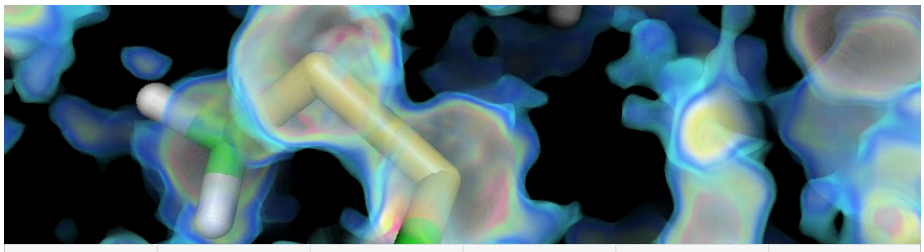
Home Products Buy News Support Contact

PyMOL

A **USER-SPONSORED** molecular visualization system on an **OPEN-SOURCE** foundation

DOWNLOAD
Version 1.3

PyMOL runs on



VIEW 3D Molecular Structures | RENDER Figures Artistically | ANIMATE Molecules Dynamically | EXPORT Geometry Data | PRESENT 3D Data in PowerPoint | DEVELOP Customized Visualizations

News

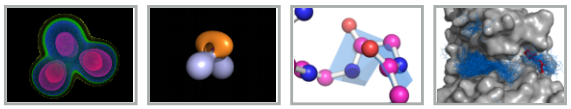
May 2, 2011: PyMOL v1.4.1 is released. Review the list of new features, and download the binaries.

March 18, 2011: PyMOL v1.4 is released. Review the new list of features, and download the binaries.

February 7, 2011: Volume visualization announced for upcoming PyMOL v1.4 release.

September 9, 2010: Congratulations to Michael Lerner for being awarded the Warren L. DeLano Memorial PyMOL Open-Source Fellowship for 2010-2011.

More news...



PyMOL is a **user-sponsored** molecular visualization system on an **open-source** foundation. Please support development of this open, effective, and affordable software by purchasing an incentive copy, which is pre-built and comes with maintenance and support.

SCHRÖDINGER

*A **USER-SPONSORED** molecular visualization system on an **OPEN-SOURCE** foundation*

PyMOL runs on

Support

View basic PyMOL installation instructions. More detailed instructions are available for incentive PyMOL subscribers.

View supported operating systems and hardware.

Professional Support

PyMOL Incentive User Area - Incentive PyMOL subscribers receive access to documentation, installation instructions, and narrated screencasts through the PyMOL Incentive User Area.

Schrödinger Technical Support - PyMOL incentive subscribers can contact us with technical questions using the online contact form or by e-mailing help@schrodinger.com.

Personalized Support and Training Sessions - PyMOL representatives are available for personalized support and training sessions at your site. Contact us for more information.

Community-Sponsored Resources

PyMOLWiki - Visit the **PyMOLWiki** for tutorials, scripts, answers to frequently asked questions, and more. A user-maintained knowledge base, the PyMOLWiki is full of helpful information.

PyMOL Users Mailing List - Users can also subscribe to the **PyMOL Users Mailing List** to participate in conversations with a dedicated base of knowledgeable users, and to stay up-to-date on the latest PyMOL tips and news. Thousands of archived conversations can also be browsed through this Mailing List.

The PyMOLWiki

Citing PyMOL, AxPyMOL, and JyMOL

Please see citation instructions for publications.

Links to External Sites

PyMOL Scripts, Programs, and Tips

Cameron Mura's PyMOL Page: includes helpful examples
eMovie: a storyboard-based tool for making molecular movies: by Eran Hodis, Gideon Schreiber, Kristian Rother and Joel L. Sussman
Robert Campbell's PyMOL Script Repository: a great collection of scripts for coloring and crystallography.
Kristian Rother's Scripts at Bioinformatik: including ScriptBox, a one-click script launching tool.
Gareth Stockwell's PyMOL Scripts at the EBI: density slider, rendering, movies, etc.
Movie HOWTO page (David Cooper)

Open-Source PyMOL Ports

Fink: Open-Source PyMOL for Mac OS X.
GNU/Darwin: Michael Love's port of Open-Source PyMOL and many other scientific software packages.
Debian:Open-Source PyMOL for Debian (Ubuntu).
Fedora Open-Source PyMOL for Fedora (RedHat).

Tools Enhanced with PyMOL: Open-Source and Open-Access

DynMap: a python- based program that generates maps of functional groups in a protein and visualizes them using dynamic parameters (Giacomo Bastianelli)
Sequence to Structure: display, manipulate, and interconnect RNA data from a sequence to structure (Fabrice Jossinet)
STRAP: sequence alignment tool (Christophe Gille)
NUCCYL: high-quality nucleic acid cartoons (Luca Jovine)

Tools Enhanced with PyMOL: Closed-Source and Restricted Access

Flexweb: Analysis of Flexibility in Biomolecules and Networks: provides PyMOL scripts for analyzing protein flexibility and dynamics.
PLANET: A Metaphorics Cabinet Server (Metaphorics LLC)

Image and Animation Galleries

The Yale Morph Server: using PyMOL for some animations.
AISMIG: An Interactive Server-side Molecular Image Generator

Software Packages that Interoperate with PyMOL

Maestro (Schrodinger): PyMOL reads Maestro files; a Maestro Plugin is available.
CCP4: a top crystallography package. PyMOL reads binary CCP4 maps.
PHENIX: another top crystallography package; PyMOL integrates with PHENIX in various ways.
CNS: another top crystallography package; PyMOL reads ASCII format CNS / X-PLOR maps.
Molscript (Avatar Software): although PyMOL superceeds Molscript in many ways, Molscript still generates uniquely beautiful geometries that can be fed into PyMOL (or Raster3D) for viewing and rendering.
MMTK: a Python-based Molecular Modeling Tool-Kit

SCHRÖDINGER.

Main Page

From PyMOLWiki

Welcome to the PyMOL Wiki!

The community-run support site for the PyMOL (https://web.archive.org/web/20110716184103/http://pymol.org/) molecular viewer.

For Educational downloads go to http://pymol.org/educational

Due to large amounts of spam, we've halted unsupervised account creation. Please contact jason . vertrees (@) gmail dot com to get a new account.

Quick Links

Tutorials	**Table of Contents**	**Commands**
Script Library	**Plugins**	**FAQ**
Gallery | Covers	**PyMOL Cheat Sheet** (*PDF*)	**GoogleSearch**

News & Updates

New Script	PluginDirectory: How to set up a personal plugin directory
New Script	Add focal blur to images FocalBlur.
New Script	Visualize VDW clashes with show bumps
New Plugin	Color by DSSP or Stride secondary structure assignment
New Script	There is a new script to calculate the Radius of gyration
New Command	Map_set Performs a given operation on a map: can create consensus volumes, for example.
New Script	ColorByDisplacement Do an (specified) alignment of residues between an open and closed form of a protein. Calculates the distance displacement between each residue and saves it as its b-factor. Then it color according to the b-factor. Quite neat feature for rotation axis in proteins.
New Script	DisplacementMap Calculates CA-CA distances between Open and Closed form of protein. Output best suggestions for site-directed mutagenesis for EPR/FRET experiments. Make distance matrix file, and output a gnuplot plot file for easy visualisation of interesting residues. Parses best suggestions back to pymol, for visual inspections.
New Script	Two new scripts: AAindex and Sidechaincenters
New Plugin	CAVER_2.0 update to the CAVER plugin.
New Script	AngleBetweenHelices calculates the angle between two helices.
Search	GoogleSearch widget fixed.
New Script	Spectrumany creates color gradients with arbitrary color sequences.
New Script	BbPlane will draw CGO planes across the backbone highlighting planarity of arrangement.
New Script	Center Of Mass has been re-written to calculate either the center-of-geometry or (mass-weighted) center-of-mass for a given selection and represents that selection as a pseudoatom (rather than a CGO sphere).

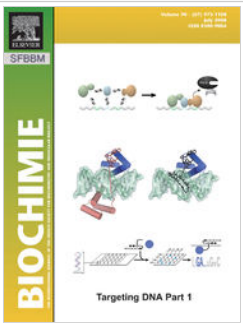
Did you know...

FilterByMol

== Overview ==

This script filters through all the PDBs in the parent dir (you can easily the the directory it scans). For each molecule, it saves just the ligands/heteroatoms (excluding the waters). This gives you a simple way to filter through a database of proteins looking only at their ligands.

This script, as noted below, works on the objects at the level of a molecule. While we can iterate over atom number (ID), residue ..→



A Random PyMOL-generated Cover. See Covers.

New Script	Jump is a tool for jumping from one frame to another when you have a movie, MD simulation, or multiple models loaded into PyMOL.
New Scripts	ResDe is a suite of programs designed to assist crystallographers in defining user defined hydrogen bond distance restraints, which can be helpful when refining low-resolution structures.
New Script	See BiologicalUnit, for a workaround to the buggy Symexp command or if you just want to learn more about symmetry expansion in PyMOL.
New Script	See Supercell, the new script for making XxYxZ supercells.
New Script	See Split_Object_Along_Axis, for a script that allows one to select a bond, and then generate 2 selections: one for the selection of all atoms that are on one side of this bond, and the other selection for the atoms on the other side of the bond.
New Script	See Consistent_View/_Map_Inspect, which is a toolkit for rapidly inspecting multiple maps and models.
Server updates	The underlying servers upon which the PyMOLWiki runs were upgraded over the weekend. We are now fully functional. A **deep** thanks to BitGnome (https://web.archive.org/web/20110716184103/http://www.bitgnome.net/) for donating time and hardware for the PyMOL project.
fetch_host setting	Fetch_Host has been added to allow users to download PDBs from their PDB server (pdb, pdb euro, or pdb japan) of choice.
Fetch	Fetch has been updated to also load electron density maps.
Schrodinger Buys PyMOL	Schrodinger has purchased PyMOL. Development, support and open-source fun to continue! Read about the sale (https://web.archive.org/web/20110716184103/http://www.schrodinger.com/news/47/) .
User Movie	One of our users has posted another interesting movie (https://web.archive.org/web/20110716184103/http://www.youtube.com/watch?v=eQWw6x3fLF0) , images from which were created with PyMOL.
New setting	surface_cavity_mode to change how PyMOL displays cavities.
Search fixed.	Thanks to a eagle-eyed user, our search has been fixed. Please let us know if you have any search-related problems.
New Command	Cache—stores information on structures, so we don't have to recompute them.
Warren	News about Warren DeLano's passing may be read on Warren's memorial page.
Setting	Fetch_Path—Sets the default path for the fetch command.
New Script	SelInside—Creates a custom selection of all atoms spatially inside some user-defined box.

navigation
- Main page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Category:Commands

This is a list of all commands for PyMol's interface and cmd module.

