Supplementary Information

AutoDock Bias: improving binding mode prediction and virtual screening using known protein-ligand interactions

Juan Pablo Arcon, Carlos P. Modenutti, Demian Avendano, Elias D. Lopez, Lucas A. Defelipe, Francesca A. Ambrosio, Adrian G. Turjanski, Stefano Forli* and Marcelo A. Marti*

*To whom correspondence should be addressed:
marti.marcelo@gmail.com, forli@scripps.edu.

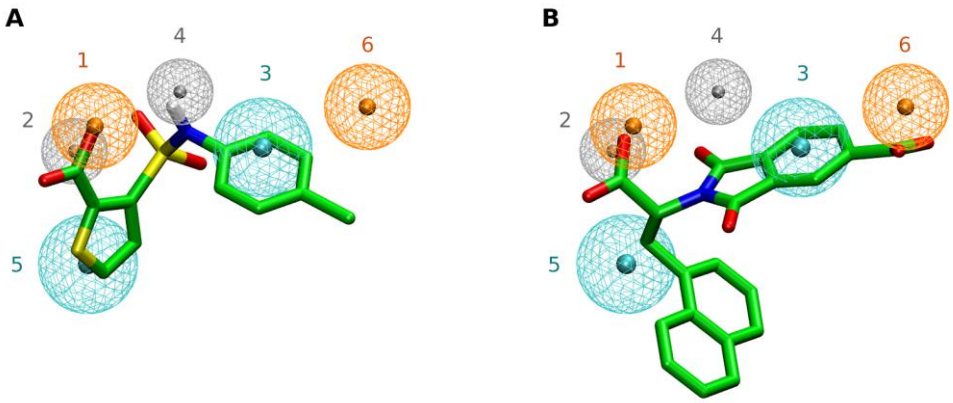


Figure S1. Ethanol sites superimposed to representative AmpC ligands from PDB ID (A) 1l2s -thiophene ligand- and (B) 2r9w -phthalimide ligand-. Hydrogen bond acceptor sites are depicted as orange spheres, hydrogen bond donor sites as gray spheres and hydrophobic sites as cyan spheres.

Table S1. β-lactamase non covalent ligands used for cross-docking experiments.

Protein	**PDB IDs for cocrystals from where the ligands were obtained**
AmpC β-lactamase	1l2s, 1xgi, 1xgj, 2pu2, 2r9w, 2r9x, 4jxs, 4jxv, 4jxw, 4kz4

Table S2. Ranking and cluster population for the correctly predicted pose (RMSD < 2 Å against reference) in the cross-docking experiments from Figure 1A.

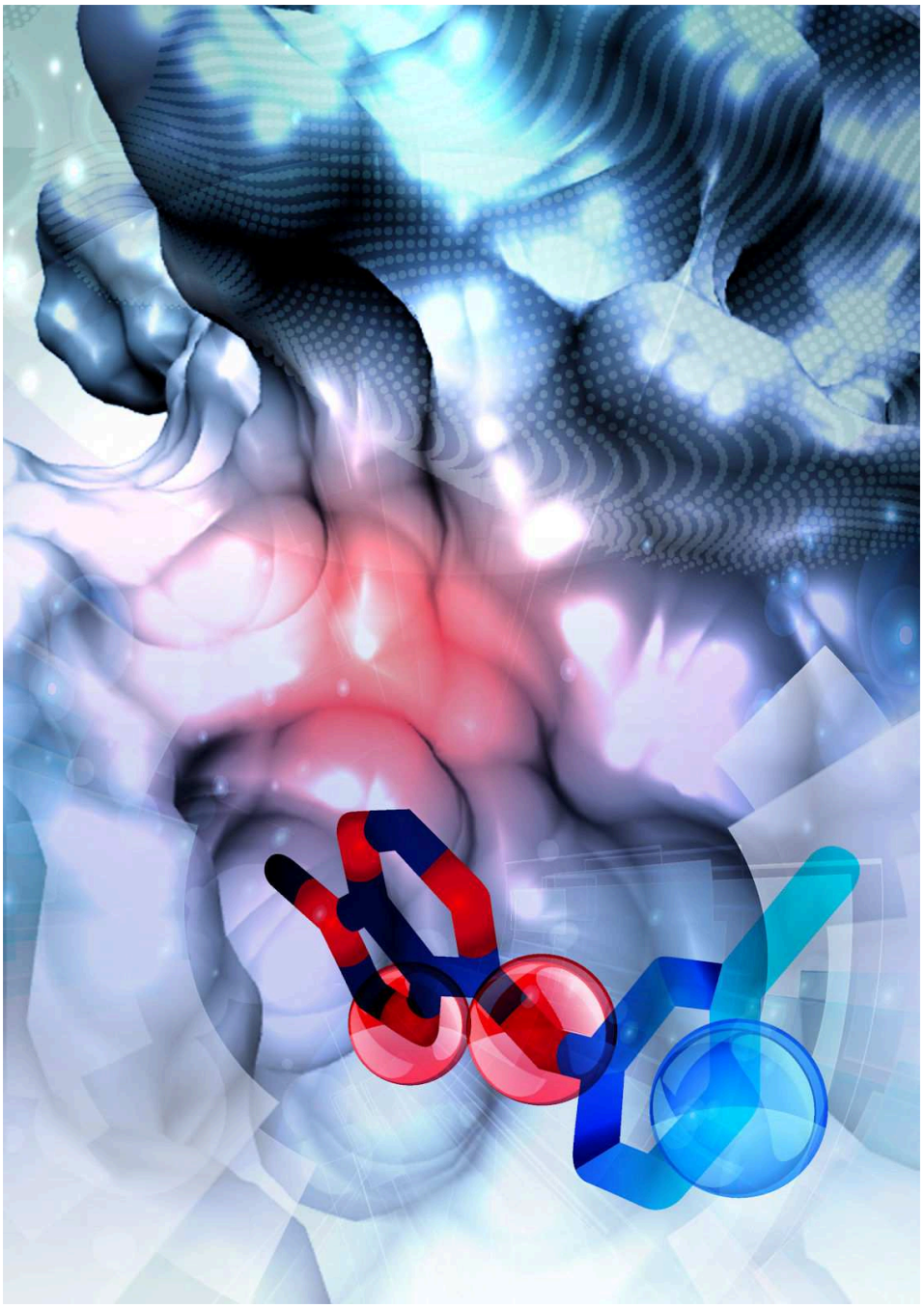
Cocrystal PDB ID for docked ligands	**ranking (population)**	
	conventional	**biased**
1l2s	-	1 (97%)
1xgi	11 (2%)	1 (96%)
1xgj	5 (7%)	1 (95%)
2pu2	8 (16%)	2 (70%)
2r9w	5 (53%)	1 (99%)
2r9x	-	1 (71%)
4jxs	-	1 (83%)
4jxv	-	4 (1%)
4jxw	9 (1%)	-
4kz4	-	1 (10%)

Text S1. Autodock Bias User guide and tutorials.

See next page.

AutoDock Bias

Biased docking with AutoDock4



USER GUIDE

Table of Contents

1. Introduction **4**

2. Use of AutoDock Bias **6**

2.1 Prepare input file for bias: the bias parameter file (BPF) 7

2.1.1 Bias parameter file format requirements 7

2.2 Modify energy maps, DPFs and ligand PDBQTs 8

2.2.1 Modify energy maps for traditional biases (don, acc, aro) 8

2.2.2 Modify DPF and ligand PDBQT for traditional biases (don, acc, aro) 9

2.2.3 Modify energy maps, DPF and ligand PDBQT for traditional biases (don, acc, aro) 10

2.2.4 Modify multiple DPFs and ligand PDBQTs for Virtual Screening 10

2.2.5 Modify energy maps, DPFs and ligand PDBQTs for Virtual Screening 10

2.2.6 Modify a specific energy map 11

2.3 Additional output: grid modification file (grid_modif_X.dat) 12

2.4 Additional script: ideal_interaction_sites.py 12

2.5 Additional script: bias2pdb.py 13

3. AutoDock Bias Tutorial **15**

3.1 Tutorial Part I: Knowledge-based biased docking 15

3.1.1 Aim 15

3.1.2 Standard AutoDock4 Run 15

3.1.2.1 Protein and ligand structure preparation 15

3.1.2.1.1 Preparing the ligand PDBQT file 16

3.1.2.1.2 Preparing the receptor PDBQT file 17

3.1.2.2 Preparing the grid parameter file (GPF) and the energy maps 18

3.1.2.3 Preparing the docking parameter file (DPF) and performing the docking calculation 19

3.1.2.4 Analyzing the results 20

3.1.3 Biased docking (ligand-derived pharmacophore) 24

3.1.3.1 Generating the bias parameter file (BPF) 24

3.1.3.2 Introducing the bias 27

3.1.3.3 Verifying the bias in the maps and DPF 27

3.1.3.4 Performing the biased docking 29

3.1.3.5 Analyzing the Results 29

3.2 Tutorial Part II: Introducing cosolvent derived bias 32

3.2.1 Aim 32

3.2.2 Standard AutoDock4 Run 32

3.2.3 Biased docking (solvent sites) 32

3.2.3.1 Generating the bias parameter file (BPF) 32

3.2.3.2 Introducing the bias 34

3.2.3.3 Verifying the bias in the maps, DPF and ligand PDBQT 35
3.2.3.4 Performing the biased docking 37
3.2.3.5 Analyzing the Results 37

1. Introduction

The aim of this tool is to provide an easy way to add into AutoDock4 scoring function a *bias* towards specific protein-ligand interactions, such as ligand-derived pharmacophores, interaction sites obtained by hotspot mapping, or simply interactions known to be essential for binding. The strategy is to increase the magnitude of the binding energy when these interactions are fulfilled. Specifically, the biases are implemented as potential energy wells that promote the placement of specific atoms or functional groups of the ligand in desired locations.

Briefly, once the standard AutoDock4 energy maps (grids) have been computed for each ligand atom type, the biases are introduced as modifications of the corresponding maps. Thus, hydrogen bond acceptor biases modify OA and NA maps, hydrogen bond donor biases modify HD maps, and aromatic biases are handled by creating a new AC atom type in the center of aromatic rings and the corresponding AC energy map (AC = Aromatic Center). It is also possible to modify any specific energy map that the user may consider relevant for his/her project (e.g. for ligands grown from fragments with known pose or ligands covalently bound to the protein).

Figure 1 shows how the map modification is accomplished. Starting from the original energy map, an additional energy reward (V_{set}) is set in the center of the site where the bias is applied (*bias site*). Thus, the calculated energy is lowered in that region. At the same time, a radius for the bias site is set to control how far from the center of that site the energy modification will still be significant. The precise form of the energy reward is shown in the equation in Figure 1. Notice that the extension through space is modulated by a distance-dependent Gaussian decay. Strictly speaking, the energy modifications are applied in the grid points locations.

