VM2
Version 2.8.2

Free energy of binding for a protein-ligand series: tutorial 1

VeraChem LLC

Copyright (c) 2015-2019, VeraChem LLC, Germantown, MD, USA. All rights reserved.

VeraChem has been issued a patent **(USPTO Patent No. 8,140,268)** for the VM2 method.

Contact:

For information regarding VM2 software package licensing contact VeraChem LLC at sales@verachem.com

For technical support contact VeraChem LLC at support@verachem.com

For general enquiries contact VeraChem LLC at info@verachem.com

VM2 Free Energy of Binding for a Protein-Ligand Series: Tutorial 1

HIV-1 protease and 38 inhibitors: Amber/GAFF/VCharge

HIV-1 protease and 38 inhibitors: Amber/GAFF/VCharge

This is a full example of setup, execution of calculations, and collection of binding affinity results for a protein plus ligand series: the target protein is human HIV-1 protease and there are 38 ligands in the inhibitor series. (*1*)

NOTE: You will need a working installation of AmberTools with the $AMBERHOME environment variable set to carry out the full procedure as described below. Please see http://ambermd.org/ to download AmberTools and for its documentation.

To proceed, first, untar the examples file vcCompChem_2_8_2_examples.tar.bz2, which is provided with the package:

tar xvf vcCompChem_2_8_2_examples.tar.bz2

The main directory for this example is:

vcCompChem_2_8_2_examples /protein_ligand/hiv1_protease_series_1/

it contains a readme file: README.hiv1p , which describes the overall process, stepping through the following three directories in turn

hiv1_protease_series_1/setup
hiv1_protease_series_1/run
hiv1_protease_series_1/results

An outline of each step now follows. You can skip the setup section by going straight to Section 2. and making use of the "-d reference" option, described in Sections 2.1.2. and 2.2.2.

1. Setup

The procedure starts with setup, namely structure preparation, typing, charge assignment of the protein target molecule and ligand inhibitors, and assignment of mobile and fixed protein atoms.

1.1. Protein setup

The basis for this setup is the crystal structure of HIV-1 protease and the co-crystalized inhibitor AD-81. The PDB access code for this structure is 2I0D. The multiple aspects to consider when preparing a protein for molecular mechanics calculations starting from PDB coordinates are described in Section V 3.1. of the main user's manual. Furthermore, the AMBER reference manual, available through the link given above, provides detailed advice for the use of AmberTools in this process - see the section titled "Preparing PDB Files".

The files used for the following steps are found in the following subdirectory:

hiv1_protease_series_1/ setup/protein

1.1.1. Remove all hetatoms and water atoms except atom 1580

For this particular receptor and set of inhibitors, it is important to explicitly include one of the water molecules (atom number 1580) present in the 2I0D crystal structure. Therefore, edit the pdb file 2i0d.pdb deleting everything prior to the first ATOM entry, all HETATOM entries except for that of atom 1580, and everything except the END record after HETATOM 1580. Name the resulting file 2i0d_1580.pdb.

1.1.2. Extract the co-crystalized ligand

The co-crystalized ligand in 2I0D is used as a reference structure, so copy and edit the original 2i0d.pdb file, deleting all atoms except the AD-81 ligand atoms, and rename the file ad_81_from_2i0d.pdb .

1.1.3. Prepare the PDB file for tleap

Prepare the pdb file for tleap by running the script run_pdb4amber_1.sh, i.e.

./run_pdb4amber_1.sh >& run_pdb4amber_1.log &

This will produce the file 2i0d_1580_p4a.pdb as well as other files required by tleap.

1.1.4. Run tleap to assign parameters

Run tleap to assign parameters using the script run_tleap_2.sh.

./run_tleap_2.sh >& run_tleap_2.log &

This will produce .incpcrd, .prmtop, .mol2, and .pdb files. These will be named 2i0d_1580_p4a_tleap.*

1.1.5. Convert .prmtop and .inpcrd to .crd, .top, and .mol files

Run the VeraChem amber pathway conversion tool prm2top.pyc using the script run_prm2top_3.sh, i.e.

