บทปฏิบัติการ: Try to Dock

(อ้างอิงจาก 2020 DOCK tutorial 1 with PDBID 3VJK

http://ringo.ams.stonybrook.edu/index.php/2020_DOCK_tutorial_1_with_PDBID_3VJK)

สำหรับบทปฏิบัติการนี้จะกล่าวถึงเทคนิคการทำนายการจับกันระหว่างโปรตีน-ลิแกนด์ (protein-ligand docking) โดยใช้วิธีโมเลกุล่าร์ ด็อกกิ้ง (molecular docking) เพื่อคำนวณค่าพลังงานระหว่างโครงสร้างสามมิติของโปรตีนและลิแกนด์ โดยใช้โปรแกรม DOCK 6 (เป็น Command line interface) ควบคู่กับ Chimera (เป็น Graphic User Interface) ซึ่งรองรับทั้งระบบปฏิบัติการ Windows Linux และ OS X สำหรับโปรแกรม DOCK เริ่มต้นการพัฒนาโดย Dr. Irwin Kuntz และทีมงาน จาก University of California San Francisco โดยการค้นหาการวางตัวของลิแกนด์ใช้ search algorithm เป็น anchor-and-grow

สำหรับชุดการสาธิตนี้ใช้โปรตีน 3C-like serine protease (3CLpro) ของไวรัส severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) ที่มีลิแกนด์ N3 ในโครงสร้าง X-ray (PDB: 6LU7)

ขั้นตอน 1 – การจัดการแฟ้มงาน

ให้ผู้ใช้งานสร้างโฟลเดอร์ 6LU7 และ 001.structure 002.surface_spheres 003.gridbox 004.dock 005.virtual_screen 006.virtual_screen_mpi 007.cartesianmin 008.rescore ภายในโฟลเดอร์ 6LU7

```
$ mkdir 6LU7
$ cd 6LU7
$ mkdir 001.structure 002.surface_spheres 003.gridbox 004.energy_min 005.dock 006.footprint 007.virtual_screen
```

ขั้นตอน 2 – การปรับปรุงรีเซปเตอร์ (ตัวรับ)

ให้ผู้ใช้งานเปิดโปรแกรม Chimera จากนั้นไปที่

```
File > Open > ตำแหน่งไฟล์ที่เก็บไฟล์ 6lu7.pdb
```

ในกรณีที่มีสายโซ่ (chain) ที่ไม่เกี่ยวข้องกับการจับกันระหว่างตัวรับและลิแกนด์ ให้ทำการลบ แต่เนื่องไฟล์ 6lu7 นี้ ไม่มีสายโซ่ที่ไม่เกี่ยวข้อง จึงไม่มีการลบสายโซ่ใด ๆ ในโครงสร้าง แต่อย่างไรก็ตาม ลิแกนด์ต้องทำการลบ โดย

Select > Chain > C Actions > Atoms/Bonds > Delete

จากนั้นให้ทำการบันทึกไฟล์ที่โฟลเดอร์ 001.structure เป็น mol2 โดย

File > Save Mol2 > “6lu7_rec_woH.mol2”

woH หมายถึง ยังไม่มีการเติมไฮโดรเจนในโครงสร้าง

จากนั้นให้การเติมอะตอมไฮโดรเจนและประจุ โดย

Tools > Structure Editing > Add H > ok

จากนั้นทำการเติม partial charge ลงไปในแต่ละอะตอมบนตัวรับ

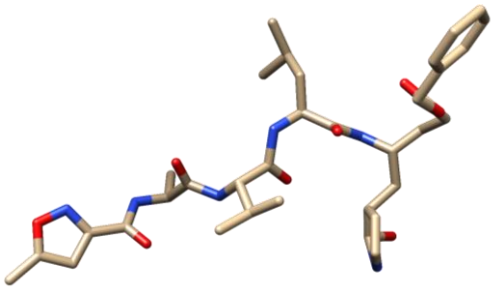
Tools > Structure Editing -> Add Charge -> (AM1BCC charges should be selected) -> Ok

ทำการบันทึกเป็นไฟล์ mol2 ชื่อ 6lu7_rec_dockprep.mol2 ที่โฟลเดอร์ 001.structure

ขั้นตอน 3 – การปรับปรุงลิแกนด์

จากนั้นให้ผู้ใช้งานเข้าโปรแกรมใหม่อีกครั้ง และเปิดไฟล์ 6lu7 จากนั้น ให้ผู้ใช้ทำการเลือกเฉพาะลิแกนด์ โดย

