working files are contained during long procedures.

-nthreads(*): Maximum number of threads to use. This affects initial alignment, model induction, model scoring, and all other aspects of QuanSA use.

Bibliography

1. Ann E Cleves and Ajay N Jain. Quantitative surface field analysis: Learning causal models to predict ligand binding affinity and pose. *Journal of Computer-Aided Molecular Design*, 32(7):731–757, 2018.
2. Ann E Cleves, Stephen R Johnson, and Ajay N Jain. Synergy and complementarity between focused machine learning and physics-based simulation in affinity prediction. *Journal of Chemical Information and Modeling*, 61 (12):5948–5966, 2021.
3. A. N. Jain. Morphological similarity: A 3D molecular similarity method correlated with protein-ligand recognition. *J Comput Aided Mol Des*, 14(2):199–213, 2000.
4. A. N. Jain. *Chemical Analysis by Morphological Similarity (US Patent 6470305)*. 2002.
5. A. N. Jain. Ligand-based structural hypotheses for virtual screening. *J Med Chem*, 47(4):947–61, 2004.
6. A. N. Jain. Virtual screening in lead discovery and optimization. *Curr Opin Drug Discov Devel*, 7(4):396–403, 2004.

7. A. E. Cleves and A. N. Jain. Robust ligand-based modeling of the biological targets of known drugs. *J Med Chem*, 49(10):2921–38, 2006.
8. A. E. Cleves and A. N. Jain. Effects of inductive bias on computational evaluations of ligand-based modeling and on drug discovery. *J Comput Aided Mol Des*, 22(3-4):147–59, 2008.
9. E. R. Yera, A. E. Cleves, and A. N. Jain. Chemical structural novelty: On-targets and off-targets. *J Med Chem*, 54(19):6771–85, 2011.
10. A.E. Cleves and A.N. Jain. Chemical and protein structural basis for biological crosstalk between PPARα and COX enzymes. *Journal of Computer-Aided Molecular Design*, 29(2):101–112, 2014.
11. A. N. Jain, T. G. Dietterich, R. H. Lathrop, D. Chapman, Jr. Critchlow, R. E., B. E. Bauer, T. A. Webster, and T. Lozano-Perez. A shape-based machine learning tool for drug design. *J Comput Aided Mol Des*, 8(6):635–52, 1994.
12. A. N. Jain, K. Koile, and D. Chapman. Compass: Predicting biological activities from molecular surface properties. performance comparisons on a steroid benchmark. *J Med Chem*, 37(15):2315–27, 1994.
13. A. N. Jain, N. L. Harris, and J. Y. Park. Quantitative binding site model generation: Compass applied to multiple chemotypes targeting the 5-HT1a receptor. *J Med Chem*, 38(8):1295–308, 1995.
14. A. N. Jain. Scoring noncovalent protein-ligand interactions: A continuous differentiable function tuned to compute binding affinities. *J Comput Aided Mol Des*, 10(5):427–40, 1996.
15. T.G. Dietterich, R.H. Lathrop, and T. Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1-2):31–71, 1997.
16. T. A. Pham and A. N. Jain. Parameter estimation for scoring protein-ligand interactions using negative training data. *J Med Chem*, 49(20):5856–68, 2006.
17. T. A. Pham and A. N. Jain. Customizing scoring functions for docking. *J Comput Aided Mol Des*, 22(5):269–86, 2008.
18. J. J. Langham, A. E. Cleves, R. Spitzer, D. Kirshner, and A. N. Jain. Physical binding pocket induction for affinity prediction. *J Med Chem*, 52(19):6107–25, 2009.
19. A. N. Jain. QMOD: Physically meaningful QSAR. *J Comput Aided Mol Des*, 24(10):865–78, 2010.
20. Rocco Varela, W Patrick Walters, Brian B Goldman, and Ajay N Jain. Iterative refinement of a binding pocket model: Active computational steering of lead optimization. *Journal of medicinal chemistry*, 55(20):8926–8942, 2012.
21. A.N. Jain and A.E. Cleves. Does your model weigh the same as a duck? *J Comput Aided Mol Des*, 26:57–67, 2012.
22. A.E. Cleves and A.N. Jain. Extrapolative prediction using physically-based QSAR. *Journal of Computer-Aided Molecular Design*, 30(2):127–152, 2016.
23. R. Varela, A.E. Cleves, R. Spitzer, and Ajay N Jain. A structure-guided approach for protein pocket modeling and affinity prediction. *Journal of Computer-Aided Molecular Design*, 27(11):917–934, 2013.
24. Jeffrey J Sutherland, Lee A O'Brien, and Donald F Weaver. A comparison of methods for modeling quantitative structure- activity relationships. *Journal of Medicinal Chemistry*, 47(22):5541–5554, 2004.

CHAPTER 7

ADVANCED APPLICATIONS

This chapter covers advanced applications, including bound ligand strain estimation from real-space refined conformer ensembles or from predicted bound poses, use of torsional restraints to guide the conformational search of complex macrocycles, and prediction of protein-ligand binding affinities based on a combination of strain estimation and intermolecular binding enthalpy.

7.1 XGEN PYMOL INTERFACE: REAL-SPACE LIGAND REFINEMENT

Real-space refinement of ligands within the electron density of protein-ligand complexes offers a means to quantitatively explore the conformational ensembles that give rise to the *snapshots* that are captured in diffraction experiments. It is strongly recommended that users read the extensive paper [1] that introduced the xGen method as well as the chapter focused on xGen usage from the command-line prior to working through this section. The prior work introducing ForceGen is also recommended [2, 3], as is the more recent work involving explicit modeling of bound ligand strain [4, 5]. The algorithms are detailed in the referenced papers, and the PyMOL interface has been implemented to simplify the real-space refinement process.

The five steps for refinement will be briefly summarized here:

1. Idealized density: Experimental density is converted into a spherical Gaussian approximation for the region of space that includes the reference ligand's Van der Waals volume plus a buffer zone of 0.5Å.

2. Restrained conformational search: The normal ForceGen conformational search procedure is carried out beginning with the ligand's reference coordinates. The MMFF94sf force field is augmented with a reward for overlap between the ligand's density function $L(x, y, z)$ and the idealized experimental density $D(x, y, z)$ (see Figures 5.1 and 5.2).

3. High-quality trio generation: Each conformer resulting from the search is re-minimized: a) under a condition in which the density overlap is strongly weighted; and b) with no density overlap reward but with a square-welled quadratic positional restraint. The three pools of conformers (high density weighting, blended weighting from the thorough conformer search, and minimized under positional constraint) are used to find high-quality trios that are characterized by high congruence to electron density and by low energy.

4. Ensemble generation: Conformers from the trios are used to construct occupancy-weighted ensembles that minimize real-space R. Calculations are done using $L(x, y, z)$ for real-space electron density (the fast approximation to ρ_{calc}) and using the full experimental density sampled on a 0.25Å grid.

5. Fit evaluation: Final statistics for real-space correlation coefficient and real-space R (RSCC and RSR, see Experimental Section) are made by comparing the 0.25Å grid-sampled experimental density to the density derived from the ligand (or ligand ensemble). For PDB reference ligand coordinates, this is done using exact, as-deposited, atom-specific B-factors, the resolution of the diffraction data, and the standard truncated Fourier approach with fitted scattering factor functions for ρ_{calc}. For xGen ensembles, the same procedure is followed, except that a constant, grossly estimated, B-factor is used for all atoms of all conformers within an ensemble (B-factor optimization does not enter into any aspect of deriving an xGen ensemble).

The entire procedure has been implemented within a PyMOL GUI. The xGen PyMOL GUI is implemented using the PyMOL plugin interface. Installation is done via the menu [Plugin/Plugin Manager/Install New Plugin], clicking [Choose File...], then navigating to select the file `bin/pymol/sf_xgen_gui/__init__.py`, and finally accepting all defaults as to installation location. The following assumes that the user has a valid license to the incentive version of PyMOL (which supports key aspects of crystallographic file reading and processing) and has installed the xGen plugin.

NOTE: The xGen GUI was initially developed for PyMol version 2.4. Later versions (including the v2.6 LTS PyMol release), contain a misnamed DLL on Windows. The program `mtz2ccp4_px.exe` requires an Intel FFT library and looks for `mkl_rt.dll`. Unfortunately, the PyMol distribution's updated library is named `mkl_rt.2.dll`. Attempting to load an MTZ file into PyMol results in nothing happening. This can be fixed by navigating to the following folder in the PyMol installation: `Schrodinger/PyMOL2/Library/bin/` and renaming `mkl_rt.2.dll` to `mkl_rt.dll`.

One can test the fix by loading an MTZ file into PyMol and looking for the omit maps that are provided in standard PDB-deposited structures. Another method is to execute the `mtz2ccp4_px` command in a shell/console window, as follows to see a usage message:

```
> cd ~/AppData/Local/Schrodinger/PyMOL2/Library/bin
> ./mtz2ccp4_px.exe
Usage: mtz2ccp4_px.exe <space_group> <a> <b> <c> <alpha> <beta> <gamma> <reso_high> ...
```

This DLL naming issue appears to be in multiple PyMol releases post v2.4.

7.1.1 Acquisition and preparation of input structures for xGen

The steps for ligand refinement from the command-line were covered in Chapter 5. Here, we will illustrate use of the xGen PyMOL interface using the 3SUE example from a paper devoted to the question of bound ligand strain [5]. 3SUE is hepatitis C virus NS3/4A protease bound to the inhibitor grazoprevir.

