Using AutoDock 4 and AutoDock Vina with AutoDockTools: A Tutorial

Written by Ruth Huey, Garrett M. Morris and Stefano Forli

Contents

Introduction

This tutorial will introduce you to docking using the **AutoDock** suite of programs. We will use a Graphical User Interface called **AutoDockTools**, or **ADT**, that helps a user easily set up the two molecules for docking, launches the external number crunching jobs in **AutoDock**, and when the dockings are completed also lets the user interactively visualize the docking results in 3D. **ADT** is available here: mgltools.scripps.edu/downloads. AutoDock Vina (Vina) is available here: http://vina.scripps.edu. AutoDock4 (AD4) along with other helpful information is available here: http://autodock.scripps.edu.

Before We Start…

And only if you are at The Scripps Research Institute… These commands are for people attending the tutorial given at Scripps. Start ADT if you find its icon in the Dashboard. Alternatively start it from the command line by opening a Terminal window and typing this at the UNIX, Mac OS X or Linux prompt:

```
% cd Desktop

% cd tutorial

% adt
```

In either case, set the startup directory for today's tutorial:
- click File in the top left corner of AutoDockTools GUI
- click Preferences in the dropdown menu
- click Set in the menu which opens
- In the General tab of the Set User Preferences widget which opens look down to find Startup Directory
- Verify that the Startup Directory entry to ~/Desktop/tutorial (or /home/user#/Desktop/tutorial)

Pmv Basics

Starting AutoDockTools **adt**, either by clicking on the adt icon or by running the "adt" script from the command line, opens a GUI containing a docked camera and control panels. Place your cursor over the icons in the **Tool bar** to find out more …

Menu bar

Tool bar

Dashboard

3D Viewer

Sequence Viewer

Info bar

4

Exercise One: Preprocessing a PDB File

Here's how	*Here's why*

1. In the **Dashboard**, place the cursor over `All Molecules` and press `Right MouseButton`.

   In the **Read Molecule:** filebrowser which opens, click on `hsg1.pdb` and press `Open`.

   We will represent those actions like this:
   `DB`→`PMV Molecules`→`RightMB`
   `Read Molecule:` → `hsg1.pdb`→`Open`

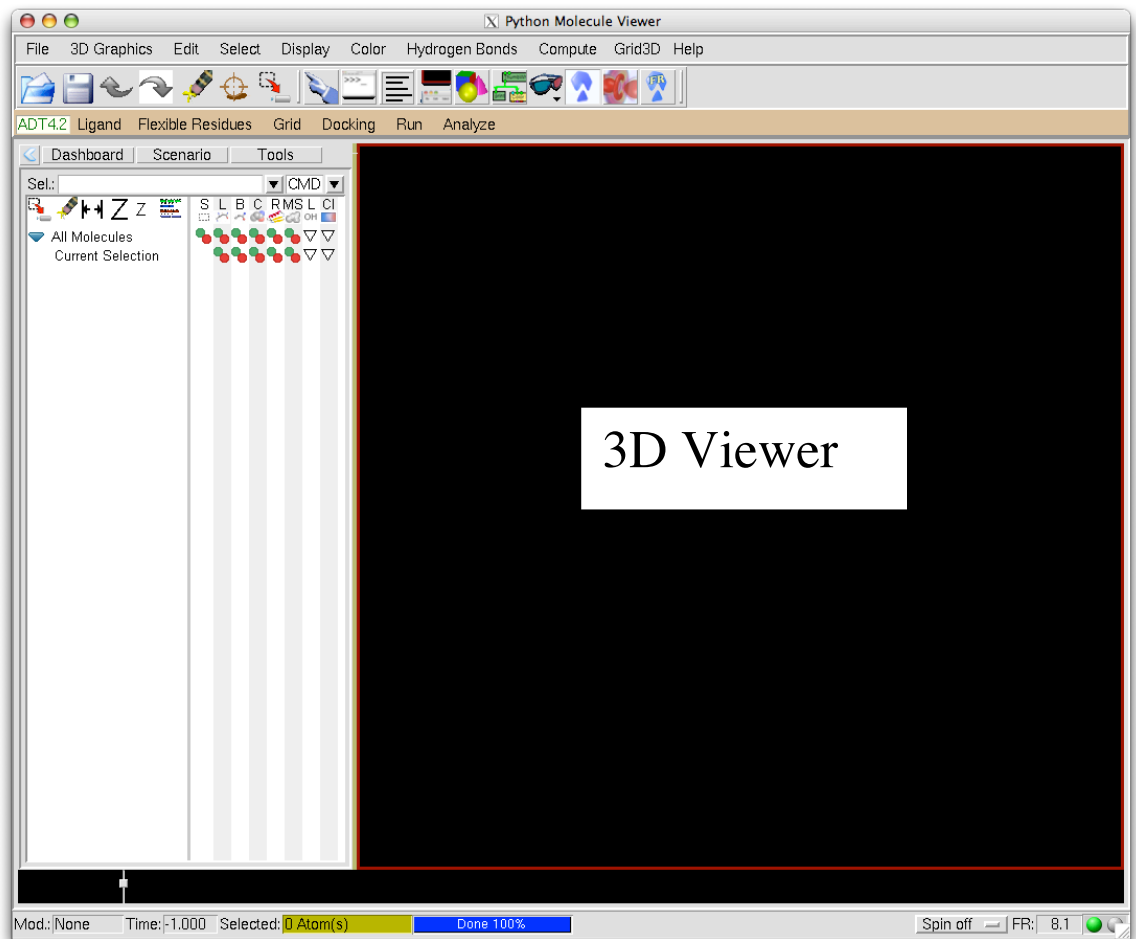Before formatting a molecule for AutoDock, various potential problems must be resolved. These can include missing atoms, chain breaks, alternate locations *etc*. Here we remove crystallographic waters from **hsg1**. Note bonds between bonded atoms are represented as lines while non-bonded atoms, here the oxygen atoms of water molecules, are shown as small squares. We will remove them in Step 4.

2. In the **Dashboard**, click on the **inverted triangle** under **Cl** to display color choices. Click on **By atom type** in the drop-down list:
   `DB`→`Cl`⬇`▽`→`By atom type`

   Click `Dismiss` to close color list.

Now the lines representing the bonded atoms are colored according to element, as follows:
- ⬤ Carbons that are aliphatic (C) - gray,
- 🟢 Carbons that are aromatic (A) - green,
- 🔵 Nitrogens (N) - blue,
- 🔴 Oxygens (O) - red,
- 🟡 Sulfurs (S) - yellow,
- ⚪ Hydrogens (H) - white.

3. Click on `Select` in the **Menu bar (MB)** to dropdown a menu. In it, locate and click on **Select From String:**
   `MB`→`Select`→`SelectFromString`→`LeftMB`

   In the **Select From String** widget, type **HOH*** in the **Residue:** entry and ***** in the **Atom** entry. Click on `Add` then click `Dismiss` to close.

   `Select From String`→`Residues:`→type in `HOH*`
   →`Atoms:` → type in `*`
   →`Add`
   →`Dismiss`

Use `Select From String` to select nodes using strings for the Molecule, Chain, Residue and/or Atom levels. These strings can be names, numbers, ranges or lambda expressions that are evaluated to build a set and can contain regular expressions including wild cards such as * which matches anything. Here we want to select all atoms (*) in residues named HOH*. Verify that you see `Selected:` `127 Atom(s)` with a yellow background in the center of the Info bar at the bottom of the ADT window.