./run_prm2top_3.sh >& run_prm2top_3.log &

This will produce the files 2i0d_1580_p4a_tleap_vm2.[crd,top,mol] These are the files that will be used to run the VM2 calculations.

Compare your results with those provided in the ./reference subdirectory to ensure that the procedure was successful.

1.2. Ligand Setup

Some remaining protein setup steps require that the AD-81 ligand be already setup, so next, the full set of ligands are prepared and parameterized. The relevant subdirectories are:

hiv1_protease_series_1/setup/ligands/source_files
hiv1_protease_series_1/setup/ligands/vconf
hiv1_protease_series_1/setup/ligands/prepare_ligands

1.2.1. Initial 2D structures

Processing with AmberTools requires an input sdf file containing the ligands in 3D, with all hydrogens present and stereochemistry properly defined with parity values. For this example, the ligands were first drawn in 2D by a chemical draw program referencing figures from the published experimental binding affinity article.(*1*) A 2D mol file was saved for each ligand.

These 2D structures can be found in the ./source_files subdirectory of ligands/. A simple python script (mol_2_sdf.py) is used to assemble them into a single sdf file called umass_1.sdf.

```
python mol_2_sdf.py -o umass_1.sdf
```

To process only a chosen subset of the prepared 2D structures a key file can be used that contains the names of the ligands, one on each line, to be processed e.g.

```
python mol_2_sdf.py -o umass_1.sdf -k ligand_key_5.txt
```

1.2.2. 2D to 3D conversion

VeraChem's Vconf program is used to convert these 2D structures to 3D. The relevant files are found in the vconf/ subdirectory. First, copy over the umass_1.sdf file generated by the last step, and then execute the run_vconf.sh script to carry out the conversion:

```
./run_vconf.sh &
```

The resulting 3D structures can be found in the file

hiv1_protease_series_1/setup/ligands/vconf/umass_1_vconf.sdf

You can compare your results against those provided in the reference/ subdirectory.

1.2.3. Generate partial charges and assign parameters to the ligands

Ambertools is used to assign bond, angle, torsion, and non-bonded Lennard-Jones parameters, while atom partial charges can be generated either by VeraChem's VCharge method or by AM1-BCC through AmberTools. The resulting prmtop and inpcrd files are then converted to the [crd,top,mol] file set used by VM2.

The prepareLigands.pyc script automates this process. First, go to the prepare_ligands directory

hiv1_protease_series_1/setup/ligands/prepare_ligands

then copy over the 3D sdf file

cp ../source_files/umass_1.sdf .

Then, to execute the script choosing VCharge partial atomic charges type:

./run_prepareLigands_vcharge.sh &

and to assign charge using AM1-BCC type:

./run_prepareLigands_am1-bcc.sh &

While VCharge takes less than a minute for the set of 28 ligands, generation of AM1-BCC partial charges requires a QM calculation, which can take a considerable amount of time, e.g., approximately 3 hours on a Xeon E5-2667, 3.2GHz cpu.

You can compare your results against those in the reference subdirectories.

1.3. Define fixed and mobile protein atoms

The choice of the included mobile and fixed protein atoms can have a significant impact on the final binding energy predictions produced by the VM2 method. VeraChem recommends inclusion of enough mobile atoms to capture relevant aspects such as loop movement on binding, while avoiding inclusion of large numbers of atoms as mobile, which are effectively spectators, so as to keep calculations manageable with respect to turnover times, and also minimize the occurrence of spurious minima that sometimes occur due to force field inadequacies.

A process for defining mobile and fixed atoms for subsequent free energy calculations is now described.