This procedure begins with two simple line commands in order to acquire the co-crystal structure and generate the correctly protonated protein and PDB deposited crystallographic ligand. The file `verifypdb.smi` is present in the directory so that PDB ligand quality checks occur. The density map in MTZ format is downloaded.

```
# Directory: examples/advanced/xgen-pymol/

> sf-dock.exe getpdb PDBList trg
> source trg-script

# visualize the prepared protein and ligand
```

```
> pym trg-pro-3SUE.mol2 trg-lig-3SUE-SUE.mol2

# Download the Map Coefficients in MTZ format at https://www.rcsb.org/structure/3SUE

# KEY FILES:
#   3SUE.pdb                  co-crystal structure in pdb format
#   trg-pro-3SUE.mol2         correctly protonated crystallographic protein
#   trg-lig-3SUE-SUE.mol2     correctly protonated crystallographic ligand
#   3sue_phases.mtz           density map coefficients
```

7.1.2 xGen PyMOL GUI

The xGen PyMOL GUI has simplified the five steps described above to generate low energy conformer ensembles that fit X-ray data. Figures 7.1 - 7.6 show the successive steps within the GUI and the associated output. The red arrows and green boxes highlight the button to click and files to be chosen at each step. The successive output automatically appears in the PyMOL session, and the session is saved as a .pse file. The small inset boxes in the Figures show the objects being displayed in the PyMOL window.

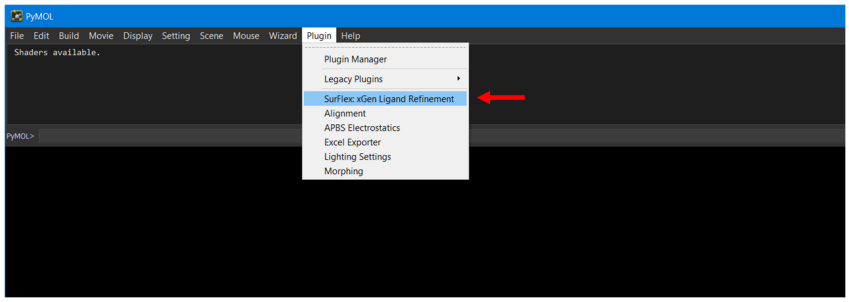
```
# Directory: examples/advanced/xgen-pymol/
> pym

# GUI Ensemble Generation Steps:
#   1:Build Density Box            move from Fourier space into real space
#   2:Calculate Idealized Density  generates idealized contours
#   3:Run xGen Search              generates trios
#   4:Run Ensemble Generation      generates final ensemble
#   5:Real-Space Fit Evaluation    RSCC and RSR of the ensemble vs xtal ligand
```

Figure 7.1 shows the xGen PyMOL GUI setup. The user launches a PyMOL session from the command line within the specified directory. From the PyMOL Plugin menu, `Surflex: xGen Ligand Refinement` is chosen and then the `Select Folder` button sets the current space as the working directory. Once the directory is chosen, the xGen GUI opens on the `xGen Density Setup` tab.

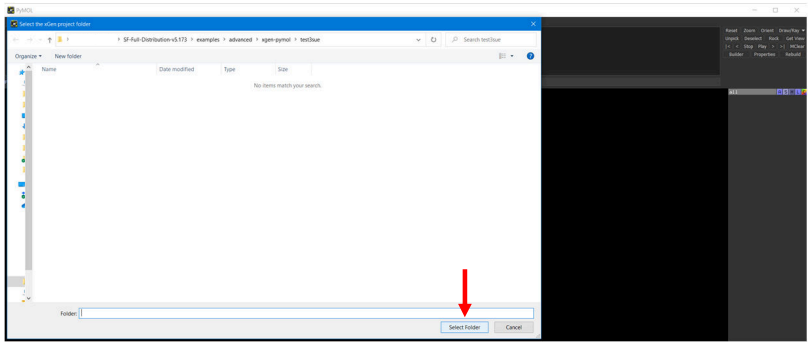
Figure 7.2 shows Step 1, the move from Fourier space into real space. The user browses and selects the protein (`trg-pro-3SUE.mol2`), the ligand (`trg-lig-3SUE-SUE.mol2`), and the density map (`3sue_phases.mtz`). The resolution should be changed from the default value (2.00Å) to the resolution provided by the PDB annotation for 3SUE (2.20Å). In this example and for many structures, the carve can be left at the default value of 5. Once these choices are set, the user selects `Build Density Box` and a command prompt window launches and the process runs. The experimental density is converted into a spherical Gaussian approximation for the region of space that includes the reference ligand's Van der Waals volume plus a buffer zone of 0.5Å. Upon completion of this step, the output raw contour appears in the PyMOL window.

Figure 7.3 shows Step 2, generation of the idealized density. The default choices are accepted: carve 0.5, outprefix xg, and contour box checked. The user selects `Calculate Idealized Density` and a command prompt window launches and the process runs (this can take several minutes, depending on hardware, for the 3SUE example). The contour files may be visualized in order to see the concordance of the experimental and idealized density functions. There can be missing areas in the contours, where there should be ligand or protein atoms but where no density appears, due to a lack of experimental density. These contours are useful in visualizing the output ligand ensembles from xGen ligand refinement and fitting procedures.

Figure 7.4 shows Step 3, generation of the conformer trios, which are, in turn, used to build conformer ensembles to fit the idealized X-ray density. The `xGen Search+Ensembles` tab will automatically open at the completion of Step 2. The default values for the Constrained Conformer Search can be accepted, most importantly the Starting Ligand Conformer (`trg-lig-3SUE-SUE.mol2`) and the mode of search depth (`pquant`). The user should select `Run xGen Search`, and a command prompt window launches and the process runs (a couple of minutes for the 3SUE example). The output of this initial search process are conformer trios, where each conformer trio includes a high density overlap conformer (biased towards fitting X-ray density), a low energy conformer (biased toward the MMFF94sf force-field,



Open the xGen GUI on the Plugin menu



Select the working directory

Figure 7.1 xGen setup involves opening the PyMOL GUI Plugin and selecting the working directory.

and a conformer with a blend of X-ray and energy bias. In the PyMOL window, the user can visualize the trios by expanding using the +Trios option in the object control panel. Three sets labeled Density-Biased, Blended, and Energy-Biased appear in the PyMOL object list.

Figure 7.5 shows Step 4, generation of the xGen ensemble. Again, the auto-filled values are accepted, the user selects `Run Ensemble Generation` and a command prompt window launches and the process runs (a few minutes for the 3SUE example). The central goal of the xGen refinement procedure is to replace a single conformer with optimized atom-specific B-factors (the input approximate ligand pose) with a conformer *ensemble* that makes use of a constant B-factor for the entire ensemble. This final step identifies the ensemble that minimizes a fast calculation of real-space R. The user may choose whether to use "strict" or "diverse" ensembles, with the former making use of smaller numbers of conformers, possibly at the expense of a slightly poorer fit to X-ray density.

Figure 7.6 shows Step 5, the real space fit evaluation. The statistical evaluation is the one of the most complex of the xGen procedures, as it must make use of atom-specific and resolution-specific truncated Fourier expansions of atomic scattering factor functions in order to accurately calculate real-space density for xGen-style ensembles and traditional PDB ligand representations. For the xGen ensembles, an approximate B-factor is identified during the evaluation process (from 10–100 in steps of 5). For specified ligand ensembles whose atomic coordinates exactly match ATOM and HETATM records within the given PDB file, those B-factors will be used.

RSCC, for a number of reasons, is more reliable than RSR in meaningfully assessing xGen ensembles compared with traditionally optimized ligand models: 1) RSCC is not sensitive to electron density scaling; 2) RSR is often overfit in a highly atom-centric manner (e.g. a single aryl halogen atom often has a very high B-factor when the remainder of the arene does not); and 3) neither xGen nor traditional refinement protocols directly optimize RSCC, but both (to some extent) optimize functions related to RSR.

During the `Run Eval` procedure, a command prompt window launches and near the end of the process, a 2nd command prompt window launches. For the 3SUE example, the process completed in ~15 min and the RSCC for the xGen ensemble was 0.9470 compared to 0.9389 for the PDB deposited crystallographic ligand. The RSR for the

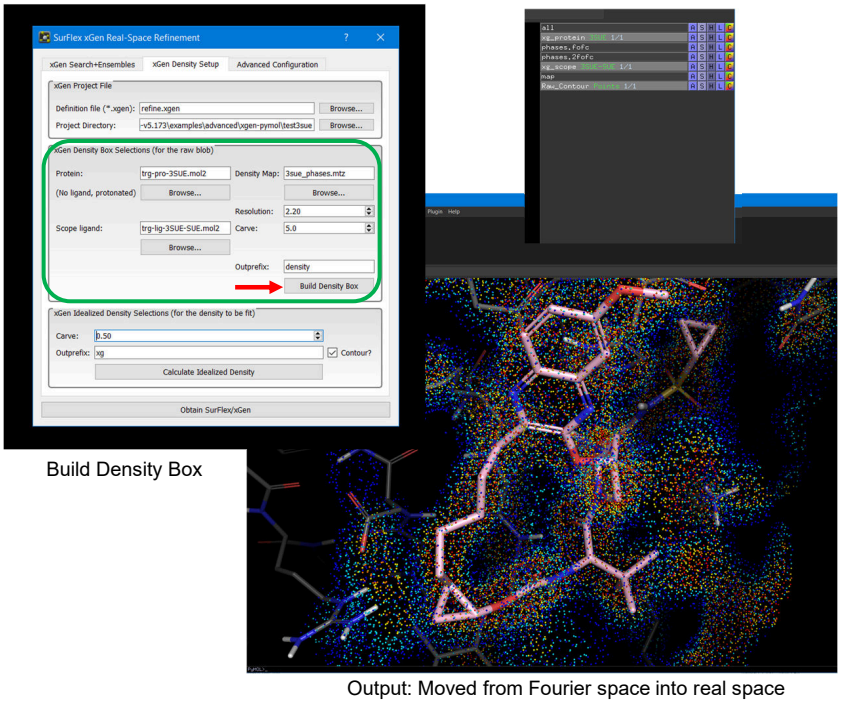


Figure 7.2 Step 1 of xGen ensemble generation: Move from Fourier space into real space.

xGen ensemble was 0.1402 compared to 0.1432 for the PDB coordinates. Thus, the xGen ensemble fit the density marginally better than the deposited ligand, and as will be shown below, was significantly lower in strain energy.