Please Read Before Adding!

For the sake of organization, please read Organizing PyMol Commands.

These all need to be reorganized based on the convention.

(previous 200) (next 200)

Subcategories

This category has the following 20 subcategories, out of 20 total.

A
- Category:Atoms Module

C
- Category:Color Module
- Category:Colour Module

D
- Category:Display Module
- Category:Distances Module

E
- Category:Editing Module

F
- Category:Fitting Module

H
- Category:Help Module

I
- Category:Imaging Module
- Category:Input Output Module

L
- Category:Language Module

M
- Category:Maps Module
- Category:Movies Module

R
- Category:Ray Tracing Module

S
- Category:Scripts Module
- Category:Selections Module
- Category:Settings Module
- Category:Stereo Module
- Category:Symmetry Module

V
- Category:View Module

Pages in category "Commands"

The following 180 pages are in this category, out of 185 total.

A
- Abort
- Alias
- Align
- Alter
- Alter State
- Attach

B
- Backward
- Bg Color
- Bond
- Button

C
- CBC
- Cache
- Capture
- Cd
- Center
- Clip
- Cls
- Color
- Commands
- Config Mouse
- Copy
- Count Atoms
- Count Frames
- Count States
- Create

G cont.
- Get Symmetry
- Get Title
- Get Type
- Get Version
- Get View
- Get chains
- Get session
- Group

H
- H Add
- H Fill
- Hide

I
- Id Atom
- Identify
- Index
- Indicate
- Intra Fit
- Intra Rms
- Intra Rms Cur
- Invert
- Isodot
- Isolevel
- Isomesh
- Isosurface
- Iterate
- Iterate State

P cont.
- Python

Q
- Quit

R
- Ramp New
- Ray
- Read Molstr
- Read Pdbstr
- Rebuild
- Recolor
- Redo
- Refresh
- Reinitialize
- Remove
- Remove Picked
- Rename
- Replace
- Reset
- Rewind
- Rms
- Rms Cur
- Rock
- Rotate
- Run

S
- Save
- Select

- Cycle Valence

D

- Deselect
- Disable
- Distance
- Draw
- Dss

E

- Edit
- Edit Keys
- Enable
- Ending
- Extend

F

- Feedback
- Fetch
- Fit
- Flag
- Fork
- Forward
- Fragment
- Frame
- Full Screen
- Fuse

G

- Get
- Get Angle
- Get Area
- Get Color Indices
- Get Dihedral
- Get Distance
- Get Extent
- Get Model
- Get Names
- Get object matrix
- Get Pdbstr
- Get Position

(previous 200) (next 200)

L

- Load
- Load Model
- Load Traj
- Ls

M

- Map Double
- Map Set Border
- Map Trim
- Map set
- Mappend
- Mask
- Matrix Copy
- Mclear
- Mdo
- Mdump
- Mem
- Meter Reset
- Middle
- Mmatrix
- Move
- Mplay
- Mpng
- Mset
- Mstop

O

- Order
- Orient
- Origin

P

- Pair Fit
- Png
- Protect
- Pseudoatom
- Push Undo
- Pwd

- Set
- Set Color
- Set Key
- Set Name
- Set Symmetry
- Set Title
- Set View
- Set bond
- Set title
- Show
- Show as
- Slice
- Slice new
- Smooth
- Sort
- Space
- Splash
- Split States
- Super
- Symexp
- Sync
- System

T

- Torsion
- Translate
- Turn

U

- Unbond
- Undo
- Unmask
- Unpick
- Unset
- Update

V

- View
- Viewport

Center

From PyMOLWiki

center translates the window, the clipping slab, and the origin to a point centered within the atom selection.

Contents

PYMOL API

```
cmd.center( string selection, int state = 0, int origin = 1 )
```

NOTES

- state = 0 (default) use all coordinate states
- state = -1 use only coordinates for the current state
- state > 0 use coordinates for a specific state

- origin = 1 (default) move the origin
- origin = 0 leave the origin unchanged

User Example

- Center around any given point

```
# define the point as x=1.0, y=2.0, z=3.0; replace [1.0, 2.0, 3.0] with your coordinate.
origin position=[1.0,2.0,3.0]
# center on it
center origin
```

SEE ALSO

Origin, Orient, Zoom

Retrieved from "http://www.pymolwiki.org/index.php/Center"
Categories: Commands | States

Color

From PyMOLWiki

color sets the color of an object or an atom selection to a predefined, named color. For an overview of predifined colors, see Color Values. For a script that enumerates all the colors see, List_Colors. If you want to define your own colors, see Set_Color.

Contents

USAGE

```
color color-name
color color-name, object-name
color color-name, (selection)
```

PYMOL API

```
cmd.color( string color, string selection )
```

EXAMPLES

Color all carbons yellow

```
color yellow, (name C*)
```

Color by Spectrum Example

Color by spectrum is in the GUI menu but did you realize that the spectrum is not limited to a simple rainbow?

```
spectrum count, x, object_name
```

x can can be anyone of the following: blue_green, green_white_magenta, red_cyan, blue_magenta, green_white_red, red_green, blue_red, green_white_yellow, red_white_blue, blue_white_green, green_yellow, red_white_cyan, blue_white_magenta, green_yellow_red, red_white_green, blue_white_red, magenta_blue, red_white_yellow, blue_white_yellow, magenta_cyan, red_yellow, blue_yellow, magenta_green, red_yellow_green, cbmr, magenta_white_blue, rmbc, cyan_magenta, magenta_white_cyan, yellow_blue, cyan_red, magenta_white_green, yellow_cyan, cyan_white_magenta, magenta_white_yellow, yellow_cyan_white, cyan_white_red, magenta_yellow, yellow_green, cyan_white_yellow, rainbow, yellow_magenta, cyan_yellow, rainbow2, yellow_red, gcbmry, rainbow2_rev, yellow_white_blue, green_blue, rainbow_cycle, yellow_white_green, green_magenta, rainbow_cycle_rev, yellow_white_magenta, green_red, rainbow_rev, yellow_white_red, green_white_blue, red_blue, yrmbcg

B-Factors

The command to color a molecule by B-Factors (B Factors) is:

```
cmd.spectrum("b", selection="SEL");
```

where **SEL** is a valid selection, for example, "protA and n. CA", for protein A's alpha carbons.

You can choose the spectrum you want with the command:

```
cmd.spectrum("b", 'rainbow', selection="SEL");
```

where **rainbow** is a valid selection from the list:

```
blue_green            green_white_magenta   red_cyan
blue_magenta          green_white_red       red_green
blue_red              green_white_yellow    red_white_blue
blue_white_green      green_yellow          red_white_cyan
blue_white_magenta    green_yellow_red      red_white_green
blue_white_red        magenta_blue          red_white_yellow
blue_white_yellow     magenta_cyan          red_yellow
blue_yellow           magenta_green         red_yellow_green
cbmr                  magenta_white_blue    rmbc
cyan_magenta          magenta_white_cyan    yellow_blue
cyan_red              magenta_white_green   yellow_cyan
cyan_white_magenta    magenta_white_yellow  yellow_cyan_white
cyan_white_red        magenta_yellow        yellow_green
cyan_white_yellow     rainbow               yellow_magenta
cyan_yellow           rainbow2              yellow_red
gcbmry                rainbow2_rev          yellow_white_blue
green_blue            rainbow_cycle         yellow_white_green
green_magenta         rainbow_cycle_rev     yellow_white_magenta
green_red             rainbow_rev           yellow_white_red
green_white_blue      red_blue              yrmbcg
```

Reassigning B-Factors and Coloring

It is commonplace to replace the B-Factor column of a protein with some other biochemical property at that residue, observed from some calculation or experiment. PyMOL can easily reassign the B-Factors and color them, too. The following example will load a protein, set ALL it's B Factors to "0", read in a list of properties for each alpha carbon in the proteins, assign those new values as the B-Factor values and color by the new values. This

example is possible because commands PyMOL does not recognize are passed to the Python interpreter --- a very powerful tool.

```
# load the protein
cmd.load("protA.pdb")

# open the file of new values (just 1 column of numbers, one for each alpha carbon)
inFile = open("newBFactors", 'r')

# create the global, stored array
stored.newB = []

# read the new B factors from file
for line in inFile.readlines(): stored.newB.append( float(line) )

# close the input file
inFile.close()

# clear out the old B Factors
alter protA, b=0.0

# update the B Factors with new properties
alter protA and n. CA, b=stored.newB.pop(0)

# color the protein based on the new B Factors of the alpha carbons
cmd.spectrum("b", "protA and n. CA")
```

If you want to save the file with the new B Factor values for each alpha carbon,

```
cmd.save("protA_newBFactors.pdb", "protA")
```

or similar is all you need.

