

## NAME

PyMOLVisualizeMacromolecules.py - Visualize macromolecules

## SYNOPSIS

```
PyMOLVisualizeMacromolecules.py [--align <yes or no>] [--alignMethod <align, cealign, super>] [
--alignMode <FirstChain or Complex>] [--alignRefFile <filename>] [--allowEmptyObjects <yes or no>] [
--BFactorChain <yes or no>] [--BFactorColorPalette <text>] [--chainIDs <First, All or ID1,ID2...>] [
--disulfideBondsChain <yes or no>] [--dockedPoses <PDBFile,ChainID,LigandID,InputFiles,...>] [
--dockedPosesDistanceContactsCutoffs <number1,number2...>] [
--dockedPosesDistanceContactsColor <text>] [--dockedPosesGroupName <text>] [
--dockedPosesName <text>] [--ignoreHydrogens <yes or no>] [--ligandIDs <Largest, All or
ID1,ID2...>] [--labelFontID <number>] [--PMLOut <yes or no>] [--pocketContactsInorganicColor
<text>] [--pocketContactsInorganicPiCationColor <text>] [--pocketContactsLigandColor <text>] [
--pocketContactsLigandHydrophobicColor <text>] [--pocketContactsLigandHalogenColor <text>] [
--pocketContactsLigandPiCationColor <text>] [--pocketContactsLigandPiPiColor <text>] [
--pocketContactsSolventColor <text>] [--pocketContactsCutoff <number>] [--pocketDistanceCutoff
<number>] [--pocketLabelColor <text>] [--pocketResidueTypes <yes or no>] [--pocketSurface <yes or
no>] [--pocketSurfaceElectrostatics <yes or no>] [--residueTypes <Type,Color,ResNames,...>] [
--residueTypesChain <yes or no>] [--saltBridgesChain <yes or no>] [--saltBridgesChainContactsColor
<text>] [--saltBridgesChainCutoff <number>] [--saltBridgesChainResidues <Type, ResNames,...>] [
--selectionsChain <ObjectName,SelectionSpec,...>] [--selectionsChainSurface <yes or no>] [
--selectionsChainStyle <DisplayStyle>] [--selectionsPocket <ObjectName,SelectionSpec,...>] [
--selectionsPocketSurface <yes or no>] [--selectionsPocketStyle <DisplayStyle>] [--surfaceChain <yes
or no>] [--surfaceChainElectrostatics <yes or no>] [--surfaceChainComplex <yes or no>] [
--surfaceComplex <yes or no>] [--surfaceColor <ColorName>] [--surfaceColorPalette <RedToWhite or
WhiteToGreen>] [--surfaceAtomTypesColors <ColorType,ColorSpec,...>] [--surfaceTransparency
<number>] [--trajectories <PDBTopologyFile,TrajFiles,...>] [--overwrite] [-w <dir>] -i
<infile1,infile2,infile3...> -o <outfile>
```

PyMOLVisualizeMacromolecules.py -h | --help | -e | --examples

## DESCRIPTION

Generate PyMOL visualization files for viewing surfaces, chains, ligands, ligand binding pockets, and interactions between ligands and binding pockets in macromolecules including proteins and nucleic acids.

The supported input file format are: PDB (.pdb), CIF (.cif)

The supported output file formats are: PyMOL script file (.pml), PyMOL session file (.pse)