1.3.1. Generate co-crystalized ligand based AD-81 conformation

First, go to the directory

setup/define_fixed_and_mobile_atoms/1_gen_coxtal_ligand_conf

Next, generate a conformation of the co-crystalized ligand AD-81 to use as the reference coordinates to carve out the mobile and fixed atoms in subsequent steps. This is achieved by 'snapping' scaffold atoms from the AD-81 structure generated in Step 2 above, to the corresponding positions of the co-Xtal AD-81 scaffold atoms in the 2I0D PDB file i.e. scaffold atoms in the file ad_81_from_2i0d.pdb generated in Step 1.2.2

The required files are:

ad_81_pdbsnap_confs.inp : VM2 input file
ad_81.crd : coordinate file generated in **Section 1.2.3.**
ad_81.top : topology/parameter file fin **Section 1.2.3.**
ad_81.mol : mol file generated in **Section 1.2.3.**
ad_81_from_2i0d.pdb : reference ad_81 coordinates from **Section 1.1.2.**

Generate the AD-81 conformations by typing:

```
./runvm2.bsh >& runvm2.log
```

The output of interest is the file:

```
ad_81.confsearch_rank1.crd
```

which contains the coordinates of lowest energy AD-81 conformer 'snapped' to the co-crystalized ligand scaffold atoms. The coordinate file is used in the next step.

1.3.2. Relax all hydrogen atoms in the system

To relieve close contacts that can occur on hydrogen atom placement, all hydrogen atom positions in the protein and AD-81 ligand are optimized according to the force field energy function.

Go to the directory

```
setup/define_fixed_and_mobile_atoms/2_opt_all_protein_h
```

then copy the file required from last step and rename it:

```
cp ../1_gen_coxtal_ligand_conf/ad_81.confsearch_rank1.crd ad_81_snap2pdb.crd
```

The required files for this step are:

2i0d_1580_p4a_tleap_hopt.inp : VM2 package input file for H atom optimization
ad_81_from_2i0d.pdb : reference ad_81 coordinates from **Section 1.1.2.**

2i0d_1580_p4a_tleap_vm2.crd | Protein coordinates, parameters etc.
2i0d_1580_p4a_tleap_vm2.top <--| generated by **Section 1.1.** above.
2i0d_1580_p4a_tleap_vm2.mol | Copied directly from ./protein

ad_81_snap2pdb.crd | ad_81_snap2pdb.crd is the just generated
ad_81.top <---| ad_81.confsearch_rank1.crd copied and
ad_81.mol | renamed. The top and mol files are as in 1.3.1.

Relax all hydrogen atom positions by typing:

./runvm2.bsh >& runvm2.log

The outputs of interest are the files

2i0d_1580_p4a_tleap_vm2.geomopt_rank1.crd
ad_81_snap2pdb.geomopt_rank1.crd

which contain the lowest energy coordinates of the protein and ligand AD-81 after hydrogen atom optimization. These coordinates are used in the next step.

1.3.3. Distance based generation of real/live set

Carve out a mobile and fixed set of protein atoms. VM2 uses so-called real and live sets, where the 'real' set are all the atoms included in the calculation (mobile and fixed) and the 'live' set is the subset of the 'real' set that is mobile. In this step, the VM2 package is used to carve out a 'real' set that comprises all residues that have an atom within 7 Angstoms any atom of the supplied AD-81 ligand coordinates, and a 'live' set of all protein atoms within 5 Angstoms of any atom of the supplied AD-81 ligand coordinates.

Go to the directory

setup/define_fixed_and_mobile_atoms/ 3_dist_based_real_live_set

then copy and rename the required files from the last step:

cp ../2_opt_all_protein_h/2i0d_1580_p4a_tleap_vm2.geomopt_rank1.crd
2i0d_1580_p4a_tleap_vm2_opth.crd

cp ../2_opt_all_protein_h/ad_81_snap2pdb.geomopt_rank1.crd
ad_81_snap2pdb_opth.crd

The required files for this step are:

2i0d_1580_p4a_tleap_genlivereal.inp <--- VM2 package input file for generation of 'real' atom set of all atoms within 7 Angstroms of any atom in the supplied AD-81 ligand crd, and a 'live' atom set within 5 Angstroms.

2i0d_1580_p4a_tleap_vm2_opth.crd | The crd file is the just generated
2i0d_1580_p4a_tleap_vm2.top <--| 2i0d_1580_p4a_tleap_vm2.geomopt_rank1.crd
2i0d_1580_p4a_tleap_vm2.mol | renamed. The top and mol are unchanged.

ad_81_snap2pdb_opth.crd | ad_81_snap2pdb_opth.crd is the just generated
ad_81.top <---| ad_81_snap2pdb.geomopt_rank1.crd from above
ad_81.mol | renamed. The top and mol files are unchanged.

