pyvol Documentation

Release 1.2.27

Ryan Smith

October 28, 2019

CONTENTS

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 standard 3D geometry visualization programs.

CHAPTER

ONE

INSTALLATION

PyVOL can be installed into any python environment; however, for most users direct installation into PyMOL will be easiest.

1.1 Installation into PyMOL

PyVOL is distributed as a GUI and a backend. Installation into PyMOL uses PyMOL's plugin manager to install the GUI and then the GUI to install the backend. The GUI is installed through the plugin manager through loading the zipped GUI file:

```
https://github.com/rhs2132/pyvol/blob/master/pyvolgui.zip
```

This creates a new `PyVOL` menu entry under plugins. The third tab of the GUI allows installation of PyVOL from PyPI along with all available dependencies. For more information, especially on Windows, see the installation page.

CHAPTER

TWO

RUNNING PYVOL

2.1 Quick Start

From within PyMOL, the simplest binding pocket calculation is simply run at the PyMOL prompt with:

```
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 adjusted to find a suitable value using the max_rad parameter:

```
pocket protein_selection, min_rad=1.4, max_rad=3.4
```

2.2 Further Examples

For extensive explanations and documentations of the GUI and command line interfaces, see the general usage page.

CHAPTER

THREE

MODULE DOCUMENTATION

For full documentation of the code, see the modules page.

3.1 Installation

PyVOL distribution is hosted by PyPI and accessed through pip. PyVOL can consequently be installed into any python environment. For convenience, the PyMOL GUI contains an installer.

3.1.1 Detailed Installation into PyMOL

PyVOL is distributed as a GUI and a backend. Installation into PyMOL uses PyMOL's plugin manager to install the GUI and then the GUI to install the backend. The GUI is installed through the plugin manager through loading the zipped GUI file:

```
https://github.com/rhs2132/pyvol/blob/master/pyvolgui.zip
```

This creates a new `PyVOL` menu entry under plugins. The third tab of the GUI allows installation of PyVOL from PyPI along with all available dependencies. On Linux and MacOS, MSMS is automatically installed from the platform-limited bioconda channel. MSMS installation instructions are otherwise below.

3.1.2 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 through PyPI. Most conveniently:

```
pip install bio-pyvol
```

3.1.3 MSMS Installation

MSMS can be installed on MacOS and Linux using the bioconda channel:

```
conda install -c bioconda msms
```

Otherwise MSMS must be installed manually by downloading it from MGLTools 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:

```
<pymol_root_dir>/pkgs/msms-2.6.1-2/bin/msms
```

3.1.4 Updating

PyVOL can be updated via the command line:

```
pip update bio-pyvol
```

If using the PyMOL GUI, the third tab has a button labeled `Check for Updates` that will query PyPI to detect whether an update is available. If one is available, that button changes to `Update PyVOL` and permits updating with a single click.

3.1.5 Uninstallation

PyVOL can be uninstalled via the command line:

```
pip uninstall bio-pyvol
```

If using the PyMOL GUI, the third tab has a button labeled `Uninstall PyVOL` that will remove the PyVOL backend. Afterwards, selecting uninstall on the plugin within the PyMOL plugin manager will the GUI.

3.2 General Usage

The GUI and shell command line prompts recapitulate the PyMOL prompt interface with few exceptions. General usage of PyVOL is shown here with specific examples and modifications for the other interfaces in the following pages. Programmatic invocation of internal functions is supported and covered through the module documentation.

3.2.1 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 (Largest Mode):

```
pocket protein_selection, mode=largest
```

Ligand Specification:

```
pocket protein_selection, mode=specific, ligand=ligand_selection
pocket protein_selection, ligand=ligand_selection
```

Residue Specification:

```
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 **:raw-html-m2r:‘<Chain>‘:raw-html-m2r:‘<Residue number>‘**. If there is only one chain in the selection, the chain ID can be excluded.

Coordinate Specification:

```
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.

Calculation of All Pockets

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.

```
pocket protein_selection, mode=all, minimum_volume=200
```

3.2.2 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.

Ligand Volume Inclusion

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:

```
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.

Ligand-defined Maximum Volume

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:

```
pocket protein_selection, ligand=ligand_selection, lig_excl_rad=2.0
```

where the value of 11lig_excl_rad11 is added to the Van der Waals radii of each atom in the ligand selection when calculating the exterior surface of the protein.