4. `Edit` → `Delete` → `Delete Selected Atoms`
   `WARNING`→`CONTINUE`.

Here you will be asked to confirm this action because deleting nodes cannot be undone.

5. `Edit` → `Hydrogens` → `Add`

   `add All Hydrogens`→`noBondOrder`→`yes`→`OK`

NOTE: You must **add all hydrogens** to a molecule *before* you select it to be either the ligand or the receptor.

6. `DB`→`L`⬇`hsg1` → `O`

   .

Pmv Mouse and Keyboard Bindings

If you have a three-button mouse, the mouse buttons can be used alone or with a modifier key to perform different operations. For example, to 'zoom the molecule' (i.e. make it look bigger or smaller) in the viewer window, press and hold down the <Shift> key and then **click and drag** with the middle mouse button pressed down also.

To summarize what the mouse buttons do:

Modifier	Left	Middle	Right
None	*Rotate*	*Scale*	*Translate left/right (X) and up/down (Y)*
Shift	*Select*	*Scale*	*Translate in/out (Z)*

As you translate a molecule out in the **Z** dimension, it will disappear into **fog** which is used for depth-cueing.

You can press the following keys when the cursor is in the viewer window to change your view of the molecule:

Key	Action
R	*Reset view*
N	*Normalize – scale molecule(s) so all visible molecules fit in the viewer*
C	*Center on the center of gravity of all the molecules*
D	*Toggle on/off Depth-cueing (blends molecule into background farther away)*
T	*Toggle between transform root (i.e. scene) and transform the Viewer's current object.*

By default, the Viewer's current object is **root** so you will not see any changes here if you toggle between *transform root* and *transfom current object*. The DejaVu GUI lets you change the current object.

Exercise Two: Preparing a Ligand for AutoDock.

Here's how	Here's why

1. Ligand → Input → Open...

   In the **Ligand File for AutoDock 4**: widget, click on the small bar at the right of PDBQT files: (*.pdbqt) to display a list of file type choices. Click on all files: to show all the files in this directory and choose ind.pdb . Click on Open . ADT automatically formats the atoms in the file opened here by adding an autodock type and a charge to each.

Formatted ligand files for AutoDock must be in pdb**qt** format and contain atom types supported by AutoDock plus extra records that specify rotatable bonds:

q: If each ligand atom already has a 'partial charge' those charges are used. If not or if each of the charges is zero, ADT computes **Gasteiger** charges for the entire ligand. For this calculation to work correctly, the molecule *must* already have hydrogen atoms added, including both polar and non-polar ones, prior to this step.

t: ADT assigns a 'autodock type' to each atom. For most elements, an atom's type is the same as its element. Two kinds of special types exist to distinguish (**1**) atoms which can hydrogen bond (**2**) aromatic carbons:

   1:By default, all hydrogens are assumed able to form a single hydrogen bond so are assigned type '**HD**' and all oxygens are assumed able to accept hydrogen-bonds so are assigned type '**OA**'. Sulphur atoms which can hydrogen-bond are assigned type '**SA**' while those that cannot are assigned type '**S**'. Nitrogens that can accept hydrogen bonds are assigned type '**NA**' while those that cannot are assigned '**N**'. In this ligand, the type of atom N5 in the heterocycle is '**NA**' while that of all other nitrogens is '**N**'.

   2: **Only carbons in planar cycles are treated as aromatic by AutoDock.** If the angle between normals to adjacent carbons in the ring is less than 7.5° for atoms in the ring, the atoms are assigned type "**A**". [Note: a look-up dictionary is used to determine aromatic carbons in peptide ligands].

Click on OK to close the summary

This summary lists the type of charge used; the numbers of non-polar hydrogens merged, of aromatic carbons, of rotatable bonds found + the number of torsional degrees of freedom detected (TORSDOF) as well as the 'total non-integral charge error' which is the amount by which the sum of the per-atom charges differs from an integer.

2. `Ligand` → `Torsion Tree` → `Detect Root...`

ADT identifies a 'central' atom in the ligand for use as the root and marks it with a green sphere. This is the atom with the *smallest* **largest sub-tree**. In the case of a tie, if either atom is in a cycle, it is picked to be root. If neither atom is in a cycle, the first found is picked. As you might imagine, this can be a slow process for large ligands. The rigid portion of the ligand includes this root atom *and* all atoms connected to it by non-rotatable bonds (which we will examine in the next section.) At this point in our example, the root portion includes only the best root atom, atom **C11** because all its bonds to other atoms are rotatable so there is no root expansion to see. If some bonds from C11 to other atoms were inactivated, you could show the entire rigid root portion with `Ligand` → `Torsion Tree` → `ShowRootExpansion`. Hide only the marker on the root with:
`Ligand` → `Torsion Tree` → `Show/Hide Root Marker`

3. `Ligand` → `Torsion Tree` → `Choose Torsions`

The **Torsion Count** widget displays the number of currently active bonds. **14/32** on the widget indicates that 14 are currently active out of the maximum allowed by AutoDock which is 32. Bonds that are currently active are colored green, bonds that cannot be rotated are colored red while bonds that could be rotated but are currently marked as inactive are colored purple. In AutoDock only single bonds which are not in cycles and not to leaves can be rotated. ADT determines which bonds could be rotated. You set which of these are to be rotatable by inactivating the others in the viewer. By default, amide bonds amide bonds are treated as non-rotatable. Note that two bonds have been inactivated, the bond between atoms **N2;6** and **C3;4** and that between atoms **C21;26** and **N4;28**. Notice that the current total number of rotatable bonds is **14**. Before you close this widget, be careful to leave all the bonds except the two amide bonds active.

> **CAUTION**! Trying to make all bonds between selected atoms inactive when there is no specific selection in **ind** will cause a problem because then the selection is expanded to include everything. This would involve processing all the bonds in **hsg1**.

set all bonds active except the 2 amide bonds, `Done`

4. `Ligand` → `Torsion Tree` → `Set Number of Torsions..`

`fewest atoms`, *type-in 6*, → `Enter`, → `Dismiss`

JUST for today's tutorial! In order to get good docked results in a small amount of time we are using this *optional* feature to reduce the number of active bonds and to keep active bonds those which move the fewest atoms. This reduces the complexity of the search problem.

5. `Ligand` → `Output` → `Save as PDBQT...`

   type-in **ind.pdbqt** , → `Save`.

Each AutoDock4 calculation requires at least 4 input files: one for the ligand, one for the receptor as well as separate parameter files for AutoGrid and AutoDock. **ind.pdbqt** is the first of these four input files.
Each AutoDock **vina** calculation uses the same ligand and receptor files along with an optional configuration file. A configuration file will be prepared in an exercise to follow.

6. `Ligand`→`Torsion Tree`→`Show/Hide RootMarker`

   DB→`Show/Hide`↧`ind` → `○` → `LeftMB`

   DB→`Show/Hide`↧`hsg1` → `○` → `LeftMB`

If it is visible, hide the root marker before going on to the next exercise.

Undisplay the ligand by clicking on the gray rectangle under **Show/Hide** for **ind** in the Dashboard.

Redisplay the receptor by clicking on the gray rectangle under **Show/Hide** for **hsg1** in the Dashboard. To reset the view, place the cursor over the viewer and type **rnc**

Exercise Three: Preparing a Macromolecule.

