Utilisation avancée

Sommaire

- Environnement
- Scripts
- Distances et angles
- Alignement et superposition
- Déplacement d'objets
- Cristallographie
- Compiled Graphic Objects

Environnement

Le fichier `pymolrc`

Le lancement de PyMOL peut être modifié avec le fichier pymolrc. L'emplacement de ce fichier est :

- `${HOME}/.pymolrc` sous Linux et Mac OS X
- `%HOMEDRIVE%%HOMEPATH%\pymolrc.pml` sous MS Windows

Une version améliorée existe, `pymolrc.py`. Il contient du code Python qui permet d'effectuer des actions plus génériques.

Le fichier `pymolrc` : exemple

```
# Affichage de l'origine
run /path/to/home/pymol/scripts/axes.py

# Suppression de la brume
set depth_cue, 0

# Un meilleur rendu pour le ray tracing
set antialias, 2

# Un fond transparent
set ray_opaque_background, 0

# Des liens souples plus proche de la chaine principale
set cartoon_smooth_loops, 0
```

Note : pas de caractère accentué dans ce fichier

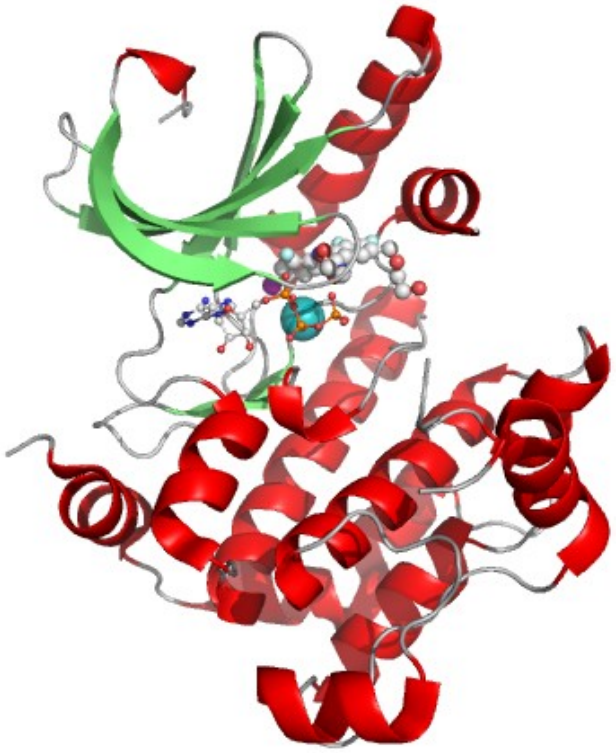
Scripts

Les scripts

Un ensemble de commande :

```
PyMOL> load 3DV3.pdb
PyMOL> hide everything
PyMOL> select mg, resn MG
PyMOL> select atp, resn ATP
PyMOL> select mek, resn MEK
PyMOL> select protein, chain a & !mg & !atp & !mek
PyMOL> show spheres, (atp or mek or mg)
PyMOL> set sphere_scale, 0.2, atp
PyMOL> show lines, atp
PyMOL> set sphere_scale, 0.4, mek
PyMOL> show sticks, mek
PyMOL> set sphere_transparency, 0.2, mg
PyMOL> show cartoon, protein
PyMOL> color red, ss h & protein
PyMOL> color lime, ss s & protein
PyMOL> color grey70, ss '' & protein
PyMOL> bg_color white
...
```

Les scripts : résultat



Les scripts

```
load 3DV3.pdb
hide everything
alter A/61-66/,ss=''
rebuild
select mg, resn MG
select atp, resn ATP
select mek, resn MEK
select protein, chain a & !(mg | atp | mek)
show spheres, (atp or mek or mg)
set sphere_scale, 0.1, atp
show lines, atp
set sphere_scale, 0.3, mek
show sticks, mek
set sphere_transparency, 0.2, mg
show cartoon, protein
```

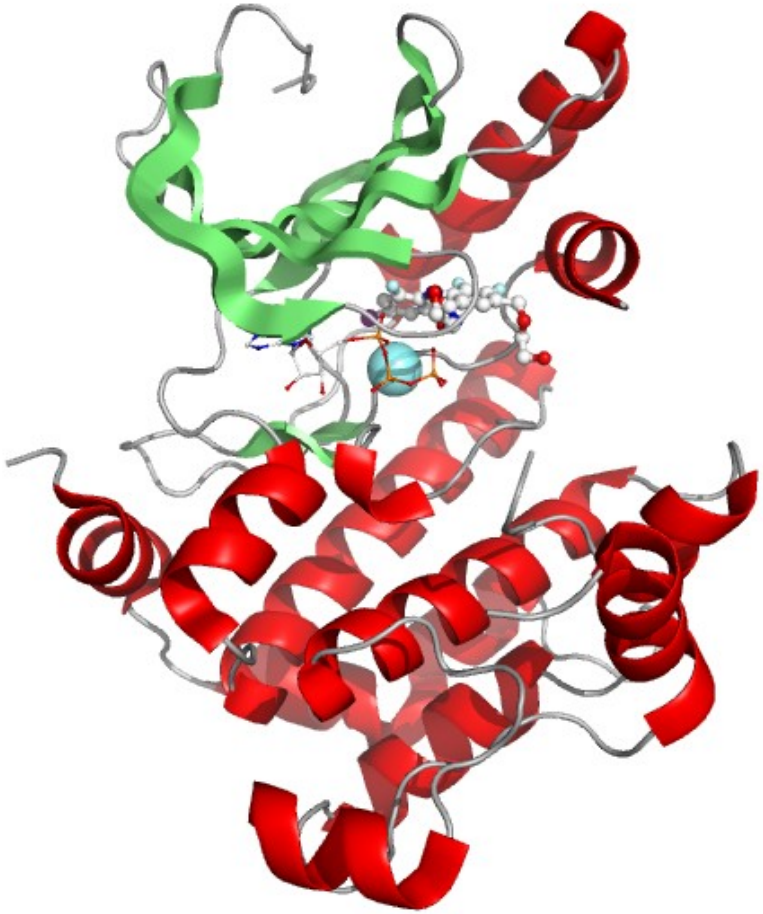
Les scripts

```
color red, ss h & protein
color lime, ss s & protein
color grey70, ss '' & protein
color red, elem o & (atp | mek)
color blue, elem n & (atp | mek)
color white, elem c & (atp | mek)
color orange, elem p & (atp | mek)
color aquamarine, mg
set cartoon_discrete_colors, 1
set cartoon_flat_sheets, 0
bg_color white
set_view (\
    -0.749093771,   -0.629502356,    0.206357747,\
     0.161474541,    0.128601521,    0.978463829,\
    -0.642484367,    0.766282082,    0.005313656,\
     0.000000000,    0.000000000, -175.443603516,\
    47.899505615,  -15.624248505,   -4.946440697,\
   138.321075439,  212.566131592,  -20.000000000 )
ray
```

