PyMOL: introductory tutorial

Introduction

PyMOL is a molecular structure visualizer, aimed primarly at structures of biological macromolecules (proteins and nucleic acids). These structures can be read from structures files of different formats, the most common being the PDB (Protein Data Bank) format, usually having a ".pdb" extension. Aside from PDB files, PyMOL can read structures mmCIF, 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 grab the structure 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 different molecule regions with different colores, apply text labels to groups ao atoms, measuare angles, distances and surface areas, align strutctures 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 Protetein 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, etc. Each PDB structure is designated by a unique code than can be used to obtain said structure, 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 and is on your computer, it cand be loaded into the software with "File→Open", and then selecting the desired file.

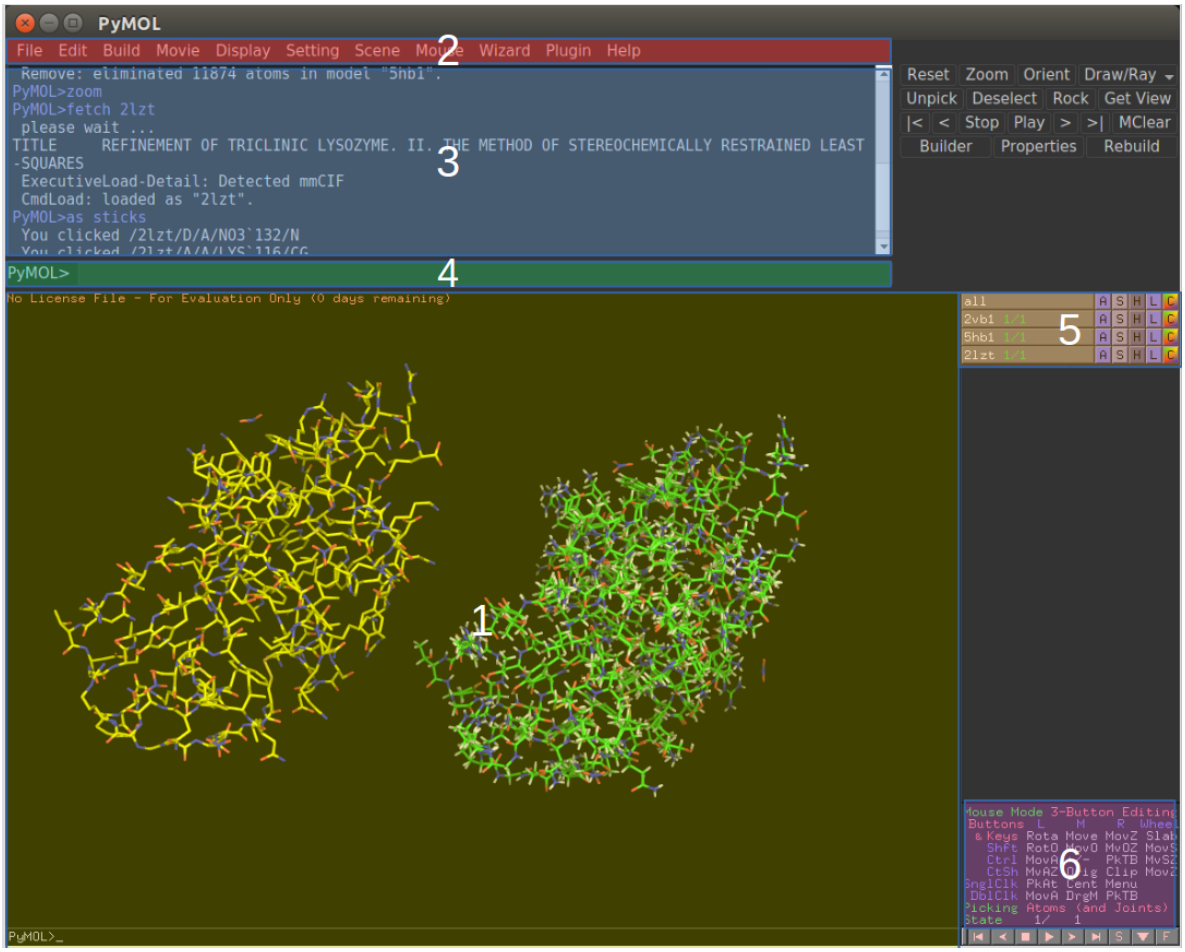
Starting PyMOL

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". The program will then present with a list of default file associations in another pop-up window. Close that window and the program window should be similar to the presented below.

The program window contains distinct zones with distinct funcionality:

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 varieity 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) aternative 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.



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 retived 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, click on it. The file should be loaded on PyMOL, and the structure visualized on the main window. visualizaton area. 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 differen 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) as represened as red dots for oxygen atoms (the water hydrogens are usually not visible in the crystallographic structures).

PyMOL uses the following default colors for chemical elements: green for carbon, red for oxygen, blue for nitrogen, yellow for sulphur, white for oxygen. The remain biological elements are colored with other colors (sodium, potassium, chloride, zinc, copper, iron, magnesisum,etc). The full set of element colours used bv 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.

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 thhree-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:

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

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 one can use a command 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-*

Either of the two options above will produce the same effect: the lisozyme molecule will be back on screen, full 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 programa). 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" represenation. If we want to keep *only* the cartoon represenation, we could go press the **H** (hide) button on the same bar, and seleect "cartoon".

To color our molecule in a different color, let's the press the C button on the "2vby1" bar, and then choose "reds", and "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 represenation.

Next we are going to load a second molécule 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 presss the "Download" button. The structure represenation for the 3b0i structure is added to the PyMOL visualization window, and a new bar labeled "3b0i" appeares 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 thep option "As:" and then "sticks". Note that all other represenations vanish, and only the sticks represenation for both molecules remain (contrast this

with the previous "show" command, adding new represenations 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 comand areas of the software (like the top menu bar or the object menu).

Let's start by deleting all loadaed 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 that is command you download the molecule straight from the Protein Data Bank, unless it is already on the folder where the PyMOL is currently looking for files)

Now let's represent 2vb1 as sticks:

`show 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 measuer 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 structurel 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.

PyMOL selection language

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

object → segment → chain → residue → atoma

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 (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). Ligandas, 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 prostetic 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 nomenclatura for small molecules.

The PyMOL selection language uses "/" as a separator betweeen levels. For example:

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

reference the object 2vb1 (normally the object code comes from the PDB code, as in this case), segment A, chain A, aminoacid residue 34, atom with name CE2 (carbon epsilon 2 on the pheninalanine ring). This selection might have been used for instance in the following command:

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

which brings atom CE2 of residue 34 of 2vb1 to the center of the screen.

Abreviated representations are permitted, such as:

```
34/CA
```

which referes 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 cabon alpha atoms in 2vb1 and could be used, for instance, in the following console command:

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

Important note: selection expressions ending on a residue nubmer or residue name will need a trailing "/" to be valid. For example:

```
color green, 123/
```

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

```
color green, 123
```

will produce an error message. It is also possible to specify residue *ranges*. The command:

'color red, 10-123/`

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

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

which will color red aminoacids 40 and 70.

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

```
color red, 2vb1//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 represent all aspartic and glutamic acids as spheres.

Exercise: Represent the molecule 2vb1 as "ribbon". Color the molecule red. Represent as green sticks the aminoacids Glu 35 and Asp 52. Represent the molecule as surface and observe the localization of the calatlyic residues in the activa site. To see it beter, 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.