Here's how	*Here's why*

1. `Grid` → `Macromolecule` → `Choose...`

   `hsg1` → `Select Molecule`.

   type-in **hsg1.pdbqt** , → `Save`

Selecting the macromolecule in this way causes the following sequence of initialization steps to be carried out automatically:

• ADT checks that the molecule has charges. If not, it adds Gasteiger charges to each atom. Remember that all hydrogens must be added to the macromolecule before it is chosen. If the molecule already had charges, ADT would ask if you want to preserve the input charges instead of adding Gasteiger charges.

• ADT merges non-polar hydrogens unless the user preference **adt_automergeNPHS** is set not to do so.

`File` → `Preferences` → `Modify Defaults` → `AutoDockTools`

• ADT also determines the types of atoms in the macromolecule. AD4 and Vina can accommodate any number of atom types in the macromolecule.

Exercise Four: Setting the Search Space

Here's how   **Here's why**

1. `Grid` → `Grid Box...`

We will use this **Grid Options Widget**: to set the location and extent of the 3D area to be searched during the AutoDock experiment. The search space is defined by specifying a center, the number of points in each dimension plus the spacing between points.

2. `60`, `60`, `60`

> **Note:** clicking with the right mouse button on a **thumbwheel** widget opens a box that allows you to type in the desired value directly. Like many other entry fields in ADT, this updates only when you press **<Enter>**.

Increase the number of points in each dimension to **60** This results in a total of **226981** because each dimension is incremented by 1 to provide a central point. Move the **Grid Options** panel to the side to see the box while you adjust its size.

3. *x* center → `2.5`
   y center → `6.5`
   z center → `-7.5`

Set the center of the search space to **2.5**, **6.5**, **-7.5**. Be careful to use **negative** 7.5 for the z-center

4. `File` → `Close saving current`

> **Setting up the Search Space:**
> If you were setting up a docking using flexible residues (Exercise Three B, p 19), make sure the specified **receptor** file is hsg1_rigid.pdbqt. **Grids must be calculated using a file for the molecule without the moving residues.**
> ALSO, be sure to **increase the number of points in the z dimension to 66** to allow room for motion of the flexible **ARG8** residues. This sets the z dimension to 24.75Angstrom to use with AutoDock Vina.

Hide the gridbox and close the widget while keeping the current search space values. [The alternative `Close w/out saving` discards your changes.]

FYI: `Center`, `View` and `Help` menubuttons at the top:

→ `Center` menu contains 4 shortcuts for setting the center of the grid box: → `Pick an atom`,
                                        → `Center on ligand`,
                                        → `Center on macromolecule`
                                        → `On a named atom`.

`View` menu lets you change the visibility of the box using `Show box`, and whether it is displayed as lines or faces, using `Show box as lines`. This menu also allows you to show or hide the center marker using `Show center marker` and to adjust its size using `Adjust marker size`.

AD4 Exercise Five: Preparing the AutoGrid Parameter File

Here's how

Here's why

1. `Grid` → `Set Map Types` → `Choose Ligand`
   `Choose Ligand`→`ind`,→ `Select Ligand`

AutoDock does not use the receptor directly. Instead it uses a set of pre-calculated 'maps' produced by AutoGrid. The set of maps must include one map for each atom type in the ligand(s) plus two extras: a 'd' map for desolvation and 'e' for electrostatics. AutoGrid records the interaction energy of a probe atom of a specific element at each point in a 3-D grid around the rigid receptor in the corresponding gridmap file. During the AutoDock calculation the energetics of a particular ligand configuration is evaluated using the values from the gridmaps. The types of maps depend on the types of atoms in the ligand(s). Thus one way to specify the types of maps is by choosing a ligand.

[**Optional** 2. `Grid` → `Set Map Types` → `Choose FlexRes....`
   `Choose Flexible Residues from...`→`hsg1`.→
   `Select molecule providing flexible residues`]

If you want to include some flexible residues in the receptor in your docking experiment, you must specify them for the AutoGrid4 calculation here. The procedure for specifying the flexible residues and creating both a **rigid** and a **flexible** file for the receptor is explained in Exercise Three B (optional) p19 .

3. `Grid` → `Output` → `Save GPF...`

   type-in **hsg1.gpf** , → `Save`

At this point in the tutorial, we have set the three pieces of information required for AutoGrid. These are (1) the rigid receptor filename, (2) the location and extent of the search space and (3) the atom types in the flexible molecule(s) to be docked. Thus now we can write the parameter file for AutoGrid. The convention is to use '**.gpf**' as the extension.

Optional 4. `Grid` → `Edit GPF...`
   `OK` or `Cancel`

If you have just written a grid parameter file, it opens in an editing window. If not, you can pick one to read in and edit via the `Read` button. If you make any changes to the content of the grid parameter file, you can save the changes via the `Write` button.
Here `Edit GPF` will open the file we wrote in step 3. Have a look without changing anything then just close this widget.
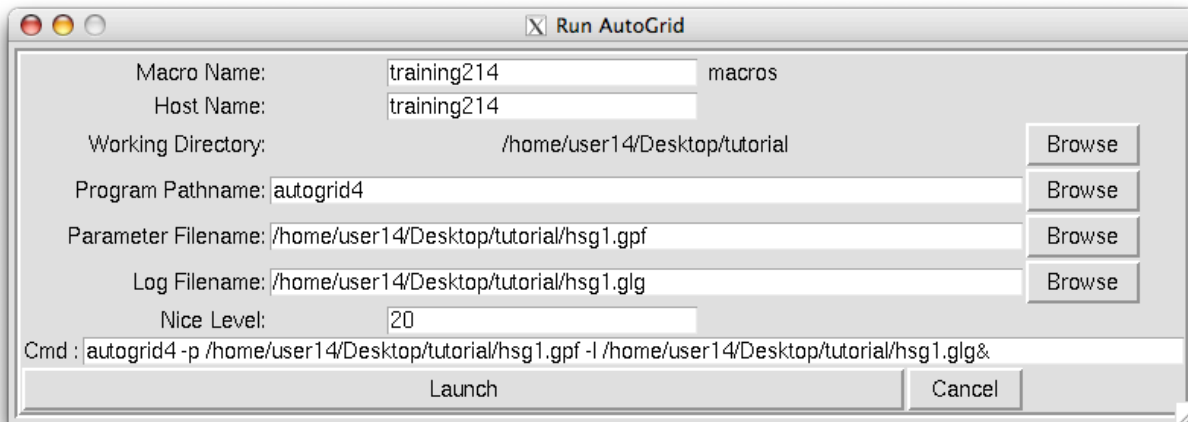
AD4 Exercise Six: Starting AutoGrid 4

Here's how

1. `Run` → `Run AutoGrid...`

2. **Set the** `Working Directory` **if you have not already done so: click** `Browse` **then locate the tutorial directory on your Desktop**

3. **Set** `Program Pathname` **: click** `Browse` **then locate** `autogrid4` **in your tutorial directory**

4. **Set** `Parameter Pathname` **: click** `Browse` **then locate** `hsg1.gpf` **in your tutorial directory.** [Note this updates the `Log Filename` and `Cmd` entries, too]

5. **Start Cmd: click** `Launch`

Optional: To follow what is written to this file during the autogrid4 execution, go to the Terminal and type:

> **tail -f   hsg1.glg**

Type **<Ctrl>-C** to interrupt the **tail** command.