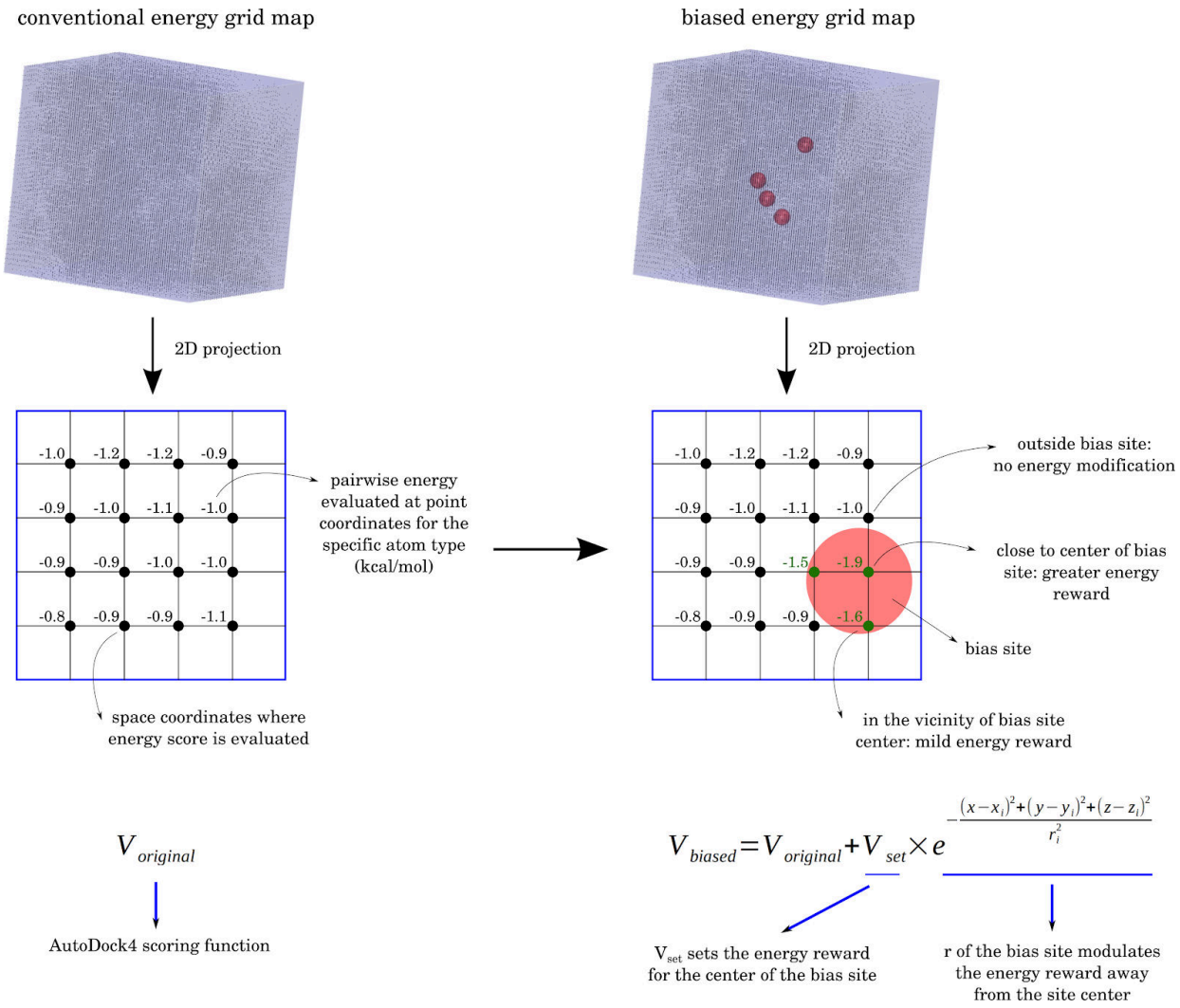


Figure 1. Energy map modification to perform biased docking with AutoDock Bias. Regarding the equation, $V_{original}$ is the original AutoDock4 energy at a certain grid point, V_{biased} corresponds to the resulting modified potential at the same grid point, V_{set} is the bias energy well maximum value (negative number), (x,y,z) are the grid point coordinates, (x_i,y_i,z_i) are the coordinates of the bias site center, and r_i is the bias site radius. V_{set}, r_i and (x_i,y_i,z_i) are user specified (see 2.1.1).

2. Use of AutoDock Bias

AutoDock Bias is based on a Python command line script that modifies *previously* created conventional AutoDock4 files. It allows two different types of bias treatment:

- ↠ Modification of energy maps, DPFs and ligand PDBQT files as required for general built-in biases, i.e. hydrogen bond donors, hydrogen bond acceptors and/or aromatic interactions. This function allows the user to define the bias for the most common molecular interactions and once the interaction type and its localization is indicated, the bias docking run is automatically prepared.

- ↠ Modification of specific energy maps to generate user defined biases. This versatile function allows the user to define a precise localization for any desired atom (e.g. metal atom) or group (e.g. substructure core of a congeneric ligand series or for fragment growth). It may also be used to set an anchor for covalent docking studies.

NOTE. Before executing the code examples below, it is necessary to define the location of the MGLTools Utilities24 path, containing the scripts from AutoDockTools. In the Bash shell, for a typical MGLTools installation it can be done executing:

```
$ export MGLUTIL=$INSTALL_DIR/MGLToolsPckgs/AutoDockTools/Utilities24/
```

Make sure to adapt it to the actual installation of MGLTools on your system. Alternatively, it is possible to execute the commands by specifying the full path of each script.

The bias script is executed as follows[1]:

```
$ pythonsh $MGLUTIL/contrib/adbias/prepare_bias.py -b bias_parameter_file [-g GPF] [-d
DPF] [-D DPFs_dir] [-m map_file]

      -b : bias parameter file (input file for bias)
      -g : AutoDock4 grid parameter file
      -d : AutoDock4 docking parameter file
      -D : modify all docking parameter files (and ligand PDBQT files if necessary)
           in the DPFs_dir directory
      -m : specific input map file
```

[1] The program is prepared to work with the pythonsh interpreter for python2 that comes with AutoDockTools 1.5.7 (http://mgltools.scripps.edu/downloads). Versions for regular python2 and python3 are also available upon request.

It requires -b and, at least, one of the other arguments (-g, -d, -D or -m). All the possible options are explained in the following paragraphs.

2.1 Prepare input file for bias: the *bias parameter file* (BPF)

The user must prepare a bias parameter file (BPF) containing all the information for the different biases to be applied. The BPF contains one line for each bias, with the following parameters:

- *(x, y, z)* coordinates in Å,
- energy reward (V_{set}) in kcal/mol,
- decay radius (r) in Å,
- type of bias (*don*, *acc*, *aro* or *map*).

The following type of biases are available:

- hydrogen bond donor = *don*,
- hydrogen bond acceptor = *acc*,
- aromatic = *aro,*
- specific bias according to the desired map = *map*.

Two examples of bias parameter files are shown below:

i) Bias parameter file with traditional interaction biases

```
x        y        z        Vset          r      type
33.682   36.327   34.471   -1.50   0.80   acc
34.557   36.028   31.933   -2.00   0.60   don
36.905   36.534   30.560   -1.75   1.00   aro
```

ii) Bias parameter file with specific map biases

```
x        y        z        Vset    r      type
5.100    1.785    20.019   -1.75   1.10   map
9.459    2.075    24.527   -2.00   0.80   map
```

2.1.1 Bias parameter file format requirements

- All lines must have 6 columns. The columns must be space or tab separated.
- Lines are ignored if the first column is not numeric (e.g., header with titles -*x, y, z,* V_{set}*, r, type*- in the examples above).
- The first three columns define the *x,y,z* coordinates of the bias site center, in Å.

- The fourth column corresponds to the energy reward (V_{set}), in kcal/mol, to be applied at the bias site center. It has to be a negative number[2]. If there is no thermodynamic information for the site, a reasonable value is -2.0 kcal/mol, which sets a relatively strong bias.
- The fifth column is the radius (r) of the bias site, in Å. It controls the extent of energy reward through space according to a Gaussian function -see equation in Figure 1-. A reasonable value range that builds a well defined bias site is between 0.6 and 1.2 Å.
- The last column indicates the type of bias and, in consequence, which energy maps will be modified:

  - *acc* modifies NA and OA maps;
  - *don* modifies HD maps;
  - *aro* creates an *ad hoc* new map (AC, aromatic center map) -see 2.2.1-;
  - *map* modifies the energy map specified in the *-m* argument. **IMPORTANT:** *map* biases cannot be combined with other types of biases (*don*, *acc*, *aro*) in the same execution of the program -see 2.2.6-.

2.2 Modify energy maps, DPFs and ligand PDBQTs

2.2.1 Modify energy maps for traditional biases (*don*, *acc*, *aro*)

Execute this line:

```
$ pythonsh $MGLUTIL/contrib/adbias/prepare_bias.py -b bias_parameter_file -g GPF
```

The script requires the input bias parameter file (see 2.1), GPF and map files in the current working directory. It will read the energy maps from the GPF and modify them according to the selected type of biases:

- If there are acceptor biases, the hydrogen bond acceptor maps (OA, NA) will be modified.
- If there are donor biases, the hydrogen bond donor map (HD) will be modified.
- If there are aromatic biases, a new atom type and map will be created (AC, atom type for the center of aromatic rings). The map will have the energy reward in the indicated locations for the aromatic bias and will be zero everywhere else. A new parameter file for the docking runs will also be created containing the parameters for this new atom type (*ad4_arom_params.dat*).

[2] **IMPORTANT.** If V_{set} is a positive number N, it will be considered as a relative density of states and will be converted to energy using -kT ln(N) -in kcal/mol-. This means you cannot set an energy *penalty*.

The new map names will add the *.biased.map* extension to the original basename.

IMPORTANT. Running the *prepare_bias.py* script just with *-b* and *-g* options will modify the energy maps, but not the DPF required for running the docking simulation, which will still be done reading the original maps (see 2.2.2 and 2.2.3 for DPF modification). In other words, to "use" the biases the DPF must also be modified.

2.2.2 Modify DPF and ligand PDBQT for traditional biases (don, acc, aro)

Execute this line:

```
$ pythonsh $MGLUTIL/contrib/adbias/prepare_bias.py -b bias_parameter_file -d DPF
```

The script requires the input bias parameter file (see 2.1) and DPF in the current directory. If there are aromatic biases, it also requires the ligand PDBQT to check if there are aromatic rings in its structure (it makes use of *openbabel* and *pybel* python bindings, already installed in AutoDockTools).

The script will modify the DPF according to the type of biases:

- ↠ If there are acceptor biases, the hydrogen bond acceptor map lines (OA, NA) will be modified.
- ↠ If there are donor biases, the hydrogen bond donor map line (HD) will be modified.