A variety of PyMOL groups and objects may be created for visualization of macromolecules. These groups and objects correspond to complexes, surfaces, chains, ligands, inorganics, ligand binding pockets, pocket, polar interactions, and pocket hydrophobic surfaces. A complete hierarchy of all possible PyMOL groups and objects is shown below:

```
<PDBFileRoot>
  .Complex
    .Complex
    .Surface
  .Trajectories
    .Topology
    .<TrajFileID>
      .Trajectory
    .<TrajFileID>
      ... ..
    .<TrajFileID>
      ... ..
  .Chain<ID>
    .Complex
      .Complex
      .Surface
    .Chain
      .Chain
      .BFactor
      .Putty
      .Cartoon
    .Selections
      .<Name>
        .Selection
        .Surface
```

```
.Surface
.Hydrophobicity
.Hydrophobicity_Charge
.<Name>
... ..
.Residues
.Aromatic
.Residues
.Surface
.Hydrophobic
.Residues
.Surface
.Polar
.Residues
.Surface
.Positively_Charged
.Residues
.Surface
.Negatively_Charged
.Residues
.Surface
.Other
.Residues
.Surface
.Surface
.Surface
.Hydrophobicity
.Hydrophobicity_Charge
.Vacuum_Electrostatics
.Contact_Potentials
.Map
.Legend
.Volume
.Disulfide_Bonds
.Residues
.Salt_Bridges
.Residues
.Positively_Charged
.Negatively_Charged
.Contacts
.Solvent
.Inorganic
.Ligand<ID>
.Ligand
.Ligand
.BallAndStick
.Pocket
.Pocket
.Polar_Contacts
.Hydrophobic_Contacts
.Pi_Pi_Contacts
.Pi_Cation_Contacts
.Halogen_Contacts
.Selections
.<Name>
.Selection
.Surface
.Surface
.Hydrophobicity
.Hydrophobicity_Charge
.<Name>
... ..
.Residues
.Aromatic
.Residues
.Surface
.Hydrophobic
.Residues
.Surface
```

```

        .Polar
            .Residues
            .Surface
        .Positively_Charged
            .Residues
            .Surface
        .Negatively_Charged
            .Residues
            .Surface
        .Other
            .Residues
            .Surface
    .Surfaces
        .Surface
            .Surface
            .Hydrophobicity
            .Hydrophobicity_Charge
            .Vacuum_Electrostatics
                .Contact_Potentials
                .Map
                .Legend
        .Cavity
            .Surface
            .Hydrophobicity
            .Hydrophobicity_Charge
            .Vacuum_Electrostatics
                .Contact_Potentials
                .Map
                .Legend
    .Pocket_Solvent
        .Pocket_Solvent
        .Polar_Contacts
    .Pocket_Inorganic
        .Pocket_Inorganic
        .Polar_Contacts
        .Pi_Cation_Contacts
.Ligand<ID>
    .Ligand
        ... ..
    .Pocket
        ... ..
    .Pocket_Solvent
        ... ..
    .Pocket_Inorganic
        ... ..
.Docked_Poses or <CustomLabel>
    .<InputFileID>
        .Poses or <CustomLabel>
        .Pocket
            .Pocket
            .Distance_Contacts
                .Contact1_At_<Distance>
                .Contact1_At_<Distance>
                ... ..
            .Polar_Contacts
            .Hydrophobic_Contacts
            .Pi_Pi_Contacts
            .Pi_Cation_Contacts
            .Halogen_Contacts
        .Pocket_Solvent
            .Pocket_Solvent
            .Polar_Contacts
        .Pocket_Inorganic
            .Polar_Contacts
            .Pi_Cation_Contacts
    .<InputFileID>
        ... ..
    .<InputFileID>
        ... ..

```

```

.Chain<ID>
... ..
.Ligand<ID>
... ..
.Ligand<ID>
... ..
.Chain<ID>
... ..
<PDBFileRoot>
.Complex
... ..
.Chain<ID>
... ..
.Ligand<ID>
... ..
.Ligand<ID>
... ..
.Chain<ID>
... ..

```

The hydrophobic and electrostatic surfaces are not created for complete complex and chain complex in input file(s) by default. A word to the wise: The creation of surface objects may slow down loading of PML file and generation of PSE file, based on the size of input complexes. The generation of PSE file may also fail.

## OPTIONS

**-a, --align <yes or no> [default: no]**

Align input files to a reference file before visualization. The docked poses and trajectories are not aligned.

**--alignMethod <align, cealign, super> [default: super]**

Alignment methodology to use for aligning input files to a reference file.

**--alignMode <FirstChain or Complex> [default: FirstChain]**

Portion of input and reference files to use for spatial alignment of input files against reference file. Possible values: FirstChain or Complex.