[At TSRI, this calculation will take 2-3 minutes.]

AD4 Exercise Seven: Preparing the AutoDock4 Parameter File

Here's how	Here's why

1. `Docking` → `Macromolecule` → `Set Rigid Filename...`
      type-in **hsg1.pdbqt** , `Open`

[Optional: `Set the Flexible Residues Filename...`]

Setting the rigid filename in ADT only specifies the stem of the gridmap filenames. This does not load a new molecule .

To specify the optional flexible residue filename in the docking parameter file.]

2. `Docking` → `Ligand` → `Choose...`
      `ind` , → `Select Ligand`
    → `Accept`

Setting the ligand sets other parameters in the dpf which could be adjusted via the **AutoDpf4 Ligand Parameters** widget. Today we'll use the defaults so just close it.

3. `Docking` → `SearchParameters` → `Genetic Algorithm...`
      Maximum Number of evals: ⬎ `short` , → `Accept`

Different search methods have different options. For today's tutorial, we are doing a short docking using 250000 evaluations per run. For harder problems, use more evals.

0. `Docking` → `Docking Parameters...`
      `Close`

Here you could choose which random number generator to use and choose seeds for it, set the energy outside the grid, set the maximum allowable initial energy and the maximum number of retries, the step size parameters, the verbosity of the output and whether or not to do a cluster analysis of the results. For today, we'll use the defaults so just click **Close**.

5. `Docking` → `Output` → `Lamarckian GA...`
      `ind.dpf` , → `Save`

DPF file *ind.dpf* contains docking parameters and instructions for a Lamarckian Genetic Algorithm (LGA) docking also known as a Genetic Algorithm-Local Search (GA-LS).

OPTIONAL 6. `Docking` → `Edit DPF...`
      `Cancel`

Take a look at the contents of the dpf file. Verify that **ind.pdbqt** appears after the **move** keyword, **6** after **ndihe** and **14** after **torsdof**. When you're done close the widget with `Cancel`.

Exercise Eight: Starting AutoDock4 and AutoDock Vina.

Here's how

Here's why

To start AutoDock4 from the ADT GUI

1. `Run` → `Run AutoDock...`

2. Set t **Working Directory** with `Browse`

3. Set **Parameter Filename:** `ind.dpf`

   %

4. Check **Log Filename:** `ind.dlg`

5. **Cmd:** `./autodock4 –p ind.dpf –l ind.dlg`

6. `Launch`

Optional: **tail -f   ind.dlg**
          <Ctrl>-C

To start AutoDock4 from the command line at the command prompt you would type:

./autodock4 –p ind.dpf –l ind.dlg

Use the **tail** command in a **terminal** to follow what is written to this file during the AD4 execution. Either wait about 5 minutes to see "Successful Completion" or use <Ctrl>–C to stop the **tail** command.

To start AutoDock Vina from the command line:

1. Start Vina by explicitly specifying all input parameters:

**% ./vina --receptor hsg1.pdbqt --ligand ind.pdbqt \
       --center_x 2.5 --center_y 6.5 --center_z -7.5 \
       --size_x 22.5 --size_y 22.5 --size_z 22.5 \
       --out ind_vina.pdbqt**

2. Start Vina using a configuration file (see Ex9):
  % ./vina --config config.txt

AutoDock Vina Exercise Nine: Preparing a Configuration File (optional)

Here's how   *Here's why*

Here is the list of commands: Vina can read a set of commands from a file.

Input:
 --receptor arg   rigid part of the receptor (PDBQT)
 --flex arg   flexible side chains, if any (PDBQT)
 --ligand arg   ligand (PDBQT)

Search space (required):
 --center_x arg   X coordinate of the center
 --center_y arg   Y coordinate of the center
 --center_z arg   Z coordinate of the center
 --size_x arg   size in the X dimension (Angstroms)
 --size_y arg   size in the Y dimension (Angstroms)
 --size_z arg   size in the Z dimension (Angstroms)

Output (optional):
 --out arg   output models (PDBQT), the default is chosen based on the ligand file name
 --log arg   optionally, write log file

Misc (optional):
 --cpu arg   the number of CPUs to use (the default is to try to detect the number of CPUs
 **   or, failing that, use 1)**
 --seed arg   explicit random seed
 --exhaustiveness arg (=8) exhaustiveness of the global search (roughly proportional to time): 1+
 --num_modes arg (=9)   maximum number of binding modes to generate
 --energy_range arg (=3)   maximum energy difference between the best binding
 **   mode and the worst one displayed (kcal/mol)**

Configuration file (optional):
 --config arg   the above options can be put here

Information (optional):
 --help   print this message
 --version   print program version

Here is the contents of a sample config.txt file:

receptor = hsg1.pdbqt
ligand = ind.pdbqt

center_x = 2.5
center_y = 6.5
center_z = -7.5

size_x = 22.5
size_y = 22.5
size_z = 22.5

out = ind_vina.pdbqt

Exercise Ten: Visualizing AutoDock Vina results….

Here's how   *Here's why*

1. `Analyze` → `Dockings` → `Open AutoDock vina result...`

    File name: `ind_vina.pdbqt`, → `Open`,

◆`Single molecule with multiple conformations`

    [if necessary: DB→ `L` ↓`O` **to hide hsg1]**

In the "**AutoDock Vina Result File**:" browser, navigate to the directory containing **ind_vina.pdbqt**. Select it and click on **Open.** This molecule has a separate set of coordinates for each docked result.

Use the arrow keys on your keyboard to move through these docked conformations. The energy of each docked pose is shown. Open the python shell for more information.

2. `Analyze` → `Macromolecule` → `Choose...`

    Choose Macromolecule: `hsg1` → `Select Molecule`

3. `Analyze` → `Dockings` → `Show Interactions`

This display is radically different: the viewer background color is white, the ligand is displayed with a solvent-excluded molecular surface, atoms in the receptor which are hydrogen-bonded or in close-contact to atoms in the ligand are shown as spheres AND pieces of secondary structure are shown for sequences of 3 or more residues in the receptor which are interacting with the ligand. The GUI for this command lets you turn on and off different parts of this specialized display as well as **list interactions** in the python shell.

4. `File`→ `Exit`

Click on ADT icon to restart before going on…

Exercise Eleven: Visualizing AD4 results....

Here's how

Here's why

1. `Analyze` → `Dockings` → `Open...`

    Docking Log File: `ind.dlg`, → `Open`

    In **Docking Log File**: browser, navigate to the directory containing **ind.dlg**. Select it and click on **Open**.

2. `Analyze` → `Conformations` → `Load`

    ind Conformation Chooser gives a concise view of energies and clusters of docked results. Double click *ind 1_1* to show docked structure. To close, you must expand the widget to access **Dismiss** button.