A script (data2bfactor.py) that loads data into the B-factor (b) or occupancy (q) columns from an external file can be found in Robert Campbell's PyMOL script repository (http://pldserver1.biochem.queensu.ca/~rlc/work/pymol/)

Expanding to Surface

See Expand_To_Surface.

If you have run the above code and would like the colors to be shown in the Surface representation, too, then you need to do the following:

```
# Assumes alpha carbons colored from above.
create ca_obj, your-object-name and name ca
ramp_new ramp_obj, ca_obj, [0, 10], [-1, -1, 0]
set surface_color, ramp_obj, your-object-name
```

Thanks to Warren, for this one.

Getting Atom Colors

```
stored.list = []
iterate all, stored.list.append(color)
print stored.list
```

Or, you can label each atom by it's color code:

```
label all, color
```

Color States Individually

```
fetch 1nmr
set all_states

# the object has 20 states, so we can set separate line colors
# for each state as follows:
for a in range(1,21): cmd.set("line_color","auto","1nmr",a)
```

Or, we can do it differently,

```
# start over,
fetch 1nmr

# break apart the object by state
split_states 1nmr

# delete the original
dele 1nmr

# and color by object (carbons only)
util.color_objs("elem c")

# (all atoms)
util.color_objs("all")
```


Categories: Objects and Selections | Commands | States | Coloring

Create

From PyMOLWiki

create creates a new molecule object from a selection. It can also be used to create states in an existing object. **NOTE**: this command has not yet been throughly tested.

Contents

USAGE

```
create name, (selection) [,source_state [,target_state ] ]
create name = (selection) [,source_state [,target_state ] ]    # (DEPRECATED)
```

- name = object to create (or modify)
- selection = atoms to include in the new object
- source_state (default: 0 - copy all states)
- target_state (default: 0)

PYMOL API

```
cmd.create(string name, string selection, int state, int target_state,int discrete)
```

NOTES

If the source and target states are zero (default), all states will be copied. Otherwise, only the indicated states will be copied.

SEE ALSO

load, copy

User Comments/Examples

Retrieved from "http://www.pymolwiki.org/index.php/Create"

Categories: Commands | States | Editing Module

- This page was last modified on 17 November 2009, at 18:29.
-

Disable

From PyMOLWiki

disable disables display of an object and all currently visible representations.

Contents

USAGE

```
disable name
disable all
```

name is the name of an object or a named selection

PYMOL API

```
cmd.disable( string name )
```

EXAMPLE

```
disable my_object
```

SEE ALSO

Show, Hide, Enable

Enable

From PyMOLWiki

enable enable display of an object and all currently visible representations.

Contents

USAGE

```
enable name
enable all
```

name = object or selection name

PYMOL API

```
cmd.enable( string object-name )
```

EXAMPLE

```
enable my_object
```

SEE ALSO

Show, Hide, Disable

Retrieved from "http://www.pymolwiki.org/index.php/Enable"
Categories: Commands | View Module

Extend

From PyMOLWiki

Extend is an API-only function which binds a new external function as a command into the PyMOL scripting language. In other words, when you write a function and want PyMOL to recognize the new command, you **extend** the command into PyMOL. Once extended, the function name is recognized like other function names (example below). Typically, **extend** is the last line of a PyMOL script.

Contents

PYMOL API

```
cmd.extend(string name,function function)
```

PYTHON EXAMPLE

```
def foo(moo=2): print moo
cmd.extend('foo',foo)
```

The following would now work within PyMOL:

```
PyMOL>foo
2
PyMOL>foo 3
3
PyMOL>foo moo=5
5
PyMOL>foo ?

Usage: foo [ moo ]
```

NOTES

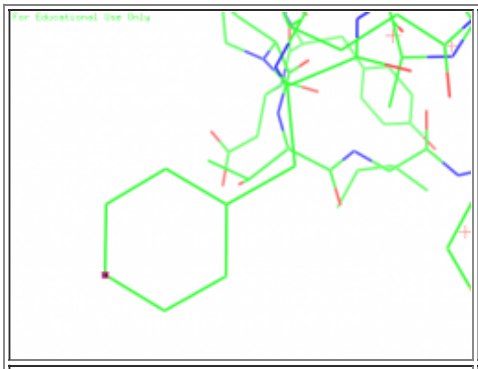
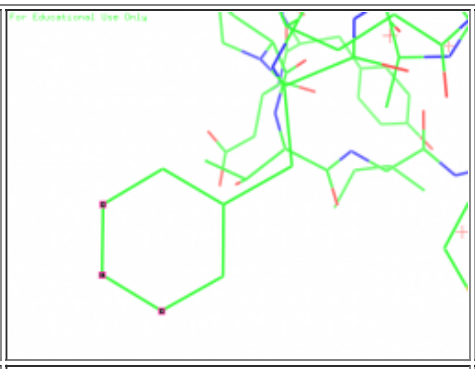
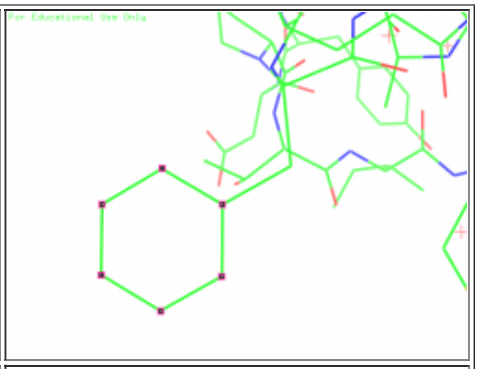
For security reasons, new PyMOL commands created using "extend" are not saved or restored in sessions.

SEE ALSO

- Alias
- Api
-

"extend" used in selections

"extend" can also be used in selection statements to grow a selection based on covalent bonds. This behavior is illustrated on residue 64 of PDB file 1KAO where we initially select one atom and then "extend" that selection by one and three covalent bonds.

		
select (resi 64 and n. cz)	select (resi 64 and n. cz) extend 1	select (resi 64 and n. cz) extend 3

Get Area

From PyMOLWiki

get_area calculates the surface area in square Angstroms of the selection given. Note that the accessibility is assessed in the context of the object(s) that the selection is part of. So, to get the surface areas of e.g. a component of a complex, you should make a new object containing a copy of just that component and calculate its area.

Contents

USAGE

```
get_area sele [,state[, load_b ]]
```

PYMOL API

```
cmd.get_area(string selection="(all)", load_b=0, state=0 )
```

Examples

Example 1 - starting with a complex in a single file

```
# load complex
# Haemoglobin in this example illustrates careful use of selection algebra
load 2HHB.pdb

# create objects for alpha1, beta1 and alpha1,beta1 pair of subunits
create alpha1, 2HHB and chain A
create beta1, 2HHB and chain B
create ab1, 2HHB and chain A+B

# get hydrogens onto everything (NOTE: must have valid valences on e.g. small organic molecules)
h_add

# make sure all atoms within an object occlude one another
flag ignore, none

# use solvent-accessible surface with high sampling density
```

```
set dot_solvent, 1
set dot_density, 3

# measure the components individually storing the results for later
alpha1_area=cmd.get_area("alpha1")
beta1_area=cmd.get_area("beta1")


# measure the alpha1,beta1 pair
ab1_area=cmd.get_area("ab1")

# now print results and do some maths to get the buried surface
print alpha1_area
print beta1_area
print ab1_area
print (alpha1_area + beta1_area) - ab1_area
```

Example 2 - starting with two components in separate files

```
# load components separately
load my_ligand.pdb
load my_target.pdb

# get hydrogens onto everything (NOTE: must have valid valences on the ligand...)
h_add

# make sure all atoms within an object occlude one another
flag ignore, none

# use solvent-accessible surface with high sampling density
set dot_solvent, 1
set dot_density, 3

# measure the components individually
ligand_area=cmd.get_area("my_ligand")
target_area=cmd.get_area("my_target")

# create the complex
create my_complex, my_ligand my_target

# measure the complex
complex_area=cmd.get_area("my_complex")

# now print results
print ligand_area
print target_area
print complex_area
print (ligand_area + target_area) - complex_area
```

Example 3 - using load_b to get surface area per atom

```
# example usage of load_b
# select some organic small molecule
select ligand, br. first organic
# get its area and load it into it's b-factor column
get_area ligand, load_b=1
# print out the b-factor/areas per atom
iterate ligand, print b
```

See Also

- For an example of **load_b** in use check out FindSurfaceResidues.
- Surface, most notably Surface#Calculating_a_partial_surface.

Categories: Commands | Biochemical Properties

Get Model

From PyMolWiki

Contents

DESCRIPTION

get_model returns a model object.

PYMOL API

```
cmd.get_model(string "selection" )
```

USAGE

```
cmd.get_model("chain A")
```

NOTES

It can be useful to loop through all the atoms of a selection (rather than using the iterate command)

```
atoms = cmd.get_model("chain A")
for at in atoms.atom:
    print "ATOM DEFINITION: "+at.chain+" "\
                             +at.resn+" "\
                             +str(at.resi)+" "\
                             +at.name+" "\
                             +str(at.index)+" "\
                             +at.b+" "\
                             +str(at.coords[0])+" "\
                             +str(at.coords[1])+" "\
                             +str(at.coords[2])
```

SEE ALSO

Retrieved from "http://www.pymolwiki.org/index.php/Get_Model"

Category: Commands

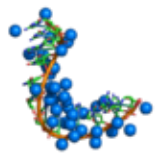
- This page was last modified 18:41, 26 September 2007.

-

Hide

From PyMOLWiki

hide conceals atom and bond representations for a certain selection or other graphical objects like distances.

	
Some normal scene, notice the waters shown as spheres	*hide spheres* issues and the spheres are now hidden.