Figure 7.7 shows the selecting of specific conformers for real space fit evaluation. After calculation of RSCC and RSR for the xGen ensemble, the user can scroll down and see a list of conformers along with the density occupancy, strain energy, and density fit. The user may have an interest in a subset of the conformers. For clarity, selected conformers are shown in the PyMOL window.

Figure 7.8 shows the real space fit evaluation for 3 selected conformers of the xGen ensemble. The RSCC for the 3 selected xGen conformers was 0.9478 compared to 0.9417 for the PDB coordinates. The RSR for the xGen conformers was 0.1337 compared to 0.1372 for the PDB coordinates. Thus, the 3 xGen conformers fit the density better than the deposited ligand.

Figure 7.9 shows the final xGen ensemble (orange) with the original crystallographic ligand (cyan) and protein (gray). Many of these visual depictions are not standard in crystallography workflows, but they have proven useful in understanding why a particular conformer or ensemble is either nominally better or worse than another. Here, it is useful to see that the 3SUE density accommodates conformational variation in the ligand.

The primary operations performed in this example demonstrate the xGen PyMol interface. The xGen GUI was used to: (1) build the density box, (2) calculate the idealized density, (3) run the xGen conformer search, (4) generate ensembles, and (5) evaluate the real-space fit.

An often important aspect of re-fitting ligands is the question of ligand strain. The final recommended step in the xGen process is to compare the bound conformational energy of the re-fit ensemble and the original PDB ligand. First, estimate the bound energy of the re-fit ensemble. The input are two key files from the GUI procedure, `xgensemble-confs.mol2` and `xgenout-*.mol2`. The key output file`xgen-bound-log` contains the energy estimate. As is typical, the Boltzmann weighted energy (143.6 kcal/mol) and the minimum energy (143.2 kcal/mol) are very close, and the minimum is recommended for strain estimation.

```
# Directory: examples/advanced/xgen-pymol/
```

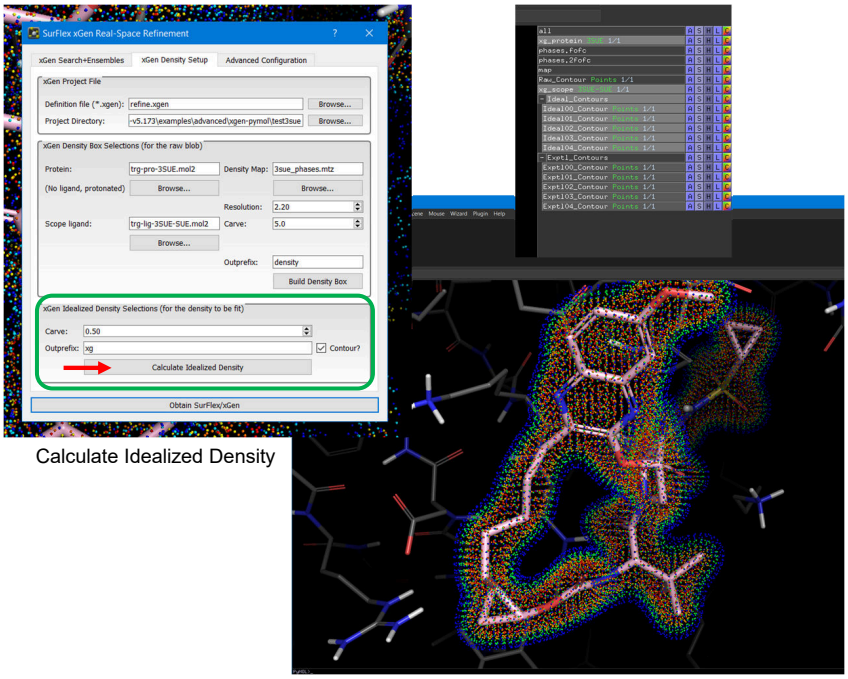


Figure 7.3 Step 2 of xGen ensemble generation: Ideal and experimental densities.

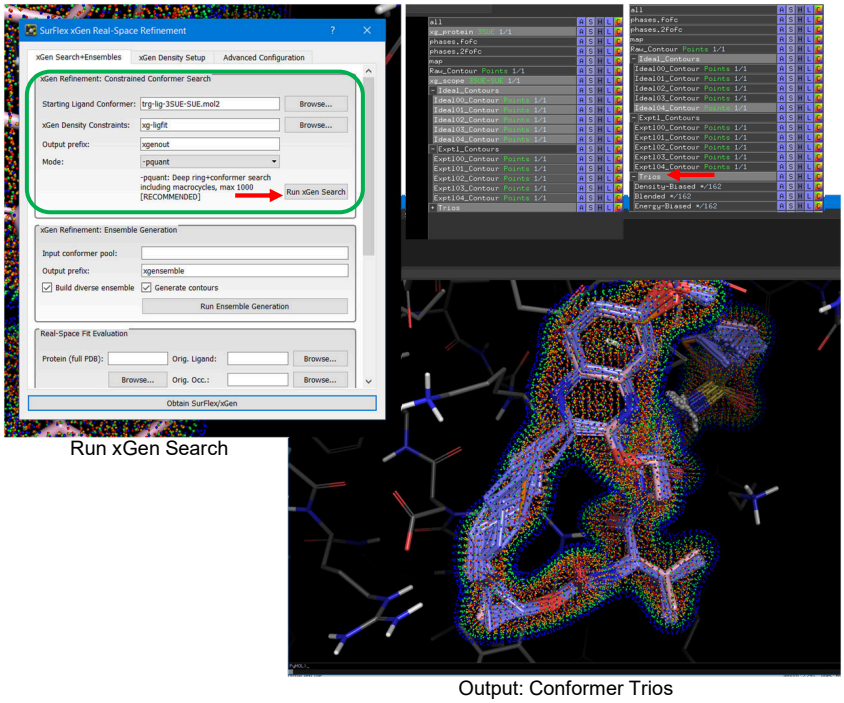


Figure 7.4 Step 3 of xGen ensemble generation: Trios.

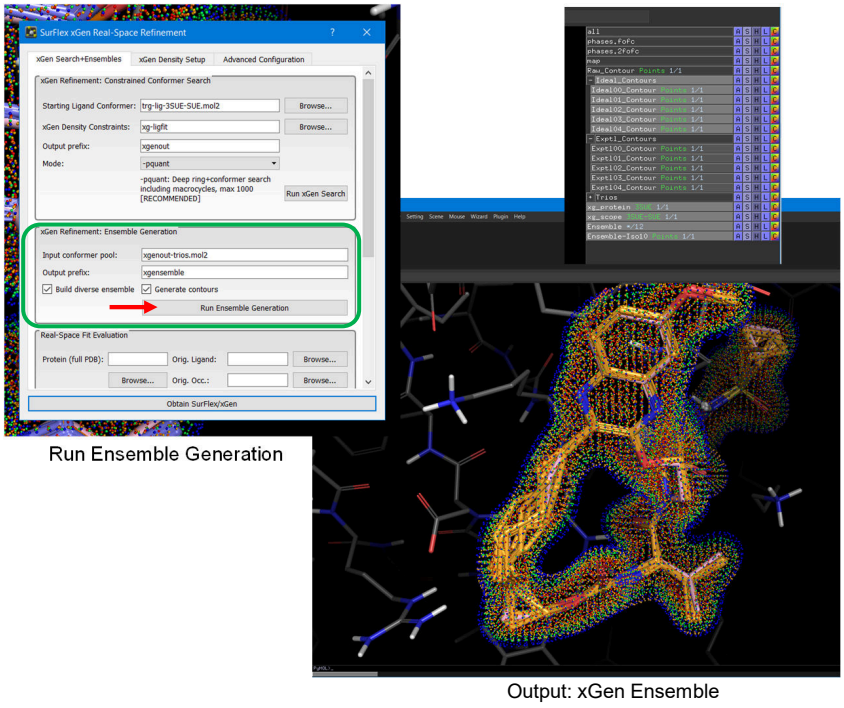


Figure 7.5 Step 4 of xGen ensemble generation: The final xGen ensemble.