Generate the real and live sets by typing:

./runvm2.bsh >& runvm2.log

The following output files allow you to visualize the 'live' set produced:

2i0d_1580_p4a_tleap_genlivereal.mol2 <--Load into visualizer to see live set produced.
2i0d_1580_p4a_tleap_genlivereal.pdb
2i0d_1580_p4a_tleap_genlivereal.sdf

To see the 'real' set of atoms defined in by these distance cutoffs, run the same calculation with the input file 2i0d_1580_p4a_tleap_genlivereal.inp changed to output 'real' atoms:

#
atomsToOutput
real
#

Generated output files required for running VM2:

2i0d_1580_p4a_tleap_vm2_opth_liverealatoms.txt <--- This file contains the atom numbers of the live and real atoms generated by the applied distance cutoffs.

Once you are happy with the defined real/live sets copy the protein data files required for VM2 runs directly into the directory define_fixed_and_mobile_atoms/ i.e.

```
cp 2i0d_1580_p4a_tleap_vm2.mol ../.
cp 2i0d_1580_p4a_tleap_vm2_opth.crd ../.
cp 2i0d_1580_p4a_tleap_vm2.top ../.
cp 2i0d_1580_p4a_tleap_vm2_opth_liverealatoms.txt ../2i0d_5_7_live_real.txt
```

NOTE: mandatory renaming of 2i0d_1580_p4a_tleap_vm2_opth_liverealatoms.txt to include the text "live_real"

The setup stage is now complete.

2. Run Calculations

The next step is to run the protein-ligand, protein, and ligand, free energy calculations. The relevant directories and readme file are:

hiv1_protease_series_1/run/1_ligand_confgen
hiv1_protease_series_1/run/2_vm2_runs
hiv1_protease_series_1/run/README.runvm2

Optionally, ligand conformations can be pre-generated in /1_ligand_confgen and used to seed the VM2 calculations in /2_vm2_runs.

2.1. Generation of Ligand Starting Conformations

Two types of pre-generated ligand conformations can be utilized in this example. One is 'snapped' conformations, where atoms in each ligand common to a, for example, co-crystalized ligand are, with an applied guiding force, superimposed, while conformational space of the remaining atoms is sampled. The other is randomly orientated conformations of the ligand, suitable for when no pose information is known, only the location of the binding site.

2.1.1. Example run

Go to the directory

run/1_ligand_confgen

This directory contains a python script to generate run directories for conformer generation, and a python script to run the conformer generation calculations. Example usage is as follows:

python build_ligand_start_conf_dirs.py -t ad_81_from_2i0d.pdb

will first populate the directories

1_ligand_confgen/gen_ligand_start_confs_snap

1_ligand_confgen/gen_ligand_start_confs_rndm

with the required subdirectories, input files, and data files to run. Then the following command

python run_ligand_confs_gen.py -r slurm

will step through all these subdirectories, generating slurm scripts, and submitting the calculations to the batch queue. See Section 2.1.3 below for additional submission options through the -r flag.

Note: Requirements for this example run are:

ad_81_from_2i0d.pdb	<---	must be present in /setup/ligands/prepareLigands
scaffold_mapping_wkey.txt	<---	must be present in the current directory and contain the mapping of each ligand onto the reference ligand

2.1.2. Options available for building conformer generation directories

The python script build_ligand_start_conf_dirs.py can take a number of arguments for non-default control the source of the system data etc.:

-d or --data reference : Populate 'input_data' directory using the data in the setup 'reference' directories e.g. /setup/ligands/prepareLigands/reference, and subsequently build the run directories with this data.

new : Populate 'input_data' directory using the new data in the setup directories e.g. /setup/ligands/prepareLigands, and subsequently build the run directories with this data. (Default behavior.)

reuse : Reuse the data from an already populated 'input_data' directory.

-s or --startconfs random : Make a run directory for each ligand in the series for generation of ligand conformers in random orientations and with their center of geometry (COG) placed at a template ligand's COG.

snap : Make a run directory for each ligand in the series for generation of ligand conformers where scaffold atoms are 'snapped' to corresponding template ligand scaffold atoms (via applied harmonic potentials).

all : Make both of the above run directories. (Default behavior.)

