PyMOL: introductory tutorial

Introduction

PyMOL is a molecular structure visualizer, aimed primarily at structures of biological macromolecules (proteins and nucleic acids). These structures can be read from structure files of different formats, the most common being the Protein Data Bank (PDB) format, which usually having a `.pdb` extension. Aside from PDB files, PyMOL can read structures in mmCIF, mmtf, mol2, sdf and a few other formats. The macromolecular structure files can be obtained from the PDB website (https://www.rcsb.org), however PyMOL has internal commands allowing to fetch PDB structures from within the software (see below).

PyMOL cannot only visualize molecules in different modes (like "sticks", "cartoon", "spheres", "surfaces", etc...), but it can also color molecules or molecular regions in different colors, apply text labels to groups of atoms, measure angles, distances and surface areas, align structures and many other tasks.

Retrieving molecular structures

In order to visualize molecular structures, we first need files with atomic coordinates, names, connectivity (chemical bonds) and other information required for structure representation. Such files can be obtained from the Protein Data Bank website (https://www.rcsb.org) (from now on referred to as "PDB"). The PDB website allows you to search structures based on a number of criteria: structure name, deposition date, atomic resolution, source organism, author names and several other properties. Each PDB structure is designated by a unique code than can be used to obtain the structure file, either by downloading it from the PDB or from within PyMOL using the console command `fetch` or the menu command `File→Get PDB`. If the structure had been previous downloaded to your computer, it could be loaded into PyMOL with `File→Open`, and then selecting the desired file.