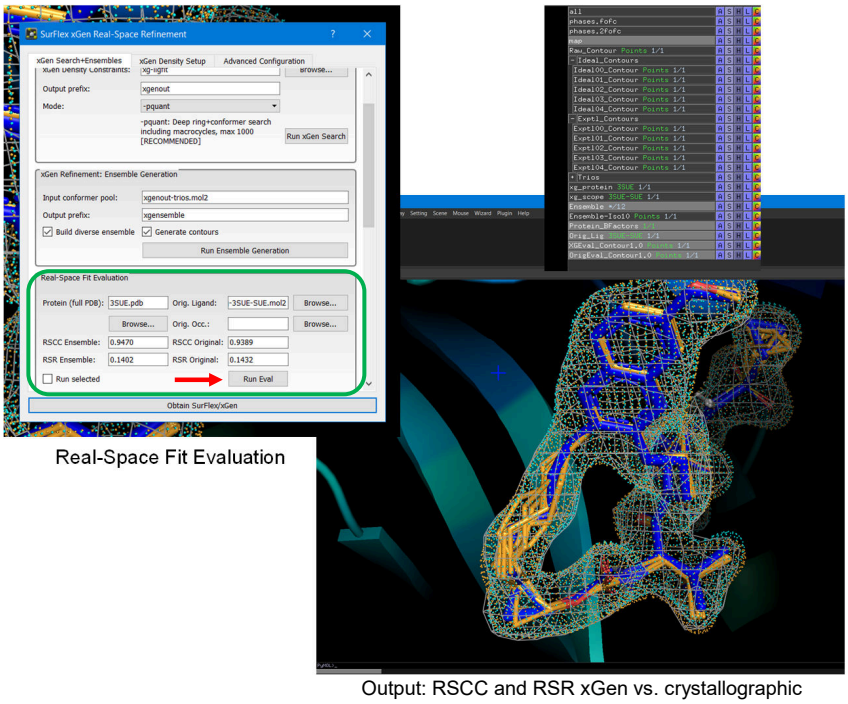


Figure 7.6 Step 5 of xGen ensemble generation: Real space fit evaluation.

```
# KEY INPUT FILES:
```

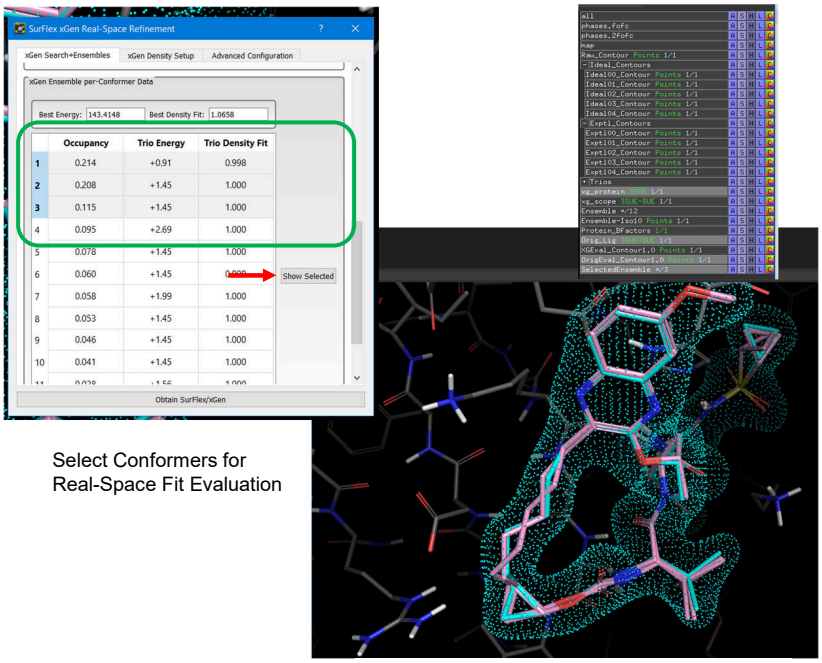


Figure 7.7 Selecting conformer subsets for real space fit evaluation.

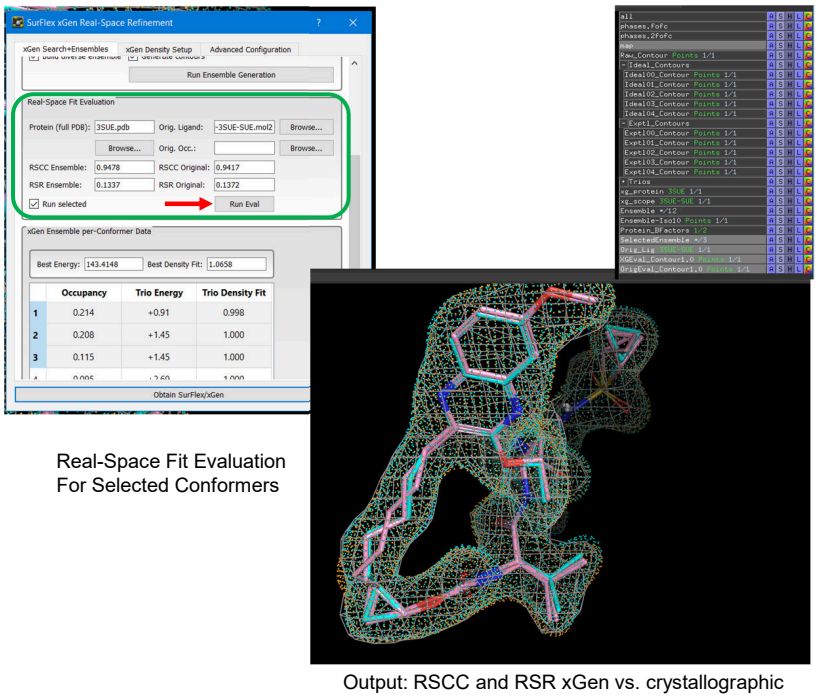


Figure 7.8 Real space fit evaluation for selected conformers.

```
# After preparing the xGen refit X-ray conformational ensemble for vaniprevir, the
    following key files exist:
```

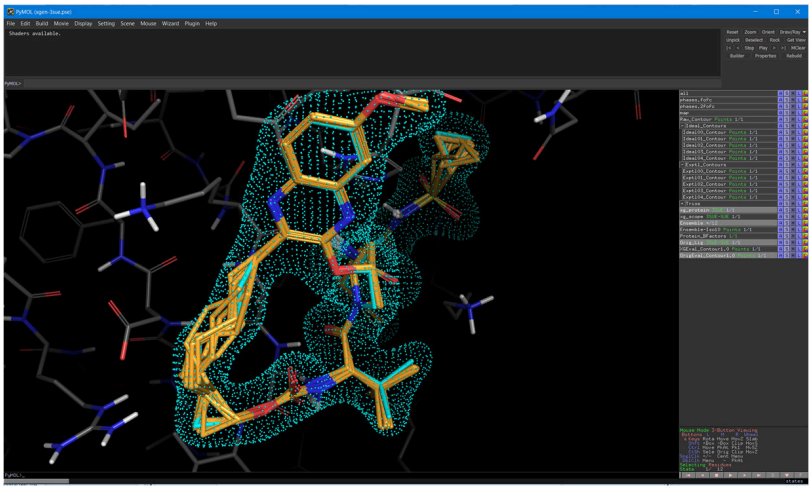


Figure 7.9 Final xGen ensemble.

```
#   xgensemble-confs.mol2     the refit conformational ensemble for vaniprevir
#   xgenout-*.mol2            the raw conformer search trios from the xGen fitting process

# To estimate the bound conformational energy of the re-fit ensemble
# note that we need to specify the "xgenout" prefix
# in order to identify the conformer trio data
# for the proper bound energy calculation:
> sf-tools.exe bound_energy xgensemble-confs.mol2 xgenout xgen-bound

# KEY OUTPUT FILE:
#   xgen-bound-log     contains the estimated bound energy for the ensemble
```

Second, estimate the bound conformational energy of the original PDB coordinates.

```
# Directory: examples/advanced/xgen-pymol/

# KEY INPUT FILES:
#   trg-lig-3SUE-SUE.mol2     the original crystallographic ligand

# To estimate the bound conformational energy of the original PDB coordinates
> sf-tools.exe bound_energy trg-lig-3SUE-SUE.mol2 none pdb-bound

# KEY OUTPUT FILE:
#   pdb-bound-log     contains the estimated bound energy for the original ligand
```

Here, the bound energy estimate for the PDB coordinates is 153.9 kcal/mol (over 10 kcal/mol higher than the re-fit ensemble).

Third, estimate the global minimum energy for the ligand. To do so, generate a conformer pool of the original ligand `trg-lig-3SUE-SUE.mol2` by performing a deep search at the `-pquant` level. Then, makes use of the `unbound_energy` command to obtain a good estimate for the global minimum. The output file `unbound-quick-log` shows 138.7 kcal/mol as the global minimum energy estimate.

```
# Directory: examples/advanced/xgen-pymol/

# KEY INPUT FILES:
#   trg-lig-3SUE-SUE.mol2     the original crystallographic ligand

# To estimate the global minimum energy of the ligand
# Perform a deep ring and conformer search
```

```
> sf-tools.exe -pquant forcegen trg-lig-3SUE-SUE.mol2 pqlig
> sf-tools.exe unbound_energy pqlig.sfdb unbound-quick

# KEY OUTPUT FILE:
#   unbound-quick-log     contains the estimated global minimum energy
```

The intramolecular strain of the xGen ensemble and that of the original crystallographic ligand are calculated by subtracting the estimated global minimum energy from each. This yields strain values of 143.2-138.7 = +4.5 kcal/mol for the xGen ensemble and 153.9-138.7 = +15.2 kcal/mol for the original PDB coordinates. Estimated energy strain was reduced more than 10 kcal/mol: from +15.2 kcal/mol for the PDB deposited crystallographic pose to +4.5 kcal/mol for the lowest energy xGen conformer. The examples in this chapter detail the importance of ligand strain and how strain can be estimated for xGen ensembles, original crystallographic ligands, and docked ligands (described in the PROTAC and PDL1 examples below).