3.2.3 Sub-pocket Partitioning

Sub-partitioning is enabled by setting the `subdivide` parameter to `True`:

```
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:

```
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.

3.2.4 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:

```
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.

```
pocket protein_selection, output_dir=chosen_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:

```
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.

3.3 GUI Interface

The GUI is divided into three tabs that respectively: 1) run new PyVOL calculations 2) load prior PyVOL calculations from disk and 3) install, update, and uninstall the backend.

3.3.1 Run Tab

test run

3.3.2 Load Tab

test load

3.3.3 Install Tab

test install

3.4 Shell Interface

PyVOL can also be run from the system command line using bash or any standard shell. 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.

3.4.1 Running from the Shell

From the command-line, PyVOL is run exclusively using a configuration file.

```
python -m pyvol <input_parameters.cfg>
```

3.4.2 Template Configuration File Generation

A template configuration file with default values supplied can be generated using:

```
python -m pyvol -t <output_template.cfg>
```

3.4.3 Notes on Output

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.

3.5 pyvol

3.5.1 pyvol package

Submodules

pyvol.cluster module

pyvol.cluster.**cluster_within_r**(*spheres*, *radius*, *allow_new=True*)
Cluster spheres with the same radius using DBSCAN, modifying input data in situ

Parameters

- **spheres** (Spheres) – complete set of input spheres
- **radius** (*float*) – radius at which clustering is to occur
- **allow_new** (*bool*) – permit new clusters? (Default value = True)

pyvol.cluster.**cluster_between_r**(*spheres*, *ref_radius*, *target_radius*)
Cluster spheres from a target radius to a reference radius, modifying input data in situ

Parameters

- **spheres** (Spheres) – complete set of input spheres
- **ref_radius** (*float*) – radius from which cluster identities will be drawn
- **target_radius** (*float*) – radius to which cluster identities will be propagated

pyvol.cluster.**cluster_improperly_grouped**(*spheres*, *radius*, *min_cluster_size=1*, *max_clusters=None*)
Reassigns improperly clustered spheres to 'proper' clusters, modifying input data in situ

Parameters

- **spheres** (Spheres) – complete set of input spheres
- **radius** (*float*) – radius at which closest groups are identified
- **min_cluster_size** (*int*) – minimum number of spheres in a 'proper' cluster (Default value = 1)
- **max_clusters** (*int*) – maximum number of 'proper' clusters (Default value = None)

pyvol.cluster.**extract_groups**(*spheres*, *surf_radius=None*, *prefix=None*)
Extracts spheres belonging to each cluster from the complete input set and optionally calculates bounded surfaces

Parameters

- **spheres** (Spheres) – complete set of input spheres
- **surf_radius** – radius used to calculate bounding spheres for individual groups (Default value = None)
- **prefix** – prefix to identify new surfaces (Default value = None)

Returns **group_list** – a list of Spheres objects each corresponding to a different cluster

Return type [Spheres]

pyvol.cluster.**hierarchically_cluster_spheres**(*spheres*, *ordered_radii*, *min_new_radius=None*, *min_cluster_size=10*, *max_clusters=None*)
Cluster spheres by grouping spheres at large radius and propagating those assignments down to smaller radii

Parameters

- **spheres** (Spheres) – complete set of input spheres
- **ordered_radii** (*[float]*) – list of radii ordered from largest to smallest
- **min_new_radius** (*float*) – smallest spheres to keep (Default value = None)
- **min_cluster_size** (*int*) – minimum number of spheres in a cluster (Default value = 10)
- **max_clusters** (*int*) – maximum number of clusters (Default value = None)

pyvol.cluster.**identify_closest_grouped**(*spheres*, *group*, *radius*)

Identifies the closest 'properly' grouped cluster to a specified group

Parameters

- **spheres** (Spheres) – complete set of input spheres
- **group** (*float*) – group for which to identify the closest clusters
- **radius** (*float*) – radius at which to perform the search

Returns **group** – passthrough of input group closest (float): id of the closest cluster magnitude (int): number of pairwise closest connections between the queried group and the closest identified cluster