Les scripts : résultat

Pour exécuter le script :

```
PyMOL> @3DV3.pml
```



Sauvegarde des commandes

Il est possible de sauvegarder toutes les commandes qui sont entrées via l'invite de commandes en utilisant un fichier de log. Pour cela :

- sélectionner File → Log...

ou

- **PyMOL>** `log_open` *`nom_du_fichier_de_log`*
- **PyMOL>** `log_close` *`nom_du_fichier_de_log`* (pour arrêter l'enregistrement)

Distances et angles

L'affichage des liens et des distances

La commande **distance** permet de calculer et d'afficher le la distance entre deux éléments. Par exemple, avec la structure 3DV3, regardons la distance entre l'atome de magnésium et les atomes d'oxygène environnants :

Sélectionner les atomes d'oxygène dans une sélection appelée (oxy)

```
PyMOL> distance coor, oxy, mg
```

Note : Effacer le style de *mg* pour voir les distances

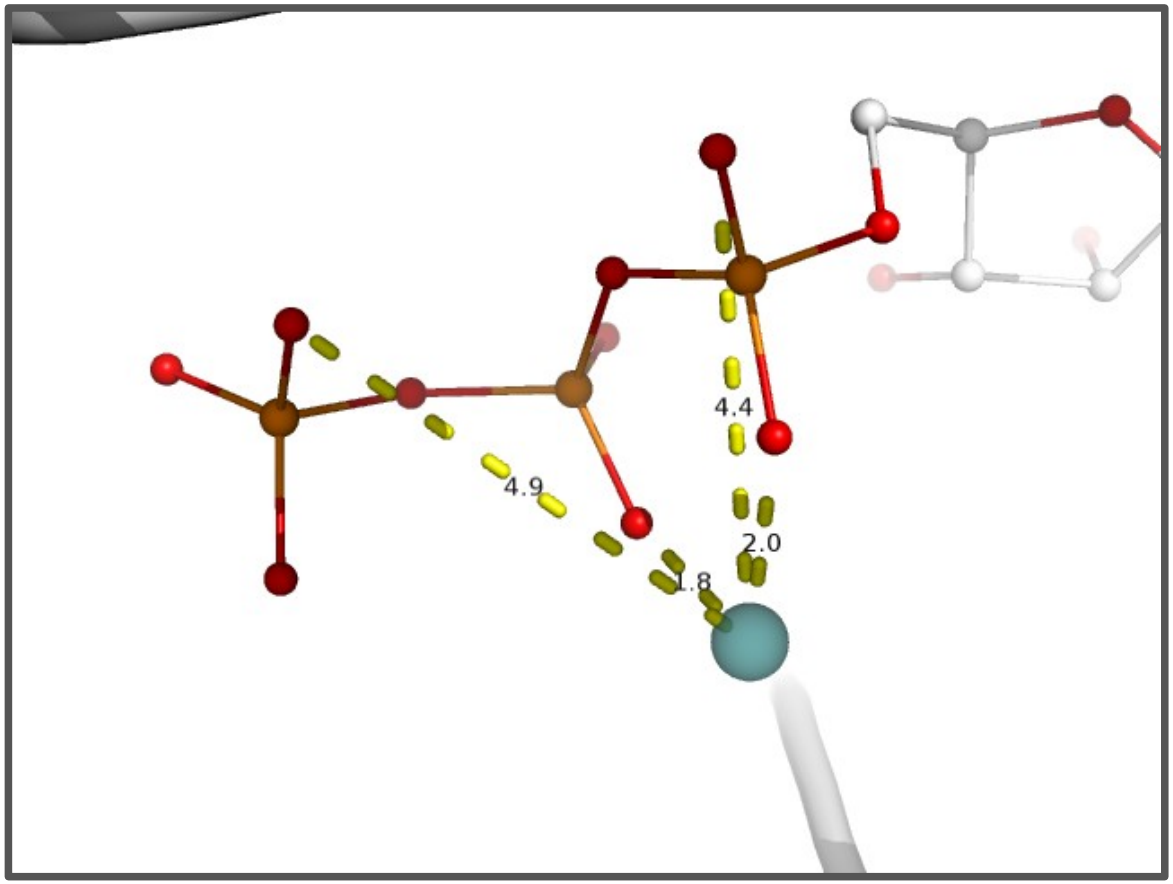
Modification du style des tirets :

```
PyMOL> hide labels, coor
PyMOL> set dash_round_ends, 1
```

Note :

Il est également possible d'utiliser le Wizard *Measurement*.