If there are aromatic biases *and* the ligand has aromatic rings, it will generate a new ligand PDBQT with a dummy atom[3] in the center of each aromatic ring (atom type AC = aromatic center). A new parameter file for the docking runs will also be created containing the parameters for the new atom type (*ad4_arom_params.dat*). Finally, the following modifications will be made on the DPF:

- ↠ the *move* line will read the new ligand PDBQT file;
- ↠ the *ligand_types* and *map* lines will include the AC atom type and map, respectively;
- ↠ a new *parameter_file* line will be added for the parameters of the new AC atom type.

The new DPF name will add the *.biased.dpf* extension to the original basename.
The new ligand PDBQT name will add the *.dum.pdbqt* extension to the original basename.

[3] The dummy atom has no charge and no volume.

IMPORTANT. Running the *prepare_bias.py* script just with *-b* and *-d* options will modify the DPF (and ligand PDBQT if necessary), but not the energy maps required for the biased docking (see 2.2.1 and 2.2.3 for maps modification).

2.2.3 Modify energy maps, DPF and ligand PDBQT for traditional biases (don, acc, aro)

Execute this line:

```
$ pythonsh $MGLUTIL/contrib/adbias/prepare_bias.py -b bias_parameter_file -g GPF -d DPF
```

This will do the same as 2.2.1 and 2.2.2, all at once.

The script requires the input bias parameter file (see 2.1), GPF, map files and DPF in the current directory. If aromatic biases are set in the BPF, it also requires the ligand PDBQT to check if here are aromatic rings in its structure (it makes use of *openbabel* and *pybel* python bindings, already installed in AutoDockTools).

2.2.4 Modify multiple DPFs and ligand PDBQTs for Virtual Screening

Execute this line line:

```
$ pythonsh $MGLUTIL/contrib/adbias/prepare_bias.py -b bias_parameter_file -D DPFs_dir
```

This will do the same as 2.2.2 for all DPFs and PDBQTs inside the indicated *DPFs_dir* directory.

The script requires the input bias parameter file (see 2.1) in the current directory and a directory (*DPFs_dir*) with the DPFs to modify. If aromatic biases are set in the BPF, it also requires the ligand PDBQTs inside the *DPFs_dir* directory to check if there are aromatic rings in their structure (it makes use of *openbabel* and *pybel* python bindings, already installed in AutoDockTools).

IMPORTANT. Running the *prepare_bias.py* script just with *-b* and *-D* options will modify the DPFs (and ligand PDBQTs if necessary), but not the energy maps required for the biased docking (see 2.2.1 and 2.2.5 for maps modification).

2.2.5 Modify energy maps, DPFs and ligand PDBQTs for Virtual Screening

Execute this line:

```
$ pythonsh $MGLUTIL/contrib/adbias/prepare_bias.py -b bias_parameter_file -g GPF -D
DPFs_directory
```

The script requires the input bias parameter file (see 2.1), GPF and map files in the current directory and an additional directory (*DPFs_dir*) with the DPFs to modify. If aromatic biases are set in the BPF, it also requires the ligand PDBQTs inside the *DPFs_dir* directory to check if there are aromatic rings in their structure (it makes use of *openbabel* and *pybel* python bindings, already installed in AutoDockTools).

The script will read the energy maps from the GPF and modify the maps, the DPFs, and the ligand PDBQTs according to the type of biases:

- ↠ If there are acceptor biases, the hydrogen bond acceptor maps (OA, NA) and their corresponding map lines in the DPFs will be modified.
- ↠ If there are donor biases, the hydrogen bond donor map (HD) and its corresponding map line in the DPFs will be modified.
- ↠ If there are aromatic biases, a new atom type and map will be created (AC, new atom type for the center of aromatic rings). The map will have the energy reward in the indicated locations for the aromatic bias and will be zero everywhere else. For each ligand bearing aromatic rings, it will generate a new ligand PDBQT with a dummy atom of type AC in the center of each aromatic ring. A new parameter file for the docking runs will also be created containing the parameters for this new atom type (*ad4_arom_params.dat*).
  The following modifications will be made on the DPFs:

  - a new *parameter_file* line will be added for the parameters of the new AC atom type;
  - the *ligand_types* and *map* lines will include the AC atom type and map, respectively;
  - the *move* line will read the new ligand PDBQT file.

The new map names will add the *.biased.map* extension to the original basename.
The new DPF names will add the *.biased.dpf* extension to the original basename.
The new ligand PDBQT names will add the *.dum.pdbqt* extension to the original basename.

2.2.6 Modify a specific energy map

Execute this line:

```
$ pythonsh $MGLUTIL/contrib/adbias/prepare_bias.py -b bias_parameter_file -m map_file
[-d DPF]
```

The script requires the input bias parameter file (see 2.1) and the input map file in the current directory. If the option *-d* is indicated, it also requires the DPF in the current directory. The input bias parameter file should only contain the keyword *map* in the type of bias.

The script will modify the input energy map indicated in the *-m* option according to the information in the bias parameter file. If the option *-d* is indicated, it will also modify the map line accordingly in the DPF.

The new map name will add the *.biased.map* extension to the original map basename.
If indicated, the new DPF name will add the *.biased.dpf* extension to the original basename.

IMPORTANT. If *-d* is not indicated, it will only modify the indicated map. The map line in the DPF should then be modified *by hand* before the docking run.

2.3 Additional output: grid modification file (*grid_modif_X.dat*)

Besides the modified maps, DPFs and ligand PDBQTs, an output LOG file *grid_modif_X.dat* is generated for each modified map (X = atom type) with the following information:

- *x,y,z* = coordinates of the grid point closest to the bias site center;
- *point* = number of the grid point closest to the bias site center;
- *dist* = distance from the bias site center to the closest grid point;
- *E_ori* = original energy in the grid point closest to the bias site center (before modification);
- *E_modif* = modified energy in the grid point closest to the bias site center;
- *E_delta* = *E_modif* - *E_ori* = energy reward in the grid point closest to the bias site center;
- *radius* = radius of the bias site.

2.4 Additional script: *ideal_interaction_sites.py*

The *ideal_interactions_sites.py* script calculates ideal locations for hydrogen bond and aromatic interactions with any residue in a protein structure. It takes a PDB file with the structure of the protein (protonated) and generates a PDB file (*interaction_sites.pdb*) with a list of possible ligand interactors with the protein residues indicated by the user.

Usage:

```
$ pythonsh $MGLUTIL/contrib/adbias/ideal_interaction_sites.py -i pdb_file -c chainID
-r resIDs
```

-*i* : input PDB file name containing the protein structure (with hydrogens)
-*c* : protein chain ID (e.g. A)
-*r* : residues IDs -comma separated- whose interactions are to be calculated (e.g. 22,54,61)

Notes:

- The script requires Maestro or Amber atom names for hydrogen atoms (e.g. HNE or HE, HH11, HH12, HH21 and HH22 for Arg hydrogens in the guanidinium group).
- The script requires Biopython for PDB parsing (already installed in AutoDockTools).
- For hydrogen bond calculations you should assign protonation explicitly and name residues accordingly (e.g. HIE, HID or HIP for histidine).

Output example: *interaction_sites.pdb*

```
ATOM      1  H   DON X   1       4.899   1.869  19.060  1.00  0.00
ATOM      2  O   ACC X   2       1.911  -0.515  15.386  1.00  0.00
ATOM      3  C   ARO X   3       9.225   1.986  23.576  1.00  0.00
```

Acceptor sites → residue name *ACC*, atom name *O*.
Donor sites → residue name *DON*, atom name *H*.
Aromatic sites → residue name *ARO*, atom name *C*.

2.5 Additional script: *bias2pdb.py*

The *bias2pdb.py* script generates a PDB file (*bias_sites.pdb*) from a bias parameter file to allow the visualization of the bias sites in the protein context.

Usage:

```
$ pythonsh $MGLUTIL/contrib/adbias/bias2pdb.py ligand.bpf
```

Output example: *bias_sites.pdb*

```
ATOM      1  H   DON X   1       5.100   1.785  20.019  1.00  0.00
ATOM      2  N   ACC X   2       6.183   2.487  22.662  1.00  0.00
```

Acceptor sites → residue name *ACC*, atom name *N*.
Donor sites → residue name *DON*, atom name *H*.

Aromatic sites → residue name *ARO*, atom name *C*.
Specific sites (“map” type) → residue name *MAP*, atom name *M*.

3. AutoDock Bias Tutorial

The objective of this tutorial is to introduce the reader to biased docking with AutoDock4. It assumes that the user has basic knowledge regarding the use of AutoDock4 software (http://autodock.scripps.edu/).

3.1 Tutorial Part I: Knowledge-based biased docking

3.1.1 Aim

In this part of the tutorial we will show how to apply a docking bias based on previous information on the target protein (*knowledge based*). The chosen target is the Fibroblast Growth Factor Receptor 1 (FGFr1), a transmembrane receptor implicated in embryonic development, angiogenesis, wound healing and malignant transformation. We will focus on its intracellular tyrosine kinase domain. Several protein-ligand complex structures are available for this target which aid in determining its key ligand interactions. These interactions are the ones we will use to bias or guide the docking.

3.1.2 Standard AutoDock4 Run

We will first run the docking with AutoDock4 using the conventional method for comparison purpose. Therefore, we will need to build the corresponding receptor, ligand, grid and docking files. If you are already familiar with AutoDock4 and want to jump right to the biased docking, go to section 3.1.3.

3.1.2.1 Protein and ligand structure preparation

The selected protein structure corresponds to PDB ID 1agw and we will dock the co-crystallized ligand SU2. Briefly, we will remove the ligand from the crystal structure, randomly alter its conformation and test if the docking program is capable of predicting the correct crystal pose. This experiment in which the protein structure is conformationally adapted for the ligand binding is known as *re-docking* or *self-docking* and represents a usual starting point.

Download the PDB file (*1agw.pdb*) and generate the following files with a text editor:

- *receptor.pdb* → initial receptor file having chain A of the crystallized structure of the protein (remove ligand, water molecules and the remaining chain).

- *ligand.pdb* → initial ligand file having the co-crystallized structure of the ligand bound to chain A of the protein (this will be the reference structure to compare the poses predicted by the docking program).

These 3 files (*1agw.pdb, receptor.pdb, ligand.pdb*) can be found in the directory *$MGLUTIL/contrib/adbias/doc/examples/initial_structures*.

Since AutoDock4 uses PDBQT input files describing atom types, atomic charges and flexibility (allowed torsions), we need to preprocess the PDB structure files.