Return type float

pyvol.cluster.**merge_sphere_list**(*s_list*, *r=None*, *g=None*)

Parameters

- **s_list** (*[Spheres]*) – list of input spheres
- **r** (*float*) – radius value to assign to output Spheres (Default value = None)
- **g** (*float*) – group value to assign to output Spheres (Default value = None)

Returns **merged_spheres** – a single Spheres object containing the merged input lists

Return type *Spheres*

pyvol.cluster.**reassign_group**(*spheres*, *source_group*, *target_group*)

Reassign a group in place

Parameters

- **spheres** (Spheres) – complete set of input spheres
- **source_group** (*float*) – group to change
- **target_group** (*float*) – new group id

pyvol.cluster.**reassign_groups_to_closest**(*spheres*, *group_list*, *radius*, *iterations=None*, *preserve_largest=False*)

Reassign a group to the closest group as identified by maximum linkage; operates in place

Parameters

- **spheres** (Spheres) – complete set of input spheres
- **group_list** (*[float]*) – list of group ids which are to be iteratively reassigned
- **radius** (*float*) – radius at which searches are to take place
- **iterations** (*int*) – number of times to attempt to reassign groups (Default value = None)

- **preserve_largest** – keep the group id of the group with more members? (Default value = False)

pyvol.cluster.**remove_interior**(*spheres*)
Remove all spheres which are completely enclosed in larger spheres; operates in place

Parameters **spheres** (Spheres) – complete set of input spheres

pyvol.cluster.**remove_overlap**(*spheres*, *radii=None*, *spacing=0.1*, *iterations=20*, *tolerance=0.02*)
Remove overlap between groups; operates in place

Parameters

- **spheres** (Spheres) – complete set of input spheres
- **radii** (*[float]*) – radii at which to perform searches for overlap (Default value = None)
- **spacing** (*float*) – binning radius (Default value = 0.1)
- **iterations** (*int*) – number of times to attempt overlap removal (Default value = 20)
- **tolerance** (*float*) – overlap tolerance (Default value = 0.02)

pyvol.identify module

pyvol.identify.**pocket**(*prot_file*, *mode='largest'*, *lig_file=None*, *coordinate=None*, *resid=None*, *residue_coordinates=None*, *min_rad=1.4*, *max_rad=3.4*, *lig_excl_rad=None*, *lig_incl_rad=None*, *subdivide=False*, *minimum_volume=200*, *min_subpocket_rad=1.7*, *min_subpocket_surf_rad=1.0*, *max_clusters=None*, *prefix=None*, *output_dir=None*, *constrain_inputs=False*)
Calculates the SES for a binding pocket

Parameters

- **prot_file** (*str*) – filename for the input pdb file containing the peptidee
- **mode** (*str*) – pocket identification mode (can be largest, all, or specific) (Default value = "largest")
- **lig_file** (*str*) – filename for the input pdb file containing a ligand (Default value = None)
- **coordinate** (*[float]*) – 3D coordinate used for pocket specification (Default value = None)
- **resid** (*str*) – residue identifier for pocket specification (Default value = None)
- **residue_coordinates** (*[float]*) – 3D coordinate of an atom in a surrounding residue used for pocket specification (Default value = None)
- **min_rad** (*float*) – radius for SAS calculations (Default value = 1.4)
- **max_rad** (*float*) – radius used to identify the outer, bulk solvent exposed surface (Default value = 3.4)
- **lig_excl_rad** (*float*) – maximum distance from a provided ligand that can be included in calculated pockets (Default value = None)
- **lig_incl_rad** (*float*) – minimum distance from a provided ligand that should be included in calculated pockets when solvent border is ambiguous (Default value = None)
- **subdivide** (*bool*) – calculate subpockets? (Default value = False)
- **minimum_volume** (*float*) – minimum volume of pockets returned when running in 'all' mode (Default value = 200)

- **min_subpocket_rad** (*float*) – minimum radius that identifies distinct subpockets (Default value = 1.7)
- **min_subpocket_surf_rad** (*float*) – radius used to calculate subpocket surfaces (Default value = 1.0)
- **max_clusters** (*int*) – maximum number of clusters (Default value = None)
- **prefix** (*str*) – identifying string for output (Default value = None)
- **output_dir** (*str*) – filename of the directory in which to place all output; can be absolute or relative (Default value = None)
- **constrain_inputs** (*bool*) – restrict quantitative input parameters to tested values? (Default value = False)