3. `Analyze` → `Conformations` → `Play, ranked by energy...`

    This player widget lets you walk through a list of docked conformations. (For more details see **Appendix 2**)

4. `Analyze` → `Macromolecule` → `Open...`

    File name: `hsg1.pdbqt`, → `Open`

    DB→ `hsg1`→`MS`⬇`O`→`Left MB`

    DB→ `ind`→`MS`⬇`O`→`LeftMB`, DB→`DG`⬇`◇`

    Use molecular surfaces to check how the docked ligand poses fit into the binding pocket on the receptor:

    Load the receptor into the viewer and display a molecular surface for it. Display a molecular surface for indinavir and color both surfaces with DG Colors to see the charge complementarity…

4. `Analyze` → `Grids` → `Open...`

    `hsg1.OA.map`, → `Open`

    `-0.5`,`Enter`, Sampling→ `1`, `Enter`

    `Select From String`→`Residues:`→type in `ARG8`
    →`Add`
    →`Dismiss`

    `Display` → `Sticks And Balls`→`Ok`

    Here we build a complicated display of the binding site showing the 0.5 isocontour of the oxygen map in red and the ARG8 residues of the receptor as sticks and balls. This is the active site of this protease. Adjust the display by setting the **isovalue** to 0.5 and the sampling to 1. Note the small bar-bell shape at the center of the binding site.

5. `Analyze` → `Dockings` → `Show as Spheres...`

    Finally we show a bird's eye view of all the dockings by marking the center of each docked result with a small sphere…

[`Analyze` → `Dockings` → `Show Interactions`]

    If you like, have a look at the AD4 results

6. `File` → `Exit`

    When you are through, close ADT.

Exercise Three B (optional): Preparing the flexible residue file

Here's how

---

1. `Flexible Residues` → `Input` → `Choose Macromolecule...`

   `hsg1` → `Select Molecule` → `Yes` → `OK`

2. `Select` → `Select From String`

   `Clear Form` , **Residue:** `ARG8` , → `Add` ,

   `Dismiss`

   Selected: `2 Residue(s)`

3. `Flexible Residues` → `Choose Torsions in Currently Selected Residues...`

Click on the rotatable bond between **CA** and **CB** in each residue to inactivate it.
This leaves a total of **6** rotatable bonds in the two flexible ARG8 residues.

   `Close` .

> **Note**: Here we inactivate this bond only to demonstrate how to do so. You can choose any of the possible torsions in the flexible sidechains of residues in the receptor to model as active. You could choose to keep them all active bearing in mind that the limit on the number of torsions including those in the ligand is 32 in AutoDock.
>
> Inactivating this bond is **not** required.

4. `Flexible Residues` → `Output` → `Save Flexible PDBQT...`

   `AutoFlex File:` , *type-in* `hsg1_flex.pdbqt` , `Save`

5. `Flexible Residues` → `Output` → `Save Rigid PDBQT...`

   `AutoFlex File:` , *type-in* `hsg1_rigid.pdbqt` , `Save`

Note: If your docking includes flexible residues, the grid parameter file prepared for the autogrid4 calculation in AD Exercise Five must include the atom types in the flexible residues.

6. `Grid` → `Set Map Types` → `Choose FlexRes....`

   `Choose Flexible Residues from...` → `hsg1` → `Select molecule providing flexible residues`

Beyond the GUI....

Here's how

Please note that this section is written for Linux and MacOS users.
Windows users please see "Basic Hints for Windows Command Line Programming"
(www.voidspace.org.uk/python/articles/command%5Fline.shtml#path)

The most complicated part of using the scripts in AutoDockTools/Utilities24 is specifying their location and specifying the location of the version of python to use with them. To invoke these scripts you must use the **pythonsh** that came with the local installation of MGLToolsPckgs. When MGLToolsPckgs is installed, an environmental variable **$MGL_ROOT** is set to the directory containing its local installation. **pythonsh** is found in the **bin** directory there: $MGL_ROOT/bin/pythonsh and the directory Utilities24 is found here: $MGL_ROOT/MGLToolsPckgs/AutoDockTools/Utilities24.

As you will see below, invoking any script in Utilities24 alone displays a helpful **usage** statement showing the required syntax and the available options.

TSRI only: for today's tutorial, use **pythonsh** and **Utilities24** located in Desktop/tutorial

Input files:

1. To show all the options for formatting a **ligand** for AutoDock without using ADT, at a terminal prompt type this line:

```
./pythonsh  Utilities24/prepare_ligand4.py
```

```
prepare_ligand4: ligand filename must be specified.
Usage: prepare_ligand4.py -l filename
   Description of command...
       -l    ligand_filename (.pdb or .mol2 or .pdbq format)
   Optional parameters:
      [-v]   verbose output
      [-o pdbqt_filename] (default output filename is ligand_filename_stem + .pdbqt)
      [-d]   dictionary to write types list and number of active torsions
      [-A]   type(s) of repairs to make:
               bonds_hydrogens, bonds, hydrogens (default is to do no repairs)
      [-C]   do not add charges (default is to add gasteiger charges)
      [-p]   preserve input charges on atom type, eg -p Zn
             (default is not to preserve charges on any specific atom type)
      [-U]   cleanup type:
               nphs_lps, nphs, lps, '' (default is 'nphs_lps')
      [-B]   type(s) of bonds to allow to rotate
             (default sets 'backbone' rotatable and 'amide' + 'guanidinium' non-rotatable)
      [-R]   index for root
      [-F]   check for and use largest non-bonded fragment (default is not to do this)
      [-M]   interactive (default is automatic output)
      [-I]   string of bonds to inactivate composed of
               of zero-based atom indices eg 5_13_2_10
               will inactivate atoms[5]-atoms[13] bond
```

and atoms[2]-atoms[10] bond
                         (default is not to inactivate any specific bonds)
        [-Z]   inactivate all active torsions
                         (default is leave all rotatable active except amide and guanidinium)
        [-g]   attach all nonbonded fragments
        [-s]   attach all nonbonded singletons
                    NB: sets attach all nonbonded fragments too
                    (default is not to do this)

To prepare IND.pdbqt with 14 active torsions (Exercise 2, leaving out 'tutorial-only' step 4)

> **./pythonsh Utilities24/prepare_ligand4.py -l ind.pdb -o IND.pdbqt**

2. To show the options for formatting a **receptor**, at a terminal prompt type:

> **./pythonsh Utilities24/prepare_receptor4.py**

prepare_receptor4: receptor filename must be specified.
Usage: **prepare_receptor4.py -r filename**
Description of command...
     -r   receptor_filename
           supported file types include pdb,mol2,pdbq,pdbqs,pdbqt, possibly pqr,cif
   Optional parameters:
     [-v] verbose output (default is minimal output)
     [-o pdbqt_filename] (default is 'molecule_name.pdbqt')
     [-A] type(s) of repairs to make:
        'bonds_hydrogens': build bonds and add hydrogens
        'bonds': build a single bond from each atom with no bonds to its closest neighbor
        'hydrogens': add hydrogens
        'checkhydrogens': add hydrogens only if there are none already
        'None': do not make any repairs
        (default is 'None')
     [-C] preserve all input charges ie do not add new charges
        (default is addition of gasteiger charges)
     [-p] preserve input charges on specific atom types, eg -p Zn -p Fe
     [-U] cleanup type:
        'nphs': merge charges and remove non-polar hydrogens
        'lps': merge charges and remove lone pairs
        'waters': remove water residues
        'nonstdres': remove chains composed entirely of residues of
            types other than the standard 20 amino acids
        'deleteAltB': remove XX@B atoms and rename XX@A atoms->XX
        (default is 'nphs_lps_waters_nonstdres')
     [-e] delete every nonstd residue from any chain
        'True': any residue whose name is not in this list:
            ['CYS','ILE','SER','VAL','GLN','LYS','ASN',
            'PRO','THR','PHE','ALA','HIS','GLY','ASP',
            'LEU', 'ARG', 'TRP', 'GLU', 'TYR','MET',
            'HID', 'HSP', 'HIE', 'HIP', 'CYX', 'CSS']
        will be deleted from any chain.
        NB: there are no nucleic acid residue names at all
        in the list and no metals.
        (default is False which means not to do this)
     [-M] interactive
        (default is 'automatic': outputfile is written with no further user input)
     [-d dictionary_filename] file to contain receptor summary information