Affichage des distances : résultat

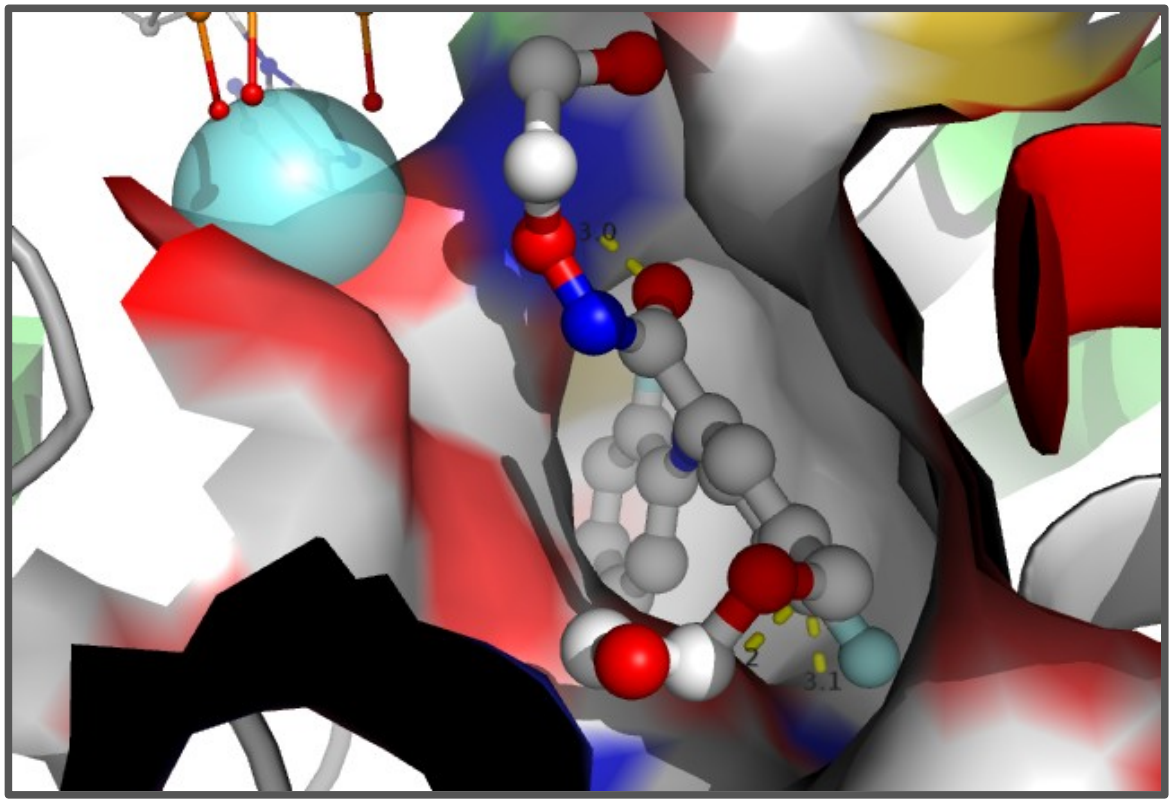


Les liaisons hydrogène

L'affichage des liaisons hydrogène est une extension de la commande précédente. La stratégie est d'afficher les liens entre donneurs et accepteurs distants de moins de 3,2 Å :

```
PyMOL> select pocket, protein & (mek around 6)
PyMOL> show surface, pocket
PyMOL> distance hbonds, pocket, mek, 3.2, mode=2
```

Les liaisons hydrogène



Les liaisons hydrogène

Il est possible d'améliorer la finesse de la précédente sélection en ne sélectionnant que les atomes donneurs et accepteurs de liaisons hydrogène :

```
PyMOL> select pocket, protein & (mek around 6)
PyMOL> show cartoon, pocket
PyMOL> h_add pocket
PyMOL> h_add mek
PyMOL> select pdonneurs, (elem n,o & (neighbor hydro)) & pocket
PyMOL> select mdonneurs, (elem n,o & (neighbor hydro)) & mek
PyMOL> select paccepteurs, (elem o or (elem n & not (neighbor
hydro))) & pocket
PyMOL> select maccepteurs, (elem o or (elem n & not (neighbor
hydro))) & mek
PyMOL> distance hbond1, mdonneurs, paccepteurs, 3.2
PyMOL> distance hbond2, pdonneurs, maccepteurs, 3.2
```

Angle et angle dièdre

Le calcul et l'affichage des angles se font avec les commandes :

- **angle**
- **dihedral**

Utilisation :

```
angle nom, atome1, atome2, atome3
dihedral nom, atome1, atome2, atome3, atome4
```

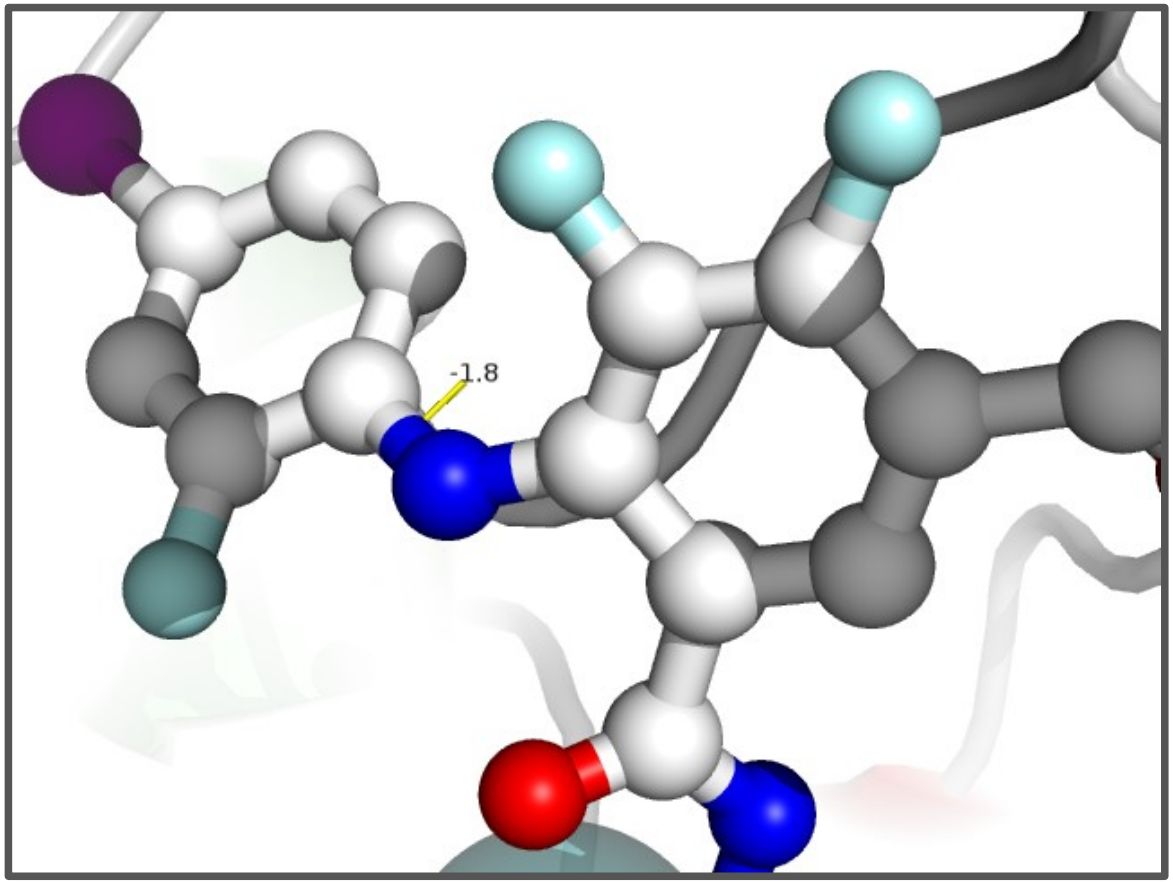
Exemple :

```
PyMOL> dihedral torsion, pk1, pk2, pk3, pk4
```