Returns **pockets** – a list of Spheres objects each of which contains the geometric information describing a distinct pocket or subpocket

Return type [Spheres]

pyvol.identify.**subpockets**(*bounding_spheres*, *ref_spheres*, *min_rad*, *max_rad*, *min_subpocket_rad=1.7*, *min_subpocket_surf_rad=1.0*, *max_subpocket_rad=None*, *sampling=0.1*, *inclusion_radius_buffer=1.0*, *min_cluster_size=50*, *max_clusters=None*, *prefix=None*)

Parameters

- **bounding_spheres** (Spheres) – a Spheres object containing both the peptide and solvent exposed face external spheres
- **ref_spheres** (Spheres) – a Spheres object holding the interior spheres that define the pocket to be subdivided
- **min_rad** (*float*) – radius for original SES calculations (Default value = 1.4)
- **max_rad** (*float*) – radius originally used to identify the outer, bulk solvent exposed surface (Default value = 3.4)
- **min_subpocket_rad** (*float*) – minimum radius that identifies distinct subpockets (Default value = 1.7)
- **min_subpocket_surf_rad** (*float*) – radius used to calculate subpocket surfaces (Default value = 1.0)
- **max_subpocket_rad** (*float*) – maximum spheres radius used for subpocket clustering (Default value = None)
- **sampling** (*float*) – radial sampling frequency for clustering (Default value = 0.1)
- **inclusion_radius_buffer** (*float*) – defines the inclusion distance for nonextraneous spheres in combination with min_rad and max_rad (Default value = 1.0)
- **min_cluster_size** (*int*) – minimum number of spheres that can constitute a proper clusterw (Default value = 50)
- **max_clusters** (*int*) – maximum number of clusters (Default value = None)
- **prefix** (*str*) – identifying string for output (Default value = None)

Returns **pockets** – a list of Spheres objects each of which contains the geometric information describing a distinct subpocket

Return type [Spheres]

pyvol.identify.**write_report**(*all_pockets*, *output_dir*, *prefix*)
Write a brief report of calculated volumes to file

Parameters

- **all_pockets** (*[Spherese]*) – a list of Spheres objects each of which contains the complete information about a distinct pocket or subpocket
- **output_dir** (*str*) – output directory, relative or absolute
- **prefix** (*str*) – identifying prefix for output files

pyvol.pymol_interface module

pyvol.pymol_interface.**load_pocket**(*spheres_file*, *name=None*, *display_mode='solid'*, *color='marine'*, *alpha=0.85*)

Parameters

- **spheres_file** (*str*) – filename
- **name** (*str*) – internal display name (Default value = None)
- **display_mode** (*str*) – display mode (Default value = "solid")
- **color** (*str*) – PyMOL color string (Default value = 'marine')
- **alpha** (*float*) – transparency value (Default value = 0.85)

pyvol.pymol_interface.**pocket**(*protein*, *mode=None*, *ligand=None*, *pocket_coordinate=None*, *residue=None*, *resid=None*, *prefix=None*, *min_rad=1.4*, *max_rad=3.4*, *lig_excl_rad=None*, *lig_incl_rad=None*, *display_mode='solid'*, *color='marine'*, *alpha=0.85*, *output_dir=None*, *subdivide=None*, *minimum_volume=200*, *min_subpocket_rad=1.7*, *min_subpocket_surf_rad=1.0*, *max_clusters=None*, *excl_org=False*, *constrain_inputs=True*)
Calculates the SAS for a binding pocket and displays it

Parameters

- **protein** (*str*) – PyMOL selection string for the protein
- **mode** (*str*) – pocket identification mode (can be largest, all, or specific) (Default value = None)
- **ligand** (*str*) – PyMOL selection string for the ligand (Default value = None)
- **pocket_coordinate** (*[float]*) – 3D coordinate used for pocket specification (Default value = None)
- **residue** (*str*) – PyMOL residue selection string for pocket specification (Default value = None)
- **resid** (*str*) – residue identifier for pocket specification (Default value = None)
- **prefix** (*str*) – identifying string for output (Default value = None)
- **min_rad** (*float*) – radius for SAS calculations (Default value = 1.4)
- **max_rad** (*float*) – radius used to identify the outer, bulk solvent exposed surface (Default value = 3.4)
- **lig_excl_rad** (*float*) – maximum distance from a provided ligand that can be included in calculated pockets (Default value = None)

- **lig_incl_rad** (*float*) – minimum distance from a provided ligand that should be included in calculated pockets when solvent border is ambiguous (Default value = None)
- **display_mode** (*str*) – display mode for calculated pockets (Default value = "solid")