3.1.2.1.1 Preparing the ligand PDBQT file

We will prepare an adequate torsion tree and assign atom types and charges for the ligand with the *prepare_ligand4.py* script provided in AutoDockTools (http://mgltools.scripps.edu/).

```
$ pythonsh $MGLUTIL/prepare_ligand4.py -l ligand.pdb -A hydrogens -o ligand.pdbqt
```

-l input ligand (PDB)
-A hydrogens adds H atoms
-o output ligand (PDBQT)

This will generate a PDBQT file for the ligand (*ligand.pdbqt*). It is important to visually check the structure, in particular the protonation state. You may open the PDBQT file with the Python Molecule Viewer (PMV) from AutoDock Tools (ADT) or with PYMOL[4].

```
$ adt ligand.pdbqt
```

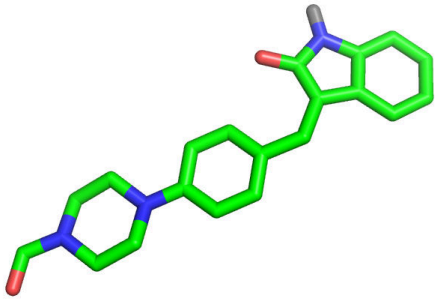


Figure 2. Ligand PDBQT.

[4] `$ pymol ligand.pdbqt`

For the current ligand (Figure 2) we will leave both N atoms from the aliphatic ring deprotonated: one forms part of an amide group, while the other is directly bound to a phenyl ring (i.e., is part of a substituted aniline). Remember AutoDock uses a *united atom* approach for hydrogen atoms that cannot be implicated in standard hydrogen bonds (those H atoms are merged, along with their charge, to their bonded heavy atom).

We will then randomize ligand torsions to avoid bias from the initial conformation with the *write_random_state_ligand.py* script provided in AutoDockTools:

```
$ pythonsh $MGLUTIL/write_random_state_ligand.py -l ligand.pdbqt -o ligand_dock.pdbqt
```

-*l* input ligand (PDBQT)
-*o* output ligand (PDBQT with randomized torsions)

Check that the new PDBQT (*ligand_dock.pdbqt*) has a different position, orientation and conformation from the original one (*ligand.pdbqt*).

3.1.2.1.2 Preparing the receptor PDBQT file

The receptor PDBQT file will be prepared from the initial receptor structure (*receptor.pdb*). We have to tackle the protonation of protein residues and the special orientation treatment for amides, histidines, alcohols and cysteines. For this sake we used the Maestro software (free academic license), but you may use whichever program of your choice (e.g OpenBabel).

The protonated receptor was saved as *receptor_H.pdb* in the *$MGLUTIL/contrib/adbias/doc/examples/initial_structures* directory.

We will now use the *prepare_receptor4.py* script provided in AutoDockTools to obtain the PDBQT file:

```
$ pythonsh $MGLUTIL/prepare_receptor4.py -r receptor_H.pdb -o receptor.pdbqt
```

-*r* receptor input (PDB)
-*o* receptor output (PDBQT)

Open the *receptor.pdbqt* file with a text editor and see that it has no torsion tree (rigid molecule).

3.1.2.2 Preparing the grid parameter file (GPF) and the energy maps

Based on the PDBQT files of the receptor and the ligand, we will now create the grid where the conformational search of the ligand will be performed, along with the precalculated energy maps for each interaction:

- 1 van der Waals (+ hydrogen bond if appropriate) map for each ligand atom type (also including the charge independent desolvation contribution);
- 1 electrostatic map;
- 1 desolvation map (charge dependent).

First, generate a file with the information required for creating the maps: the GPF (grid parameter file). This is done using the *prepare_gpf4.py* script from AutoDockTools:

```
$ pythonsh $MGLUTIL/prepare_gpf4.py -y -l ligand.pdbqt -p npts="70,70,70" -r receptor.pdbqt -o receptor.gpf
```

-*y* centers the grid in the center of mass of the reference ligand (ligand.pdbqt)
-*l* input ligand (PDBQT)
-*p npts="70,70,70"* gives a grid covering the whole ATP binding site of FGFr1
-*r* input receptor (PDBQT)
-*o* output GPF

Now we are ready to build the energy maps using the AutoGrid4 program (from AutoDock4 package) and the parameters from the GPF:

```
$ autogrid4 -p receptor.gpf -l receptor.glg
```

-*p* input GPF
-*l* output log file

This creates the different energy maps (*receptor.X.map*). You may visualize the grid size (Figure 3) using ADT (or VMD[5]) and selecting one of the maps:

```
$ adt receptor.pdbqt ligand.pdbqt
```

> Click “Grid3D”, “Read...” and open one of the maps (e.g. receptor.A.map).

[5] `$ vmd -m receptor.pdbqt ligand.pdb receptor.A.map`

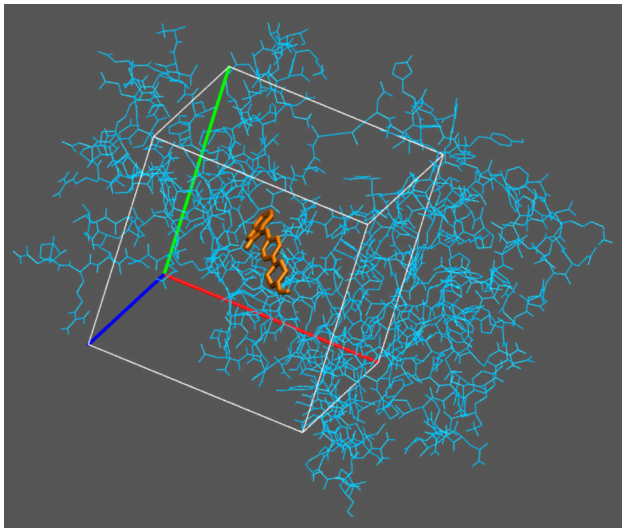


Figure 3. Grid centered on reference ligand.

3.1.2.3 Preparing the docking parameter file (DPF) and performing the docking calculation

We will first prepare the parameter file for the docking runs (DPF = docking parameter file) using the *prepare_dpf42.py* script from AutoDockTools:

```
$ pythonsh $MGLUTIL/prepare_dpf42.py -p ga_run=100 -l ligand_dock.pdbqt -r receptor.pdbqt -o ligand_dock.dpf
```

-p ga_run=100 indicates 100 independent docking runs
-l input with the ligand to be docked (PDBQT)
-r input receptor (PDBQT)
-o: output with the parameters for the docking runs (DPF)

By now, we should have created the following files[6], that are all we need to perform the conventional docking calculation:

- LIGAND PDBQT: *ligand_dock.pdbqt*
- MAPS: *receptor.A.map, receptor.C.map, receptor.HD.map, receptor.N.map, receptor.OA.map, receptor.e.map, receptor.d.map, receptor.maps.xyz, receptor.maps.fld*
- DPF: *ligand_dock.dpf*

Now lets run the conventional docking:

```
$ autodock4 -p ligand_dock.dpf -l ligand_dock.dlg
```

[6] Some example files are provided as a guide in the *$MGLUTIL/contrib/adbias/doc/examples/conventional* directory of the present tutorial (*outputs* subdirectory).

-p input DPF
-l output log file with the docking results

3.1.2.4 Analyzing the results

We will illustrate the analysis with our previously calculated result (see *ligand_dock.dlg* in the *$MGLUTIL/contrib/adbias/doc/examples/conventional/outputs* folder), but you may follow the steps with your own results.

First open the docking log file (*ligand_dock.dlg*) with a text editor and go to the "CLUSTERING HISTOGRAM" section. In our case, 6 solutions or ligand poses were found with the first ranked pose bearing both the best free energy of binding (-9.15 kcal/mol) and clustering population (35/100).

Let us look at the poses with ADT (or VMD[7]) and compare them with the reference one (*ligand.pdbqt*):

```
$ adt receptor.pdbqt ligand.pdbqt
```

> On the ADT bar, click on "Analyze", "Dockings", "Open…" and choose the *ligand_dock.dlg* file.
> Click on "Analyze", "Clusterings", "Show…".

An histogram of the energy distribution for the resulting poses, clustered according to their RMSD, will pop up. See Figure 4 as an example -yours is probably slightly different due to the stochastic nature of docking runs-.

[7] `$ vmd -m receptor.pdbqt ligand.pdb ligand_dock.dlg`

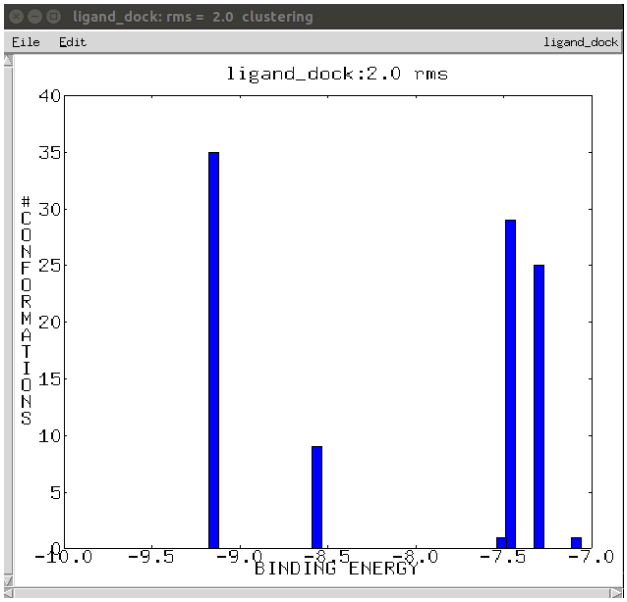


Figure 4. Binding energy distribution for 100 docking runs.

Click on each bar to see the 3D structure of the poses belonging to that cluster in the visualization panel. Figure 5 shows representative structures of the clusters ranked 1st and 5th.

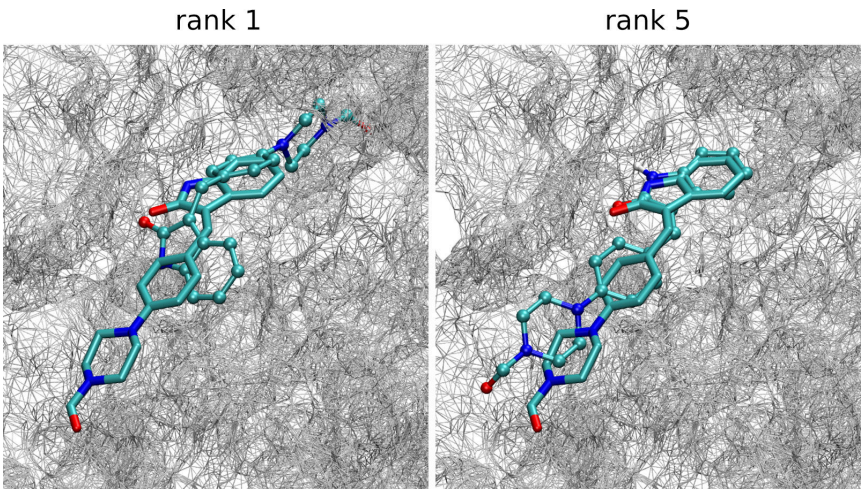


Figure 5. Ranked 1st and 5th poses (in CPK) compared to crystal structure (in cylinders).