ou

```
PyMOL> dihedral
```

Angle dièdre : résultat



Alignement et superposition

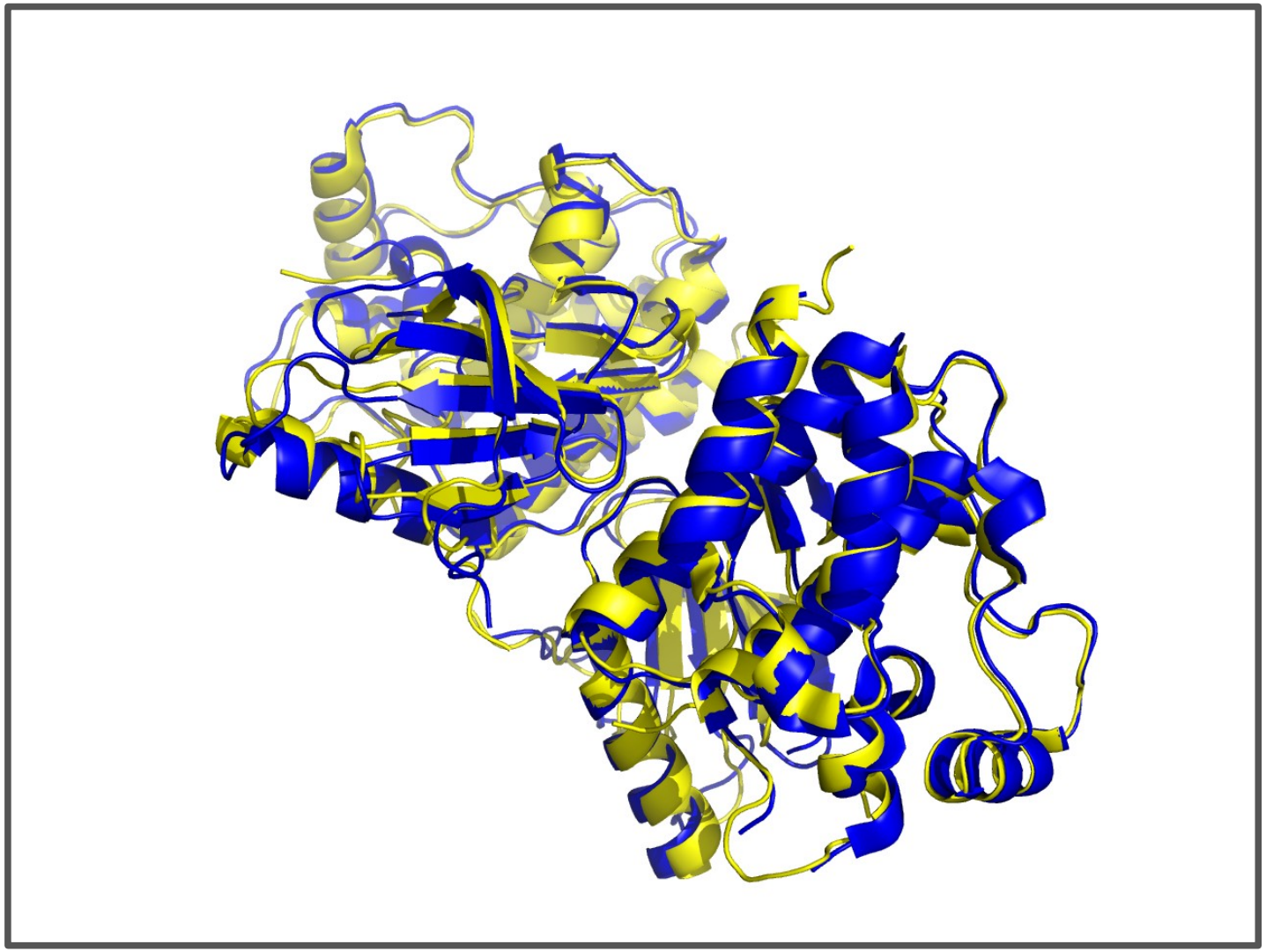
Alignement de structures

PyMOL permet d'effectuer des alignements de structure via la commande **align**. Dans un premier temps, le logiciel effectue un alignement de séquence, puis il essaye de superposer les deux structures afin de minimiser la valeur RMSD entre chaque résidu.

Exemple d'application :

```
PyMOL> load 3IDP.pdb
PyMOL> load 3D4Q.pdb
PyMOL> align 3IDP, 3D4Q
```

Alignement de structures



Alignement de structures : ressources complémentaires

Voir aussi :

http://www.pymolwiki.org/index.php/Align

Pour aller plus loin :

http://www.pymolwiki.org/index.php/Cealign
http://www.pymolwiki.org/index.php/Kabsch

Mutagenèse

Mutagenèse

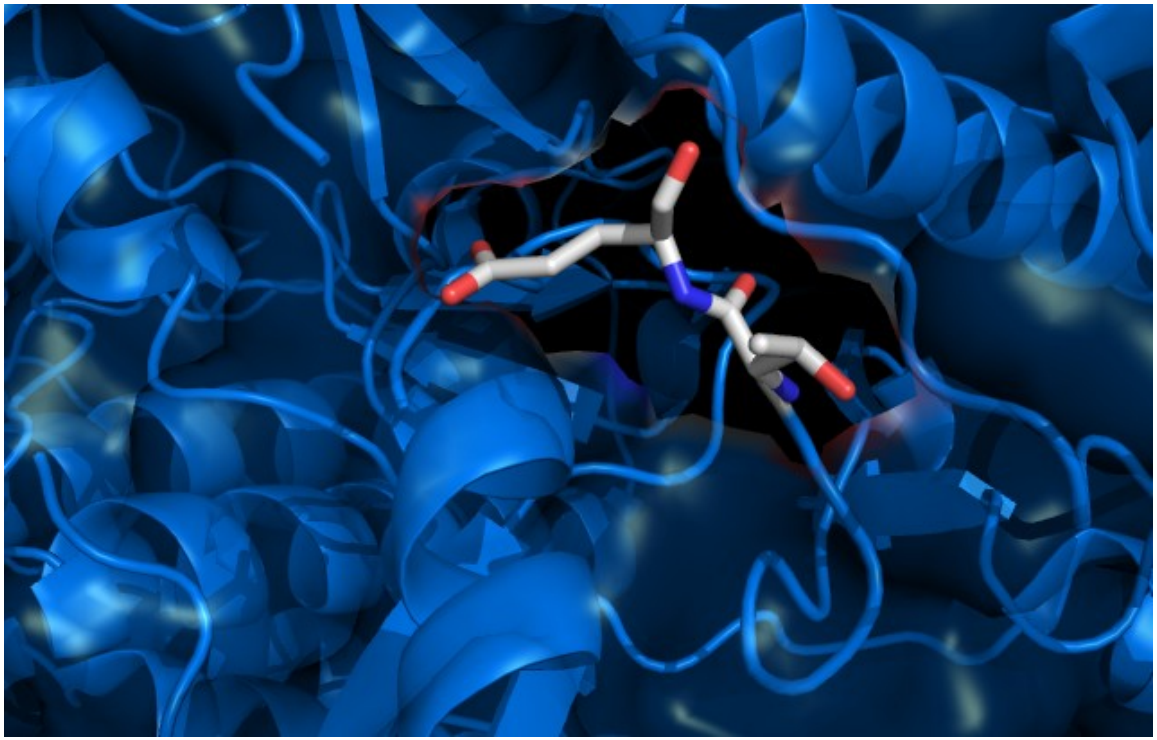
L'outil *Mutagenesis* du menu *Wizard* permet de modifier un ou plusieurs acides aminés dans une protéine. L'utilisation est relativement simple :