- **color** (*str*) – PyMOL color string (Default value = 'marine')
- **alpha** (*float*) – transparency value (Default value = 0.85)
- **output_dir** (*str*) – filename of the directory in which to place all output; can be absolute or relative (Default value = None)
- **subdivide** (*bool*) – calculate subpockets? (Default value = None)
- **minimum_volume** (*float*) – minimum volume of pockets returned when running in 'all' mode (Default value = 200)
- **min_subpocket_rad** (*float*) – minimum radius that identifies distinct subpockets (Default value = 1.7)
- **min_subpocket_surf_rad** (*float*) – radius used to calculate subpocket surfaces (Default value = 1.0)
- **max_clusters** (*int*) – maximum number of clusters (Default value = None)
- **excl_org** (*bool*) – exclude non-peptide atoms from the protein selection? (Default value = False)
- **constrain_inputs** (*bool*) – constrain input quantitative values to tested ranges? (Default value = True)

pyvol.pymol_utilities module

pyvol.pymol_utilities.**construct_palette**(*color_list=None*, *max_value=7*, *min_value=1*)

Construct a palette

Parameters

- **color_list** (*[str]*) – list of PyMOL color strings (Default value = None)
- **max_value** (*int*) – max palette index (Default value = 7)
- **min_value** (*int*) – min palette index (Default value = 1)

Returns **palette** – list of color definitions

Return type [str]

pyvol.pymol_utilities.**display_pseudoatom_group**(*spheres*, *name*, *color='gray60'*, *palette=None*)

Displays a collection of pseudoatoms

Parameters

- **spheres** (Spheres) – Spheres object holding pocket geometry
- **name** (*str*) – display name
- **color** (*str*) – PyMOL color string (Default value = 'gray60')
- **palette** (*[str]*) – palette (Default value = None)

pyvol.pymol_utilities.**display_spheres_object**(*spheres*, *name*, *state=1*, *color='marine'*, *alpha=0.7*, *mode='solid'*, *palette=None*)

Loads a mesh object into a cgo list for display in PyMOL

Parameters

- **spheres** (Spheres) – Spheres object containing all geometry
- **name** (*str*) – display name
- **state** (*int*) – model state (Default value = 1)
- **color** (*str*) – PyMOL color string (Default value = 'marine')
- **alpha** (*float*) – transparency value (Default value = 0.7)
- **mode** (*str*) – display mode (Default value = "solid")
- **palette** (*[str]*) – palette (Default value = None)

pyvol.pymol_utilities.**mesh_to_solid_CGO**(*mesh*, *color='gray60'*, *alpha=1.0*)
Creates a solid CGO object for a mesh for display in PyMOL

Parameters

- **mesh** (*Trimesh*) – Trimesh mesh object
- **color** (*str*) – PyMOL color string (Default value = 'gray60')
- **alpha** (*float*) – transparency value (Default value = 1.0)

Returns **cgobuffer** – CGO buffer that contains the instruction to load a solid object

Return type str

pyvol.pymol_utilities.**mesh_to_wireframe_CGO**(*mesh*, *color='gray60'*, *alpha=1.0*)
Creates a wireframe CGO object for a mesh for display in PyMOL

Parameters

- **mesh** (*Trimesh*) – Trimesh mesh object
- **color** (*str*) – PyMOL color string (Default value = 'gray60')
- **alpha** (*float*) – transparency value (Default value = 1.0)

Returns **cgobuffer** – CGO buffer that contains the instruction to load a wireframe object

Return type str

pyvol.spheres module

class pyvol.spheres.**Spheres**(*xyz=None*, *r=None*, *xyzr=None*, *xyzrg=None*, *g=None*, *pdb=None*, *bv=None*, *mesh=None*, *name=None*, *spheres_file=None*)
Bases: object

__add__(*other*)
Create a new Spheres object by overloading addition to concatenate xyzr contents Does not add meshes (just spheres)

copy()

calculate_surface(*probe_radius=1.4*, *cavity_atom=None*, *coordinate=None*, *all_components=False*, *exclusionary_radius=2.5*, *largest_only=False*, *noh=True*, *minimum_volume=200*)
Calculate the SAS for a given probe radius

Parameters

- **probe_radius** (*float*) – radius for surface calculations (Default value = 1.4)
- **cavity_atom** (*int*) – id of a single atom which lies on the surface of the interior cavity of interest (Default value = None)

- **coordinate** (*[float]*) – 3D coordinate to identify a cavity atom (Default value = None)
- **all_components** (*bool*) – return all pockets? (Default value = False)
- **exclusionary_radius** (*float*) – maximum permissibile distance to the closest identified surface element from the supplied coordinate (Default value = 2.5)
- **largest_only** (*bool*) – return only the largest pocket? (Default value = False)