The available representations are:

- lines
- spheres
- mesh
- ribbon
- cartoon
- sticks
- dots
- surface
- labels
- nonbonded
- nb_spheres

Contents

USAGE

```
hide representation [,object]
hide representation [,(selection)]
```

```
hide (selection)
```

PYMOL API

```
cmd.hide( string representation="", string selection="")
```

EXAMPLES

```
# hides all lines
hide lines,all

# hides all ribbons
hide ribbon

# hides all distances
hide dashes

# hides sticks in protA and all residues that aren't in the range of 40-65
hide sticks, protA and not i. 40-65
```

SEE ALSO

Show, Enable, Disable

Retrieved from "http://www.pymolwiki.org/index.php/Hide"
Categories: Commands | View Module

Select

From PyMOLWiki

select creates a named selection from an atom selection. Selections are one of the most powerful aspects of PyMOL and learning to use selections well is paramount to quickly achieving your goals in PyMOL.

Contents

USAGE

```
select (selection)
select name, (selection)
select name = (selection)            # (DEPRECATED)
```

PYMOL API

```
cmd.select(string name, string selection)
```

EXAMPLES

```
select near , (ll expand 8)
select near , (ll expand 8)
select bb, (name ca,n,c,o )
```

```
cmd.select("%s_%s"%(prefix,stretch), "none")
```

NOTES

Type **help selections** for more information about selections.

SEE ALSO

- Selection Algebra
- Property Selectors

Retrieved from "http://www.pymolwiki.org/index.php/Select"

Categories: Commands | Selecting

- This page was last modified on 17 November 2009, at 18:48.
-

Set

From PyMOLWiki

set is one of the most utilized commands. PyMOL representations, states, options, etc. are changed with **set**. Briefly, **set** changes one of the PyMOL state variables. Currently there are over *600* PyMOL settings!

Contents

USAGE

```
# set '''name''' to '''value'''
set name, [,value [,object-or-selection [,state ]]]

# alternative way to do the above.
set name = value  # (DEPRECATED)
```

PYMOL API

```
cmd.set ( string name,
    string value=1,
    string selection='',
    int state=0,
    int updates=1,
    quiet=1)
```

EXAMPLES

```
set surface_color, red

set ray_trace_mode, 3

set ribbon_width, 4

# set the label size to 2Ang.
set label_size, -2
```

NOTES

The default behavior (with a blank selection) changes the global settings database. If the selection is 'all', then the settings database in all individual objects will be changed. Likewise, for a given object, if state is zero, then the object database will be modified. Otherwise, the settings database for the indicated state within the object will be modified.

If a selection is provided, then all objects in the selection will be affected.

SEE ALSO

Get

Categories: Commands | Settings | States

Set Name

From PyMOLWiki

set_name can be used to change the name of an object or selection.

Not only can you simply rename an object or selection, but this command is also a powerful tool for those who deal with multiple structures in one file --- say a collection of NMR models. The user can execute the Split_States command and then rename the molecule of choice in the state of choice. For example, if one loads an NMR structure (with, say, 20 states) and aligns it to another structure, the balance of the alignment will (most likely) be off due to the weighting of the 19 other structures you probably don't see. To overcome this problem, one simply executes Split_States and then renames one of the states and then aligns that newly renamed object.

USAGE

```
set_name old_name, new_name
```

PYMOL API

```
cmd.set_name(string old_name, string new_name)
```

User Comments/Examples

```
cmd.set_name("example", "nicename")
```

Retrieved from "http://www.pymolwiki.org/index.php/Set_Name"
Categories: Commands | States

navigation
- Main page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Show

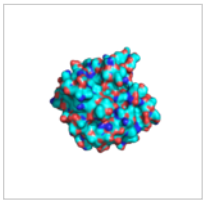
Show displays atom and bond representations for certain selections.

Contents [hide]

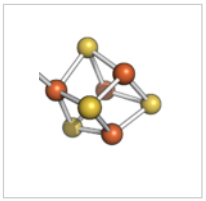
Details

The **Show** command, is one of the most often used commands in PyMOL. For example, you can *show* certain atoms as *Lines* or *Sticks* or *Cartoons* or any of the following representations:

- lines
- spheres
- mesh
- ribbon
- cartoon
- sticks
- dots
- surface
- label
- extent
- nonbonded
- nb_spheres
- slice
- cell



Example of a shown surface.



Ball and sticks shown.

A car show

USAGE

```
show
show reprentation [,object]
show reprentation [,(selection)]
show (selection)
```

PYMOL API

```
cmd.show( string representation="", string selection="" )
```

EXAMPLES

Example

```
# show the backbone using lines.
show lines,(name ca or name c or name n)
```

Example

```
# show the ribbon representation for all objects
show ribbon
```

Example

```
# show all hetero atoms as spheres
show spheres, het
```

Example

NOTES

selection can be an object name.

show alone will turn on lines and nonbonded for all bonds.

show cell will show the triclinic unit cell, provided it's described in the PDB. For cell packing, you need a script. See The Script Library or Robert Campbell's Site

SEE ALSO

Hide, Enable, Disable, Lines, Cartoon, Spheres, Mesh, Ribbon, Sticks, Dots, Surface, Label, Extent, Slice, Cell, Select, Show_as

Categories: Commands | View Module

Cartoon

From PyMOLWiki

Contents

Cartoon Command

DESCRIPTION

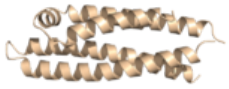
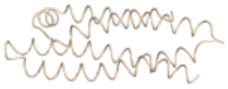
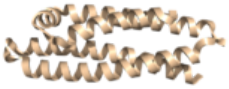
cartoon changes the default cartoon for a set of atoms.

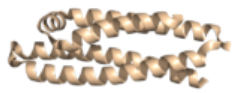
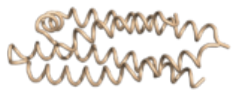
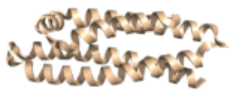
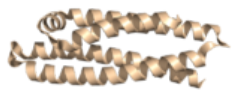
USAGE

```
cartoon type, (selection)
```

type:

1. skip
2. automatic
3. loop
4. rectangle
5. oval
6. tube
7. arrow
8. dumbbell

cartoon_skip; yes, blank.	 cartoon_automatic	 cartoon_loop	 cartoon_rectangle

			
cartoon_oval	cartoon_tube	cartoon_arrow	cartoon_dumbbell

PYMOL API

```
cmd.cartoon(string type, string selection )
```

EXAMPLES

```
cartoon rectangle,(chain A)
cartoon skip,(resi 145:156)
```

NOTES

the "automatic" mode utilizes ribbons according to the information in the PDB HELIX and SHEET records.

Adjusting width of cartoon

Try varying the following.

For β-strands:

```
cartoon_rect_length
cartoon_rect_width
```

For α-helices:

```
cartoon_oval_length
cartoon_oval_width
```

For loops:

```
cartoon_loop_radius
```

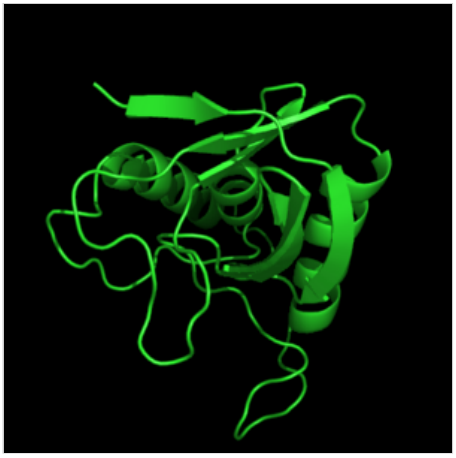
For nucleic acid backbones which resemble 'loops'; however, are not classified as such by Pymol (see more about nucleic acid representation settings at bottom of page):

```
cartoon_tube_radius,0.8
```

For "fancy" α-helices:

```
cartoon_dumbbell_length
cartoon_dumbbell_width
cartoon_dumbbell_radius  (radius of cylinder at edge of helix ribbon)
```

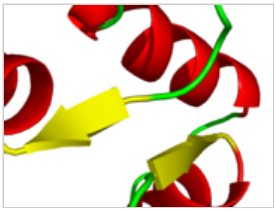
In each case "length" refers to what some might call the width and "width" refers to what some might call the thickness.



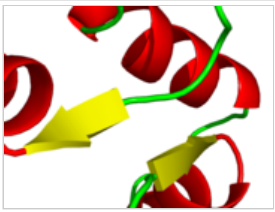
Cartoon Representation Example

Forcing Exact Boundaries in Coloring Secondary Structures

To force PyMol to respect secondary structural elements color-wise (PyMol smooths out colors near color chagnes for a prettier image) use the following PyMol command: `set cartoon_discrete_colors, on`



Discrete Coloring Off



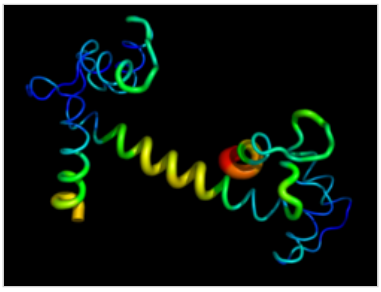
Discrete Coloring On

Sausage Representation

The familiar sausage representation in PyMol is called, "putty". To enable the putty/sausage view simply do,

```
show cartoon
cartoon putty
unset cartoon_smooth_loops
unset cartoon_flat_sheets
```

As of v 0.98 or so, there's a Putty option. Use this.