1. Appeler l'outil *Mutagenesis*
2. Sélectionner l'acide aminé à muter
3. Choisir la mutation
4. Choisir le rotamère (utilisation des contrôles *film* pour voir les possibilités)
5. Appliquer la mutation

Exemple d'application :

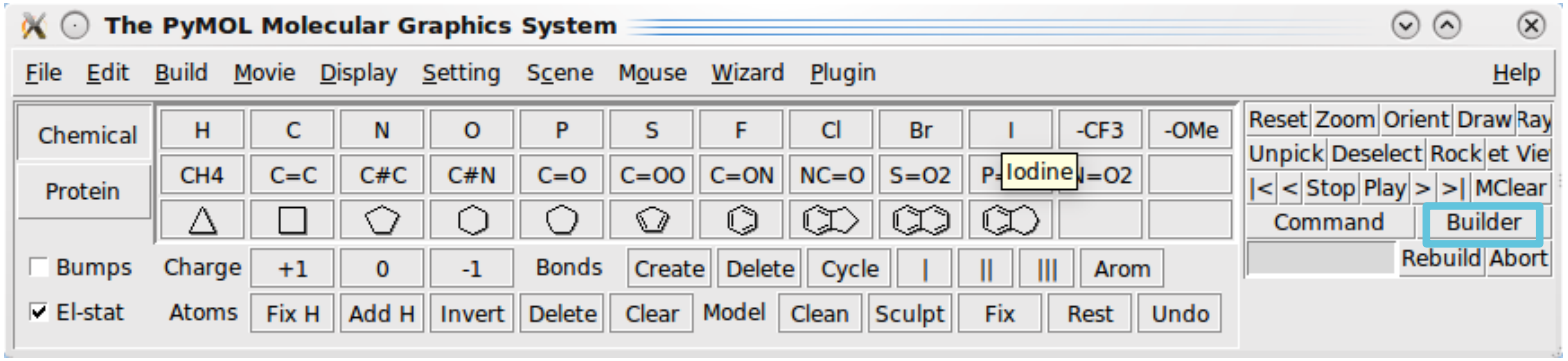
Mutation V600E de B-Raf (structure 3D4Q)

Mutagenèse : résultat



Construction de molécules (*builder*)

Le logiciel PyMOL permet de construire des molécules et intègre un outil de minimisation d'énergie. L'interface a été améliorée avec la version 1.2r2 :



De nombreuses options dépendent de Freemol :

`http://freemol.org`

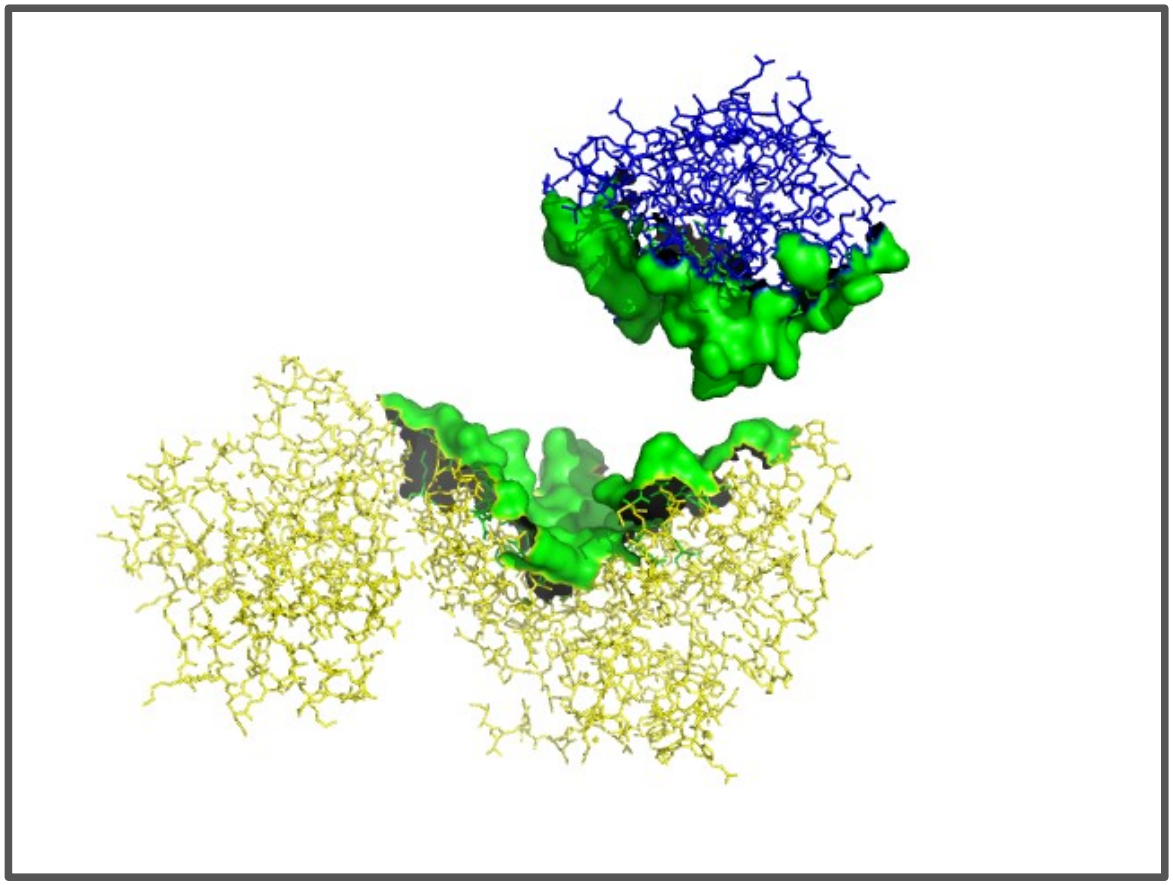
Déplacement d'objets

Déplacement des objets

Les objets peuvent être déplacés avec la commande **translate** :

```
load 1BKD.pdb
create ras =(chain R)
create sos=(chain S)
delete 1BKD
color blue, ras
color yellow, sos
select inter = (byres ((ras within 5 of sos)\
  or (sos within 5 of ras)))
color green, inter
bg_color white
orient
origin position=[0,0,0]
translate [10,10,10], ras
translate [-10,-10,-10], sos
show surface, inter
disable inter
```

