pyvol Documentation

Release 1.2.27

Ryan Smith

October 28, 2019

CONTENTS

Contents:

# PyVOL

PyVOL is a python library packaged into a PyMOL GUI for identifying protein binding pockets, partitioning them into sub-pockets, and calculating their volumes. PyVOL can be run as a PyMOL plugin through its GUI or the PyMOL prompt, as an imported python library, or as a commandline program. Visualization of results is exclusively supported through PyMOL though exported surfaces are compatible with all standard 3D geometry viewers.

## Basic PyMOL Installation PyVOL can be installed into PyMOL by using the plugin manager to install directly from the PyMOL wiki or by manually installing the zip file downloaded from: `bash https://github.com/rhs2132/pyvol/blob/master/pyvolgui.zip ` Either option adds a menu option under plugins called "PyVOL." Opening this menu will launch the PyVOL GUI. The third tab manages the installation and update of the PyVOL backend. Simply clicking "Install PyVOL" will use pip to install the code and all dependencies. However, installation of MSMS is unavailable through this path. On MacOS and Linux, MSMS is subsequently installed using conda. However, Windows requires independent installation as described below.

## Basic Manual Installation PyVOL minimally requires biopython, msms, numpy, pandas, scipy, scikit-learn, and trimesh in order to run. PyVOL is available for manual installation from github or from PyPI. `bash pip install bio-pyvol `

## Manual MSMS Installation MSMS can be installed on MacOS and Linux using the bioconda channel: ```bash conda install -c bioconda msms

~~~
` Otherwise MSMS must be installed manually by downloading it from
[MGLTools](http://mgltools.scripps.edu/packages/MSMS/) and adding it to the
path.  PyMOL distributions from Schrodinger have MSMS included; however,
it must still be added to the path manually.  The executable is located at:
```bash <pymol_root_dir>/pkgs/msms-2.6.1-2/bin/msms `
~~~

## Quick Start From within PyMOL, the simplest binding pocket calculation is simply run either with the provided GUI or with: `python pocket protein_selection ` The two parameters that most dramatically affect calculations are the maximum and minimum radii used to respectively define the exterior surface of the protein and the boundary of the binding pocket itself. In practice, the minimum radius does not need to be changed as its default (1.4) is broadly useful. The maximum radius does often need to be adjust to find a suitable value using the max_rad parameter: `python pocket protein_selection, min_rad=1.4, max_rad=3.4 `

## Basic Usage ### Pocket Specification PyVOL by default recognizes the largest pocket and returns the volume and geometry for it. However, manual identification of the pocket of interest is generally preferable. This can be done through specification of a ligand, a residue, or a coordinate. If a specification is given, the mode is changed to specific by default.

Default behavior: `python pocket protein_selection, mode=largest ` Ligand specification: `python pocket protein_selection, mode=specific, ligand=ligand_selection pocket protein_selection, ligand=ligand_selection ` Residue specification: `python pocket protein_selection, mode=specific, resid=A15 pocket protein_selection, resid=A15 pocket protein_selection, mode=specific, residue=residue_selection pocket protein_selection, residue=residue_selection ` where the resid is written as <Chain><Residue number>. If there is only one chain in the selection, the chain ID can be excluded.

Coordinate specification: `python pocket protein_selection, mode=specific, pocket_coordinate="5.0 10.0 15.0" pocket protein_selection, pocket_coordinate="5.0 10.0 15.0" ` where the coordinate is provided as three floats separated by spaces and bounded by quotation marks.

Alternatively, PyVOL can return the surfaces and volumes for all pockets above a minimum volume that are identified. By default, this volume cutoff is set at 200 A^3. `python pocket protein_selection, mode=all, minimum_volume=200 `

### Extra Ligand Options When a ligand is provided, the atoms of the ligand can be used to identify both minimum and maximum extents of the calculated binding pocket. To include the volume of the ligand in the pocket volume (useful for when the ligand extends into bulk solvent), use the lig_incl_rad parameter: `python pocket protein_selection, ligand=ligand_selection, lig_incl_rad=0.0 ` where the value of lig_incl_rad is added to the Van der Waals radii of each atom in the ligand selection when calculating the exterior surface of the protein.

The atoms of the ligand can also be used to define a maximum boundary to the calculated pocket by specifying the lig_excl_rad parameter: `python pocket protein_selection, ligand=ligand_selection, lig_excl_rad=2.0 ` where the value of lig_excl_rad is added to the Van der Waals radii of each atom in the ligand selection when calculating the exterior surface of the protein.

### Sub-pocket Partitioning Sub-partitioning is enabled by setting the subdivide parameter to True: `python pocket protein_selection, subdivide=True `

Parameters controlling the number of sub-pockets identified generally perform well using defaults; however, they can be easily adjusted as needed. The two most important parameters are the minimum radius of the largest sphere in each sub-pocket (this excludes small sub-pockets) and the maximum number of clusters: `python pocket protein_selection, subdivide=True, min_subpocket_rad=1.7, max_clusters=10 ` If the number of clusters must be reduced, sub-pockets are merged on the basis of connectivity between the defining sets of tangent spheres. Practically, sub-pockets with a greater surface area boundary are merged first.

### Display and Output Options By default, PyVOL simply outputs a log containing volumes and, when invoked through PyMOL, displays pocket boundaries as semi-translucent surfaces. This behavior can be extensively customized.

The output name for all computed PyMOL objects and the base filename for any output files can be specified using the prefix option: `python pocket protein_selection, prefix=favprot ` PyVOL can also write the input and output files to a directory if given an output directory. In this case it writes out the input protein and ligand structures, a csv report of all calcuated volumes, and paired csv/obj files containing tangent sphere collections and 3D triangulated mesh files respectively. `python pocket protein_selection, output_dir=best_out_dir `

Calculated surfaces can be visualized in three different ways by setting the display_mode parameter. The following three commands set the output as a solid surface with transparency, a wireframe mesh, and a collection of spheres. Color is set with the color parameter and transparency (when applicable) with the alpha parameter: `python pocket protein_selection, display_mode=solid, alpha=0.85, color=skyblue pocket protein_selection, display_mode=mesh, color=red pocket protein_selection, display_mode=spheres, color=firebrick ` where alpha is [0, 1.0] and the color is any color defined within pymol. The presets should generally be sufficient, but custom colors can be chosen using the commands given on the PyMOL wiki.

### Command-line Interface PyVOL can also be run from the command-line. If installed using pip, a *pyvol* entry point should be automatically installed and made available on the path. Otherwise, manual invocation of *pyvol/__main__.py* should work. From the command-line, PyVOL is run with a standard configuration file. `bash python -m pyvol <input_parameters.cfg> ` A template configuration file with default values supplied can be generated using: `bash python -m pyvol -t <output_template.cfg> ` Currently, PyVOL only reports standard log output to stdout when run this way. So if an output directory is not provided, there is no easy way to retrieve the results.

CHAPTER

ONE

INDICES AND TABLES

- genindex
- modindex
- search