The FirstChain mode allows alignment of the first chain in each input file to the first chain in the reference file along with moving the rest of the complex to coordinate space of the reference file. The complete complex in each input file is aligned to the complete complex in reference file for the Complex mode.

**--alignRefFile <filename> [default: FirstInputFile]**

Reference input file name. The default is to use the first input file name specified using '-i, --infile' option.

**--allowEmptyObjects <yes or no> [default: no]**

Allow creation of empty PyMOL objects corresponding to solvent and inorganic atom selections across chains and ligands in input file(s). By default, the empty objects are marked for deletion.

**-b, --BFactorChain <yes or no> [default: yes]**

A cartoon and putty around individual chains colored by an arbitrary set of B factor values. The minimum and maximum values for B factors are automatically detected. These values may indicate spread of electron density around atoms or correspond to any other property mapped to B factors in input file.

**--BFactorColorPalette <text> [default: blue\_white\_red]**

Color palette for coloring cartoon and putty around chains generated using B factor values. Any valid PyMOL color palette name is allowed. No validation is performed. The complete list of valid color palette names is available at: [pymolwiki.org/index.php/Spectrum](http://pymolwiki.org/index.php/Spectrum). Examples: blue\_white\_red, blue\_white\_magenta, blue\_red, green\_white\_red, green\_red.

**-c, --chainIDs <First, All or ID1,ID2...> [default: First]**

List of chain IDs to use for visualizing macromolecules. Possible values: First, All, or a comma delimited list of chain IDs. The default is to use the chain ID for the first chain in each input file.

**-d --disulfideBondsChain <yes or no> [default: auto]**

Disulfide bonds for chains. By default, the disulfide bonds group is automatically created for chains

containing amino acids and skipped for chains only containing nucleic acids.

--dockedPoses <PDBFile,ChainID,LigandID,InputFiles,...> [default: none]

PDB file name, pocket chain ID, ligand specification, and input files to use for creating pockets to visualize docked poses or any arbitrary set of molecules. Any valid PyMOL input file format is allowed.

It's a quartet of comma limited values corresponding to PDF file name, pocket chain ID, ligand ID, and input files. Multiple input file names are delimited by spaces.

The supported values for docked poses are shown below:

```
PDBFile: A valid PDB file name
ChainID: A valid Chain ID
LigandID: A valid Ligand ID or UseInputFile
InputFiles: A space delimited list of input file names
```

All docked pose values must be specified. No default values are assigned.

The 'ChainID' and 'LigandID' are used for creating pocket to visualize docked poses.

The 'LigandID' must be a valid ligand ID in 'ChainID'. Alternatively, you may use input files by specifying 'UseInputFile' to select residues in 'ChainID' for creating pockets to visualize docked poses.

--dockedPosesDistanceContactsCutoffs <number1,number2...> [default: none]

A comma delimited list of distances in Angstroms for identifying and displaying distance contacts between heavy atoms in pocket residues and docked poses. A PyMOL distance contact object is created for each specified distance. You may find it helpful for identifying steric clashes between docked poses and pocket residues. The maximum distance cutoff value must be less than the specified value for '--pocketDistanceCutoff' option.

--dockedPosesDistanceContactsColor <text> [default: red]

Color for drawing distance contacts between docked poses and pocket residues. The specified value must be valid color. No validation is performed.

--dockedPosesGroupName <text> [default: Docked\_Poses]

PyMOL object name for docked poses group. You may use an arbitrary name to reflect data in input file(s) specified in '--dockedPoses' option. It must be a valid PyMOL object name. No validation is performed.

--dockedPosesName <text> [default: Poses]

PyMOL object name for docked poses object. You may use an arbitrary name to reflect data in input file(s) specified in '--dockedPoses' option. It must be a valid PyMOL object name. No validation is performed.

-e, --examples

Print examples.

-h, --help

Print this help message.

-i, --infile1 <infile1,infile2,infile3...>

Input file names.