Déplacement des objets : résultat

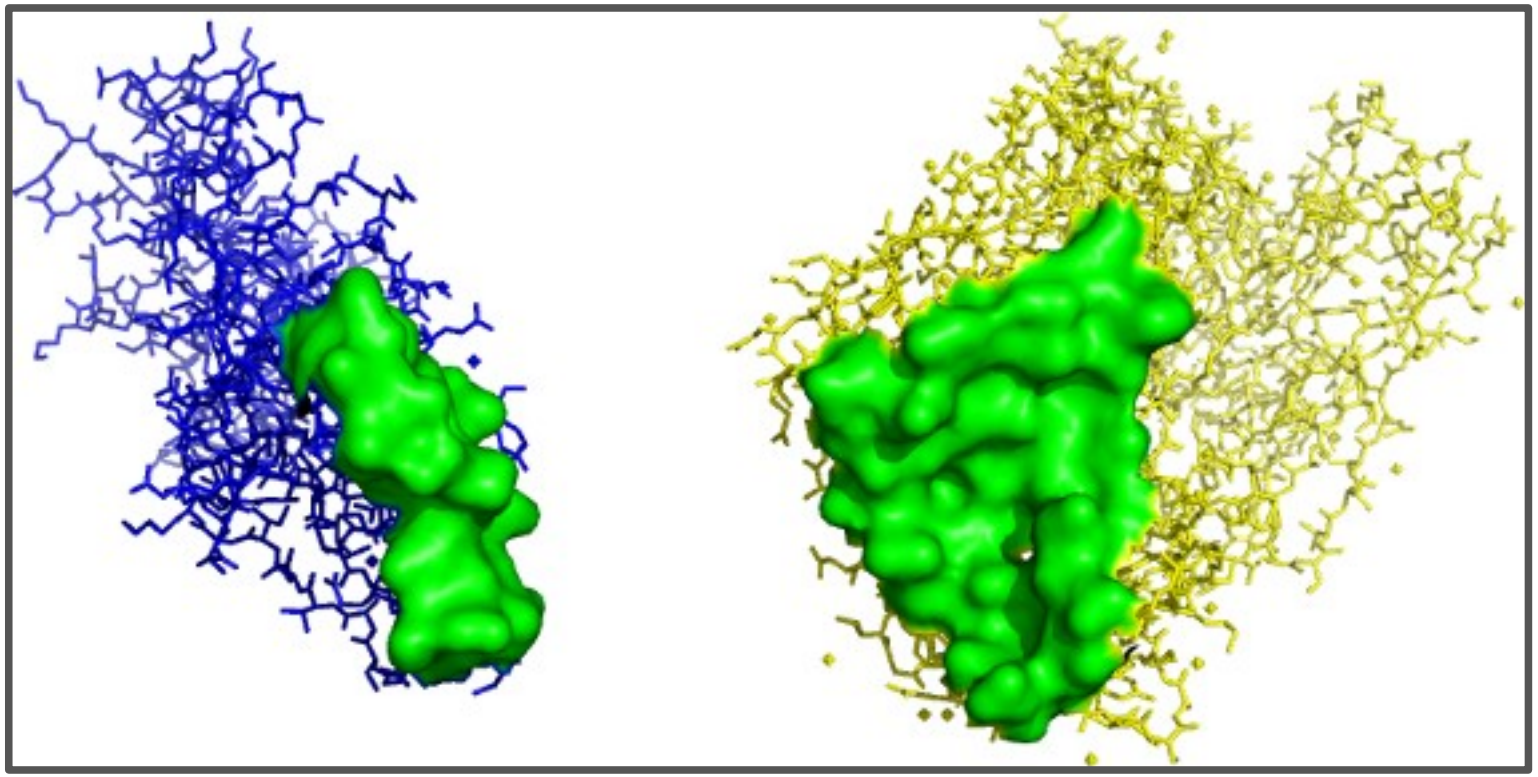


Déplacement des objets

Les objets peuvent également être déplacés avec la commande **rotate** :

```
load 1FJ1.pdb
create anti=(chain F)
create fab=(chain A,B)
delete 1FJ1
color yellow,fab
color blue,anti
bg_color white
select inter = (byres ((fab within 5 of anti)\
   or (anti within 5 of fab)))
color green,inter
orient
origin fab
rotate y,60,fab
origin anti
rotate y,-60, anti
zoom
show surface, inter
disable inter
```

Déplacement des objets : résultat



Cristallographie

Symexp

La fonction **symexp** reconstruit le voisinage cristallin d'une unité asymétrique à partir des données de l'expérience cristallographique qui a permis d'obtenir la structure. Cette fonction implique l'utilisation d'un fichier PDB ou équivalent comportant suffisamment d'information pour reproduire le réseau cristallin.

Syntaxe :

```
symexp nom_nouvel_objet,nom_objet_asym.,(nom_objet_asym.),distance
```

Voir aussi :