-t or --template 'template_filename' : Name of file containing template ligand coordinates e.g. co-xtal ligand or previously docked ligand. Required unless '-d reuse' option set.

-c or --clear input : Delete the contents of 'input_data' directory.

rundirs : Delete the contents of the run directories 'gen_ligand_start_confs_rndm' and 'gen_ligand_start_confs_snap'.

all : Delete content from the 'input_data' directory and the run directories.

Example usage:

```
python build_ligand_start_conf_dirs.py -c rundirs -d reuse
```

This will clear the contents of previously generated run directories and use the data already present in ./input_data to regenerate the run directories i.e. data will not be taken from the setup directories in this case.

2.1.3. Options available for running conformer generation

The python script run_ligand_confs_gen.py can take a number of arguments:

-s or --startconfs	random	: Step through each ligand directory in /gen_ligand_start_confs_rndm and submit a calculation for generation of ligand conformers in random orientations and with their center of geometry (COG) placed at a template ligand's COG.
	snap	: Step through each ligand directory in gen_ligand_start_confs_snap and submit a calculation for generation of ligand conformers where scaffold atoms are 'snapped' to corresponding template ligand scaffold atoms (via applied harmonic potentials).
	all	: Carry out both sets of calculations. (Default behavior.)
-r or --runscript	bsh	: Generate and use bash shell scripts for submission of each calculation. (Default behavior.)
	csh	: Generate and use c-shell scripts for submission of each calculation.
	pbs	: Generate a pbs script for submission of each calculation to a queue.
	slurm	: Generate a slurm script for submission of each calculation to a queue.
-q or --partition	'queue name'	: For pbs and slurm run scripts, the name of the queue or partition if the default queue is not being used.
-p or --prepmode		: If present the run scripts are generated and placed

in every directory, but the calculations are not submitted.

2.2. Protein-ligand calculations

Two main types of VM2 protein-ligand free energy calculation are available. One is regular VM2, which carries out iterative rounds of conformational searching until convergence; the other type carries out geometry optimizations of protein-ligand conformations constructed from ligand conformers read-in and processes them for free energy. The latter is much faster, but much less exhaustive in terms of sampling conformational space. In combination, there are three ways to seed these two VM2 calculation types with ligand conformers: multiple conformers with selected atoms 'snapped' to a reference ligand – see Section 2.1. above; multiple conformers randomly orientated in space, but placed at the location of the binding site – see Section 2.1. above, and a single conformer, based on the position and geometry in which it was prepared originally. This provides for six different overall VM2 calculation schemes, which cover various types of use scenarios.

2.2.1. Example run

Go to the directory

run/2_vm2_runs

This directory contains a python script to generate run directories for protein-ligand VM2 free energy calculations, and a python script to step through the directories and run the calculations. Example usage is as follows:

python build_vm2_run_dirs.py -t ad_81_from_2i0d.pdb

will first populate the following six directories, which cover the calculation types described above, with the required subdirectories, input files, and data files to run.

/2_vm2_runs/fast_vm2_snap
/2_vm2_runs/fast_vm2_rndm
/2_vm2_runs/fast_vm2_single
/2_vm2_runs/vm2_snap
/2_vm2_runs/vm2_rndm
/2_vm2_runs/vm2_single

Note: For "_snap" and "_rndm" types, the corresponding pre-generation of ligand conformers – Section 2.1. - must already have occurred.

Then the following command:

python run_vm2_calculations.py -s snap -v fast -r slurm

will step through the subdirectories of /2_vm2_runs/fast_vm2_snap, generating slurm scripts, and submitting the calculations to the batch queue. Similarly, any of the other five calculations types may be run by setting the appropriate flags – see Section 2.2.2 below. See Section 2.2.3 below for additional submission options through the -r flag.