As Figure 5 shows, the first ranked pose does not resemble the crystal pose (*ligand.pdbqt*) at all (Figure 5, left panel). The predicted ligand is buried deeper in the pocket forming different interactions with the receptor. It is not until the fifth ranked pose that the program finds the crystal pose (which we assume is the correct one), except for the outermost aliphatic ring that has no interaction with the protein (Figure 5, right panel).

We can also generate PDB files for each cluster representative pose with the *pdb_poses.py* script and calculate their RMSD against the crystal pose with the compute_rms_between_conformations.py scripts.

```
$ pythonsh $MGLUTIL/contrib/adbias/pdb_poses.py ligand_dock.dlg
```

This will generate rank1.pdb, rank2.pdb, …, rank6.pdb structures. Recall that rank5.pdb has the correctly predicted pose. Then, just calculate the rmsd for the first and fifth ranked poses:

```
$ pythonsh $MGLUTIL/compute_rms_between_conformations.py -f ligand.pdbqt -s rank1.pdb
$ pythonsh $MGLUTIL/compute_rms_between_conformations.py -f ligand.pdbqt -s rank5.pdb
```

The script will generate a file called *summary_rms_results.txt* which contains the RMSD values calculated for each pose with respect to the reference structure. The values obtained in our experiment were:

- RMSD rank 1 vs. reference = 10.61 Å
- RMSD rank 5 vs. reference = 1.68 Å

With all this information and the "CLUSTERING HISTOGRAM" section of the DLG, we may build the typical cluster population vs docking score plot shown in Figure 6 (remember that yours is probably slightly different).

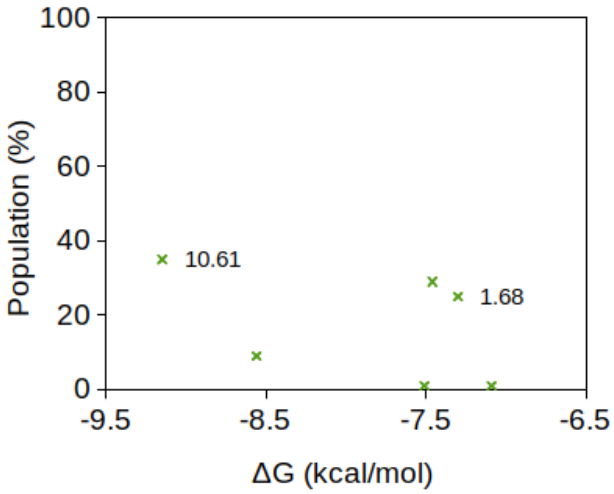


Figure 6. Cluster population vs. free energy of binding (AutoDock4 score) for the conventional redocking. RMSD (in Å) of selected poses against the reference structure is indicated as numbers besides the points.

It becomes clear from Figure 6 that the docking algorithm was unable to correctly discriminate the crystal pose (RMSD = 1.68 Å) from the other predicted poses, which are assumed to be false positives. In other words, the crystal pose has neither the minimum binding energy nor the highest popùlation.

We will now turn to the biased docking!

3.1.3 Biased docking (ligand-derived pharmacophore)

As previously stated, we are going to use known information about protein-ligand interactions in FGFr1 to bias the docking of the desired ligand (SU2). The main ligand-derived pharmacophore for this target is located in the so called *hinge* region, showing an hydrogen bond acceptor moiety interacting with the backbone NH of Ala564 and an hydrogen bond donor moiety interacting with the backbone C=O of Glu562 (see Figure 7).

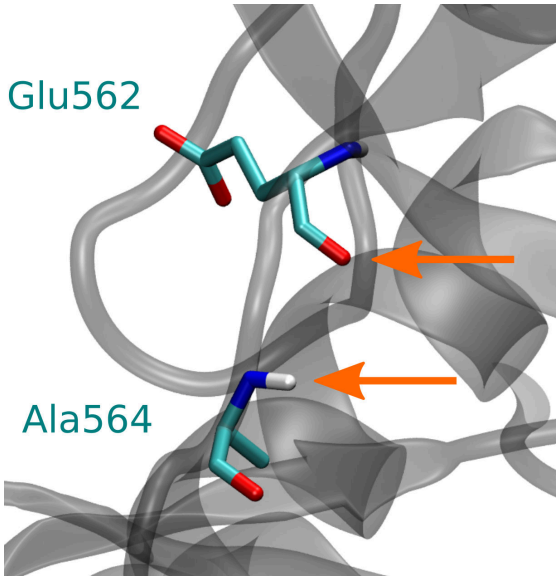


Figure 7. Hinge region of FGFr1.
Arrows indicate protein interactors obtained from ligand-derived pharmacophore.

NOTE. All the files for this section can be found in the *$MGLUTIL/contrib/adbias/doc/examples/biased_pharma* directory.

3.1.3.1 Generating the bias parameter file (BPF)

First we need to determine the bias position and parameters that will promote the formation of the hydrogen bonds with the FGFr1 residues marked in Figure 7. To get the bias position we will use the protonated receptor structure (*receptor_H.pdb*) and an additional script (*ideal_interaction_sites.py*) that calculates the location of *ideal* interactions for selected residues of interest from the receptor (Glu562 and Ala564 in our case).

```
$ pythonsh $MGLUTIL/contrib/adbias/ideal_interaction_sites.py -i
receptor_H.pdb -c A -r 562,564
```

-i input PDB file name containing the protein structure (with hydrogens)
-c protein chain ID (e.g. A)
-r residues IDs -comma separated- whose *ideal* interaction positions are to be calculated (562,564)

NOTE. The script requires Maestro or Amber atom names for Hs (e.g. HNE or HE, HH11, HH12, HH21 and HH22 for Arg hydrogens in the guanidinium group) and Biopython for PDB parsing (included in AutoDockTools).

Open the generated ideal interaction sites (*interaction_sites.pdb*):

```
$ adt receptor.pdbqt interaction_sites.pdb
```

You will see the structure shown in Figure 8 with the calculated *ideal* interaction sites for the indicated residues (both for their backbone and sidechain).

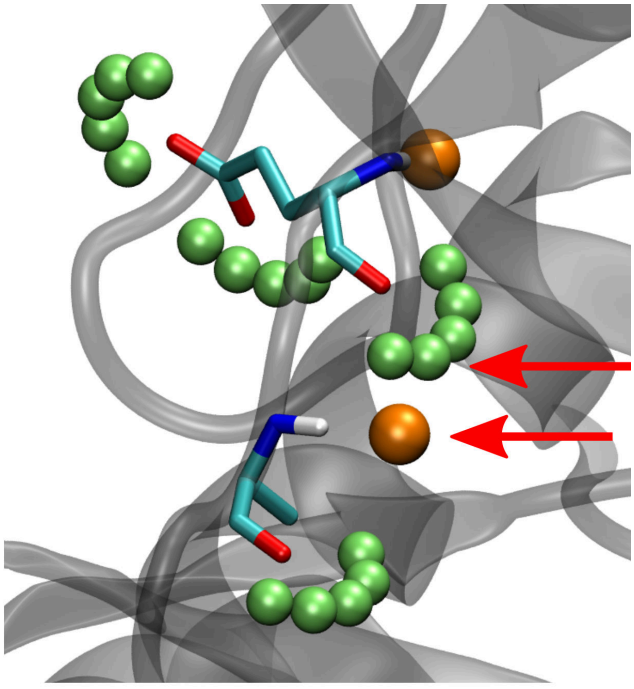


Figure 8. Ideal interaction sites. The receptor depicted as gray ribbons. The specified residues are shown as cylinders. The calculated interaction sites are shown as spheres: orange = hydrogen bond acceptor, green = hydrogen bond donor.

According to the known protein-ligand pharmacophore, the conserved interaction sites are the ones shown with red arrows in Figure 8, a hydrogen bond acceptor from Ala564 NH backbone and a hydrogen bond donor to Glu562 C=O backbone. Their coordinates are:

Acceptor site: (6.183, 2.487, 22.662)

Donor site: (5.100, 1.785, 20.019)

With the coordinates and type of sites, generate the input bias parameter file (*ligand_dock.bpf*) as follows:

```
x      y      z       Vset   r     type
5.100  1.785  20.019  -2.00  1.20  don
6.183  2.487  22.662  -2.00  1.20  acc
```

The V_{set} and radius (r) are user defined. We will use -2.0 kcal/mol, a strong bias, because we want those interactions to be satisfied as much as possible. If you want softer biases you may use lower energy values (*moderate bias = -1.5 to -1.0 kcal/mol, weak bias = -1.0 to -0.5 kcal/mol*). A radius of 1.2 Å will control the decay of the energy reward (see equation from Figure 1).

Use the *bias2pdb.py* script to convert the input bias file to a PDB file (*bias_sites.pdb*) and check that the sites coordinates are properly placed, by opening the file with ADT (Figure 9).

```
$ pythonsh $MGLUTIL/contrib/adbias/bias2pdb.py ligand_dock.bpf
```

```
$ adt receptor.pdbqt bias_sites.pdb
```

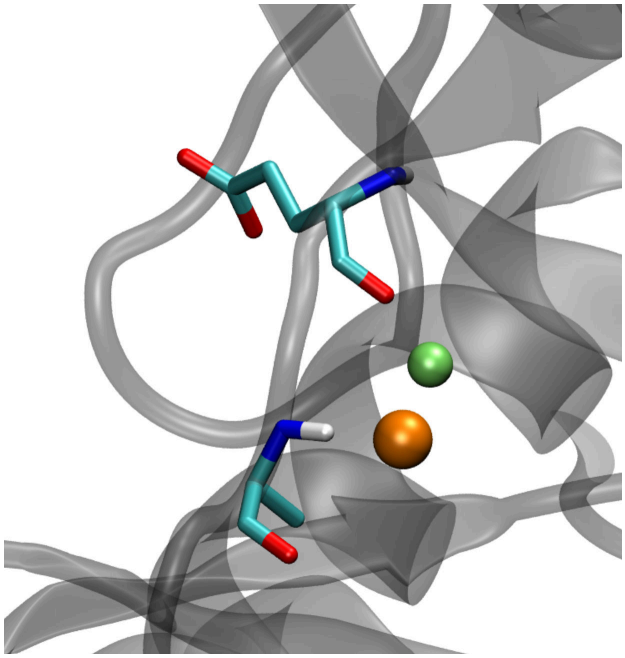


Figure 9. Bias sites shown as spheres:
orange = hydrogen bond acceptor, green = hydrogen bond donor.

The *bias_sites.pdb* has the acceptor bias site named as ACC residue, and the donor bias site named as DON residue. Their coordinates will set the locations where the corresponding energy maps will be modified.