To prepare HSG1.pdbqt with added hydrogens (Exercises 1 and 3)

> **./pythonsh Utilities24/prepare_receptor4.py -r hsg1.pdb -A hydrogens -o HSG1.pdbqt**

3. To create a **grid parameter file** for AutoGrid4, at a terminal prompt type:

```
./pythonsh Utilities24/prepare_gpf4.py
```

prepare_gpf4.py: ligand and receptor filenames must be specified.
Usage: prepare_gpf4.py -l pdbqt_file -r pdbqt_file
    -l ligand_filename
    -r receptor_filename

Optional parameters:
    [-i reference_gpf_filename]
    [-o output_gpf_filename]
    [-x flexres_filename]
    [-p parameter=newvalue. For example: -p ligand_types='HD,Br,A,C,OA']
    [-d directory of ligands to use to set types]
    [-y boolean to center grids on center of ligand]
    [-n boolean to NOT size_box_to_include_ligand]
    [-I increment npts in all 3 dimensions by this integer]
    [-v]

Prepare a grid parameter file (GPF) for AutoDock4.
The GPF will by default be <receptor>.gpf. This may be overridden using the -o flag.

Currently the only way to set the gridcenter to something other than the center of the ligand is to use a template gpf containing the desired value for the gridcenter.

Here is the content of a template gpf "sample.gpf":

```
npts 40 40 46                          # num.grid points in xyz
gridfld hsg1_rigid.maps.fld            # grid_data_file
spacing 0.375                          # spacing(A)
receptor_types A C HD N OA SA          # receptor atom types
ligand_types C HD N A NA OA            # ligand atom types
receptor hsg1_rigid.pdbqt              # macromolecule
gridcenter 2.5 6.5 -7.5                # xyz-coordinates or auto
smooth 0.5                             # store minimum energy w/in rad(A)
map hsg1_rigid.C.map                   # atom-specific affinity map
map hsg1_rigid.HD.map                  # atom-specific affinity map
map hsg1_rigid.N.map                   # atom-specific affinity map
map hsg1_rigid.A.map                   # atom-specific affinity map
map hsg1_rigid.NA.map                  # atom-specific affinity map
map hsg1_rigid.OA.map                  # atom-specific affinity map
elecmap hsg1_rigid.e.map               # electrostatic potential map
dsolvmap hsg1_rigid.d.map                # desolvation potential map
dielectric -0.1465                     # <0, AD4 distance-dep.diel;>0, constant
```

To prepare HSG1.gpf (Exercise 5)

```
./pythonsh Utilities24/prepare_gpf4.py -l IND.pdbqt  -r HSG1.pdbqt  -p npts="60,60,66" -i sample.gpf
```

4. To create a **docking parameter file** for AutoDock4, at a terminal prompt type:

```
./pythonsh Utilities24/prepare_dpf4.py
```

prepare_dpf4.py: ligand and receptor filenames must be specified.
Usage: prepare_dpf4.py -l pdbqt_file -r pdbqt_file
    -l ligand_filename
    -r receptor_filename

Optional parameters:
   [-o output dpf_filename]
   [-i template dpf_filename]
   [-x flexres_filename]
   [-p parameter_name=new_value]
   [-k list of parameters to write]
   [-v] verbose output
   [-L] use local search parameters
   [-S] use simulated annealing search parameters
   [-s] seed population using ligand's present conformation

Prepare a docking parameter file (DPF) for AutoDock4.
The DPF will by default be <ligand>_<receptor>.dpf. This may be overridden using the -o flag

To prepare ind.dpf (Exercise 7)

./pythonsh Utilities24/prepare_dpf4.py –l IND.pdbqt –r HSG1.pdbqt -p ga_num_evals=250000

Output files:

1. To write summary files for result directories containing multiple dlg files:

./pythonsh Utilities24/summarize_results4.py

summarize_results4: directory must be specified.
Usage: summarize_results4.py -d directory

  Description of command...
    -d   directory
  Optional parameters:
   [-t]   rmsd tolerance (default is 1.0)
   [-f]   rmsd reference filename
   (default is to use input ligand coordinates from docking log)
   [-b]   print best docking info only (default is print all)
   [-L]   print largest cluster info only (default is print all)
   [-B]   print best docking and largest cluster info only (default is print all)
   [-o]   output filename
       (default is 'summary_of_results')
   [-a]   append to output filename
       (default is to open output filename 'w')
   [-k]   build hydrogen bonds and report number built
   [-e]   compute estat, vdw, hb + desolv energies and report breakdown
   [-r]   receptor filename
   [-u]   report unbound energy
   [-i]   subtract internal energy
   [-p]   report depth of torsion tree
   [-v]   verbose output

./pythonsh Utilities24/summarize_results4.py –d hsg1_ind –t 2.0 –b –a –o summary_2.0.txt

cat summary_2.0.txt

```
lowestEnergy_dlgfn #runs #cl #LEC LE rmsd_LE  #ats #tors #h_ats #lig_eff
hsg1_ind/ind4, 10,  2,  8, -11.2300, 10.0409, 49, 6, 4, -0.2496
......
```

2. To write VSResult files (pdbqt+) for directories containing multiple dlg files:

./pythonsh Utilities24/process_VSResults.py

process_VSResults: directory must be specified.
Usage: process_VSResults.py -d directory

Description of command...
    -d   directory
Optional parameters:
    [-t]   rmsd tolerance (default is 2.0)
    [-f]   rmsd reference filename
    (default is to use input ligand coordinates from a docking log)
    [-r]   receptor filename (default is set from gridmap names)
    [-B]   create best docking pdbqt only (default is create both best energy and largest cluster)
    [-L]   create largest cluster docking pdbqt only (default is create both best energy and largest cluster)
    [-l]   stem for largest cluster docking pdbqt file (default is 'ligandname_lc')
    [-x]   maximum number of clusters to report (default is 10, use '-1' for all)
    [-D]   include interactions in output pdbqt file
    [-n]   do not build hydrogen bonds (default is to build hydrogen bonds + report)
    [-c]   do not detect atoms in close contact (default is to detect + report)
    [-p]   include detection of pi-pi interactions in close contact (default is not to detect pi-pi and pi-cation)
    [-v]   verbose output

./pythonsh Utilities24/process_VSResults.py -d ZINC02026663_xJ1_xtal -r ZINC02026663_xJ1_xtal /xJ1_xtal.pdbqt -x -1 -D -p