Understanding whether a strain estimate is physically reasonable is important. There is a limited amount of energy that can be obtained from non-covalent association of a protein and ligand. As a rule of thumb, roughly -0.3 kcal/mol/heavy-atom is a generous estimate for enthalpic energy per atom (for a more detailed discussion, including aspects of ligand efficiency, refer to this extensive study of bound ligand strain [5]). Grazoprevir (the 3SUE ligand) has 54 non-hydrogen atoms, and its experimentally determined free energy of binding is -12.3 kcal/mol (K_i=0.84 nM). With 54 heavy atoms and a ballpark value of -0.3 kcal/mol per heavy atom for enthalpy, we would optimistically expect roughly -16.2 kcal/mol for the overall enthalpy of protein-ligand binding.

However, the bound ligand strain works *against* the energy of protein-ligand association. In the original deposited PDB coordinates, the estimated strain is of the *same* rough magnitude (+15.2 kcal/mol) as a generous estimate of possible enthalpic energy of binding (-16.2 kcal/mol). Considering entropic losses as well, grazoprevir would not bind the protease at all if its strain were that large. The lower strain energy of the xGen ensemble seems much more plausible (+4.5 kcal/mol), as it is approximately one-quarter the magnitude of our optimistic estimate of possible enthalpic protein-ligand association energy.

7.2 PROTAC DOCKING

PROTACs are a type of molecular glue designed to bring an E3 ubiqitin ligase in close proximity to a protein of interest (POI) such that the ubiquitination of the target protein leads to degradation by the proteasome. PROTACs are challenging to study using structure-based drug design (SBDD) due to the large size of the ligands and the complexities of crystallizing ternary complexes. Accurate docking requires that ternary complexes of good resolution are available and that the conformational space of the PROTAC to be docked has been sufficiently searched. Ideally, the docking method should have quantitative output that can be related to the half-maximal degradation concentration (DC50) within a series of PROTACs.

The steps for PROTAC Docking will be briefly summarized here:

1. Acquire and prepare the crystallographic proteins and ligands, and align the proteins if more than one complex is available for the target.

2. Prepare the protein targets for docking by generating protomols.

3. Using the cognate ligand for the protein target of docking, generate molecular fragments that represent the warheads for the E3 ligase and the POI.

4. Generate a constrained conformer pool of the ligand to be docked using the warhead fragments as tight or loose restraints.

5. Generate an unconstrained deeply-searched conformer pool to identify the global minimum energy.

6. Dock the warhead-constrained conformer pools and generate pose families from the docked conformers.

7. Estimate the strain of the docked conformers by subtracting the energy of the global minimum from the calculated energy of the top docked pose family.

Here, we will use an example of the von Hippel-Lindau (VHL) E3 ligase in complex with a PROTAC and the BRD4 target protein (PDB Code 8BDS) as the target for docking a non-cognate PROTAC (the ligand of 8BEB). This procedure begins with two simple line commands in order to acquire the co-crystal structures and generate the correctly protonated proteins and crystallographic ligands. The file `verifypdb.smi` is present in the directory so that PDB ligand quality checks occur. Protein pocket alignment is performed using the `psim_align_all` command.

```
# Directory: examples/advanced/protac-docking/

# acquire, prepare, and align the proteins
> sf-dock.exe getpdb PDBList trg
> source trg-script
> sf-dock.exe psim_align_all trg-plist trg_align

# visualize the prepared and aligned proteins, and the transformed ligands
> pym trg_align-c0-*.mol2

# KEY OUTPUT FILES:
#   trg_align-c0-pro-8bds-QIY.mol2     correctly protonated crystallographic protein
#   trg_align-c0-lig-8bds-QIY.mol2     correctly protonated crystallographic ligand
#   trg_align-c0-pro-8beb-QIK.mol2     correctly protonated crystallographic protein
#   trg_align-c0-lig-8beb-QIK.mol2     correctly protonated crystallographic ligand
```

Figure 7.10 shows the prepared and aligned crystallographic proteins as well as the transformed ligands. In preparation for docking, the binding site is scoped and a protomol (idealized ligand) is generated:

```
# Directory: examples/advanced/protac-docking/

# prepare the 8BDS protein for docking
> echo trg_align-c0-pro-8bds-QIY.mol2 trg_align-c0-lig-8bds-QIY.mol2 > proto-list
> sf-dock.exe mproto proto-list p1

# KEY OUTPUT FILES:
#   p1-targets          text file list of protein targets
#   p1-protomol.mol2    idealized ligand
#   p1-corevox.mol2     marks deepest area of pocket
```

Figure 7.11 shows the protomol and corevox for the 8BDS protein. The protomol is an idealized ligand that scopes the binding pocket and the corevox mark the most deeply buried area of the pocket.

Generating deeply-searched, warhead-constrained conformer pools is key to docking PROTACs. Molecular fragments representing the two warhead moieties that bind the POI and ligase respectively can be generated from a known crystallographic ligand. In order to match a diverse set of analogs that may include linker and warhead variation, the fragments should have protons removed and substituents trimmed. The exact definition of the warhead fragments can be somewhat tricky, particularly as regards the matching between an apparently SP3 site on a fragment and a non-SP3 site on the PROTAC of interest. Users should be careful to make sure that both fragments are correctly matching the PROTAC (see below).

In this exercise, QIY warhead fragments are generated and three QIK conformer pools are generated. First is a constrained search using the warhead fragments as tight conformational restraints (default search behavior). Second, since linkers are varied, warhead movement might occur. Loosening the torsional restraint during search (`pwiggle` parameter) allows for warhead movement, but also requires a deeper search using `fgen_deep` to be sure to thoroughly cover the conformational space. As a control, the loosely constrained pool was profiled against the crystallographic pose of QIK resulting in the `profpqdeep-confreport.tab` output table. Sorting low to high on column G_RMS shows the best RMSD was 0.9Å. Third, in order to obtain a global energy minimum conformer, a conformer pool was generated through constraint-agnostic conformational search. The agnostic pool was also profiled against the crystallographic pose of QIK resulting in the `profpqdeepagnostic-confreport.tab` output table which showed the best RMSD was 1.3Å.

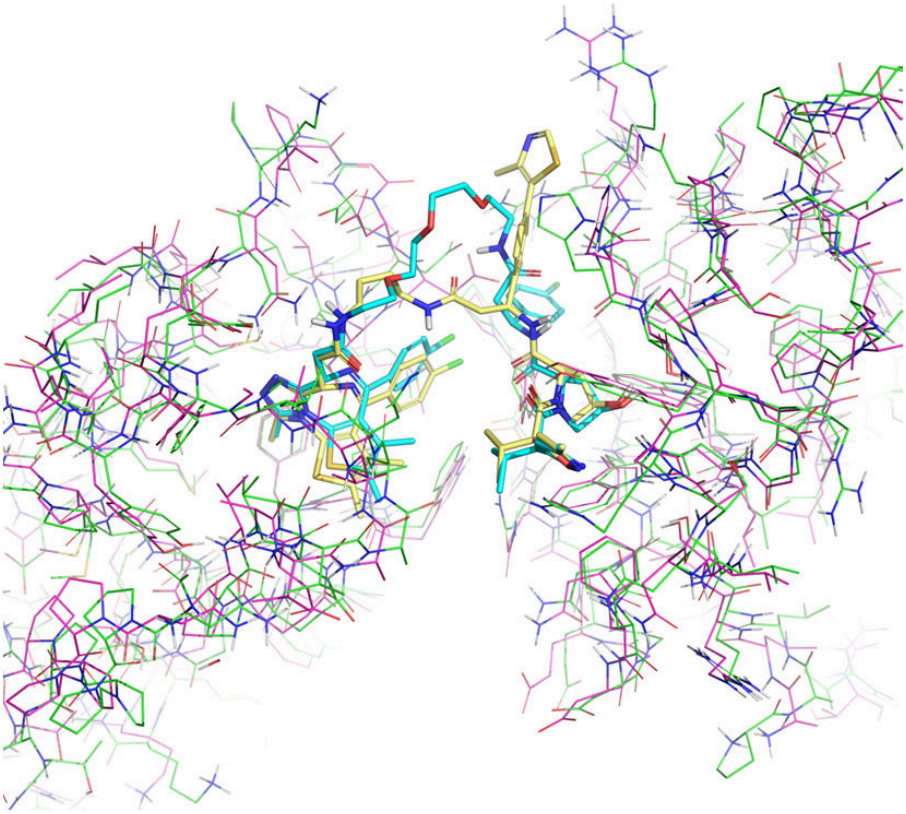


Figure 7.10 PDB ternary complexes 8BDS and 8BEB were acquired and prepared, and the protein binding pockets were aligned.

```
# Directory: examples/advanced/protac-docking/

# Manually create frag1.mol2 and frag2.mol2 from the aligned 8BDS-QIY ligand
# --> frag1.mol2 and frag2.mol2
# These should have protons removed and have substituents trimmed off.
> cat frag1.mol2 frag2.mol2 > frags.mol2
# KEY OUTPUT FILE: frags.mol2

# Randomize the cognate conformation of QIK
> sf-tools.exe regen3d trg_align-c0-lig-8beb-QIK.mol2 qik
# KEY OUTPUT FILE: qik-random.mol2

# Regenerate the starting conformation using the fragments
> sf-tools.exe -torcon frags.mol2 regen3d qik-random.mol2 qik-fg3d

# Here, the output to the console will show something like this:
#
# User specified -torcon: 2 mol fragments
# ------------------------------------------------
# Read 8beb-QIK
# Initial energy/atom: 1.32 (nmacroflex = 0)
# Torcon min energy: total 4.29e+11   torcon 20847.30 (ntor = 50)
# POSITIONAL CONSTRAINTS (N TORCON FRAGS = 2): (56: force 5.0 wiggle 0.2  ...
#   (21: force 5.0 wiggle 0.2  pos 9.13 -5.45 -14.26)
# ........!!!!!!!!==!!==!===$>$$$$$>
# FINAL best energy for 8beb-QIK: 165.881 (1.32 per atom)
```