3.1.3.2 Introducing the bias

For introducing the bias we will need to copy to the working directory the following files prepared in the conventional way (see 3.1.2):

- Energy maps (i.e., receptor.*.map, and receptor.maps.*);
- GPF;
- DPF;
- ligand PDBQT;

Additionally, we will need:

- AutoDock Bias script, *prepare_bias.py*
- input bias parameter file, *ligand_dock.bpf*

Execute AutoDock Bias (*prepare_bias.py*) to modify the energy maps and input files according to the bias parameter file prepared in the previous section:

```
$ pythonsh $MGLUTIL/contrib/adbias/prepare_bias.py -b ligand_dock.bpf -g
receptor.gpf -d ligand_dock.dpf
```

This will modify the acceptor and donor maps (OA and HD in the present case) and the input DPF (the maps to read will be the biased ones), so that everything is set to run the biased docking.

3.1.3.3 Verifying the bias in the maps and DPF

Use ADT to check the map modifications (Figure 10):

```
$ adt bias_sites.pdb
```

> Click the “C” circle on the bias_sites line for activating “sphere” representation.
> Click the “L” circle on the bias_sites line for deactivating “lines” representation.
> Right click the “C” circle > “Rendering” tab (lower panel) > “Points” representation.
> If necessary, zoom out with the middle scroll to see both sites.
> Open 3D Grid/Volume Rendering (Grid3D > Show Control Panel).

> Add the biased map you want to check (e.g. receptor.HD.biased.map) by clicking the “Add” (+) button and selecting the corresponding map.
> Click “Isocontour”.
> Set “max” to 0.00
> SHIFT + click in the panel to set the isocontour value.
> Move the bar and check that the **lower energy values are on the bias site**.
> Repeat for the remaining biased map (receptor.OA.biased.map).

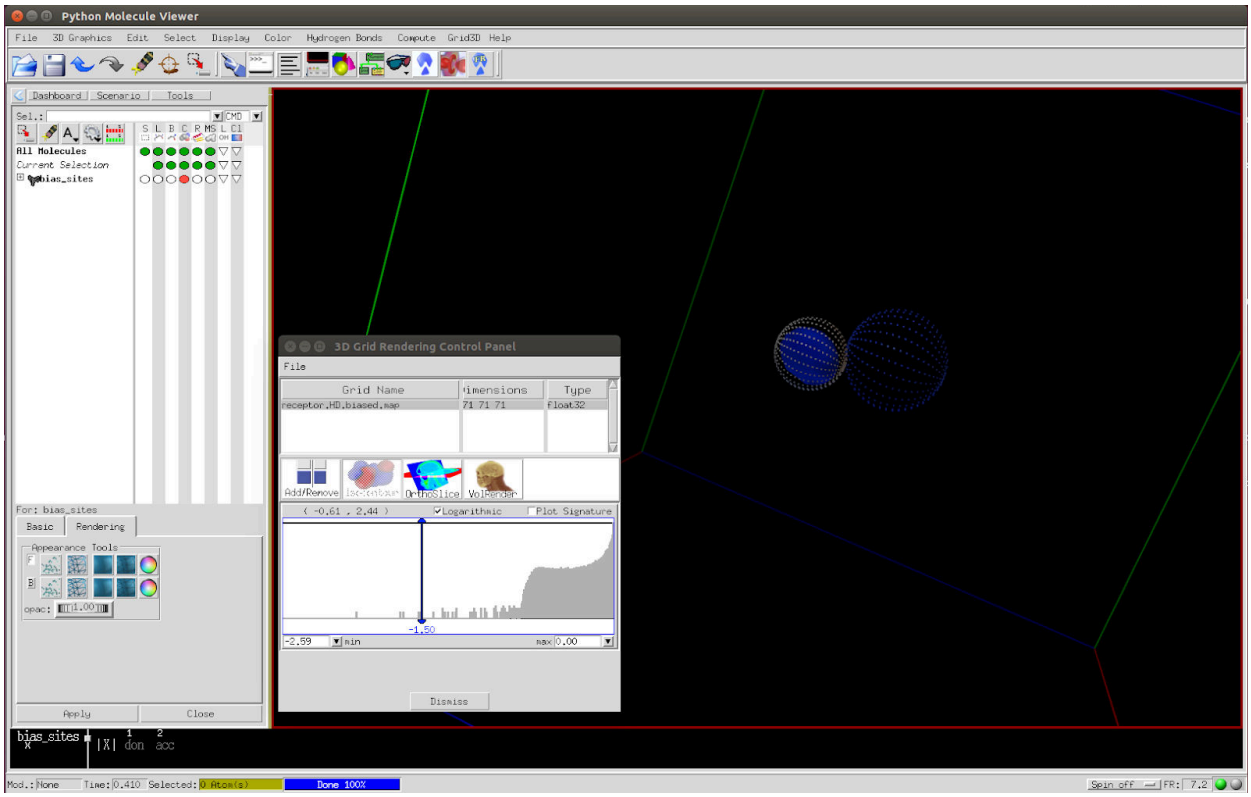


Figure 10. Grid Control Panel of ADT, showing receptor.HD.biased.map, at an isoenergetic value of -1.5 kcal/mol. The point representation is for the bias sites.

Now check the changes in the map lines between the original DPF and the one newly created *(ligand_dock.biased.dpf)*. You may use a text editor to open the two files, or text comparison programs such as *diff*, *vimdiff* or *meld*, for example:

```
$ vimdiff ligand_dock.dpf ligand_dock.biased.dpf
```

Verify that the new DPF reads the biased maps instead of original ones for HD and OA.

3.1.3.4 Performing the biased docking

The docking is executed as usual, indicating the biased docking parameter file:

```
$ autodock4 -p ligand_dock.biased.dpf -l ligand_dock.dlg
```

3.1.3.5 Analyzing the Results

As we did before, we will illustrate the analysis with previously calculated results (see the file *ligand_dock.dlg* in the folder *$MGLUTIL/contrib/adbias/doc/examples/biased_pharma/outputs*), but you may follow the steps with your own results.

First open the docking log file (*ligand_dock.dlg*) with a text editor and go to the "CLUSTERING HISTOGRAM" section. In our case, four different ligand poses were found, with the first ranked pose bearing both the best free energy of binding (-10.99 kcal/mol) and cluster population (55/100).

Let us look at the poses on ADT and compare them with the reference one (*ligand.pdbqt*):

```
$ adt receptor.pdbqt ligand.pdbqt
```

> On the ADT bar, click on "Analyze", "Dockings", "Open…" and choose the *ligand_dock.dlg* file.
> Click on "Analyze", "Clusterings", "Show…".

Again, a histogram of the energy distribution for the resulting poses, clustered according to their RMSD, will pop up (see Figure 11 as an example, yours is probably slightly different). Click on each bar to see the 3D structure of the poses belonging to each cluster in the visualization panel. Figure 12 shows representative structures of each cluster.

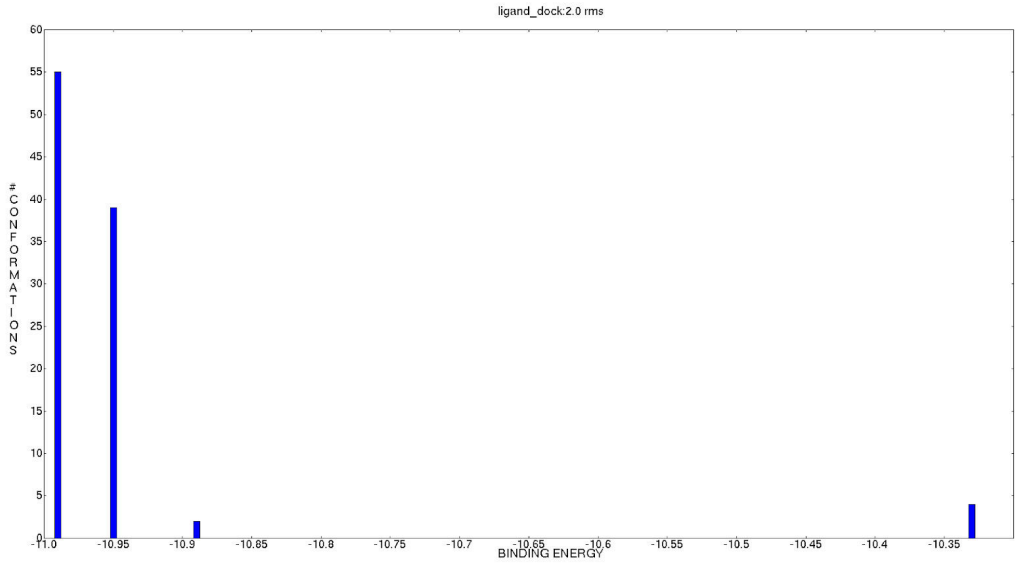


Figure 11. Binding energy distribution for 100 docking runs.

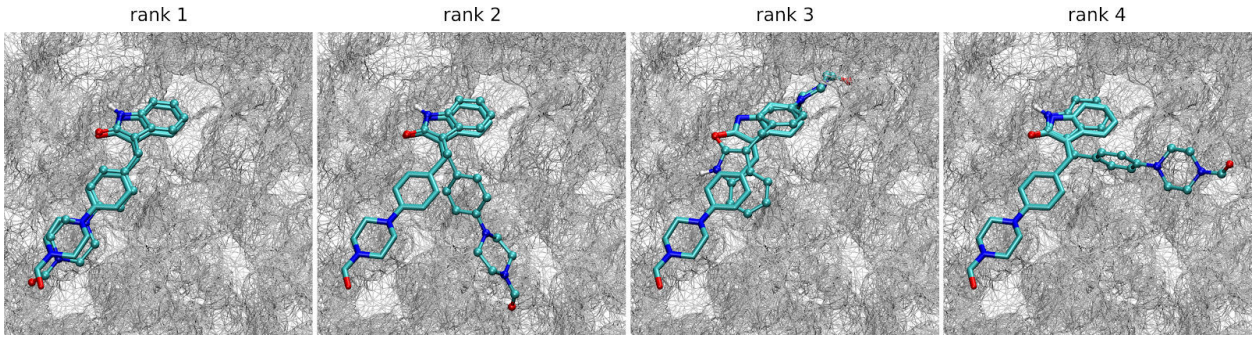


Figure 12. Predicted poses (in CPK) compared to crystal structure (in cylinders). The rank is specified above each panel.

As clearly shown in Figure 12, now the first ranked pose nicely resembles the crystal pose (*ligand.pdbqt*). The poses ranked 2 and 4 also satisfy the biased interactions, but establish a different interaction pattern with the rest of molecule. Finally, the third ranked pose is the same as the one ranked first with the conventional method (Figure 5, left panel).