- **noh** (*bool*) – remove waters before surface calculation? (Default value = True)
- **minimum_volume** (*int*) – minimum volume of pockets returned when using 'all_components' (Default value = 200)

identify_nonextraneous (*ref_spheres*, *radius*)
Returns all spheres less than radius away from any center in ref_spheres using cKDTree search built on the non-reference set

Parameters

- **ref_spheres** (Spheres) – object that defines the pocket of interest
- **radius** (*float*) – maximum distance to sphere centers to be considered nonextraneous

Returns **nonextraneous** – a filtered Spheres object

Return type *Spheres*

nearest (*coordinate*, *max_radius=None*)
Returns the index of the sphere closest to a coordinate; if max_radius is specified, the sphere returned must have a radius <= max_radius

Parameters

- **coordinate** (*float nx3*) – 3D input coordinate
- **max_radius** (*float*) – maximum permissibile distance to the nearest sphere (Default value = None)

Returns index of the closest sphere

Return type nearest_index

nearest_coord_to_external (*coordinates*)
Returns the coordinate of the sphere closest to the supplied coordinates

Parameters **coordinates** (*float nx3*) – set of coordinates

Returns **coordinate** – coordinate of internal sphere closest to the supplied coordinates

Return type float 1x3

remove_duplicates (*eps=0.01*)
Remove duplicate spheres by identifying centers closer together than eps using DBSCAN

Parameters **eps** (*float*) – DBSCAN input parameter (Default value = 0.01)

remove_ungrouped ()
Remove all spheres that did not adequately cluster with the remainder of the set

remove_groups (*groups*)
Remove all spheres with specified group affiliations

Parameters **groups** (*[float]*) – list of groups to remove

write (*filename*, *contents='xyzrg'*, *output_mesh=True*)
Writes the contents of _xyzrg to a space delimited file

Parameters

- **filename** (*str*) – filename to write the report and mesh if indicated
- **contents** (*str*) – string describing which columns to write to file (Default value = "xyzrg")
- **output_mesh** (*bool*) – write mesh to file? (Default value = True)

xyzrg
Retrieve the coordinates, radii, and group ids

xyzr
Retrieve coordinates and radii

xyz
Retrieve the coordinates

r
Retrieve the radii

g
Retrieve the group indices

pyvol.utilities module

pyvol.utilities.**check_dir**(*location*)
Ensure that a specified directory exists

Parameters **location** (*str*) – target directory

pyvol.utilities.**coordinates_for_resid**(*pdb_file*, *resid*, *chain=None*, *model=0*)
Extract the 3D coordinates for all atoms in a specified residue from a pdb file

Parameters

- **pdb_file** (*str*) – filename of the specified pdb file
- **resid** (*int*) – residue number
- **chain** (*str*) – chain identifier (Default value = None)
- **model** (*int*) – model identifier (Default value = 0)

Returns **coordinates** – 3xN array containing all atomic positions

Return type [[float]]

pyvol.utilities.**run_cmd**(*options*, *in_directory=None*)
Run a program using the command line

Parameters

- **options** (*[str]*) – list of command line options
- **in_directory** (*str*) – directory in which to run the command (Default value = None)

pyvol.utilities.**surface_multiprocessing**(*args*)
A single surface calculation designed to be run in parallel

Parameters **args** – a tuple containing: spheres (Spheres): a Spheres object containing all surface producing objects probe_radius (float): radius to use for probe calculations kwargs (dict): all remaining arguments accepted by the surface calculation algorithm

Returns **surface** – the input Spheres object but with calculated surface parameters

Return type *Spheres*

pyvol.utilities.**sphere_multiprocessing**(*spheres*, *radii*, *workers=None*, ***kwargs*)

A wrapper function to calculate multiple surfaces using multiprocessing

Parameters

- **spheres** (Spheres) – input Spheres object
- **radii** (*[float]*) – list of radii at which surfaces will be calculated
- **workers** (*int*) – number of workers (Default value = None)
- ****kwargs** (*dict*) – all remaining arguments accepted by surface calculation that are constant across parallel calculations

Returns **surfaces** – a list of Spheres object each with its surface calculated

Return type [Spheres]

Module contents

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

p

Symbols

C

D

E

G

H

I

L

M

N

P

R

S

W

X