2.2.2. Options available for building VM2 directories

The python script build_vm2_run_dirs.py can take a number of arguments for non-default control of the source of the system data etc.:

-d or --data reference : Populate 'input_data' directory using the data in the setup 'reference' directories e.g. /setup/ligands/prepareLigands/reference and /setup/define_fixed_and_mobile_atoms/reference, and the ligand start conformer generation reference directory /run/1_ligand_confgen/reference and subsequently build the run directories with this data.

new : Populate 'input_data' directory using the new data in the setup directories e.g. /setup/ligands/prepareLigands and /setup/define_fixed_and_mobile_atoms/ and the ligand start conformer generation directories /run/1_ligand_confgen/gen_ligand_start_confs_rndm and /run/1_ligand_confgen/gen_ligand_start_confs_snap and subsequently build the run directories with this data. (Default behavior.)

reuse : Reuse the data from an already populated 'input_data' directory.

-s or --startconfs random : Requests run directory set up for VM2 free energy calculations where randomly oriented ligand conformers are placed in the active site and are used to generate starting protein-ligand conformations.

snap : Requests run directory set up for VM2 free energy calculations where ligand conformers in which scaffold atoms have been 'snapped' to corresponding scaffold atoms of a template ligand (e.g. co-xtal ligand) are used to generate starting protein-ligand conformations.

single : Requests run directory set up for VM2 free energy calculations where a single ligand starting conformation and placement is used based on the supplied ligand .crd file coordinates. The placement can be adjusted if a template ligand is supplied and the place ligand flag set; see -t, --template and -p, --placelig below. Only used a

non-adjusted ligand .crd if you prepared the ligand in a very good placement and pose in the receptor binding site.

all : Requests both types of directory to be set up. (Default behavior.)

-t or --template 'template_filename' : Name of file containing template ligand coordinates e.g. co-xtal ligand or previously docked ligand. Could simply be coordinates that signifiy the loacation of the binding site. Not required unless random start conformers are in use or the place ligand option just below is set.

-p or --placelig tcog : Place ligand .crd coordinates center of geometry at template ligand's center of geometry.

-c or --clear input : Delete the contents of 'input_data' directory.

rundirs : Delete the contents of the run directories.

all : Delete content from the 'input_data' directory and the run directories.

-v or --vm2type regular : Requests run directory set up for regular VM2 protein-ligand free energy calculations, which carry out extensive conformational searching.

fast : Requests run directory set up for fast VM2 protein-ligand free energy calculations, which calculate free energies via geometry optimizing protein-ligand conformations generated from read-in ligand conformers previously snapped to a template scaffold.

all : Requests set up for both types of VM2 calculation.

-k or --keyfile 'ligand_key_filename' : Name of text file containing the subset of ligands in the series - one on each line (see ligand_key_5.txt.)

2.2.3. Options available for running VM2 calculations

The python script run_ligand_confs_gen.py can take a number of arguments:

-s or --startconfs random : Requests that VM2 free energy calculations are run for the series where randomly oriented ligand conformers

are placed in the active site and are used to generate starting protein-ligand conformations.

snap : Requests that VM2 free energy calculations are run for the series where ligand conformers in which scaffold atoms have been 'snapped' to corresponding scaffold atoms of a template ligand (e.g. co-xtal ligand) are used to generate starting protein-ligand conformations. (Default behavior.)

all : Requests both types of run be carried out.

-r or --runscript bsh : Generate and use bash shell scripts for submission of each calculation. (Default behavior.)

csh : Generate and use c-shell scripts for submission of each calculation.

pbs : Generate a pbs script for submission of each calculation to a queue.

slurm : Generate a slurm script for submission of each calculation to a queue.

-q or --partition 'queue name' : For pbs and slurm run scripts, the name of the queue or partition if the default queue is not being used.

-p or --prepmode : If present the run scripts are generated and placed in every directory, but the calculations are not submitted.

-v or --vm2type regular : Requests regular VM2 protein-ligand free energy calculations for the series, which carry out extensive conformational searching.

fast : Requests fast VM2 VM2 protein-ligand free energy calculations for the series, which calculate free energies via geometry optimizing protein-ligand conformations generated from read-in ligand conformers snapped to a template scaffold. (Default behavior.)

all : Requests both types of VM2 calculation are run for the series.