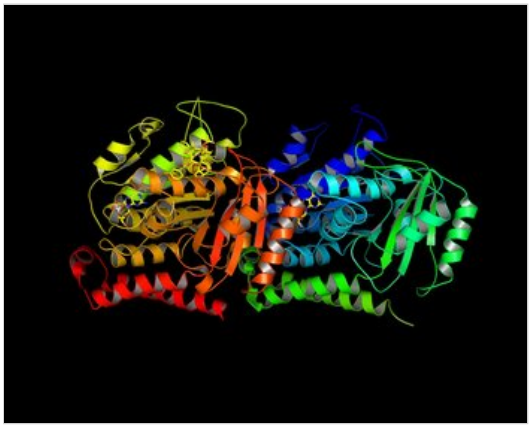


Example of B-factor Putty

Black and White Representation

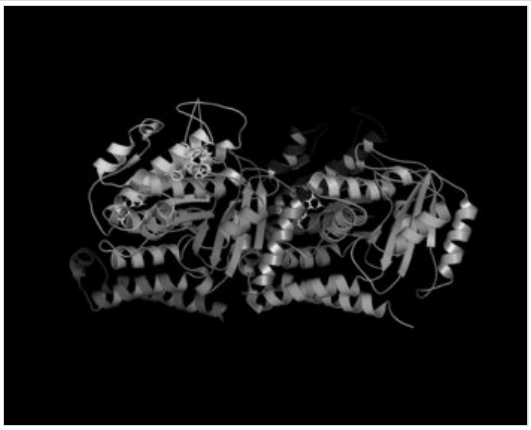
UPDATE: This method is essentially obseleted by the new setting **set ray_trace_mode,2**. More information on this at Ray. For those who want a nifty black and white representation of their protein try the following:

1. Ray trace your protein of choice in a cartoon representation use a BLACK background
2. Save the image
3. Load the image in GIMP.



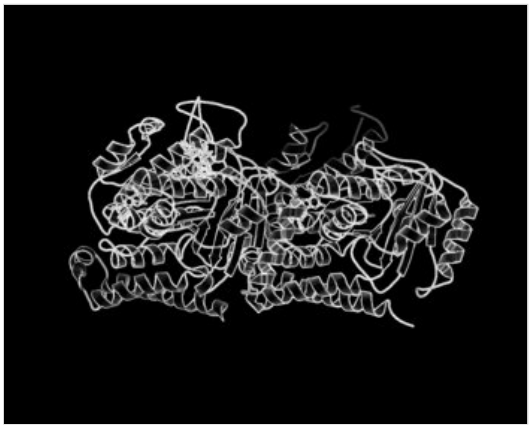
Black BG Ribbon

4. Desaturate or Grayscale the image.



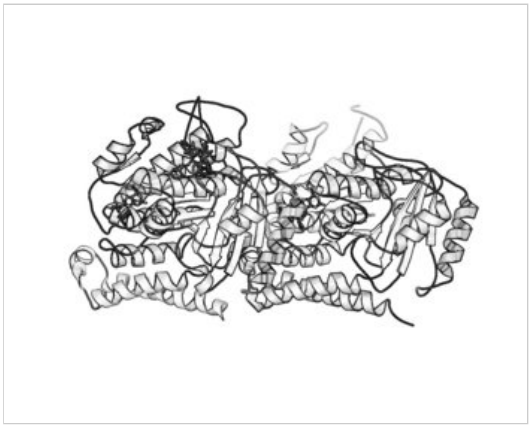
Grayscale

5. Run the filter: Filter->Edge-Detect->Edge.



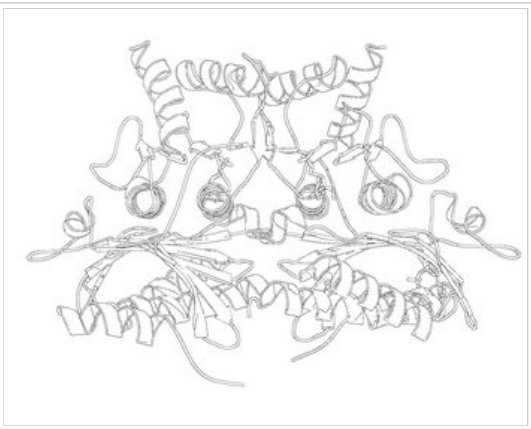
Edge Detect

6. Select: Layers->Color->Invert.



Invert Color

7. Different methods of edge detection will give you different results. In the last example, I used Laplace Edge-Detect, then painted an all white layer beneath the current layer to achieve the results.



Comments

I'm sure there are other ways to do this. If you want to include it in a publication make sure you ray traced it large enough. For that, see Creating Publication Quality Images.

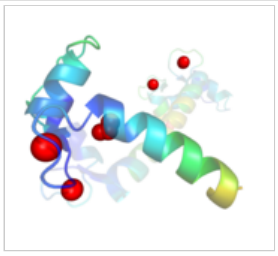
CA (Alpha Carbon) Trace

If you have a structure with just a alpha carbon trace, you can get a cartoon by

```
set cartoon_trace,1
show cartoon
```

If your structure is more than just the CA backbone, the cartoon representation will look incorrect, so use it just with CA trace.

Various Transparency Levels

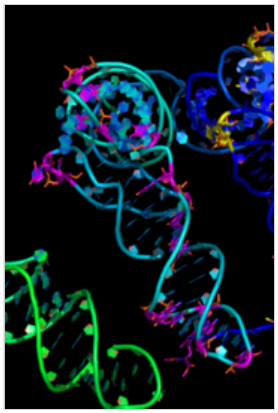


Example of Cartoon Multi-level Transparency. The near cartoon has transparency setting **0.2**, the segment in the BG **0.5**.

One can make different cartoon selections have different transparency values, in PyMol. The trick here is to use "create" instead of "select". Create makes new objects that can have independent settings.

```
load mol_obj.pdb

# transfer a piece of the molecule into a new object

create new_obj, chain A
remove mol_obj in new_obj

# adjust trasparency for the new object

set cartoon_transparency, 0.5, new_obj
```

Nucleic Acid Representation



Showing Nucleic Acids

To control radius of nucleic acids default backbone cartoon:

```
set cartoon_tube_radius,0.8   #0.5 seems close to the default setting
```

To show nucleic acids in a nicer format do:

```
set cartoon_ring_mode,1
 show cartoon
```

Other Nucleic Acids & Cartoon Settings

Here are some things to try:

```
set cartoon_ring_mode, 1   # (or 2 or 3)
set cartoon_ring_finder, 1 # (or 2 or 3 or 4)
set cartoon_nucleic_acid_mode, 0   # (or 1 or 2 or 3 or 4)

set cartoon_side_chain_helper
rebuild

set cartoon_ring_transparency, 0.5

set cartoon_ladder_mode, 0 # or 1

set cartoon_ladder_color, color-name
set cartoon_nucleic_acid_color, color-name

cartoon oval
set cartoon_oval_width, 0.8

cartoon rect

cartoon dumbbell
set cartoon_dumbbell_width, 0.4
set cartoon_dumbbell_radius, 0.4
```

Overview of nucleic acid cartoons

Examples of nucleic acid cartoons

See Also

Displaying_Biochemical_Properties

Retrieved from "http://www.pymolwiki.org/index.php/Cartoon"
Categories: Representations | Nucleic Acids | Putty

Lines

From PyMOLWiki

Lines is name of the basic representation for atoms and bonds in PyMOL. **Lines** is a very simple representation, where each atom bond is displayed as a single colored line, and each atom is displayed as the intersection of any two or more non-terminal bonds.

Contents

Usage

```
# show everything as lines
show lines

# only show residues 50-80 as lines
show lines, i.50-80
```

Examples

Example: Displaying dashed lines between two atoms

The following commands will create a dashed line between two atoms.

```
# first, create two named selections
select a, ///A/501/02
select b, ///B/229/N
# calculate & show the distance from selection a to selection b.
distance d, a, b
# hide just the distance labels; the
# dashed bars should still be shown
hide labels, d
```

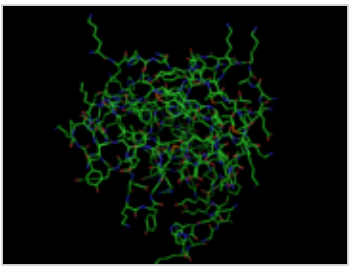
Technically, the object *d* is a labelled distance, only the label is hidden. When ray-tracing the image, the dashes come out a bit fat. You can slim them with

```
set dash_gap, 0.5
set dash_radius, 0.1
```

before the 'ray' command.

See Also

Please read about other representations in the **Representation Category**.
Measure_Distance
Distance

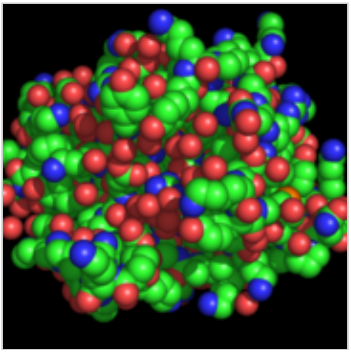


Lines Representation Example

Spheres

From PyMOLWiki

Contents



Normal Sphere Representation Example

Representation

To enable the **spheres** representation do the following for any selection SEL,

```
show spheres, SEL
```

Adjusting Sphere Size

```
alter selection, vdw=number
```

Examples

Shrink the size of all Iron atoms:

```
alter elem fe, vdw=1.0
rebuild
```

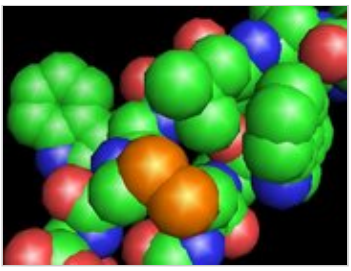
Dramatically enlarge all spheres in an object

```
alter object, vdw=4.0
rebuild
```

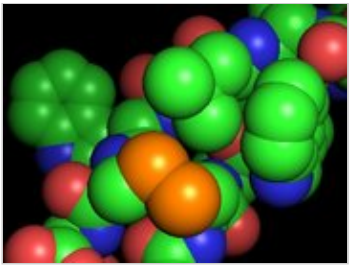
OpenGL Shaders & Spheres

Newer OpenGL supported cards (like the NVidia 5950 Ultra, or the 6800 GT Ultra) support **Shaders**. Shaders are best used for massive numbers of molecules that are to be represented as spheres. Typical ranges will now include 500 000 to 3 000 000 atoms! Take a look, the following example is of a viral nucleocapsid: 261 240 atoms! Performance and visual quality -- without rendering -- are far improved.