--ignoreHydrogens <yes or no> [default: yes]

Ignore hydrogens for ligand, pocket, selection, and residue type views.

-l, --ligandIDs <Largest, All or ID1,ID2...> [default: Largest]

List of ligand IDs present in chains for visualizing macromolecules to highlight ligand interactions. Possible values: Largest, All, or a comma delimited list of ligand IDs. The default is to use the largest ligand present in all or specified chains in each input file.

Ligands are identified using organic selection operator available in PyMOL. It'll also identify buffer molecules as ligands. The largest ligand contains the highest number of heavy atoms.

--labelFontID <number> [default: 7]

Font ID for drawing labels. Default: 7 (Sans Bold). Valid values: 5 to 16. The specified value must be a valid PyMOL font ID. No validation is performed. The complete lists of valid font IDs is available at: [pymolwiki.org/index.php/Label\\_font\\_id](http://pymolwiki.org/index.php/Label_font_id). Examples: 5 - Sans; 7 - Sans Bold; 9 - Serif; 10 - Serif Bold.

-o, --outfile <outfile>

Output file name.

-p, --PMLOut <yes or no> [default: yes]

Save PML file during generation of PSE file.

--pocketContactsInorganicColor <text> [default: deepsalmon]

Color for drawing polar contacts between inorganic and pocket residues. The specified value must be valid color. No validation is performed.

--pocketContactsInorganicPiCationColor <text> [default: limon]

Color for drawing pi cation contacts between inorganic and pocket residues. The specified value must be valid color. No validation is performed. The pi cation contacts are drawn using PyMOL distance command with support for mode 7 and may require incentive version of PyMOL.

--pocketContactsLigandColor <text> [default: orange]

Color for drawing polar contacts between ligand and pocket residues. The specified value must be valid color. No validation is performed.

--pocketContactsLigandHalogenColor <text> [default: magenta]

Color for drawing halogen contacts between ligand and pocket residues. The specified value must be valid color. No validation is performed.

--pocketContactsLigandHydrophobicColor <text> [default: purpleblue]

Color for drawing hydrophobic contacts between ligand and pocket residues. The specified value must be valid color. No validation is performed. The hydrophobic contacts are shown between pairs of carbon atoms not connected to hydrogen bond donor or acceptors atoms as identified by PyMOL.

--pocketContactsLigandPiCationColor <text> [default: yelloworange]

Color for drawing pi cation contacts between ligand and pocket residues. The specified value must be valid color. No validation is performed. The pi cation contacts are drawn using PyMOL distance command with support for mode 7 and may require incentive version of PyMOL.

--pocketContactsLigandPiPiColor <text> [default: cyan]

Color for drawing pi pi contacts between ligand and pocket residues. The specified value must be valid color. No validation is performed. The pi pi contacts are drawn using PyMOL distance command with support for mode 6 and may require incentive version of PyMOL.

--pocketContactsSolventColor <text> [default: marine]

Color for drawing polar contacts between solvent and pocket residues.. The specified value must be valid color. No validation is performed.

--pocketContactsCutoff <number> [default: 4.0]

Distance in Angstroms for identifying polar, hydrophobic contacts, pi pi, pi cation, and halogen contacts between atoms in pocket residues and ligands.

--pocketDistanceCutoff <number> [default: 5.0]

Distance in Angstroms for identifying pocket residues around ligands.

--pocketLabelColor <text> [default: magenta]

Color for drawing residue or atom level labels for a pocket. The specified value must be valid color. No validation is performed.

--pocketResidueTypes <yes or no> [default: auto]

Pocket residue types. The residue groups are generated using residue types, colors, and names specified by '--residueTypes' option. It is only valid for amino acids. By default, the residue type groups are automatically created for pockets containing amino acids and skipped for chains only containing nucleic acids.