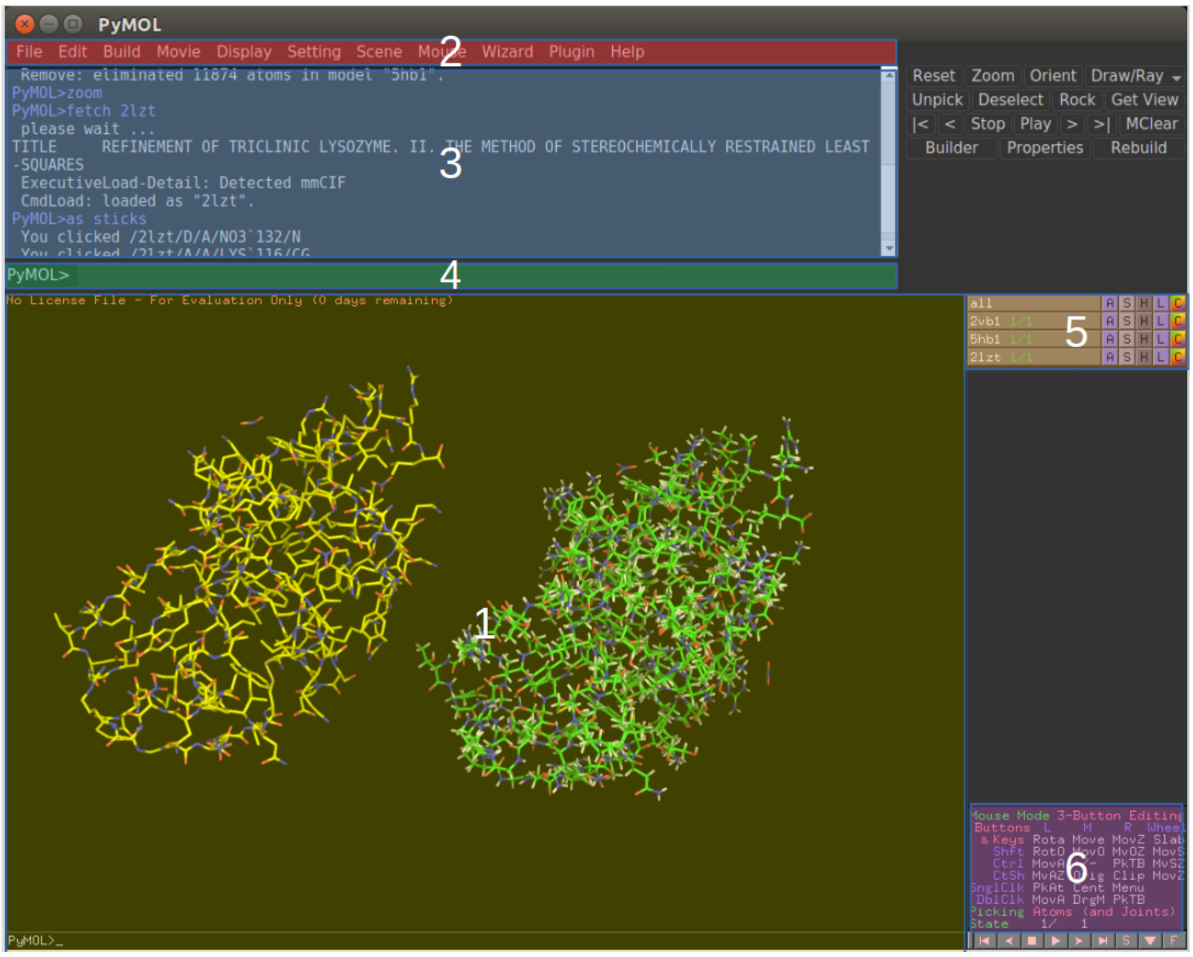
Starting PyMOL

Look for the icon of the PyMOL software, or search for its name using the Windows search bar. When you start the program, a window will pop up, where you should press the button labeled "Skip Activation" (you will have to do is every time you start the program). The first time you run PyMOL, you will be presented with a list of default file associations in a second pop-up window. Close that and you should be left in a program window similar to what is described below.

The program window contains several zones with distinct functionality (see figure):

1. **Visualization area:** the area where the structure (or structures) are displayed for visualization.
2. **Menu Bar:** this bar contains a number of entries named "Edit", "Build", "Movie", "Display","Setting", "Scene", "Mouse", "Wizard", "Plugin" e "Help". The corresponding pull-down menus offer many PyMOL commands, the majority of which can also be executed from the *input* console.

3. **Output Console:** this is the area where PyMOL writes the result of running commands (if they produce result at all), and produces a variety of informative messages. If command a produces and error, it will be displayed here.
4. **Input Console:** this region can be used to type many different PyMOL commands (terminated by pressing the ENTER key), and provides an (often much faster) alternative to using menu commands.
5. **Object Menu:** This area features horizontal bars with buttons for applying commands all loaded object (the "all" bar) or to each object separately (there is a bar for each loaded object). The buttons are **A** (action), **S** (show), **H** (hide), **L** (label) e **C** (color). When you start PyMOL, there are no objects loaded, so only the "(all)" bar will be shown, and it does nothing.



```
 Remove: eliminated 11874 atoms in model "5hb1".
PyMOL>zoom
PyMOL>fetch 2lzt
 please wait ...
TITLE     REFINEMENT OF TRICLINIC LYSOZYME. II. THE METHOD OF STEREOCHEMICALLY RESTRAINED LEAST
-SQUARES
 ExecutiveLoad-Detail: Detected mmCIF
 CmdLoad: loaded as "2lzt".
PyMOL>as sticks
 You clicked /2lzt/D/A/NO3`132/N
 You clicked /2lzt/A/A/LYS`116/CG
```

PyMOL
File Edit Build Movie Display Setting Scene Mouse Wizard Plugin Help
Reset Zoom Orient Draw/Ray
Unpick Deselect Rock Get View
|< < Stop Play > >| MClear
Builder Properties Rebuild
PyMOL>
No License File - For Evaluation Only (0 days remaining)
all
2vb1 1/1
5hb1 1/1
2lzt 1/1
A S H L C
Mouse Mode 3-Button Editing
PyMOL>_
1
2
3
4
5
6

Basic PyMOL Usage: mouse controls and object menu

Let's assume the PDB structure with code 2VB1 as has been retrieved from the PDB, and is stored in some folder in your computer (pdb files can be retrieved by code at the RSCB Protein Data Bank Site, https://www.rcsb.org). Let's start PyMOL as described above, and then select the `File→Open` option on the top menu bar (see Figure 1). Now navigate the file system until the folder where you stored the `2vb1.pdb` file, and select it. The file should be loaded in PyMOL, with a graphical representation appearing on the visualization area of the main window. Also, some information regarding the loaded structure will be posted on the output console (e.g. the structure name). By default, PyMOL will represent the structure in "cartoon" mode (schematic representation of the polypeptide chain, emphasizing the different types of secondary structure present, but with no atomic detail), while ions and other "foreign" molecules will be represented as spheres. The crystallographic water molecules (trapped in the crystal structure at relatively fixed conformation near the protein surface) are represented as red dots for the oxygen atoms (the water hydrogens are often not visible in the crystallographic structures, due to experimental limitations).

PyMOL uses the following default colors for chemical elements: green for carbon, red for oxygen, blue for nitrogen, yellow for sulfur, white for hydrogen. The remain biological elements are colored with other colors (sodium, potassium, chloride, zinc, copper, iron, magnesium,etc). The full set of element colors used by PyMOL can be checked at this page: https://pymolwiki.org/index.php/Color_Values#Chemical_element_colours

Note: this are the default colors, but they can easily be changed by the user with simple console commands (see below).