-i or --mpiprocs n (integer) : Sets the number of MPI processes to run. Currently all processes must run on the same node - though hand editing of run scripts can remove this restriction. The default is 8.

-g or --gpu : If present requests use of CUDA enabled VM2 executable.

-o or --ompthreads 1 : If -g not set results in MPI parallelism only. Enforced for ligand only runs.

2 : If set will result in MPI+OpenMP run (8 MPI processes (default), 2 OpenMP threads per process). If -g also set will result in MPI+OpenMP+CUDA parallelism.

4 : Same as previous, but 4 OpenMP threads.

-m or --molsystems complexes+ligands |
|
complexes+protein |
|
protein+ligand |
|
complexes |----> Run subset of the moleculer system
| types.
|
ligands |
|
protein |

all : Default. Run ligands, complexes, and protein.

Example usage:

```
nohup python run_vm2_calculations.py -g -o 2
```

Run default fast-snap set of calculations (fast_vm2_snap directory) with 8 MPI process calculations for ligand calculations, but MPI+OpenMP+CUDA calculations for the complexes and the protein.

This run utilizes 8 MPI processes with 1 GPU per MPI process and 2 OpenMP threads per MPI process. It therefore requires 16 compute cores and 8 GPUs.

3. Results Collection

When the protein-ligand, protein, and ligand VM2 free energy calculations for the complete ligand series have completed, the binding free energies may then be calculated, and the formatted files, e.g., .mol2, .pdb, .sdf, containing the associated molecular structures collected.

The relevant directories and readme file are:

hiv1_protease_series_1/results
hiv1_protease_series_1/results/conformers
hiv1_protease_series_1/results/README.results

3.1. Generate binding free energy spreadsheets and collect conformer files

Go to the directory

hiv1_protease_series_1/results

To generate spreadsheets and collect molecule conformer files for the "fast_vm2_snap" calculations from Section 2.2.1 type:

python create_vm2_summaries.py -c fast_vm2_snap -n 2i0d -l ad_81

Requirements:

File containing experimental data: experimental_data.csv

The filename must contain "experimental_data".
The format is <proteinname_ligandname>, <value> e.g.

2i0d_ad_12,-9.367
2i0d_ad_17,-14.203
2i0d_ad_23,-11.559
2i0d_ad_24,-10.126
2i0d_ad_32,-10.337
2i0d_ad_33,-12.458
:

Output spreadsheets:

results/2i0d_fast_vm2_snap_complex.csv
results/2i0d_fast_vm2_snap_protein.csv
results/fast_vm2_snap_ligand.csv
results/2i0d_fast_vm2_snap_SUMMARY.csv

The last of these contains the binding free energies.

Output conformer files:

For the protein, each ligand, and each protein-ligand complex, formatted files (e.g. mol2, pdb, sdf, xyz) containing the lowest energy conformer, and the eight lowest energy conformers are written to:

results/conformers/fast_vm2_rndm/complexes
results/conformers/fast_vm2_rndm/ligands
results/conformers/fast_vm2_rndm/protein

3.2. Results generation options

For the script create_vm2_summaries.py the following two commandline arguments are mandatory with the following options:

-c or --calctype fast_vm2_snap : Identify the calculation type to collect and summarize run data for.
fast_vm2_rndm
fast_vm2_single
vm2_snap
vm2_rndm
vm2_single

-n or --receptorname : Provide the name of the receptor e.g. for this case the protein is named "2i0d"

There are two additional non mandatory arguments:

-d or --data new : Sets the source of the calculation data to be extracted and summarized as ../run/2_vm2_runs/fast_vm2_snap etc. (Default behavior.)

reference : Sets the source of the calculation data to be extracted and summarized as ../run/2_vm2_runs/reference/fast_vm2_snap etc.

-l or --refligand : Provide the name of the reference ligand to be used in relative binding affinity calculation i.e. for Delta(DeltaG) The default is no reference.

1. A. Ali *et al.*, Discovery of HIV-1 protease inhibitors with picomolar affinities incorporating N-aryl-oxazolidinone-5-carboxamides as novel P2 ligands. *J. Med. Chem.* **49**, 7342-7356 (2006).