Comparing Shaders and No-Shaders



Normal spheres: no shaders



(sphere_mode=5) Shader Spheres

To turn on Sphere Shaders use

```
set sphere_mode, 5
as spheres, SEL
```

where **SEL** is the name of your selection. Getting normal sphere mode back is easy:

```
set sphere_mode, 4
as spheres, SEL
```

Enabling Shaders

If the above doesn't work, then you may need to rebuild PyMol so that it builds the shaders source code. To do this, you simply need to edit the **setup.py** file before you build PyMol.

Find the appropriate line in your **setup.py** file depending on your system. The relevant lines are, first for Windows,

```
if sys.platform=='win32':
```

and for Windows using Cygwin

```
elif sys.platform=='cygwin':
```

and finally for *nix or other systems as the following

```
else:
```

Under this code, find the

```
def_macros=[("_PYMOL_MODULE",None),
            ("_PYMOL_INLINE",None),
#           ("_PYMOL_NUMPY",None),
            ("_HAVE_LIBPNG",None)]
```

and make it

```
def_macros=[("_PYMOL_MODULE",None),
              ("_PYMOL_INLINE",None),
#             ("_PYMOL_NUMPY",None),
              ("_HAVE_LIBPNG",None),
              ("_PYMOL_OPENGL_SHADERS",None)]
```

I just added the

```
("_PYMOL_OPENGL_SHADERS",None)]
```

line.

See the Installation Page to find out how to build PyMol.

Retrieved from "http://www.pymolwiki.org/index.php/Spheres"
Categories: Representations | Performance

Sticks

From PyMOLWiki

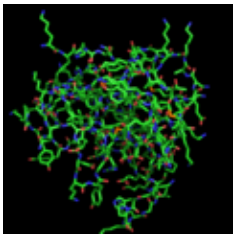
Contents

Overview

A simple PyMol representation where bonds are drawn as sticks. Use

```
# using the show command, for some SELECTION
show sticks, SELECTION

# using the as command
as sticks, SELECTION
```

where SELECTION is a valid selection or previously defined selection name.



Example Sticks Representation

Settings

- stick_ball
- stick_nub
- stick_transparency
- stick_ball_ratio

- stick_overlap
- stick_valence_scale
- stick_color
- stick_quality
- stick_fixed_radius
- stick_radius
- set_bond

Example Settings

Color Sticks

Use set_bond to set stick-bond settings, like color:

```
set_bond 1foo and i. XYZ, color red
```

Sticks Radius (Sticks Weight)

To change the radius for sticks, enter the following:

```
set stick_radius, VALUE
```

where

```
0.0<=VALUE<=1.0
```

- **1.0** is 100% or full radius
- **0.0** is 0% or invisible -- so use at least 0.1 or greater
- The default value is: ~0.3

Sticks Transparency

To enable transparency for sticks, enter the following:

```
set stick_transparency, VALUE
```

where *0.0<=VALUE<=1.0*

- **1.0** is 100% transparent -- so invisible
- **0.0** is 0% transparent -- so opaque

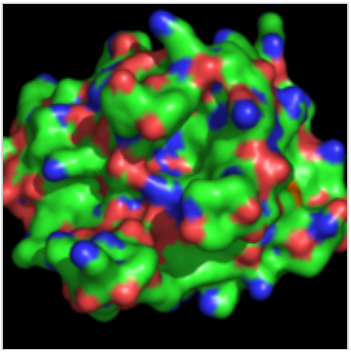
```
set stick_transparency, 0.45
```

Retrieved from "http://www.pymolwiki.org/index.php/Sticks"
Categories: Representations | Sticks

Surface

From PyMOLWiki

The surface representation of a protein, in PyMol, shows the "Connolly" surface (https://web.archive.org/web/20110607231738/http://en.wikipedia.org/wiki/Connolly_surface) or the surface that would be traced out by the **surfaces** of waters in contact with the protein at all possible positions.



Surface Representation Example

Contents

Enabling

To enable the surface representation do

```
show surface, SEL
```

for any proper selection SEL.

Settings

- cavity_cull
- surface_best
- surface_negative_color
- surface_carve_cutoff
- surface_negative_visible

- surface_carve_normal_cutoff
- surface_normal
- surface_carve_selection
- surface_optimize_subsets
- surface_carve_state
- surface_poor
- surface_circumscribe
- surface_proximity
- surface_clear_cutoff
- surface_quality
- surface_clear_selection
- surface_ramp_above_mode
- surface_clear_state
- surface_solvent
- surface_color
- surface_trim_cutoff
- surface_debug
- surface_trim_factor
- surface_miserable
- surface_type
- surface_mode

Examples

Transparency

To adjust the transparency of surfaces try:

```
set transparency, 0.5
```

Where 1.0 will be an invisible and 0.0 a completely solid surface.

Quality

To smooth your surface representation try:

```
set surface_quality, 1
```

or higher if you wish, though it will take longer and might look odd.

Probe Radius

To change the probe radius other than default 1.4 Å, you need to change the solvent radius, say, 1.6 Å:

```
set solvent_radius, 1.6
```

If the surface does not change correspondingly, use:

```
rebuild
```

Tips

Exporting Surface/Mesh Coordinates to File

PyMOL can export its coordinates as WRL wireframe model files for VRML input.

Older PyMOL Versions

```
# export the coordinates to povray
open("surface.inc","w").write(cmd.get_povray()[1])
```

Newer PyMOL Versions

```
# export the coordinates to .wrl file
save myscene.wrl
```

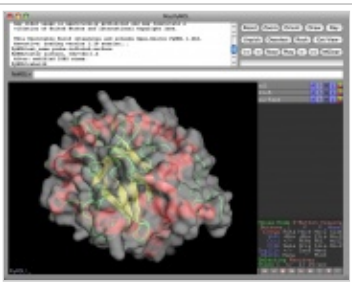
or

```
# export the coordinates to .obj file
save myscene.obj
```

Representation-independent Color Control

To color the surface representation a different color than the underlying cartoon or ligand representations, simply duplicate the object, show only the surface in the duplicate, and show only the cartoon and/or ligands in the original object.

Or use the surface_color setting that is available.



Representation-independent Color Control Example

Displaying a protein as surface with a ligand as sticks

An easy way to do this is to create separate objects for each type of display.

1 Load your protein

2 Select the ligand

3 Create a separate object for the ligand

4 Remove ligand atoms from the protein

5 Display both objects separately

Example:

```
load prot.ent,protein
select ligand,resn FAD
create lig_sticks,ligand
```

```
remove ligand
show sticks,lig_sticks
show surface,protein
```

Even easier is to:

1 Load the protein

2 S (Show) > organic > stick

3 S (Show) > surface

Calculating a partial surface

There is, until now, an undocumented way to calculate a surface for only a part of an object without creating a new one:

```
flag ignore, not A/49-63/, set
delete indicate
show surface
```

If the surface was already computed, then you'll also need to issue the command:

```
rebuild
```

See Get_Area for more information on surface area calculations.

Displaying surface inside a molecule

As far as I can tell, setting ambient to zero alone doesn't quite do the job, since some triangles still get lit by the light source. The best combination I can find is:

```
set ambient=0
set direct=0.7
set reflect=0.0
set backface_cull=0
```

Which gives no shadows and only a few artifacts.

As an alternative, you might just consider showing the inside of the surface directly...that will create less visual artifacts, and so long as ambient and direct are sufficiently low, it will look reasonable in "ray".

```
util.ray_shadows("heavy")
set two_sided_lighting=1
set backface_cull=0
```

Creating a Closed Surface

 Example OPEN Surface	 Example CLOSED Surface

To create what I'll call a **closed surface** (see images), you need to first make your atom selections, then create a new object for that selection then show the surface for that object. Here's an example.

```
sel A, id 1-100
 create B, A
 show surface, B
```

Huge Surfaces

If your protein or complex is too large to render (ray runs out of RAM, for example) then check out the tip for huge surfaces.

Performance

To optimize performance and responsiveness, PyMOL tends to defer compute-intensive tasks until their results are actually needed. Thus,

```
cmd.show("surface")
```

doesn't actually show a surface, it only sets the surface visibility flag on the atoms present (for future reference). An actual surface won't be computed until PyMOL is asked to refresh or render the display. When running a script, you can force an update by calling:

```
cmd.refresh()
```

after cmd.show.

Categories: Representations | VRML | Surfaces | Performance

navigation
- Main page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Category:Plugins

Overview

A list of available plugins written by the community, for the community.

Need Help?

The Plugins_Tutorial is a great place to start.

Subcategories

This category has the following 2 subcategories, out of 2 total.

P

- Category:Plugins PDB

r

- Category:Plugins rtools

Pages in category "Plugins"

The following 37 pages are in this category, out of 37 total.

A

- AAindex
- APBS
- AngleBetweenHelices
- Annocryst
- Autodock Plugin

C

- CASTp
- Caver
- Colorama

D

- DSSP
- DivScore

E

- EZ-Viz
- EMovie

H

- Helicity check

M

- MSMS
- Mole

N

- NsSNP Loader

O

- DYNMAP

P

- PDB Index Search
- PDB Loader Service
- PluginDirectory
- Plugins
- Plugins Tutorial
- PocketPicker
- Polarpairs
- ProMOL
- Proxy Config

P cont.