The object menu should contain two lines, one name "(all)" and another "2vb1" (because the current scene contains only one object).

Let's assume we have a three-button mouse (highly recommended) connected to the computer. We can use the following mouse commands to move, rotate, scale and orient the structure presented in the visualization area:

- **left mouse button:** rotates the viewing camera around the screen center
- **middle mouse button:** moves the viewing camera laterally (*panning*)
- **right mouse button:** moves the camera away or towards the scene (*zooming*)
- **mouse wheel:** controls the depth of the viewing area (*viewing slab*)

NOTE: Though it may appear as if it is the molecule that is moving, it is in fact the viewing camera that changes position with the above mouse commands. This will be more obvious once you load more than one molecule into PyMOL.

NOTE: The above commands are for a system equipped with a 3-button mouse. If you are on a laptop with a 2-button track pad, you can switch the mouse command layout using the menu command `Mouse→2 Button Viewing`. In that case, the mouse commands are:

- **left mouse button:** rotates the viewing camera around the screen center
- **ctrl+left mouse button:** moves the viewing camera laterally (*panning*)
- **right mouse button:** moves the camera away or towards the scene (*zooming*)

After some camera manipulations, it is very possible that the molecule ends up off centered, or even totally out of the visualization area. In that case, the "zoom" command can be used to bring the objects back centered and in full view. This command can be issued in two different ways:

1. In the input console, write "zoom" and press ENTER.
2. In the object menu, select the "(all)" bar and click the button marked with an **A**. A menu with various options will pop up, one of them being *zoom*. Just select it.

Either of the two options above will produce the same effect: the lysozyme molecule will be back on screen, fully centered and filling most of the screen area.

To change the molecule representation type, we can use the "S" button on the "2vb1" or "(all)" bars on the object menu (presently they have exactly the same effect, because there's only one object loaded on the program). Let's select the option "sticks" on the "S" button. The result of this command is to show the stick representation of the molecule. which will appear superimposed to the previous "cartoon" representation. If we want to keep *only* the cartoon representation, we could go and press the **H** (hide) button on the same bar, and select "cartoon".

To color our molecule in a different color, let's the press the "C" button on the "2vb1" bar, and then choose "reds", and then "tv red".

Exercise: Represent the molecule as "spheres", and color it blue. Can you see the sticks ? .. Hide the stick representation anyway, and also cartoon. Show the molecule as surface. It probably looks a bit weird, because you can see the sphere representation protruding through the surface. Hide the spheres representation.

Next we are going to load a second molecule in PyMOL, but this time we will do it directly through the program command interface, using the option "Get PDB" on the top menu bar option "File". In the box that pops up, make sure that only the box labeled "Pdb structure" is ticked. Type in the code "3b0i" (human lactalbumin) in the box labeled "PDB ID" and press the "Download" button. The structure representation for the 3b0i structure is added to the PyMOL visualization window, and a new bar labeled "3b0i" appears in the object menu. Presently, we have two molecules on screen, represented in different modes. To view both molecules as "sticks", go to the "(all)" bar, click the **S** button and select the option "As:" and then "sticks". Note that all other representations vanish, and only the sticks representation for both molecules remains (contrast this with the previous "show" command, adding new representations to what is already on the screen).

Exercise: Represent the two molecules as "ribbon" and color one in red and the other in green.

Using PyMOL console commands

As previously explained, the input console command area (number 4 in Figure 1), allows for sending text commands to the software. Many of these commands are alternatives, often faster, to commands than can be issued from other command areas of the software (like the top menu bar or the object menu).

Let's start by deleting all loaded molecules with the following console command:

```
delete all
```

And now, lets again load the molecule 2vb1 into PyMOL, using the console command:

```
fetch 2vb1
```

NOTE: this command will download the molecule straight from the Protein Data Bank, by default in mmCIF format (extension `.cif`) unless it is already on the folder where PyMOL is currently looking for files. You can check "where" PyMOL is by typing the command `pwd` (print working directory) on the console. This is the place where it will create or read files, unless you specify otherwise.