Generate PDB files for each cluster representative pose with the *pdb_poses.py* script and calculate the RMSD of the first ranked pose against the crystal structure with the *compute_rms_between_conformations.py* script in the same manner that we did with the conventional method:

```
$ pythonsh $MGLUTIL/contrib/adbias/pdb_poses.py ligand_dock.dlg
```

```
$ pythonsh $MGLUTIL/compute_rms_between_conformations.py -f ligand.pdb -r rank1.pdb
```

The value obtained in our experiment was:

- RMSD rank 1 vs. reference = 1.15 Å

With all this information and the "CLUSTERING HISTOGRAM" section of the DLG, build the cluster population vs. docking score plot. An example with our obtained results is shown in Figure 13.

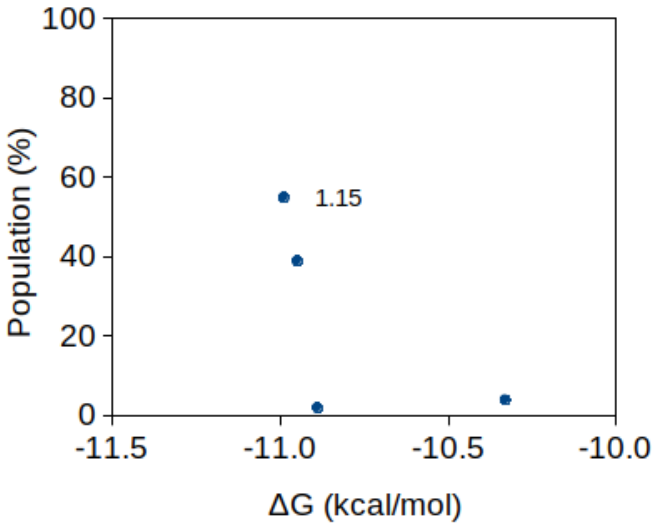


Figure 13. Cluster population vs. free energy of binding (AutoDock4 biased score) for the biased redocking. RMSD (in Å) of the first ranked pose against the reference structure is indicated besides the plotted point.

Conclusion. After applying the bias towards two well known protein-ligand interactions, the docking method was capable of predicting the crystal pose with best energy and highest population, thus effectively discriminating the correct pose among the false positives (compare with the conventional method, Figure 6).

3.2 Tutorial Part II: Introducing cosolvent derived bias

3.2.1 Aim

In this part of the tutorial we will show how to apply biases based on interaction sites derived from Molecular Dynamics (MD) in mixed solvents. Assume that we have run a MD simulation of the protein target in an aqueous solution of ethanol. In the binding site, for example, there will be regions where the ethanol preferentially interact with the protein surface, with either its hydroxyl (OH) end or its hydrophobic methyl (CH3) end. These regions are called *solvent sites* or interaction hot spots. The general idea will be to guide the docking of ligand groups capable of forming hydrogen bonds towards locations where the OH from ethanol preferentially interact with the protein and, at the same time, guide hydrophobic groups from the ligand to hydrophobic hot spots obtained from the methyl end of ethanol.
The solvent sites that will define the bias position and parameters (V_{set} and r) can be obtained using any of the available strategies, such as WATCLUST, Watermap, MDMix, SILCS, MixMD, etc. In the present example we will use sites determined by WATCLUST. The chosen target is the same as in the previous part of the tutorial, FGF receptor 1 kinase domain (FGFr1), PDB ID 1agw, and we will again dock the co-crystallized ligand SU2.

3.2.2 Standard AutoDock4 Run

The standard AutoDock4 run is the same as that thoroughly explained and performed previously in section 3.1.2.

3.2.3 Biased docking (solvent sites)

3.2.3.1 Generating the bias parameter file (BPF)

To determine the bias position and parameters we previously performed three short 20 ns MD simulations of FGFr1 in explicit water/ethanol mixture and used WATCLUST to determine hydrophilic and hydrophobic ethanol solvent sites. The users are referred to WATCLUST tutorials (http://sbg.qb.fcen.uba.ar/watclust) if they want to determine the sites by themselves. Here we provide the obtained results for a brief analysis.

Available files in the *$MGLUTIL/contrib/adbias/doc/examples/biased* directory of this tutorial:

Reference FGFr1 receptor structure = *receptor_H.pdb*
WATCLUST output file = *solvent_sites.pdb*

The *solvent_sites.pdb* file has the interaction sites determined with ethanol probe for this target.

H bond acceptor sites > atom name O1
H bond donor sites > atom name H1
Hydrophobic sites > atom name C1

As previously described,[8] the hydrophobic sites from ethanol capture the location of aromatic rings from common ligands and will therefore be used as aromatic biases.

Visualize these sites with ADT (Figure 14):

```
$ adt receptor.pdbqt ligand.pdbqt solvent_sites.pdb
```

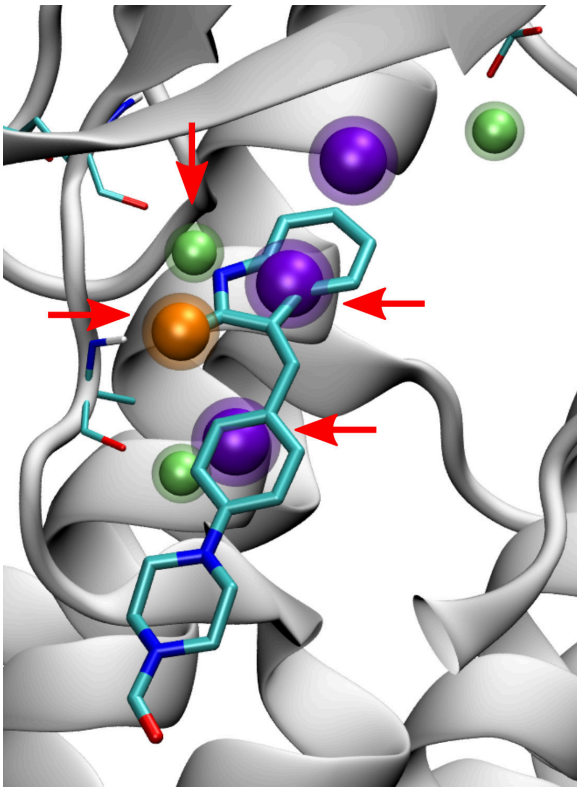


Figure 14. Solvent sites derived from Ethanol/Water MD. Hydrogen bond donor sites are depicted in green, acceptors in orange and hydrophobic/aromatic in purple.

[8] Arcon J.P. *et al.* Molecular Dynamics in Mixed Solvents Reveals Protein–Ligand Interactions, Improves Docking, and Allows Accurate Binding Free Energy Predictions. *J. chem. Inf. Model.* **2017**, 57 (4), 846-863.

As can be seen from Figure 14, the orange acceptor site and one of the green donor sites, both marked with red arrows, coincide with appropriate ligand groups (and with the sites derived from ligand pharmacophore, see Part I of the tutorial). There are also 2 purple hydrophobic/aromatic sites nicely overlaying aromatic rings from the ligand, also marked with red arrows. It should be noted, however, that there are 3 extra sites that may be considered false positives for this ligand (the ones not marked with a red arrow in Figure 14). The robustness of the method will be to improve the docking of this ligand despite the presence of those other bias sites.

Generate the input bias parameter file (*ligand_dock.bpf*) with the coordinates and type of sites extracted from the *solvent_sites.pdb*:

```
x      y      z       Vset   r     type
5.100  1.785  20.019  -2.72  1.20  don
5.962  7.220  13.075  -2.29  1.10  don
9.459  2.075  24.527  -2.27  0.60  don
6.183  2.487  22.662  -2.28  0.80  acc
4.244  6.136  17.593  -2.16  1.30  aro
6.684  3.905  19.437  -2.09  1.20  aro
9.365  3.153  23.215  -2.08  1.40  aro
```

The V_{set} and radius (r) comes from the MD derived solvent sites. The V_{set} is the free energy of binding of the ethanol site and the radius is related to the dispersion and translational entropy of the site (see WATCLUST web page or ref. on footnote 8 for details on their calculation).

3.2.3.2 Introducing the bias

For introducing the bias we will need to copy to the working directory the following files prepared in the conventional way (see 3.1.2):

- Energy maps (i.e., receptor.*.map, and receptor.maps.*);
- GPF;
- DPF;
- ligand PDBQT;

Additionally, we will need:

- AutoDock Bias script, *prepare_bias.py*
- input bias parameter file, *ligand_dock.bpf*

Execute AutoDock Bias (*prepare_bias.py*) to modify the energy maps and input files according to the bias parameter file prepared in the previous section:

```
$ pythonsh $MGLUTIL/contrib/adbias/prepare_bias.py -b ligand_dock.bpf -g
receptor.gpf -d ligand_dock.dpf
```

This will modify the acceptor and donor maps (OA and HD in our case) and create a new map for the aromatic bias (AC). It will also modify the DPF and the ligand PDBQT to insert a dummy atom in the center of aromatic rings (see 3.2.3.3).

3.2.3.3 Verifying the bias in the maps, DPF and ligand PDBQT

Use ADT to check the map modifications (Figure 15):

```
$ adt solvent_sites.pdb
```

> Click the “C” circle on the solvent_sites line for activating “sphere” representation.
> Click the “L” circle on the solvent_sites line for deactivating “lines” representation.
> Right click the “C” circle > “Rendering” tab (lower panel) > “Points” representation.
> If necessary, zoom out with the middle scroll to see all the sites.
> Open 3D Grid/Volume Rendering (Grid3D > Show Control Panel).
> Add the biased map you want to check (e.g. receptor.AC.biased.map) by clicking the “Add” (+) button and selecting the corresponding map.
> Click “Isocontour”.
> Set “max” to 0.00
> SHIFT + click in the panel to set the isocontour value.
> Move the bar and check that the **lower energy values are on the solvent sites** (the AC map should show a spherical pattern).
> Repeat for all the biased maps.

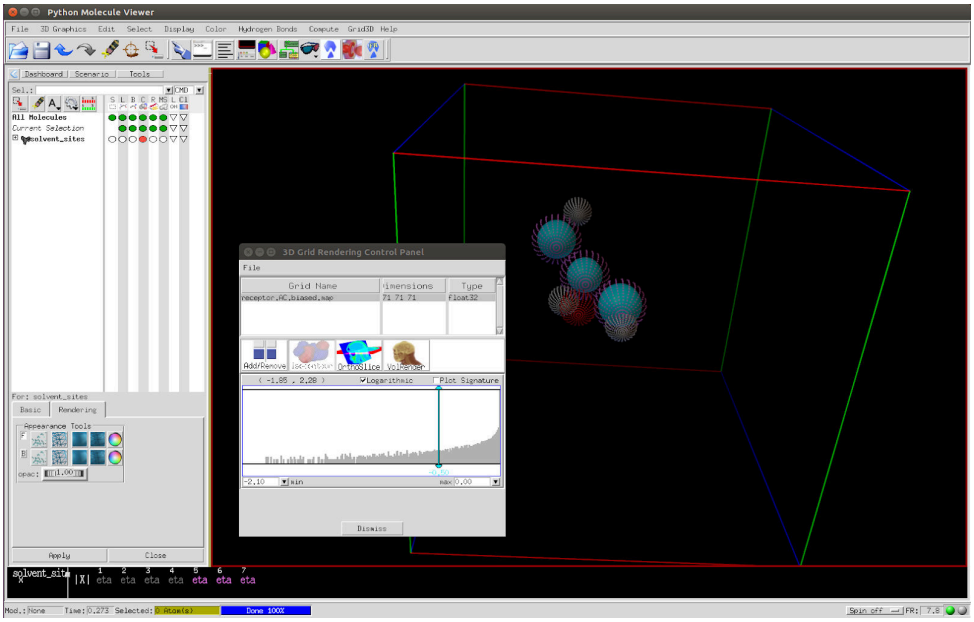


Figure 15. Grid Control Panel of ADT, showing receptor.AC.biased.map, at an isoenergetic value of -0.5 kcal/mol. The point representation is for the bias sites.