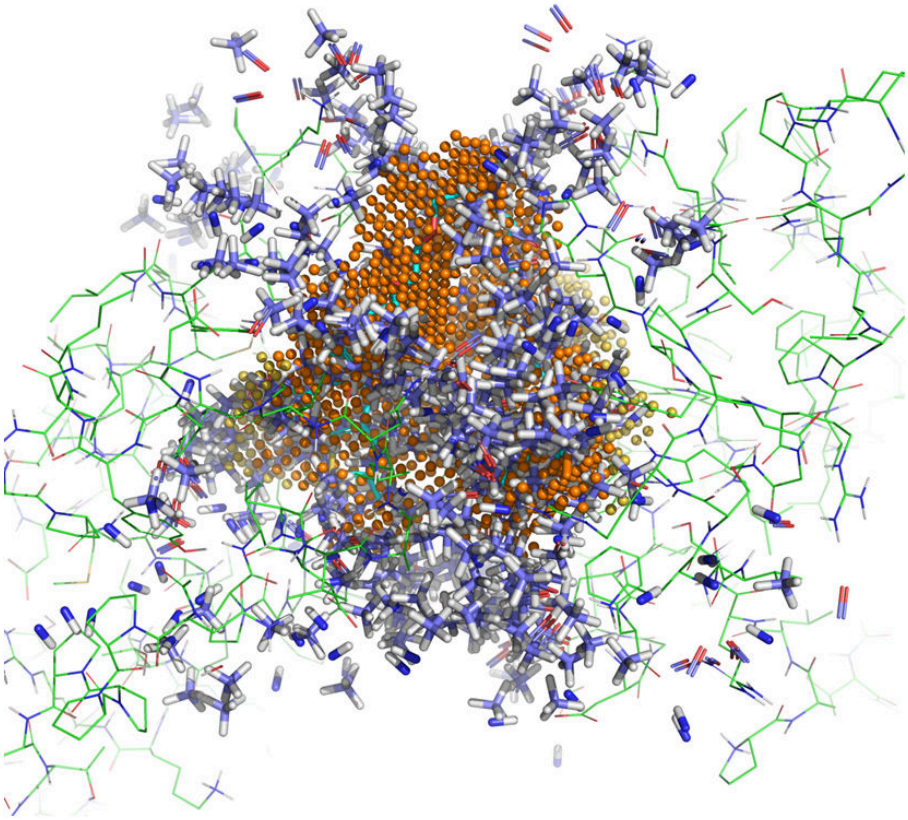


Figure 7.11 In preparation for docking, the protomol and corevox were generated for the 8BDS protein.

```
#   (torcon_energy = 0.123)
#  (pos_energy = 0.839)

# KEY OUTPUT FILE: qik-fg3d-random.mol2
# Visualize that the restraints were applied correctly:
> pym frags.mol2 qik-fg3d-random.mol2
```

In the initial re-generation of the 3D structure subject to the warhead fragment restraints, we see that two fragments have been parsed from the input restraint file, *and* that positional restraints, in addition to torsional restraints, have been applied to the input ligand. Displaying the warhead fragments with the regenerated 3D coordinates will show tight alignment between the fragments and the matching parts of the input ligand if the matching has proceeded correctly.

```
# Perform conformational search using the warhead fragments as a guide using
# tight (default) restraints
> sf-tools.exe -torcon frags.mol2 -pquant forcegen qik-fg3d-random.mol2 pq-torcon-tight
# KEY OUTPUT FILES: pq-torcon-tight.sfdb
# Visualize the resulting pool. The tight restraints hold the fragments in place strongly.
> sf-tools.exe extract_sfdb pq-torcon-tight.sfdb disp
> pym frags.mol2 disp-8beb-QIK.mol2

# Perform conformational search using the warhead fragments as a guide using
# loose restraints
> sf-tools.exe -torpen 0.01 -pwiggle 2.0 -torcon frags.mol2 -pquant fgen_deep
    qik-fg3d-random.mol2 pq-torcon-deep
# KEY OUTPUT FILES: pq-torcon-deep-final.sfdb
# Profile the loosely constrained pool against the bound form. Best RMSD is 0.9 Angstroms.
```

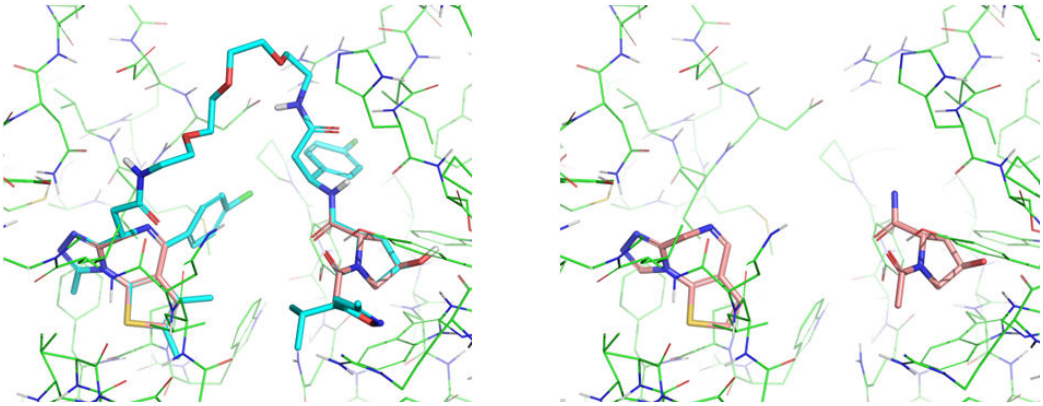


Figure 7.12 Warhead fragments from the crystallographic 8BDS ligand QIY were generated as torsional restraints for QIK conformer generation.

```
> sf-tools.exe profile pq-torcon-deep-final.sfdb trg_align-c0-lig-8beb-QIK.mol2 profpqdeep
# KEY OUTPUT FILE: profpqdeep-confreport.tab

# Perform an agnostic conformational search
> sf-tools.exe -pquant fgen_deep qik-random.mol2 pq-agnostic-deep
# KEY OUTPUT FILES: pq-agnostic-deep-final.sfdb
# Profile the agnostic pool against the known bound form(best RMSD is 1.3 Angstroms)
> sf-tools.exe profile pq-agnostic-deep-final.sfdb trg_align-c0-lig-8beb-QIK.mol2
    profpqdeepagnostic
# KEY OUTPUT FILE: profpqdeepagnostic-confreport.tab
```

As before in displaying the initial regenerated 3D ligand structure, the resulting conformer pool from the restrained conformer search can be displayed and should show good adherence of the ligand warhead substructures to the given fragments. The degree of restraint can be varied using the `-pwiggle` parameter of the ForceGen search. Figure 7.12 shows the warhead fragments derived from 8BDS, which are the binding moieties for the POI and the E3 ubiquitin ligase. Again, checking for proper matching between these fragment and the PROTAC to be docked is important, as described above.

Each of the three QIK conformer pools described above were docked to the 8BDS protein. The pool loosely constrained by the warheads is docked first here as this is the pool most relevant to a real-world PROTAC program. The docking commands for each of the three conformer pools are as follows.

```
# Directory: examples/advanced/protac-docking/

# Dock the loosely constrained conformer pool and make pose families
> sf-dock.exe -lmatch trg_align-c0-lig-8bds-QIY.mol2 -pquant gdock_list
    pq-torcon-deep-final.sfdb p1-targets pqdeepdock
> sf-dock.exe -posehints trg_align-c0-lig-8bds-QIY.mol2 posefam pqdeepdock
# KEY OUTPUT FILE: pqdeepdock-topfam.mol2
# Visualize the docking results
> pym disp.pml

# Dock the tightly constrained conformer pool and make pose families
> sf-dock.exe -lmatch trg_align-c0-lig-8bds-QIY.mol2 -pquant gdock_list
    pq-torcon-tight.sfdb p1-targets pqtightdock
> sf-dock.exe -posehints trg_align-c0-lig-8bds-QIY.mol2 posefam pqtightdock
# KEY OUTPUT FILE: pqtightdock-topfam.mol2

# Dock the agnostic conformer pool and make pose families
```

```
> sf-dock.exe -lmatch trg_align-c0-lig-8bds-QIY.mol2 -pquant gdock_list
    pq-agnostic-deep-final.sfdb p1-targets pqdeepdockagnostic
> sf-dock.exe -posehints trg_align-c0-lig-8bds-QIY.mol2 posefam pqdeepdockagnostic
# KEY OUTPUT FILE: pqdeepdockagnostic-topfam.mol2
```

Figure 7.13 shows the docking results. Shown in grey is the top pose family for the loosely constrained QIK conformer pool relative to the crystallographic pose of QIK (pale yellow). The conformers in the docked top pose family range from 0.8 - 1.5Å RMSD relative to the native pose.