Now let's represent 2vb1 as sticks:

```
as sticks, 2vb1
```

and let's load the second molecule, 3b0i, with the command:

```
fetch 3b0
```

Let's show both molecules as ribbons, deleting all other representations:

```
as ribbon
```

Now color the first molecule in red:

```
color red, 2vb1
```

and the second molecule in green:

```
color green, 3b0i
```

And now let's use the `align` command to superimpose the two molecules, allowing us to compared the structures and obtain a quantitative measure of their similarity:

```
align 3b0i, 2vb1
```

The molecules will be superimposed on the screen and the returned result of this command shows on the output console (number 4 in Figure 1) indicating an RMSD of 0.9 Angstrom. The RMSD is a measure of structural similarity, and a 0.9 A value indicates a high degree of similarity between the structures of lysozyme and lactalbumin. Note that the 3b0i molecule moves on top of 2vb1, making the image off-centered. You can recenter using the following console command:

```
zoom
```

Exercise: using console commands, remove the molecule 3b0i and represent 2vb1 as surface. Then hide the surface and represent the molecule as sticks, to be used in the next sections.

PyMOL selection language

PyMOL console commands only become really powerful when one is able to select specific molecules, molecule regions or atoms for the objects in the scene. For that purpose, there's a selection language available, based on the following hierarchical organizing of data objects:

```
object → segment → chain → residue → atom
```

meaning that an object can have one or more segments, and those segments in turn can have one or more chains, and those chains can have one or more residues, and residues in turn will have one or more atoms.

1. Object: generally a molecule, but could contain more than one copy of the same molecule, or be an oligomer of different molecules (water and small molecule ligands bound to a protein are generally part of the same object)
2. Segment: use to designate different regions of a molecule, corresponding to chain groups, ligands, etc. Many PDB objects don't define segments or define them like chains (one segment -> one chain)
3. Chain: this hierarchic level is normally use for each of the polypeptide chains present in an object (no matter if the object has one or more proteins). Ligands, ions, and other species may also have their own chain code.
4. Residue: one of the aminoacid residues of a protein, or a nucleotide in a nucleic acid, but also any small molecule or non-aminoacidic prosthetic group. Can be designated by name or number (see below)
5. Atom: an atom present in one of the molecular objects in the system. It is designated according to the standard nomenclature of protein and nucleic acids used by the Protein Data Bank, and based on the organic chemistry nomenclature for small molecules.

The most general type of selection expression will be of the form:

```
/object/segment/chain/residue/atom
```

with the `/` being used as a separator between the identifiers of the different levels.

This specific example:

```
/2VB1/A/A/34/CE2
```

references the atome named `CE2` in residue number `34` of chain `A` of segment `A` of object `2VB1`. This will be the carbon ϵ2 of the phenylalanine ring in residue 34 of lysozyme. Note that the object identifier (`2vb1` in this case) generaly comes from the name of the corresponding PDB file or code.

This selection can be used for instance in the following console command:

```
zoom /2VB1/A/A/34/CE2
```

which zooms in on the atom `CE2` of residue `34` of object `2vb1`, bringing it to the center of the screen. In this case the leading `/` can be omitted:

```
zoom 2VB1/A/A/34/CE2
```

because the four intervening `/` are enough for PyMOL to guess the correct meaning of all the five identifiers.

Even more abbreviated representations are permitted, such as:

```
34/CA
```

which refers to the alpha carbon of residue 34, of *all* chains of all objects loaded in PyMOL. In the present case there is only one object with one chain loaded in PyMOL, so if you issue:

```
zoom 34/CA
```

the carbon alpha of residue `34` will be brought to the center of the screen.

You can also use specifications of the type:

```
2vb1////CA
```

the above expression refers to all carbon alpha atoms in 2vb1 and could be used, for instance, in the following console command:

```
color red, 2vb1////CA
```

NOTE: selections in PyMOL that do not start with a `/` will require a trailing `/` if they end on a residue identifer (name or nunmber). For example:

```
color green, 123/
```

will color green all atoms of residue `123` of all chains of all loaded objects, but:

```
color green, 123
```

will produce an error message. On the other hand, if a selection expression starts with a `/`, then the trailing `/` after a residue id won't be needed. For example:

```
color red, /2vb1///100
```

will not produce an error. The sequence `///` is used to *omit* the segment and chain identifiers, ensuring that 100 really is at the correct position for a residue id. This means that a residue number in *any* segment or *any* chain will be matched (in this case there will be only one match, because the `2vb1` object contains only one segment with a single chain).

NOTE: To better understand the difference between selection expressions with a leading or trailing `/`, think of it this ways: expressions with a leading `/` are read left to right, while expressions with a trailing `/` are read right to left.

The following expression:

```
color red, 100/CA/
```

will produce an error - there cannot be anything after the atom identifier, therefore the trailing "/" is invalid.