`http://www.pymolwiki.org/index.php/SuperSym`

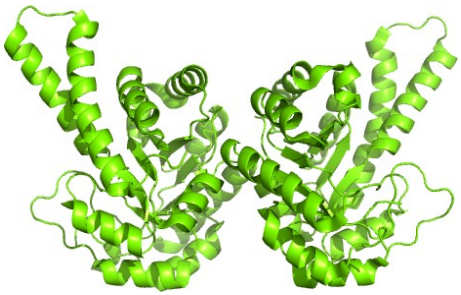
Symexp

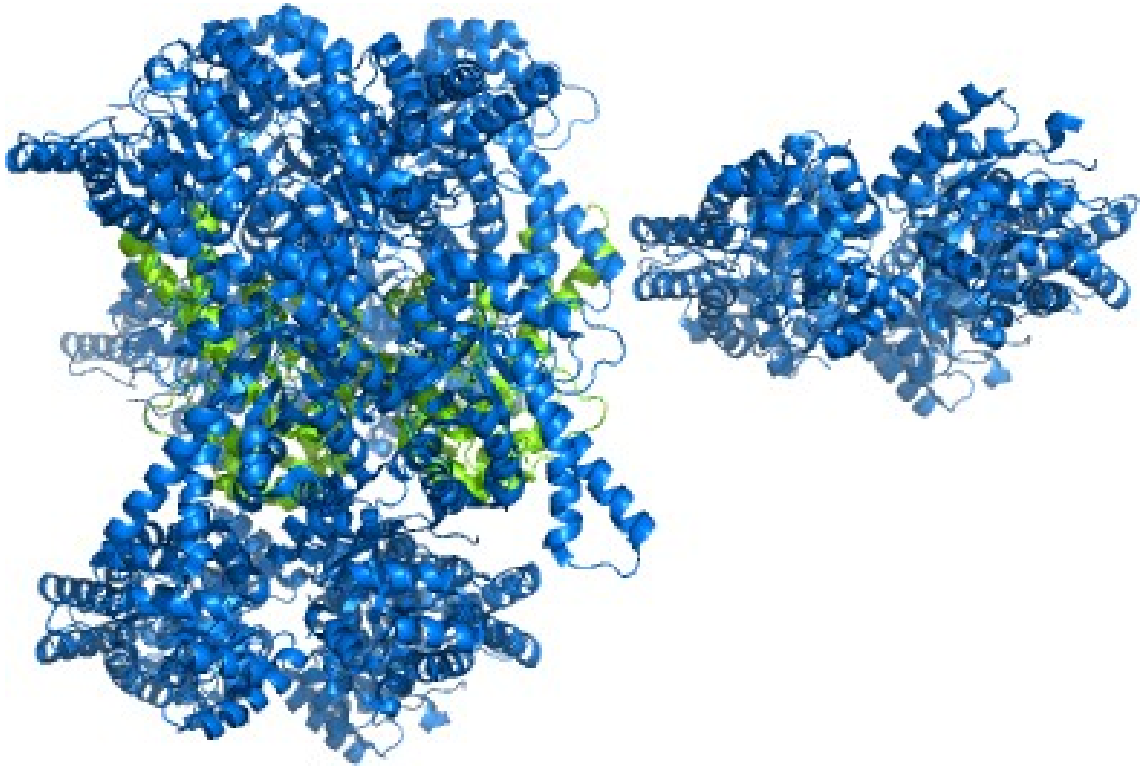
Exemple :

1. Charger le fichier `1GVF.pdb`
2. Utiliser la fonction ci-dessous pour générer une maille complémentaire :

```
symexp sym,1GVF,(1GVF),5
```

3. Faire varier la distance





Unité biologique

Le module `biological_unit.py` permet de reconstruire une unité biologique. Cette fonction implique que le fichier PDB contiennent les champs `BIOMT` décrivant l'unité biologique.

Voir aussi :

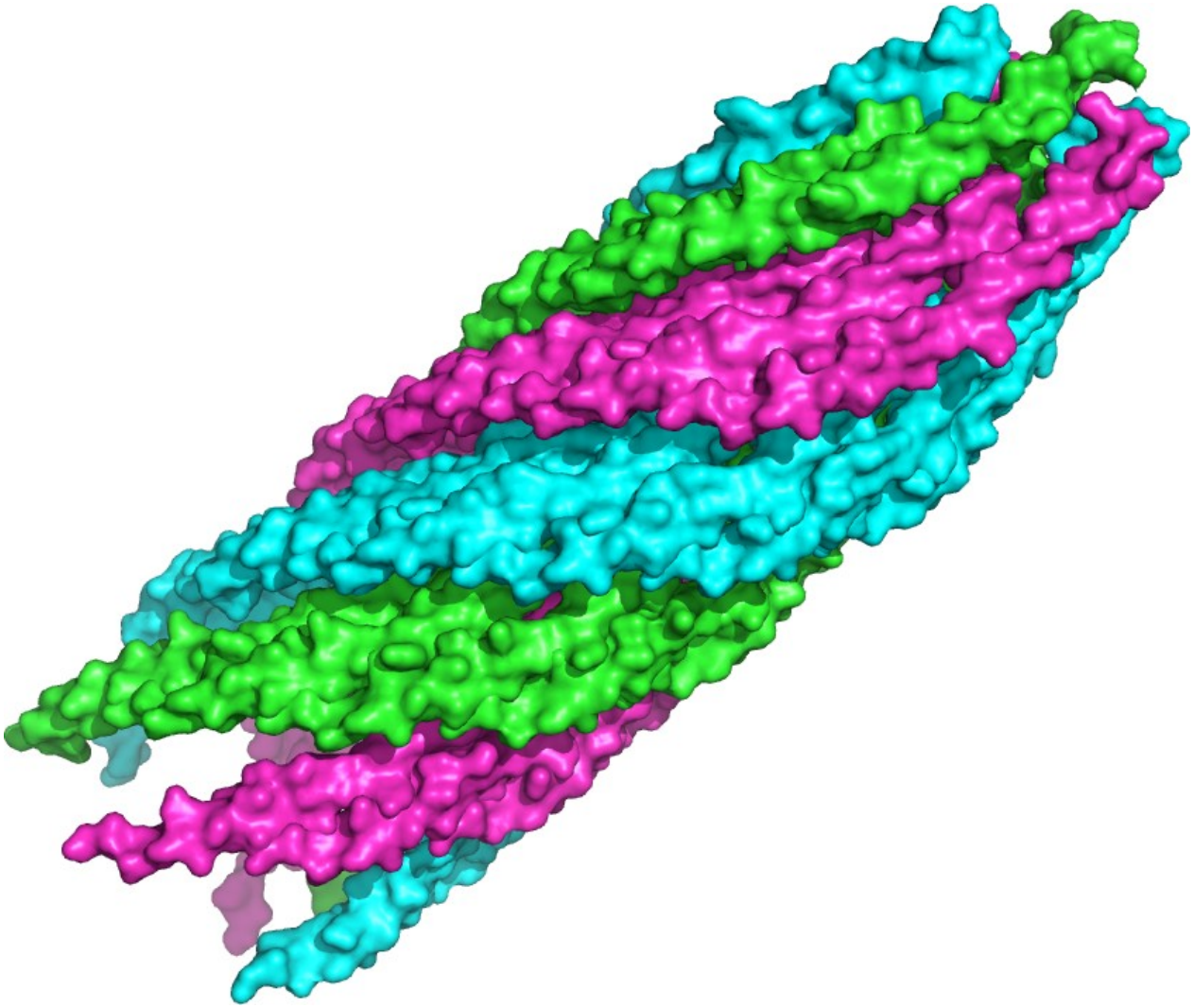
`http://www.pymolwiki.org/index.php/BiologicalUnit`

Exemple :

1. Charger le script `biologicalUnit.py`
2. Utiliser les fonctions de ce script :

```
PyMOL> load 1QL2.pdb
PyMOL> symMat = readSymmetry("1QL2.pdb","1QL2")
PyMOL> biologicalUnit("helice", "1QL2", symMat)
```