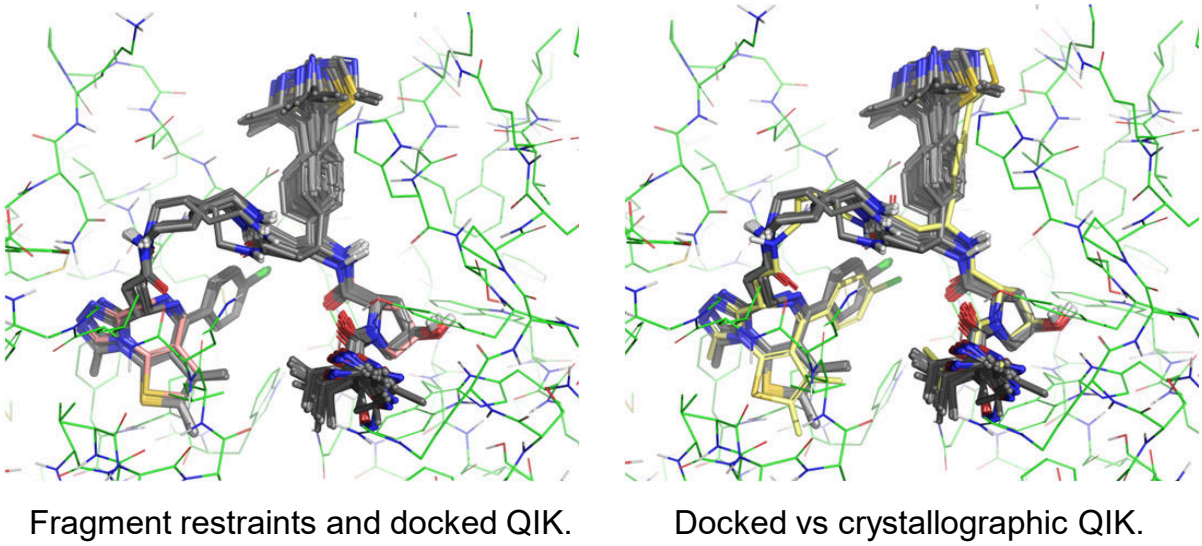


Figure 7.13 Docked QIK relative to the QIY warhead fragments and the crystallographic QIK ligand.

As was discussed above in the xGen ensemble example, it is important to estimate strain for docked ligand poses. The strain estimation protocol for the docked poses of the loosely constrained QIK conformer pool is as follows.

```
# Directory: examples/advanced/protac-docking/

# To calculate the strain for the docked top pose family
# of the loosely constrained conformer pool,
# we will apply a simple procedure to reduce singularities
# and perform local minimization on the docked and global conformer pools
> sf-tools.exe bound_energy pqdeepdock-topfam.mol2 none pqdeepdock-bounden
# KEY OUTPUT FILE: pqdeepdock-bounden-log
# The minimum energy value in the log file shows a bound energy of 164.8 kcal/mol

# Estimate the strain for the global minimum from the agnostic constrained conformer pool
> sf-tools.exe unbound_energy pq-agnostic-deep-final.sfdb globen
# The minimum energy value in the log file shows a global minimum of 151.7 kcal/mol

# so the strain estimate is 164.8-151.7, roughly 13 kcal/mol
```

As seen in the output file `pqdeepdock-bounden-log`, the Boltzmann weighted energy minimum for the top docked pose family of the loosely constrained pool was 164.8 kcal/mol. And, as seen in the output file `globen-log`, the minimum energy for the agnostic conformer pool was 151.7. Subtracting the strain of the global minimum from that of the docked pool yields a strain estimate of 164.8-151.7, roughly 13 kcal/mol. This is a reasonable strain estimate for a ligand as large as the PROTAC QIK.

7.3 MACROCYCLIC PEPTIDE RESTRAINED SEARCH, DOCKING, AND STRAIN ESTIMATION

Lead optimization of large macrocyclic peptides in real-word pharmaceutical applications is challenging. There are significant computational obstacles in modeling these molecules including: (i) generating conformer pools that include the biologically-relevant conformations, (ii) docking such complex molecules, and (iii) accurate strain estimates. With respect to macrocyclic conformer generation, `forcegen` and `fgen_deep` have both been successful with deep unrestrained search, NMR-restrained search, and torsionally restrained search. The example presented here is taken from work on a real-world macrocyclic lead optimization project [6]. Macrocyclic peptides were designed to block the PD-1/PD-L1 interaction as potential anti-cancer therapeutics. The computational approach began with applying NMR restraints to yield a low energy 3D conformer of a lead molecule. From the lead compound conformer, a backbone was derived to torsionally constrain the conformer search of the follow-on molecules including the clinical candidate. The conformer pools were docked into a single available crystal structure. The estimated intermolecular noncovalent binding enthalpies and intramolecular strains were used to estimate the free energy of binding. The estimated binding energies showed a statistically significant correlation to the experimental values. Figure 7.14 shows the 2D structures of the lead molecule Pep-01 and the clinical candidate BMT-174900. Shown in red numbers on the right-hand structure are the 6 changes from Pep-01 to make BMT-174900. Systematic exploration at those 6 positions with just 5 alternate amino acids would require over 15,000 analogs. The computational procedure outlined here was designed to make lead optimization of such molecules much more efficient.

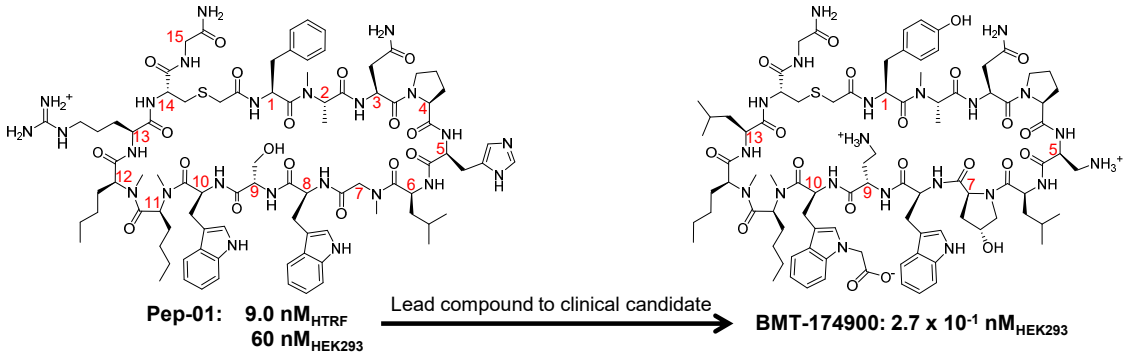
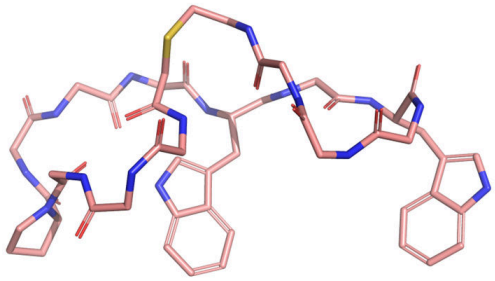


Figure 7.14 Lead molecule Pep-01 and clinical candidate BMT-174900.

The following shows the sequence of commands to run compound `bmt-174900` through the full workflow in order to obtain an estimate of how well it binds. The `RunQuick` and `RunThorough` bash-style scripts can be used to run any of the compounds provided in `pep.smi`.

```
# Directory: examples/advanced/macro-peptide/
# See file:  RunAll

# Grab the peptide in question from the SMILES file
> grep bmt-174900 pep.smi > bmt-174900.smi
# Generate a conforming 3D to the full backbone frags, then do a loose -torcon ForceGen
> sf-tools.exe -torcon allfrags.mol2 fgen3d bmt-174900.smi bmt-174900-fg3d
# sf-tools.exe -findbeta -torpen 0.01 -torcon allfrags.mol2 -pquant forcegen
    bmt-174900-fg3d.mol2 pq-bmt-174900

# 6PV9 is the co-crystal structure of human PD-L1 and Pep-01
# generate the protomol for 6PV9 using the xGen refit ligand
> echo pro-6PV9.mol2 lig-6PV9.mol2 > mproto-list
> sf-dock.exe mproto mproto-list p1

# Do the docking and pose family construction
> sf-dock.exe -lmatch lig-6PV9.mol2 -pquant gdock_list pq-bmt-174900.sfdb p1-targets
    log-bmt-174900
```

```
> sf-dock.exe -posehints lig-6PV9.mol2 posefam log-bmt-174900

# Compute the bound conformational energy and the intermolecular score
> sf-tools.exe bound_energy log-bmt-174900-topfam.mol2 bmt-174900-en
> sf-dock.exe -ntweak 10 opt log-bmt-174900-topfam.mol2 pro-6PV9.mol2 opt-bmt-174900

# Dump the values to output files
> grep Boltzmann bmt-174900-en-log | awk '{print $6}' > bmt-174900-boundenergy
> grep Intermolecular opt-bmt-174900-log | awk '{print $3}' > bmt-174900-interscore

# Do a ForceGen search from the original 3D (backbone-restrained) conformer to find the
    global min
# sf-tools.exe -findbeta -pquant forcegen bmt-174900-fg3d.mol2 pq-bmt-174900-glob

# Find the nominal global minimum
> sf-tools.exe unbound_energy pq-bmt-174900-glob.sfdb bmt-174900-glob-en
> grep Boltzmann bmt-174900-glob-en-log | awk '{print $6}' > bmt-174900-globmin

# Now, the bound energy (kcal/mol) is in bmt-174900-boundenergy, the global min is in
    bmt-174900-globmin
# and the intermolecular score is in bmt-174900-interscore (pKd units)
# To compute an estimate of the binding enthalpy (kcal/mol): (bound_en - globmin_en) +
    (-1.36 * interscore)

> paste bmt-174900-boundenergy bmt-174900-globmin bmt-174900-interscore |
        awk '{print (($1-$2) + (-1.36 * $3))}' > bmt-174900-enthalpy

# Note that this could also be run as ./RunQuick bmt-174900
# and the more thorough search variation by ./RunThorough bmt-174900

# as an example, determine if 6PV9 is a high or low contact ligand
>sf-dock.exe opt lig-6PV9.mol2 pro-6PV9.mol2 contact
# See contact-log (orig_lig_coverage)
# orig_lig_coverage = 0.417
```



Backbone derived from NMR-restrained conformer of Pep-01.