To view the content of the resulting pdbqt+ file

cat ZINC02026663_vs.pdbqt

(Note: lines omitted here for brevity):
REMARK VirtualScreeningResult Mon Jul 19 11:41:15 2010
USER AD> lig_hb_atoms : 3
USER AD> xJ1_xtal:B:ARG57:NE,HE~ZINC02026663_vs:d:<0>:O3
USER AD> xJ1_xtal:B:ARG57:NH1,HH11~ZINC02026663_vs:d:<0>:O2
USER AD> xJ1_xtal:B:LYS55:NZ,HZ3~ZINC02026663_vs:d:<0>:O4
USER AD> macro_close_ats: 18
USER AD> xJ1_xtal:B:VAL77:O
USER AD> xJ1_xtal:B:VAL56:C
USER AD> xJ1_xtal:B:GLY78:CA
.....
USER AD> lig_close_ats: 16
USER AD> ZINC02026663_vs:d:<0>:C7,C15,N4,O4,O3,C5,N3,C3,C6,C14,C12,C13,O1,C4,C11,O2
USER AD> pi_cation: 1
USER AD> xJ1_xtal:B:LYS55:NZ~~ZINC02026663_vs:d:<0>:C7,C4,C5,C8,C3,C6

USER AD> binding 2.00 129 runs 31 clusters
USER AD> ligand efficiency -0.3112
USER AD> rmsd, LE, clu_size, clu_e_range, dlgfilename, run#, b_curCRDs, b_LE, b_LC
USER AD> 0.000,-7.470,39,0.60,faah8621_ZINC02026663_xJ1_xtal_00.dlg,30,1,1,1
USER AD> 16.810,-7.330,3,0.03,faah8621_ZINC02026663_xJ1_xtal_02.dlg,38,0,0,0
.....
REMARK 3 active torsions:
REMARK status: ('A' for Active; 'I' for Inactive)
    REMARK   1 A   between atoms: C5_6 and N4_22
    REMARK   2 A   between atoms: N2_12 and C10_13

24

```
REMARK   3  A   between atoms: C15_18  and  N3_19
ROOT
ATOM      1  C1  <0> d        15.212  12.586  -2.152  0.00  0.00     0.092 C
ATOM      2  C2  <0> d        15.052  12.346  -3.631  0.00  0.00     0.110 A
.....
```

These scripts can be used in foreach loops to process multiple files. Here are two examples.

1. To format many ligands: in the directory containing many ligand pdb files:

```
foreach f (`ls *pdb`)
./pythonsh Utilities24/prepare_ligand4.py -l $f
echo $f
end
```

2. To summarize dockings of many different ligands to the same target: in the directory containing the directories of results:

```
foreach f (`ls`)
pythonsh Utilities24/summarize_results4.py -d $f -t 2.0 -b -a -o summary_2.0.txt
echo $f
end
```

Files for Exercises:

Input Files:

```
hsg1.pdb
ind.pdb
config.txt
```

Results Files

Ligand

```
ind.pdbqt (6 torsions moving fewest atoms)
```

Macromolecule

```
hsg1.pdbqt
```

   or

```
hsg1_flex.pdbqt
```

```
hsg1_rigid.pdbqt
```

AutoGrid

```
hsg1.gpf
hsg1.glg
hsg1.*.map
hsg1.maps.fld,hsg1.maps.xyz
```

AutoDock

```
hsg1.dpf
ind.dlg
```

Useful Scripts in AutoDockTools/Utilities24

```
prepare_ligand4.py
```

```
prepare_receptor4.py
```

```
prepare_flexreceptor4.py
```

```
prepare_gpf4.py
```

```
prepare_dpf4.py
```

```
summarize_results4.py
```

Customization Options for ADT

```
adt_automergeNPHS: default is 1
adt_autoCtoA: default is 1
adt_editHISprotonation: default is 'No Change'
autotors_userProteinAromaticList
```

Appendix 1: Dashboard Widget

Select entry + **Command** Buttons →

Tree Widget
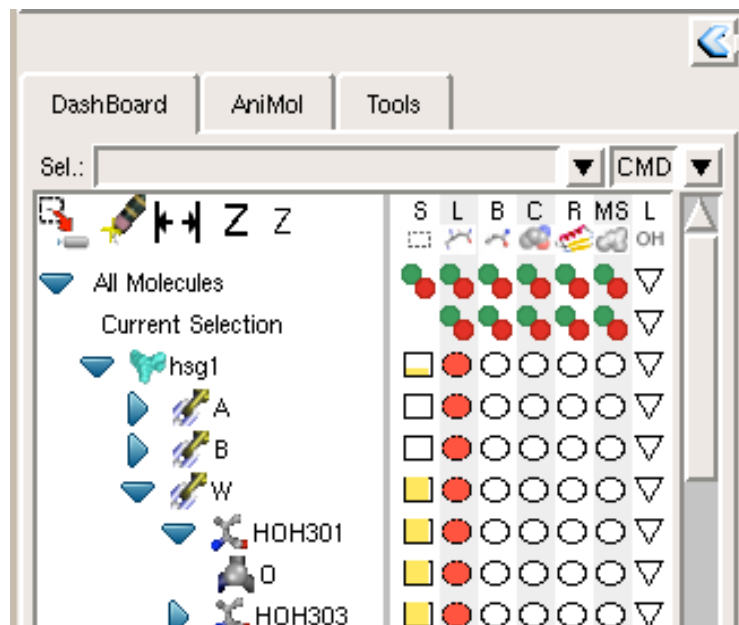
Note: Clicking on a **sh**ape - rectangle, circle, square or diamond - under a command causes the command linked to the shape to be applied to each node in the corresponding row. If the shape is off (colored white), the command will be applied to nodes and the shape will be colored red. If the shape is on (colored red), clicking on the command button will undo the command and the shape will be colored white. Circles are used for **display** commands, squares for **label** commands and diamonds for **color** commands. Coloring can be replaced by a different coloring scheme but cannot be undone. The gray rectangle is used for **show/hide** and the white rectangle for **select**.

The **Tree Widget** on the left lists all molecules currently loaded in PMV. Click on the arrows ▶ to navigate between molecules ✹, chains ✎ , residues ✖ and atoms ⚒. Clicking on a **shape** in one of the columns in the right section executes the PMV command corresponding to the label at the top of the column on the group of nodes corresponding to the row. There 16 different commands that can be executed this way - gray rectangle (Show/Hide), select/unselect (**Sel**.), display lines (**Lines**), display CPK (**CPK**), display sticks and balls (**S&B**), display secondary structure (**Rib.**), display molecular surface (**MS**), display labels (**Lab**.), color by atom type (**Atom**), color by molecule (**Mol**), color by chain (**Chain**), color by residue according Rasmol (**RAS**), color by residue according Shapely (**SHA**), color according to David Goodsell colors (**DG**), color by secondary structure element type (**Sec.Str**.) and color by instance (**Inst**).

To help users see the connection between molecular fragments and PMV commands, a crosshair is drawn when cursor is inside the Dashboard widget.

Right-clicking on a shape displays an input parameter panel for the command and allows the user to customize specific input parameters for the command.

The **Sel:** entry in the top left corner of the Dashboard can be used to select entries in the Tree using a Pmv compound selector. Nodes matching the specified string will be selected. Selected nodes are outlined with a yellow selection box. When a shape is clicked for a selected node, the corresponding command is applied to all currently selected nodes. Hovering over this entry shows samples of the required syntax.

Right-click on **S L B C R MS L CL** to access a menu which allows you to specify what the display commands will display: backbone atoms only (**BB**), the side chain atoms only (**SC**), the sidechain atoms and **CA** atoms (**SC+CA**) or the full molecular fragment (**ALL**). This can be overridden for each column (**CMD**).