Now check the changes between the original DPF and the one newly created *(ligand_dock.biased.dpf)*. You may use a text editor to open the two files, or text comparison programs such as *diff*, *vimdiff* or *meld*, for example:

```
$ vimdiff ligand_dock.dpf ligand_dock.biased.dpf
```

- First line *parameter_file ad4_arom_params.dat* reads the created file with parameters for the AC dummy atom type.
- AC is added as a ligand atom type (*ligand_types*) with its corresponding map.
- Biased maps are read instead of original maps for HD and OA.
- The docking ligand is updated (*move ligand_dock.dum.pdbqt*)

Finally, check with ADT (or PYMOL[9]) that *ligand_dock.dum.pdbqt* is the same ligand as the original one (*ligand_dock.pdbqt*), but with dummy atoms added in the center of each aromatic ring:[10]

```
$ adt ligand_dock.pdbqt ligand_dock.dum.pdbqt
```

To sum up the effect of the aromatic bias, we just saw that the AC energy map has lower energies in the locations corresponding to hydrophobic ethanol sites. Therefore, aromatic rings from the

[9] `$ pymol ligand_dock.pdbqt ligand_dock.dum.pdbqt`
[10] The aromatic rings are the ones detected by OpenBabel.

ligands (with AC dummy atoms in their center of mass) will be guided during the docking towards those same locations.

3.2.3.4 Performing the biased docking

The docking is executed as usual, indicating the biased docking parameter file:

```
$ autodock4 -p ligand_dock.biased.dpf -l ligand_dock.dum.dlg
```

3.2.3.5 Analyzing the Results

As we did in Part I of the tutorial, we will illustrate the analysis with previously calculated results (see *ligand_dock.dum.dlg* in the *$MGLUTIL/contrib/adbias/doc/examples/biased/outputs* folder), but you may follow the steps with your own results.

First open de docking log file (*ligand_dock.dum.dlg*) with a text editor and go to the "CLUSTERING HISTOGRAM" section. In our case, three different ligand poses were found. The first ranked pose has both the best free energy of binding (-13.22 kcal/mol) and population (84/100).

Let's look at the poses with ADT and compare them with the reference one (*ligand.pdbqt*):

```
$ adt receptor.pdbqt ligand.pdbqt
```

> On the ADT bar, click on "Analyze", "Dockings", "Open…" and choose the *ligand_dock.dum.dlg* file.
> Click on "Analyze", "Clusterings", "Show…".

Again, a histogram of the energy distribution for the resulting poses, clustered according to their RMSD, will pop up (see Figure 16 as an example, yours is probably slightly different). Click on each bar to see the 3D structure of the poses belonging to each cluster in the visualization panel. Figure 17 shows representative structures of all the clusters.

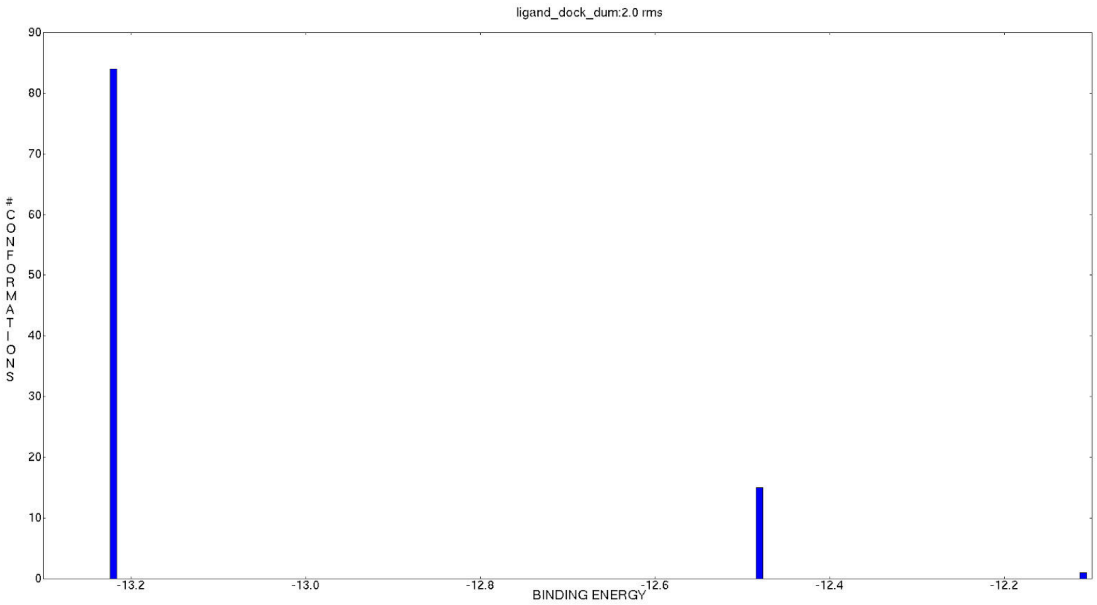


Figure 16. Binding energy distribution for 100 docking runs.

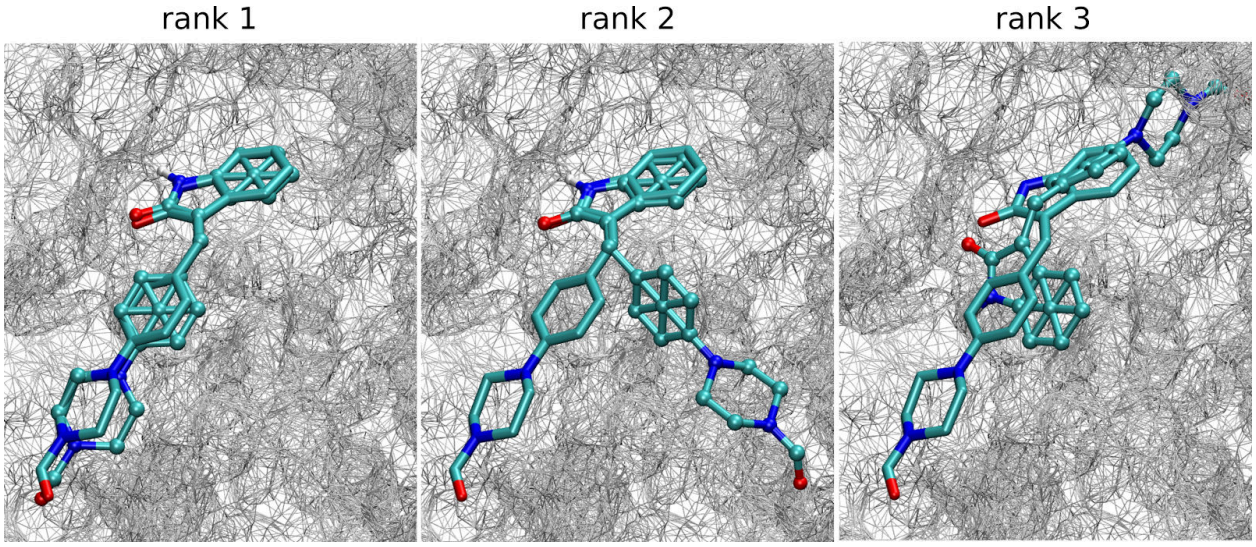


Figure 17. Predicted poses (in CPK) compared to crystal structure (in cylinders). The rank is specified above each panel.

Figure 17 shows that the first ranked pose nicely resembles the crystal pose (*ligand.pdbqt*). The second ranked pose does not satisfy one of the aromatic biases with the consequent loss in energy. The third ranked pose is the same as the one ranked first with the conventional method (Figure 5, left panel).

We now want to calculate the RMSD of the first ranked pose against the crystal structure in the same manner that we did with the conventional method. First generate PDB files for each cluster representative pose:

```
$ pythonsh $MGLUTIL/contrib/adbias/pdb_poses.py ligand_dock.dum.dlg
```

This will generate rank1.pdb, rank2.pdb and rank3.pdb.

Manually delete the dummy atoms from rank1.pdb with a text editor and get the RMSD:

```
$ pythonsh $MGLUTIL/compute_rms_between_conformations.py -f ligand.pdbqt -s rank1.pdb
```

The value obtained in our experiment was:

- RMSD rank 1 vs. reference = 1.14 Å

With all this information and the "CLUSTERING HISTOGRAM" section of the DLG, build the cluster population vs. docking score plot. An example with our obtained results is shown in Figure 18.

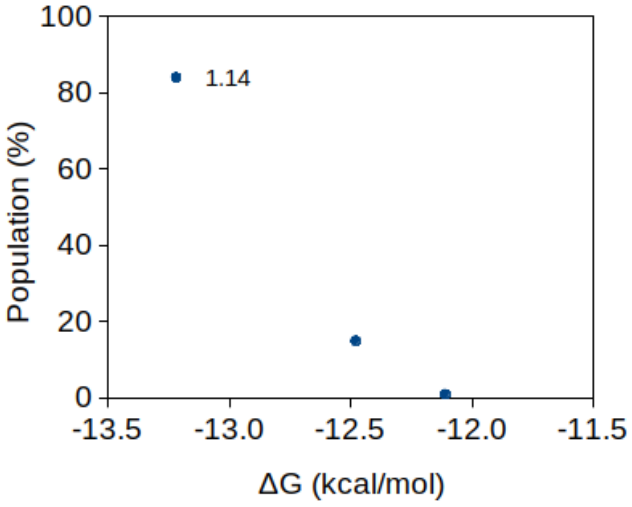


Figure 18. Cluster population vs. free energy of binding (AutoDock4 biased score) for the biased redocking. RMSD (in Å) of the first ranked pose against the reference is indicated besides the plotted point.

Conclusion. After applying the bias towards the solvent sites from MD, the docking method was capable of predicting the crystal pose with best energy and highest population, thus effectively discriminating the correct pose among the other false positives (compare with the conventional

method, Figure 6). In addition, the aromatic bias aided in energy separation between the crystal pose and the other ones (compare with the docking experiment just with hydrogen bond bias, Figure 13).