--pocketSurface <yes or no> [default: auto]

Surfaces around pocket residues colored by hydrophobicity alone and both hydrophobicity and charge. The hydrophobicity surface is colored at residue level using Eisenberg hydrophobicity scale for residues and color gradient specified by '--surfaceColorPalette' option. The hydrophobicity and charge surface is colored [ Ref 140 ] at atom level using colors specified for groups of atoms by '--surfaceAtomTypesColors' option. This scheme allows simultaneous mapping of hydrophobicity and charge values on the surfaces.

The cavity surfaces around ligands are also generated. These surfaces are colored by hydrophobicity along and both hydrophobicity and charge.

This option is only valid for amino acids. By default, both surfaces are automatically created for pockets containing amino acids and skipped for pockets containing only nucleic acids.

--pocketSurfaceElectrostatics <yes or no> [default: no]

Vacuum electrostatics contact potential surface around pocket residues. A word to the wise from PyMOL documentation: The computed protein contact potentials are only qualitatively useful, due to short cutoffs, truncation, and lack of solvent "screening".

The cavity surface around ligands is also generated. This surface is colored by vacuum electrostatics contact potential.

This option is only valid for amino acids. By default, the electrostatics surface is automatically created for chains containing amino acids and skipped for chains containing only nucleic acids.

-r, --residueTypes <Type,Color,ResNames,...> [default: auto]

Residue types, colors, and names to generate for residue groups during '--pocketResidueTypes' and '--residueTypesChain' option. It is only valid for amino acids.

It is a triplet of comma delimited list of amino acid residues type, residues color, and a space delimited list three letter residue names.

The default values for residue type, color, and name triplets are shown below:

```
Aromatic,brightorange,HIS PHE TRP TYR,
Hydrophobic,orange,ALA GLY VAL LEU ILE PRO MET,
Polar,palegreen,ASN GLN SER THR CYS,
Positively_Charged,marine,ARG LYS,
Negatively_Charged,red,ASP GLU
```

The color name must be a valid PyMOL name. No validation is performed. An amino acid name may appear across multiple residue types. All other residues are grouped under 'Other'.

--residueTypesChain <yes or no> [default: auto]

Chain residue types. The residue groups are generated using residue types, colors, and names specified by '--residueTypes' option. It is only valid for amino acids. By default, the residue type groups are automatically created for chains containing amino acids and skipped for chains only containing nucleic acids. --saltBridgesChain <yes or no> [default: auto] Salt bridges for chains. By default, the salt bridges group is automatically created for chains containing amino acids and skipped for chains only containing nucleic acids. The salt bridges correspond to polar contacts between positively and negatively charged residues in a chain.

--saltBridgesChainContactsColor <text> [default: brightorange]

Color for drawing polar contacts between positively and negatively charged residues involved in salt bridges. The specified value must be valid color. No validation is performed.

--saltBridgesChainCutoff <number> [default: 4.0]

Distance in Angstroms for identifying polar contacts between positively and negatively charged residues involved in salt bridges in a chain. --saltBridgesChainResidues <Type, ResNames,...> [default: auto] Residue types and names to use for identifying positively and negatively charged residues involved in salt bridges.

It is a pair of comma delimited list of amino acid residue types and a space delimited list three letter residue names.

The default values for residue type and name pairs are shown below:

```
Positively_Charged,ARG LYS HIS HSP
Negatively_Charged,ASP GLU
```

The residue names must be valid names. No validation is performed.

--selectionsChain <ObjectName,SelectionSpec,...> [default: none]

Custom selections for chains. It is a pairwise list of comma delimited values corresponding to PyMOL object names and selection specifications. The selection specification must be a valid PyMOL specification. No validation is performed.

The PyMOL objects are created for each chain corresponding to the specified selections. The display style for PyMOL objects is set using value of '--selectionsChainStyle' option.

The specified selection specification is automatically appended to appropriate chain specification before creating PyMOL objects.

For example, the following specification for '--selectionsChain' option will generate PyMOL objects for chains containing Cysteines and Serines:

```
Cysteines,resn CYS,Serines,resn SER
```

--selectionsChainSurface <yes or no> [default: auto]

Surfaces around individual chain selections colored by hydrophobicity alone and both hydrophobicity and charge. This option is similar to '--surfaceChain' options for creating surfaces for chain. Additional details are available in the documentation section for '--surfaceChain' options.