- PyDet

R

- Rendering Plugin
- Resicolor plugin

S

- Sidechaincenters
- Spectrumany
- Stride
- SuperSym
- SuperSymSource

T

- TCONCOORD

V

- VASCo
- Voronoia

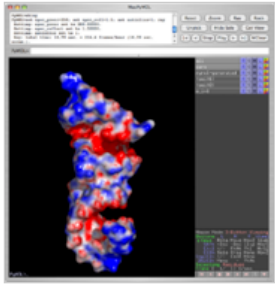
APBS

From PyMOLWiki

Contents

Introduction

APBS (https://web.archive.org/web/20110728103647/http://apbs.sourceforge.net/) , the Adaptive Poisson-Boltzmann Solver, is a freely (https://web.archive.org/web/20110728103647/http://www.oreilly.com/openbook/freedom/) available macromolecular electrostatics calculation program released under the GPL (https://web.archive.org/web/20110728103647/http://www.gnu.org/copyleft/gpl.html) . It is a cost-effective but uncompromised alternative to GRASP (https://web.archive.org/web/20110728103647/http://trantor.bioc.columbia.edu/grasp/) , and it can be used within PyMOL. PyMOL can display the results of the calculations as an electrostatic potential molecular surface.



APBS-generated electrostatic surface displayed in PyMOL

APBS Plugin with New Features

Pre-release version

There is often a more current pre-release version available on my user page. If you're experiencing bugs, please test the pre-release version to see if they've already been fixed. Thanks! --michael 19:43, 29 October 2010 (UTC)

PyMol currently supports the **APBS plugin** written by Michael Lerner (https://web.archive.org/web/20110728103647/http://pymolwiki.org/index.php/User:Mglerner) . This plugin makes it possible to run APBS from within PyMOL, and then display the results as a color-coded electrostatic surface (units K_bT / e_c) in the molecular display window (as with the image to the right). This wiki page has updated instructions on how to download, install and use the plugin.

Nucleic acids may prove problematic for the apbs plugin. If so, use the pdb2pqr (https://web.archive.org/web/20110728103647/http://pdb2pqr.sourceforge.net/) command-line tool to create a pqr file manually, instead of using the plugin to generate it. Then direct the APBS GUI on the main menu (https://web.archive.org/web/20110728103647/http://www-personal.umich.edu/~mlerner/PyMOL/images/main.png) to read the pqr file you **externally generated.**

There is a new version of the PyMOL-APBS plugin and it's now ready for pre-release. There are several big advantages of the new version:

- It's been tested modern OS X, Windows and Linux systems and fixes several long-standing bugs.
- It allows you to call through to PDB2PQR directly.
- It allows you to show the electric field lines.

- It has two visualization panels to aid in showing multiple potential surfaces at once.
- It defaults to using PDB2PQR for PQR generation and APBS's psize.py for grid sizing/spacing.
- It has also an increased default maximum allowed memory since typical users have bigger and faster computers these days.
- It calls on the correct paths and binaries for multi-threaded APBS (apbs-mpi-openmpi).

The main reason it is not included in the latest PyMOL release is to receive bug reports. Once it's shown to be stable, it'll be included in the next PyMOL release. See Michael Lerner (https://web.archive.org/web/20110728103647/http://pymolwiki.org/index.php/User:Mglerner) 's page.

How to get it

There are two ways to get the new plugin

- If you have subversion installed, you can always get the latest version via

```
svn co http://pymolapbsplugin.svn.sourceforge.net/viewvc/pymolapbsplugin/trunk/src/apbsplugin.py
```

- You can download it from http://pymolapbsplugin.svn.sourceforge.net/viewvc/pymolapbsplugin/trunk/src/apbsplugin.py

That should give you a file called apbsplugin.py

Once you have the plugin, you can install it via PyMOL's plugin installer: Plugin --> Manage Plugins --> Install

You may have to run PyMOL with sudo privileges.

Note that the plugin will be installed as "APBS Tools2.1" so that you can continue to use your old version.

Further details, as well as screen shots, are given elsewhere in this wiki (https://web.archive.org/web/20110728103647/http://www.pymolwiki.org/index.php/MAC_Install#Install_APBS_and_friends_with_fink) .

Required Dependencies

APBS (https://web.archive.org/web/20110728103647/http://apbs.sourceforge.net/) and its dependencies like pdb2pqr (https://web.archive.org/web/20110728103647/http://pdb2pqr.sourceforge.net/) and maloc (https://web.archive.org/web/20110728103647/http://scicomp.ucsd.edu/~mholst/codes/maloc/) are freely (https://web.archive.org/web/20110728103647/http://www.oreilly.com/openbook/freedom/) available under the GPL (https://web.archive.org/web/20110728103647/http://www.gnu.org/copyleft/gpl.html) . The author of the software however asks that users register (https://web.archive.org/web/20110728103647/http://agave.wustl.edu/apbs/download/) with him to aid him in obtaining grant funding.

Installing the Dependencies on OS X

1. First, register (https://web.archive.org/web/20110728103647/http://agave.wustl.edu/apbs/download/) your use of the software. This will keep everyone happy.
2. Second, if you don't already have the fink package management system (https://web.archive.org/web/20110728103647/http://fink.sourceforge.net/) , now is a good time to get it. Here is a quick-start set of instructions (https://web.archive.org/web/20110728103647/http://xanana.ucsc.edu/~wgscott/xtal/wiki/index.php/Quick_Start) for getting X-windows, compilers, and fink all installed.
3. Once you are up and going, activate the unstable branch in fink (https://web.archive.org/web/20110728103647/http://xanana.ucsc.edu/~wgscott/xtal/wiki/index.php/How_to_Activate_the_Unstable_Branch) , and then issue the commands

```
fink self-update
fink install apbs
```

or if you want to use the multi-processor version, issue

```
fink self-update
fink install apbs-mpi-openmpi
```

Then install the X-windows based version of pymol using the command

```
fink install pymol-py25
```

It is recommended to install the latest pdb2pqr as well as the APBS plugin makes use of it

```
fink install pdb2pqr
```

Note that the fink version of PyMOL as of 1.3-4 does not have the latest version of the APBS plugin. Make sure you get the new version!

Installing the Dependencies on Linux

From Scratch

Note that this tutorial assumes you're using the bash shell and have root privileges

1. Obtain APBS and MALOC from...
   APBS = http://apbs.sourceforge.net (currently 0.4)
   MALOC = http://www.fetk.org/codes/maloc/index.html#download (currently 0.1-2)
2. Set up some environment variables & directories (temporary for building)

```
$ export FETK_SRC=/<building directory>/temp_apbs
$ export FETK_PREFIX=/usr/local/apbs-0.4.0  (or wherever you want it to live)
$ export FETK_INCLUDE=${FETK_PREFIX}|>/include
$ export FETK_LIBRARY=${FETK_PREFIX}|>/lib
$ mkdir -p ${FETK_SRC}|> ${FETK_INCLUDE}|> ${FETK_LIBRARY}|>
```

3. Unpack the source packages

```
$ cd ${FETK_SRC}|>
$ gzip -dc maloc-0.1-2.tar.gz | tar xvf -
$ gzip -dc apbs-0.4.0.tar.gz | tar xvf -
```

4. Compile MALOC

```
$ cd ${FETK_SRC}|>/maloc
$ ./configure --prefix=${FETK_PREFIX}|>
```

   If everything went well, then

```
$ make; make install
```

5. Go get a coffee. Compilation/installation takes about 15 minutes on a 3GHz computer with 1GB of RAM.
6. Now on to compiling APBS itself

```
$ cd ${FETK_SRC}|>/apbs-0.4.0
$ ./configure --prefix=${FETK_PREFIX}|>
```

   If all goes well:

```
$ make all; make install
```

7. No time for coffee. Takes about 5 minutes on that fast computer.
8. There will now be an APBS binary at

```
/usr/local/apbs-0.4.0/bin/i686-intel-linux/apbs
```

9. Make appropriate links

```
$ ln -s /usr/local/apbs-0.4.0/bin/i686-intel-linux/apbs /usr/local/bin/apbs
```

10. Get rid of /<building directory dir>/temp_apbs
11. Open PyMOL and make sure that the APBS plugin points to /usr/local/bin/apbs
12. Rock and or Roll.

Pre-Packaged

RPMs

A variety of RPMs are available from the APBS downloads website (https://web.archive.org/web/20110728103647/http://sourceforge.net/project/showfiles.php?

group_id=148472&package_id=163734&release_id=378273) . Again, please register (https://web.archive.org/web/20110728103647/http://agave.wustl.edu/apbs/download/) your use of the software if you have not yet done so.

Debian packages

For ubuntu and other debian linux distributions, probably the simplest thing is to download a promising looking rpm, convert it with the program alien (https://web.archive.org/web/20110728103647/http://kitenet.net/programs/alien/) , and then install the newly generated debian package (https://web.archive.org/web/20110728103647/http://xanana.ucsc.edu/linux) with the command

```
sudo dpkg -i apbs*.deb
```

Gentoo

You have to install apbs and pdb2pqr. Both are masked via keywords atm. Type as root:

```
echo sci-chemistry/pdb2pqr >> /etc/portage/package.keywords
echo sci-chemistry/apbs >> /etc/portage/package.keywords
emerge -av sci-chemistry/apbs sci-chemistry/pdb2pqr
```

Troubleshooting

- If the B-factor is ≥ 100, then APBS doesn't properly read in the PDB file and thus outputs garbage (or dies). To fix this, set all b factors to be less than 100.

  ```
  alter all, b=min(b,99.9)
  ```

  The problem stems from how to parse a PDB file. The PDB file originally was written when most people used FORTRAN programs, and so the file format was specified by columns, not by the more modern comma separated value format we tend to prefer today. For the latest on the PDB format see the new PDB format docs (https://web.archive.org/web/20110728103647/http://www.wwpdb.org/docs.html) .
- APBS has problems, sometimes, in reading atoms with alternate conformations. You can remove the alternate locations with a simple script removeAlt.
- ObjectMapLoadDXFile-Error: as of this writing (9-23-2008) a known problem exists, and the Baker lab is working on it. It is typically caused by the use of directories with spaces in their names under Windows.