The selection language allows for the specification of residue *ranges*. For example, the command:

```
color red, 10-123/
```

will color all residues from 10 to 123. To specify non-contiguous ranges, the following notation can be used:

```
color red, 40+70/
```

which will color in red aminoacids 40 and 70.

It is also possible to specify residue *names* instead of numbers. One must note, however, that name specifications tend to be ambiguous, since there will be (in general) more than one residue with a given name. The following command, for instance:

```
color red, //A/HIS/
```

will color red all histidine residues present in chains labeled "A" of all loaded objects. The command:

```
show spheres, 2vb1//A/ASP+GLU/
```

will show all aspartic and glutamic acid residues as spheres in chain `A` of any segment of object `2vb1`.

Exercise: Represent the molecule 2vb1 as "ribbon". Color the molecule red. Represent as green sticks the catalytic Glu 35 and Asp 52. Represent the molecule as surface and observe the localization of the catalytic residues in the active site. To see it better, turn the surface transparent with the following console command:

```
set transparency, 0.2
```

so that you can see the active site residues through the surface.

The PyMOL selection language has many more commands allowing the specification of atoms by other properties, and operators like `and` or `or` to combine expressions. To see a full list of keywords and operators and their meanings, please check https://pymolwiki.org/index.php/Selection_Algebra .

Selection commands

PyMOL has the ability to *select* regions of objects, so that those regions can be processed as a group by different commands. These regions can even be given specific names to be used for later work

Selections in PyMOL can be created in two ways: interactively with the mouse, or using the "select" console command. Let us describe each of this mechanisms in turn:

1. **Selection with Left Mouse Button:** by clicking with the left mouse button on an atom, different groups of atoms can be *selected* depending on the current **selection level**. The selection level can be checked and changed in region 6 of PyMOL interface (see Figure 1). There you will find a line starting with "Selecting: " and after that the word "residues". This indicates that the current selection level is residue - clicking on an atom will select the entire residue it belongs to. By clicking on the "Selecting:", it will circulate through the following selection levels:

   ```
   Residues→Chains→Segments→Objects→Molecules→C-alphas→Atoms
   ```

   The current selection level indicates what happens when you pick an click on an atom. If the selection level is "chain", click an atom will select the entire chain. If the selection level is "residue", only the atoms of the residue to which the clicked atom belong will be selected. The lowest selection level is "Atom" which will select *only* the clicked atom.

Any time that a selection as been created in this way, there will be a "(sele)" line in the object menu window. That line has the exact same options (Action, Show, Hide, Label and Color) as any PyMOL object - any commands applied trough these options will affect *only* the selected region.

Selections can be expanded my clicking on multiple regions of the objects. As long as you keep clicking, the new selections will be added to the selection object - for instance, if the selection level is "residues", any clicks will add the new residues to the "(sele)" object. If you click on the "void" outside objects, the selection will be erased and you will start from scratch.

2. **Console command "select":** the "select" console command can be used in two ways to generate selections:
    - Unnamed selection: type "select" followed by a selection language expression, for instance `select ASP+GLU/`. A selection object "(sele)" will appear in the object menu. Click on it to see the Glu and Asp residues highlighted in the molecule representation. Any operations on the selection object will affect *only* the Asp and Glu residues in the protein (try representing them as spheres).
    - Name selections: type "select negatives, ASP+GLU/" on the console. Now a new object name "(negatives)" will appear on the object menu. As before, any operation on this object will affect only the Glu and Asp residues. But now, if you by accident click on some part of the protein, you will reset the "(sele)" object, but not the "(negative)". In this way you can create multiple named selections in your object and represent them in different ways. (as an exercise, try to create a selection for the positive residues Lys, Arg and His, and the color the positive residues in "cyan" and the negative residues in "red").

    NOTE: object selection names appear enclosed in "()" in the object list. You don't need, in fact, to use the "()" to refer to them, so for instance doing "color green, negatives" will color Asp and Glu in green.

Saving your work

If you by any reason need to interrupt your working session and have already loaded a number of structures, created different representations and named selections, etc... there is a way to save your work so that you can restart PyMOL and pick up exactly where you left. All you have to do is to create a PyMOL session file (extension `.pse`) before exiting PyMOL, for instance:

`save work.pse`

NOTE: you can use any name you want before the `.`, but after it you must use the `pse` extension for PyMOL to create a session file.

When you are ready to continue, just restart PyMOL and type the following console command:

`load work.pse`

and PyMOL will load all the objects and their current representations, plus any named selections into the working space.