--selectionsChainStyle <DisplayStyle> [default: sticks]

Display style for PyMOL objects created for '--selectionsChain' option. It must be a valid PyMOL display style. No validation is performed.

--selectionsPocket <ObjectName,SelectionSpec,...> [default: none]

Custom selections for pocket residues. It is a pairwise list of comma delimited values corresponding to PyMOL object names and selection specifications. The selection specification must be a valid PyMOL specification. No validation is performed.

The PyMOL objects are created for each pocket corresponding to the specified selections. The display style for PyMOL objects is set using value of '--selectionsChainStyle' option.

The specified selection specification is automatically appended to appropriate pocket specification before creating PyMOL objects.

For example, the following specification for '--selectionsPocket' option will generate PyMOL objects for pockets containing Tyrosines and Serines:

```
Tyrosines,resn TYR,Serines,resn SER
```

--selectionsPocketSurface <yes or no> [default: auto]

Surfaces around individual pocket chain selections colored by hydrophobicity alone and both hydrophobicity and charge. This option is similar to '--surfaceChain' options for creating surfaces for chain. Additional details are available in the documentation section for '--surfaceChain' options.

--selectionsPocketStyle <DisplayStyle> [default: sticks]

Display style for PyMOL objects created for '--selectionsPocket' option. It must be a valid PyMOL display style. No validation is performed.

--surfaceChain <yes or no> [default: auto]

Surfaces around individual chain colored by hydrophobicity alone and both hydrophobicity and charge. The hydrophobicity surface is colored at residue level using Eisenberg hydrophobicity scale for residues and color gradient specified by '--surfaceColorPalette' option. The hydrophobicity and charge surface is colored [ Ref 140 ] at atom level using colors specified for groups of atoms by '--surfaceAtomTypesColors' option. This scheme allows simultaneous mapping of hydrophobicity and charge values on the surfaces.

This option is only valid for amino acids. By default, both surfaces are automatically created for chains containing amino acids and skipped for chains containing only nucleic acids.

--surfaceChainElectrostatics <yes or no> [default: no]

Vacuum electrostatics contact potential surface and volume around individual chain. A word to the wise from PyMOL documentation: The computed protein contact potentials are only qualitatively useful, due to short cutoffs, truncation, and lack of solvent "screening".

This option is only valid for amino acids. By default, the electrostatics surface and volume are automatically created for chains containing amino acids and skipped for chains containing only nucleic acids.

--surfaceChainComplex <yes or no> [default: no]

Hydrophobic surface around chain complex. The surface is colored by hydrophobicity. It is only valid for amino acids.

--surfaceComplex <yes or no> [default: no]

Hydrophobic surface around complete complex. The surface is colored by hydrophobicity. It is only valid for amino acids.

--surfaceAtomTypesColors <ColorType,ColorSpec,...> [default: auto]

Atom colors for generating surfaces colored by hydrophobicity and charge around chains and pockets in proteins. It's a pairwise comma delimited list of atom color type and color specification for groups of



atoms. The default values for color types [ Ref 140 ] along with color specifications are shown below:

```
HydrophobicAtomsColor, yellow,
NegativelyChargedAtomsColor, red,
PositivelyChargedAtomsColor, blue,
OtherAtomsColor, gray90
```

The color names must be valid PyMOL names.

The color values may also be specified as space delimited RGB triplets:

```
HydrophobicAtomsColor, 0.95 0.78 0.0,
NegativelyChargedAtomsColor, 1.0 0.4 0.4,
PositivelyChargedAtomsColor, 0.2 0.5 0.8,
OtherAtomsColor, 0.95 0.95 0.95
```

--surfaceColor <ColorName> [default: lightblue]

Color name for surfaces around chains and pockets. This color is not used for surfaces colored by hydrophobicity and charge. The color name must be a valid PyMOL name.