Figure 7.15 Backbone of the low energy conformer from an NMR-restrained pool for the lead molecule.

Figure 7.15 shows the backbone derived from the lowest energy conformer of the lead molecule Pep-01. The conformer pool from which this conformer was derived was generated using NMR restraints. The backbone `allfrags.mol2` was employed as a torsional constraint using the `-torcon` parameter in both 2D -> 3D conversion and the conformer search for the set of peptides in this study. The `-findbeta` parameter that searches for and enforces beta hairpin hydrogen bonds was turned off so as not to unduly influence the search.

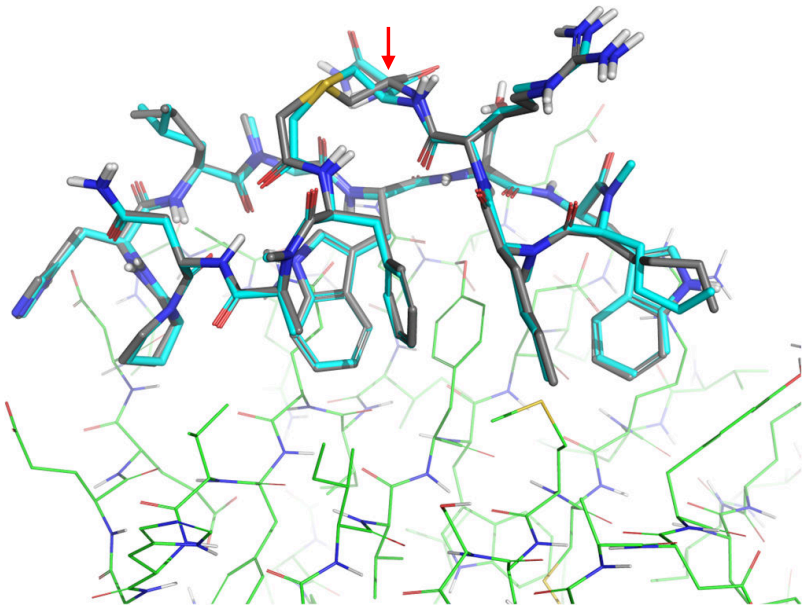
Although it was not done here, it would also have been valid to generate the backbone fragment from a crystallographic ligand. 6PV9 is the co-crystal structure of human PD-L1 and Pep-01. To demonstrate the validity of deriving the torsional constraint from an NMR restrained Pep-01 conformer, the NMR restrained ensemble was compared to the xGen re-fit (`lig-6PV9.mol2`) of the crystallographic ligand (`lig-6PV9-orig.mol2`). It was shown that there

was a low energy NMR restrained conformer within 0.9Å RMSD for all non-hydrogen atoms and 0.4Å RMSD for ring backbone atoms of the xGen refit ligand ([6]). Although the 6PV9 structure was not used as the source of the Pep-01 backbone, it was used as the target for docking Pep-01 analogs in this exercise.

Figure 7.16 shows the crystallographic 6PV9 protein (green) and the xGen re-fit (cyan) of the coordinates for Pep-01. The original ligand had been modeled well (grey sticks, Figure 7.16) with the exception of one incorrect chiral center that caused a distortion to the ring-closing thioether linkage (red arrow, Figure 7.16). The xGen refinement was important as the corrected ligand was then used as a known pose in the docking of a series of peptides to the 6PV9 structure.

For these large macrocyclic peptides, the conformer search and the docking were performed at the `-pquant` level to ensure sufficient depth. The xGen re-fit of the crystallographic ligand was employed as a known pose in the docking and in the generation of docked pose families. Figure 7.17 shows the top pose family (wheat color) from the docking of clinical candidate BMT-174900. The predicted bound pose showed two salt bridges to the protein at positions 5 and 10 (red arrows) which could contribute to the significant improvement in affinity over the lead molecule. As discussed in the xGen example, the binding enthalpy was estimated by combining the intermolecular noncovalent binding enthalpy (here, an optimized docking score) with the the intramolecular strain. The intermolecular score was 24.8 pK_d, yielding -33.7 kcal/mol (= 24.8 * -1.36). The calculated strain was 2.8 kcal/mol (= 296.1 kcal/mol [bound energy] - 293.3 kcal/mol [global min]). So, the estimated binding enthalpy was -30.9 kcal/mol (= -33.7 kcal/mol [intermolecular score] + 2.8 kcal/mol [strain]). Finally, as discussed in [5], large peptidic macrocycles such as those in this study tend to have less buried binding sites compared to those of small molecule ligands. For example, the macrocycle in 6PV9 has orig_lig_coverage = 0.417, and is thus a low contact ligand which are defined as having <70% of the ligand atoms within 1.0Å of any protein atom.

A much more extensive discussion of this example along with implications for molecular design strategies is presented in the full paper [6]. In particular, it turns out that strain estimates can be more important in estimating binding affinity than the magnitude of the enthalpic aspect of protein-ligand interactions.



6PV9 crystallographic protein, original ligand (grey), and xGen refit ligand (cyan).

Figure 7.16 6PV9 structure.

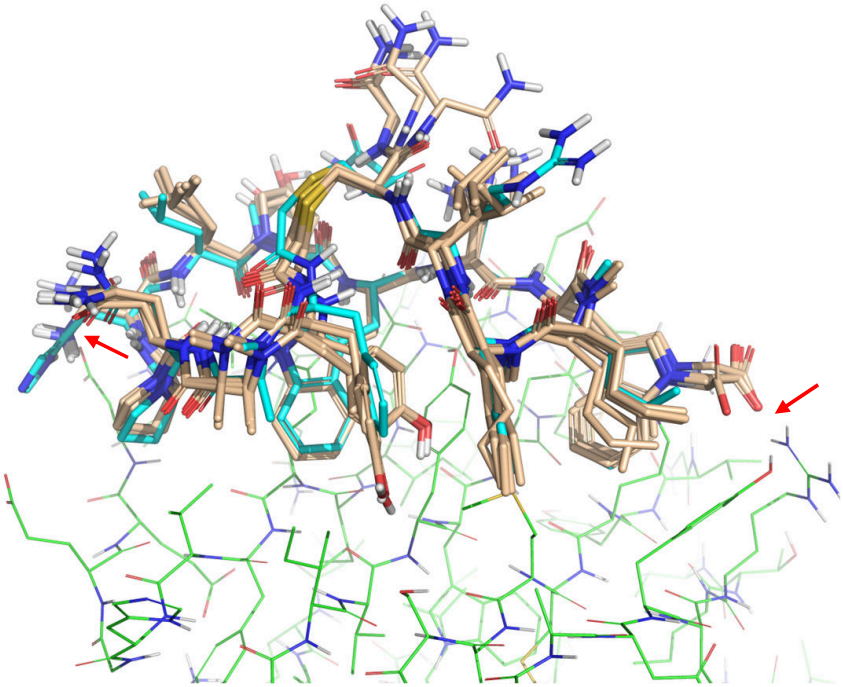


Figure 7.17 Docked top pose family for BMT-174900.

Bibliography

1. Ajay N Jain, Ann E Cleves, Alexander C Brueckner, Charles A Lesburg, Qiaolin Deng, Edward C Sherer, and Mikhail Y Reibarkh. xgen: Real-space fitting of complex ligand conformational ensembles to x-ray electron density maps. *Journal of Medicinal Chemistry*, 63(18):10509–10528, 2020.
2. Ann E Cleves and Ajay N Jain. ForceGen 3D structure and conformer generation: From small lead-like molecules to macrocyclic drugs. *Journal of Computer-Aided Molecular Design*, 31(5):419–439, 2017.
3. Ajay N Jain, Ann E Cleves, Qi Gao, Xiao Wang, Yizhou Liu, Edward C Sherer, and Mikhail Y Reibarkh. Complex macrocycle exploration: Parallel, heuristic, and constraint-based conformer generation using forcegen. *Journal of Computer-Aided Molecular Design*, 33(6):531–558, 2019.
4. Alexander C Brueckner, Qiaolin Deng, Ann E Cleves, Charles A Lesburg, Juan C Alvarez, Mikhail Y Reibarkh, Edward C Sherer, and Ajay N Jain. Conformational strain of macrocyclic peptides in ligand–receptor complexes based on advanced refinement of bound-state conformers. *Journal of Medicinal Chemistry*, 64(6):3282–3298, 2021.
5. Ajay N Jain, Alexander C Brueckner, Ann E Cleves, Mikhail Reibarkh, and Edward C Sherer. A distributional model of bound ligand conformational strain: From small molecules up to large peptidic macrocycles. *Journal of Medicinal Chemistry*, 66(3):1955–1971, 2023.
6. Ajay N Jain, Alexander C Brueckner, Christine Jorge, Ann E Cleves, Purnima Khandelwal, Janet Caceres Cortes, and Luciano Mueller. Complex peptide macrocycle optimization: Combining nmr restraints with conformational analysis to guide structure-based and ligand-based design. *Journal of Computer-Aided Molecular Design*, 37(11): 519–535, 2023.