Problems with the bundled version of APBS

There is an issue with the freemol version of APBS shipped with PyMOL 1.2r2 for OS X.

Leopard and Snow Leopard (10.5 and 10.6)

There are three fairly easy ways to resolve it

1. Download and install the most recent (post Dec. 1 2009) version of APBS from [1] (https://web.archive.org/web/20110728103647/http://www.poissonboltzmann.org/apbs/downloads) . Then copy the apbs binary into the freemol directory (mv it to /Applications/PyMOLX11Hybrid.app/pymol/freemol/bin/apbs.exe, overwriting the version that comes installed with PyMOL).

2. Download File:Libgfortran.3.dylib.bz2, unzip it ("bunzip2 libgfortran.3.dylib.gz2") and move it to /usr/local/lib ("mv libgfortran.3.dylib /usr/local/lib" ... on some machines, you may need "sudo mv libgfortran.3.dylib /usr/local/lib").

```
bunzip2 libgfortran.3.dylib.gz2
mv libgfortran.3.dylib /usr/local/lib
```

3. Use macports to install gcc 4.4.2 and link the appropriate library ("ln -s /opt/local/lib/gcc44/libgfortran.3.dylib /usr/local/lib/libgfortran.3.dylib" ... on some machines, you may need "sudo ln -s /opt/local/lib/gcc44/libgfortran.3.dylib /usr/local/lib/libgfortran.3.dylib").

```
ln -s /opt/local/lib/gcc44/libgfortran.3.dylib /usr/local/lib/libgfortran.3.dylib
sudo ln -s /opt/local/lib/gcc44/libgfortran.3.dylib /usr/local/lib/libgfortran.3.dylib
```

If you're curious, the problem is that APBS is dynamically linked, but Apple does not provide FORTRAN libraries.

The version of libgfortran above is covered by the GNU General Public License (GPL). A copy of the GPL may be found at [2] (https://web.archive.org/web/20110728103647/http://www.gnu.org/licenses/licenses.html) , and the source may be obtained from MacPorts ([3] (https://web.archive.org/web/20110728103647/http://www.macports.org/)).

Tiger (10.4)

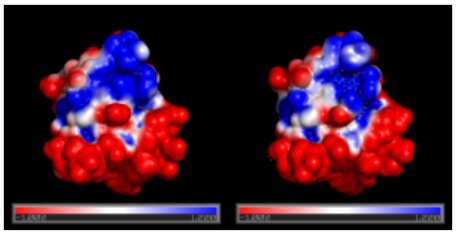
You'll need to install APBS yourself via MacPorts or via fink. Fink instructions may be found on the **APBS** page. Installation via MacPorts requires first installing MacPorts from [4] (https://web.archive.org/web/20110728103647/http://www.macports.org/) and then typing

```
sudo port install apbs
```

from the command line. This process could easily take several hours on an older machine, as MacPorts will recompile gcc, gfortran, and several other packages along the way.

Using APBS

There is a nice tutorial on the APBS homepage: [5]



PyMOL visualizing two maps at once

(https://web.archive.org/web/20110728103647/http://www.poissonboltzmann.org/apbs/examples/visualization/apbs-electrostatics-in-pymol) For further help, there is a mailing list [6] (https://web.archive.org/web/20110728103647/https://lists.sourceforge.net/lists/listinfo/apbs-users) with the corresponding archive [7] (https://web.archive.org/web/20110728103647/http://sourceforge.net/mailarchive/forum.php?forum_name=apbs-users)

Further contributions and edits are needed.

Retrieved from "http://www.pymolwiki.org/index.php/APBS"
Categories: Electrostatics | Biochemical Properties | Plugins

TOPTOC

- **See Also:** Old Table of Contents
- **Warning: This page is new and needs more updating.**

navigation
- Main page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Installation, Configuration & Learning

1. **Installation**
    1. Linux
    2. MAC
    3. Windows
2. **Stereo 3D Display**
    1. Linux XFree86 Configuration

1. **Using PyMol**
    1. **Launching**
        1. Launching PyMOL
        2. Command Line Options
        3. Launching From a Script
    2. Mouse Controls
    3. **Objects and Selections**
        1. **Working with Objects**
        2. **Working with Selections**
        3. **Selector Quick Reference**
            1. Single-word Selectors
            2. Property Selectors
            3. Selection Algebra
            4. Selection Macros

Commands, Representations & Setti

1. **PyMOL Commands**
2. **Settings (Documented)**
    1. All

1. **Representations**
    1. Lines
    2. Sticks
    3. Ball and Stic
    4. Cartoon
    5. Surface
    6. Mesh
    7. Spheres
    8. Dots
2. **Coloring**
    1. Color

Physico-chemical Modeling

1. **Biochemical Properties**
    1. Displaying biochemical properties
    2. Surfaces and Voids

1. Modeling and editing structures
    1. Editing Atoms
    2. Molecular sculpting
    3. Homology modeling

Images & Movies

1. **Image Manipulation**
    1. Labels
    2. Photoshop/GIMP
    3. Stereo Figures
    4. Publication-quality images
    5. PovRay

1. Making Mov
    1. Code

Extending, Embedding & Scripting

1. PyMOL Development
2. Scripting
    1. Example Scripts
    2. **Script Library**
3. **Plugins**
    1. **Electrostatics and energy minimization.**
        1. APBS Plugin
        2. Protein_contact_potential
    2. rtools
    3. Tutorial on writing plugins

1. High-Level Applications
    1. Crystallography Applications
        1. Symmetry|Symmetry
        2. Electron Density
    2. NMR Applications
        1. Working with Structure Families
    3. **Superimposition and RMSD**
        1. NMR Restraints and Analysis

Third Party Applications

1. O
2. SURFNET

1. Python Integratic
2. **PyMol_Integrat**
    1. Nuccyl -- PyMol ex
    2. CASTp -- site predi
    3. CCTBX - Crystallo
    4. ProMOL
    5. S2S
    6. **Arbitrary**

Category:Biochemical Properties

Information about Biochemical Properties in PyMOL.

Pages in category "Biochemical Properties"

The following 14 pages are in this category, out of 14 total.

A
- APBS

C
- Caver

D
- Displaying Biochemical Properties
- Dss

G
- Get Area

M
- Modeling and Editing Structures
- Mole

N
- Nonstandard Amino Acids
- Nuccyl

P
- Protein contact potential

P cont.
- Pucker

S
- Solvent radius
- Surfaces and Vc

V
- Vdw

navigation
- Main page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

navigation
- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Nonstandard Amino Acids

Overview

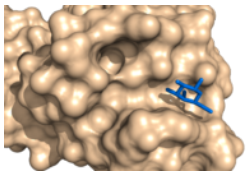
This page talks a little about how PyMOL deals with nonstandard amino acids, and the various representations and options available.

By default, PyMOL considers nonstandard amino acids as HETERO atoms. Therefore, when you draw a default surface, the heteroatoms are not included. Also, no out of the backbone representation in the cartoon and ribbon representations.

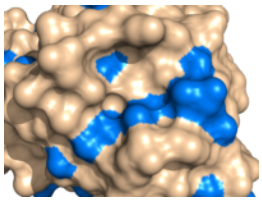
Workarounds

- If you're looking to represent the backbone via ribbon or cartoon, then just use the mutagenesis to modify the nonstandard amino acid to something standard, like G The draw/ray your structure as needed.
- If you want the nonstandard amino acid to be included in the surface representation, then set surface_mode to 1. For example, consider the images below. We hav in the image at left, not included in the surface, become part of the surface when we set the surface_mode to 1.

Surface Mode Examples



surface_mode set to 0, the default. The galactose (blue) is not considered part of the surface.



surface_mode set to 1 -- now including heteroatoms. The galactose and all heteroatoms (blue) are now considered part of the surface and colored blue.

- Another work around is to try

```
flag ignore, bymol polymer, set
rebuild
```

Categories: Biochemical Properties | Objects and Selections

navigation
- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

Mutagenesis

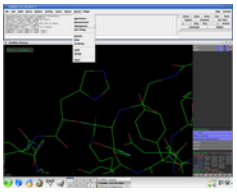
Mutagenesis

PyMol has a **Mutagenesis Wizard** to make mutagenesis very easy for the end user.

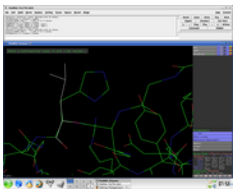
Walk-through

To mutate a residue follow these easy steps:

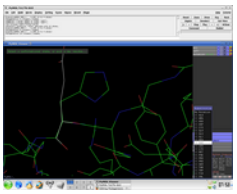
1. Load a PDB file



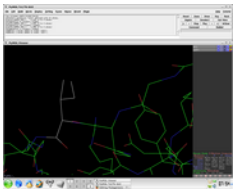
2. Under the **Wizard** menu select **Mutagenesis**



3. In the PyMol viewer window select a residue



4. Select **No Mutation** and select resultant residue



5. Selecting the rotamer you think better fits your structure.

   Several side chain orientations (rotamers) are possible. You can use the back-and-forth movie controls (lower right corner) to display (in white) each of the rotan residue in PyMOL, whose current and total numbers are shown in the (green) Frame info. The rotamers are ordered according to their frequencies of occurrenc a red percentage at the mutation object, which exists while mutagenesis is being performed.

6. Select Apply
7. Select Done

Explanation of colour codes

From [1] :

The visible disks & colors in the Mutagenesis Wizard indicate pairwise overlap of atomic van der Waals radii. The intent is to provide a qualitative feedback regarding c

Short green lines or small green disks are shown when atoms are almost in contact or slightly overlapping. Large red disks indicate signficant van der Waals overlap. E between those extremes.

Categories: Tutorials | Wizards