--surfaceColorPalette <RedToWhite or WhiteToGreen> [default: RedToWhite]

Color palette for hydrophobic surfaces around chains and pockets in proteins. Possible values: RedToWhite or WhiteToGreen from most hydrophobic amino acid to least hydrophobic. The colors values for amino acids are taken from color\_h script available as part of the Script Library at PyMOL Wiki.

--surfaceTransparency <number> [default: 0.25]

Surface transparency for molecular surfaces.

-t, --trajectories <PDBTopologyFile,TrajFiles,...> [default: none]

PDB topology file name and MD trajectories files for visualizing trajectories for PDB files.

It's a pair of comma limited values corresponding to a PDB file name and MD trajectory files. Multiple trajectory file names are delimited by spaces.

The supported values for trajectories are shown below:

```
PDBTopologyFile: A valid PDB file name
TrajFiles: A space delimited list of MD trajectory file names
```

The trajectory files must correspond to the specified PDB topology file. In addition, the format of trajectory files must be a valid PyMOL format. PyMOL uses Molfile Plugin, which supports a variety of trajectory file formats. For example: HARMM, NAMD, X-PLOR (.dcd), Gromacs TRR/XTC (.trr, .xtc), XYZ (.xyz) etc.

--overwrite

Overwrite existing files.

-w, --workingdir <dir>

Location of working directory which defaults to the current directory.

## EXAMPLES

To visualize the first chain, the largest ligand in the first chain, and ligand binding pockets to highlight ligand interaction with pocket residues, solvents and inorganics, in a PDB file, and generate a PML file, type:

```
% PyMOLVisualizeMacromolecules.py -i Sample4.pdb -o Sample4.pml
```

To visualize the first chain along with all cysteines and serines, the largest ligand in the first chain, and ligand binding pockets to highlight ligand interaction with pocket residues, solvents and inorganics, in a PDB file, and generate a PML file, type:

```
% PyMOLVisualizeMacromolecules.py -i Sample4.pdb -o Sample4.pml
--selectionsChain "Cysteines,resn cys,Serines,resn ser"
```

To visualize the first chain along with all serines and tyrosines in binding pockets, the largest ligand in the first chain, and ligand binding pockets to highlight ligand interaction with pocket residues, solvents and inorganics, in a PDB file, and generate a PML file, type:

```
% PyMOLVisualizeMacromolecules.py -i Sample4.pdb -o Sample4.pml
--selectionsPocket "Serines,resn ser,Tyrosines,resn tyr"
```

To visualize docking poses from a SD file in a pocket corresponding to a SD file for a chain, along with visualization of other information, in a PDB file, and generate a PML file, type:

```
% PyMOLVisualizeMacromolecules.py -c All -l "N3" -i SampleMpro6LU7.pdb
-o SampleMpro6LU7.pml --dockedPoses "SampleMpro6LU7.pdb,A,UseInputFile,
SampleMproDockedPosesTop100.sdf"
```

To visualize docking poses from a SD file in a pocket corresponding to SD file for a chain, along with visualization of distance contacts at 3.0 and 3.5 Angstroms, in a PDB file, and generate a PML file, type:

```
% PyMOLVisualizeMacromolecules.py -c All -l "N3" -i SampleMpro6LU7.pdb
-o SampleMpro6LU7.pml --dockedPoses "SampleMpro6LU7.pdb,A,UseInputFile,
SampleMproDockedPosesTop100.sdf"
--dockedPosesDistanceContactsCutoffs "3.0, 3.5"
```

To visualize docking poses from a SD file in a pocket corresponding to a specific ligand a chain, along with visualization of other information, in a PDB file, and generate a PML file, type:

```
% PyMOLVisualizeMacromolecules.py -c All -l "N3" -i SampleMpro6LU7.pdb
-o SampleMpro6LU7.pml --dockedPoses "SampleMpro6LU7.pdb,A,N3,
SampleMproDockedPosesTop100.sdf"
```