Unité biologique



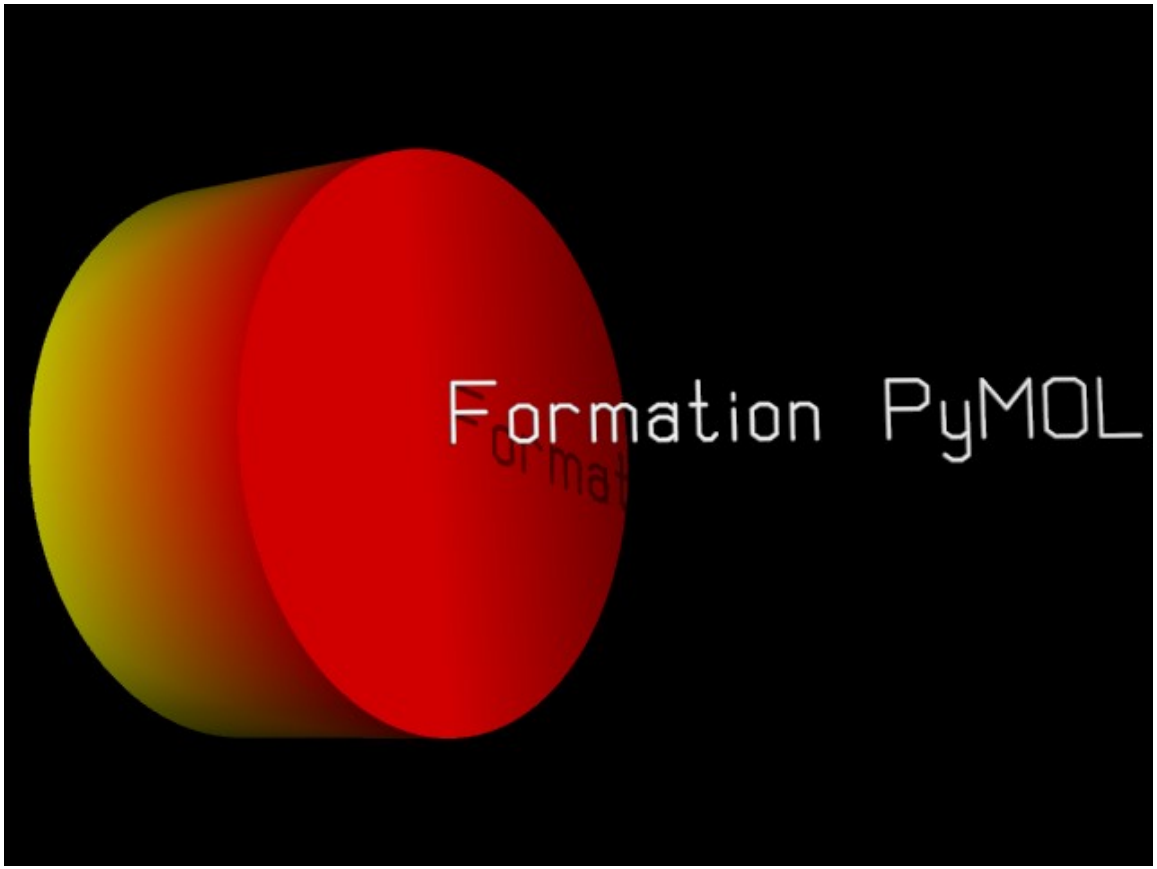
Compiled Graphic Objects

Compiled Graphic Objects

```
# Script cgo.py
from pymol import cmd
from pymol.cgo import *
from pymol.vfont import plain

x1,y1,z1 = 10, 0, 0 # debut du cylindre
r1,g1,b1 = 1,0,0 # couleur (red)
x2,y2,z2 = 0.1, 0, 0 # find du cylindre
r2,g2,b2 = 1,1,0 # couleur (yellow)
radius = 10
cmd.load_cgo( [ 9.0, x1, y1, z1, x2, y2, z2, radius, r1, g1, b1,\
 r2, g2, b2 ], "cylinder1" )
cgo = []
axes = [[2.0,0.0,-2.0],[0.0,2.0,0.0],[0.0,0.0,2.0]]
pos = [10.1,0.0,0.0]
wire_text(cgo,plain,pos,'Formation PyMOL',axes)
cmd.set("cgo_line_radius",0.1)
cmd.load_cgo(cgo,'txt')
cmd.zoom("all",2.0)
cmd.ray()
```

Compiled Graphic Objects : résultat



Pour aller plus loin :

http://pymol.sourceforge.net/newman/user/S0500cgo.html#14
http://www.liv.ac.uk/~afcalder/apt3.html

bbPlane

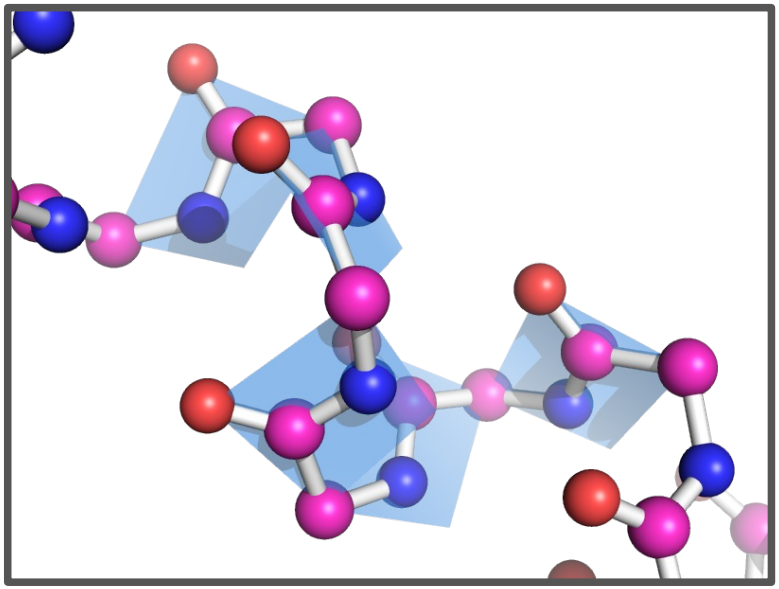
La fonction **bbPlane** permet de dessiner les plans formés par deux acides aminés consécutifs. Elle permet de représenter la planéité des atomes du squelette protéique.

Syntaxe :

```
bbPlane selection,couleur,transparence
```

Voir :

`http://pymolwiki.org/index.php/BbPlane`



Ressources complémentaires

Ressources complémentaires

Général :

http://www.pymol.org

http://www.pymolwiki.org

Scripts :

http://www.pymolwiki.org/index.php/Category:Script_Library

http://pldserver1.biochem.queensu.ca/~rlc/work/pymol/

Astuces :

http://www-cryst.bioc.cam.ac.uk/members/zbyszek/figures_pymol