Click on any colored oval to **Show/Hide** a specific geometry of a molecule. Notice that the molecule in the viewer disappears. Click on the same oval again to redisplay it. Click on the rectangle under the **Sel** level to select or deselect the molecule. Experiment by clicking on each of the other buttons. These are short cuts to a basic set of commands for displaying and coloring various molecular representations.

Appendix 2: Conformation Player

- `Type-in entry` at center random access to any conformation by its id. Valid ids depend on which menubutton was last used to start the player.

- Click on `black arrow` buttons next to entry to change to next or previous conformation in current list.

- `White arrow` buttons start play according to current play mode parameters (see below). Clicking on an active white arrow button stops play. [While a play button is active, its icon is changed to double vertical bars.]

- `Double black arrow` buttons start play as fast as possible in the specified direction.

- `Double black arrow plus line` buttons advance to beginning or end of conformation list.

- `Ampersand` button opens the **Set Play Options** widget (see next).

- `Quatrefoil` button closes the player.

Next, a tour of the **Set Play Options** widget and its buttons:
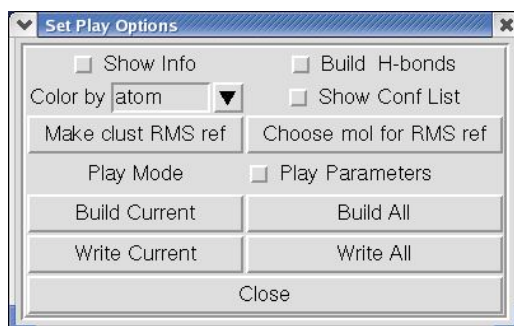
- `Show Info` opens and closes a separate panel **Conformation # Info** that displays additional information about current conformation (see following).

- **Build H-bonds** turns on and off building and displaying hydrogen bonds between the macromolecule and the ligand in its current conformation. Note: building hydrogen bonds requires that the receptor molecule be present in the viewer and that you have either chosen it using: **Analyze**→**Macromolecule**→**Choose** or read it in specifically using **Analyze**→**Macromolecule**→**Open...**

- **Color by** allows you to choose how to color the ligand from a list of available coloring schemes.

- **Show Conf List** opens and closes a separate **Choose Conformations** widget showing current idlist (see below).

- **Make clust RMS ref** sets the reference coordinates for RMS to those of the current conformation. [This RMS value is shown in Info panel as clRMS]

- **Choose mol for RMS ref** lets you select a different molecule from list of those in Viewer to use as reference for a new RMS computation.

- **Play Mode** opens a separate **Play Mode** widget (see next).

- **Play Parameters** exposes controls for setting parameters governing how the conformations are played:

Note: To set the value of a thumbwheel click on it with the left mouse button and hold the mouse button down while you drag the mouse to the right to increase the value or to the left to decrease the value. Alternatively, you can right-click on a thumbwheel to open a separate widget which lets you type in a new value.
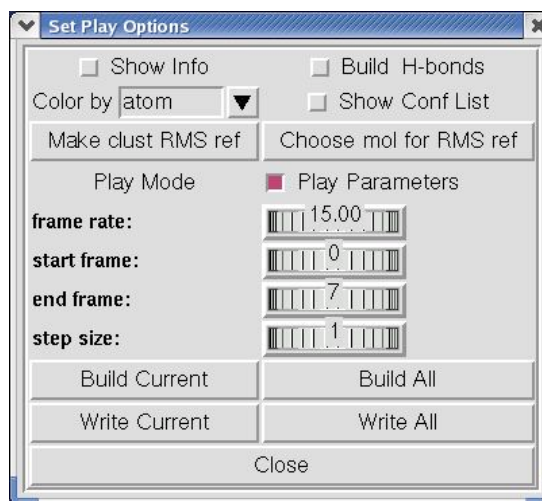
Four **thumbwheel** widgets are used to set **Play Parameters**.
- **frame rate:** set the relative speed of the player [absolute rate is cpu dependent]
- **start frame:** index into current conformation list. **Note** the input conformation is always inserted at index 0 in sequence list.
- **end frame:** index into current conformation list.
- **step size:** determines next conformation in list. For example, step size 1 plays every available conformation whereas step size 2 every other…

Clicking on **Play Parameters** hides the thumbwheels.

- **Build Current** adds a new molecule to the viewer with current conformation's coordinates providing that a molecule hasn't already been built with this conformation's id.

- **Build All** builds a new molecule for each conformation in the current sequence of conformations bound to the player.

- **Write Current** lets you choose a filename for writing a formatted file using the current coordinates.

- **Write All** writes a formatted file for each set of coordinates in the current sequence. This uses default filenames based on the id of each Conformation.

- **Close** button closes **Set Play Options** widget.

Next, the **Play Mode** widget and its buttons:
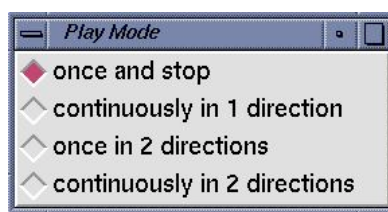
These 4 radiobuttons are used to set the current play mode. [Note that at any time, the current **endFrame** and the current **startFrame** depend on the direction of play.]

- **once and stop** plays from the current conformation in the current direction up to and including the endFrame.

- **continuously in 1 direction** plays in the current direction up to and including the endFrame and then restarts with startFrame , again and again....

- **once in 2 directions** plays from the current conformation in the current direction up to and including the endFrame and then plays back to startFrame and stops.

- **continuously in 2 directions** plays from current conformation in current direction up to endFrame then back to the beginning then back to the end, again and again....

-

The Choose Conformation widget:

31
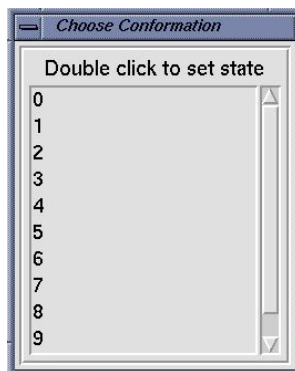
Choose Conformation widget has list of ids for each conformation in the current sequence list. Double clicking on an entry in this list updates the ligand to the corresponding conformation. This widget is closed by clicking on the checkbutton **Show Conf List** in **Set Play Options** widget.

Conformation # Info widget shows information about a specific conformation from a docking experiment.
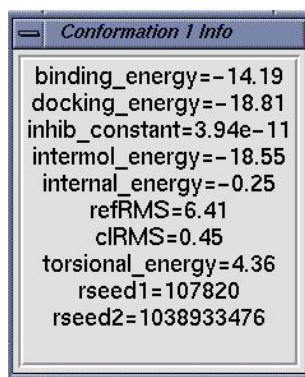
- **binding energy** is the sum of the intermolecular energy and the torsional free-energy penalty.

- **docking energy** is the sum of the intermolecular energy and the ligand's internal energy.

- **inhib_constant** is calculated in AutoDock as follows:

$$Ki=exp((deltaG*1000.)/(Rcal*TK)$$

where **deltaG** is docking energy, **Rcal** is 1.98719 and **TK** is 298.15

- **refRMS** is rms difference between current conformation coordinates and current reference structure. By default the input ligand is used as the reference.

- **clRMS** is rms difference between current conformation and the lowest energy conformation in its cluster.

- **torsional_energy** is the number of active torsions * .3113
  [.3113 is AutoDock 3 forcefield torsional free energy parameter]

- **rseed1** and **rseed2** are the specific random number seeds used for current conformation's docking run.