To visualize docking poses from multiple SDs file in a pocket corresponding SD file a for a chain, along with visualization of other information, in a PDB file, and generate a PML file, type:

```
% PyMOLVisualizeMacromolecules.py -c All -l "N3" -i SampleMpro6LU7.pdb
-o SampleMpro6LU7.pml --dockedPoses "SampleMpro6LU7.pdb,A,UseInputFile,
SampleMproDockedPosesTop100.sdf SampleMproDockedPosesDiverse100.sdf"
```

To visualize trajectory from a DCD file, along with visualization of other information, corresponding to a PDB topology file and generate a PML file, type:

```
% PyMOLVisualizeMacromolecules.py -t "Sample10.pdb, Sample10.dcd"
-i Sample10.pdb -o Sample10.pml
```

To visualize all chains, all ligands in all chains, and all ligand binding pockets to highlight ligand interaction with pocket residues, solvents and inorganics, in a PDB file, and generate a PML file, type:

```
% PyMOLVisualizeMacromolecules.py -c All -l All -i Sample4.pdb -o
Sample4.pml
```

To visualize all chains, ligands, and ligand binding pockets along with displaying all hydrophobic surfaces and chain electrostatic surface, in a PDB file, and generate a PML file, type:

```
% PyMOLVisualizeMacromolecules.py -c All -l All
--surfaceChainElectrostatics yes --surfaceChainComplex yes
--surfaceComplex yes -i Sample4.pdb -o Sample4.pml
```

To visualize chain E, ligand ADP in chain E, and ligand binding pockets to highlight ligand interaction with pocket residues, solvents and inorganics, in a PDB file, and generate a PML file, type:

```
% PyMOLVisualizeMacromolecules.py -c E -l ADP -i Sample3.pdb
-o Sample3.pml
```

To visualize chain E, ligand ADP in chain E, and ligand binding pockets to highlight ligand interaction with pocket residues, solvents and inorganics, in a PDB file, and generate a PSE file, type:

```
% PyMOLVisualizeMacromolecules.py -c E -l ADP -i Sample3.pdb
-o Sample3.pse
```

To visualize the first chain, the largest ligand in the first chain, and ligand binding pockets to highlight ligand interaction with pocket residues, solvents and inorganics, in PDB files, along with aligning first chain in each input file to the first chain in first input file, and generate a PML file, type:

```
% PyMOLVisualizeMacromolecules.py --align yes -i
"Sample5.pdb,Sample6.pdb,Sample7.pdb" -o SampleOut.pml
```

To visualize all chains, all ligands in all chains, and all ligand binding pockets to highlight ligand interaction with pocket residues, solvents and inorganics, in PDB files, along with aligning first chain in each input file to the first chain in first input file, and generate a PML file, type:

```
% PyMOLVisualizeMacromolecules.py --align yes -c All -l All -i  
"Sample5.pdb,Sample6.pdb,Sample7.pdb" -o SampleOut.pml
```

To visualize all chains, all ligands in all chains, and all ligand binding pockets to highlight ligand interaction with pocket residues, solvents and inorganics, in PDB files, along with aligning first chain in each input file to the first chain in a specified PDB file using a specified alignment method, and generate a PML file, type:

```
% PyMOLVisualizeMacromolecules.py --align yes --alignMode FirstChain  
--alignRefFile Sample5.pdb --alignMethod super -c All -l All -i  
"Sample5.pdb,Sample6.pdb,Sample7.pdb" -o SampleOut.pml
```

## AUTHOR

Manish Sud(msud@san.rr.com)

## SEE ALSO

DownloadPDBFiles.pl, PyMOLVisualizeCavities.py, PyMOLVisualizeCryoEMDensity.py,  
PyMOLVisualizeElectronDensity.py, PyMOLVisualizeInterfaces.py, PyMOLVisualizeSurfaceAndBuriedResidues.py

## COPYRIGHT

Copyright (C) 2025 Manish Sud. All rights reserved.

The functionality available in this script is implemented using PyMOL, a molecular visualization system on an open source foundation originally developed by Warren DeLano.

This file is part of MayaChemTools.

MayaChemTools is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.