Select > Chain > C Select > Invert (selected model) Actions->Atoms/Bonds->Delete



จากนั้นทำการบันทึกไฟล์ที่โฟลเดอร์ 001.structure ที่โฟลเดอร์ 001.structure

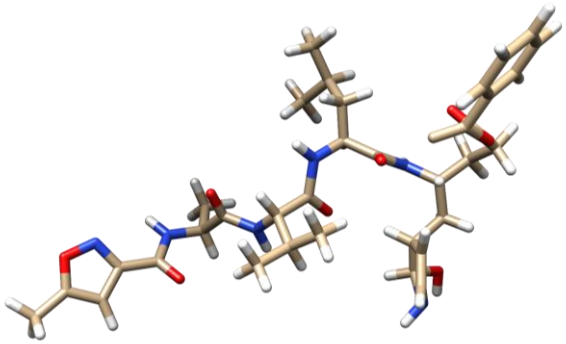
File > save as mol2 > 6lu7_ligand_noH.mol2

จากนั้นทำการเติมอะตอมไฮโดรเจนและประจุให้กับลิแกนด์ โดย

Tools > Structure Editing > Add H > ok

จากนั้นทำการเติม partial charge ลงไปในแต่ละอะตอมของลิแกนด์

Tools > Structure Editing -> Add Charge -> (AM1BCC charges should be selected) -> Ok



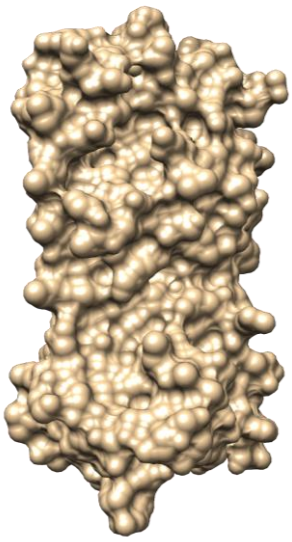
ทำการบันทึกเป็นไฟล์ mol2 ชื่อ 6lu7_ligand_wH.mol2 ที่โฟลเดอร์ 001.structure

ขั้นตอน 4 – การสร้างพื้นผิวและขอบเขต

ในขั้นตอนนี้จะเป็นการสร้างพื้นผิวให้กับโปรตีน เพื่อใช้สำหรับเป็นพื้นที่ในการทำโมเลกุล่าร์ ด็อกกิ้ง โดยเป็นการสร้าง van der Waals surface ของโมเลกุล

ทำการเปิดไฟล์ 6LU7_rec_woH.mol2 จากนั้นไปที่

Actions > Surface > Show
Tools > Structure Editing -> Write DMS -> "6LU7_rec_surface.dms" (บันทึกที่โฟลเดอร์ 002.surface_spheres)



จากนั้นจะเป็นขั้นตอนการสร้างขอบเขตสำหรับการด็อกกิ้ง โดยขั้นตอนนี้จะเป็นการสร้างพื้นที่ว่างข้างในโปรตีน ซึ่งเป็นบริเวณที่คาดว่าจะเกิดการจับกับลิแกนด์ สำหรับการสร้างนั้นจะใช้ sphgen ในการสร้าง ซึ่งเป็นแพ็คเกจในโปรแกรม DOCK 6 (อยู่ที่ตำแหน่ง dock6/bin/sphgen) แต่ก่อนอื่นให้ทำการสร้างสคริปเพื่อสร้างขอบเขตดังกล่าวก่อน โดยใช้ชื่อ INSPH

$ cd 002.surface_spheres
$ vi INSPH

ให้ทำการใส่ข้อมูลนี้

```
6LU7_rec_surface.dms
R
X
0.0
4.0
1.4
6LU7_receptor_woH.sph
```

โดยแต่ละบรรทัดมีความหมาย ดังนี้

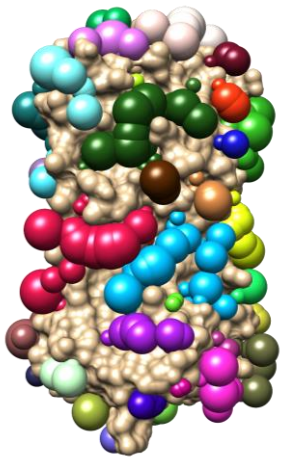
[your_receptor].dms - molecular surface file that we got from previous step
<flag> - sphere outside of surface (R) or inside surface (L)
<flag> - specifies subset of surface points to be used (X = all points)
<double> - prevents generation of large spheres with close surface contacts (default = 0.0)
<double> - maximum sphere radius in angstroms (default = 4.0 Angstroms)
<double> - minimum sphere radius in angstroms (default = radius of probe = 1.4 Angstroms = Size of a water molecule)
[your_receptor].sph - clustered spheres file that we want to generate

จากนั้นทำการสร้างขอบเขต โดยใช้คำสั่ง

```
$ sphgen -i INSPH -o OUTSPH
```

-i is the flag that give sphgen the input file **INSPH**
INSPH is the file created above that gives sphgen instructions
-o is the flag to create the oputput file
OUTSPH is the output file with the information of the spheres generated from sphgen

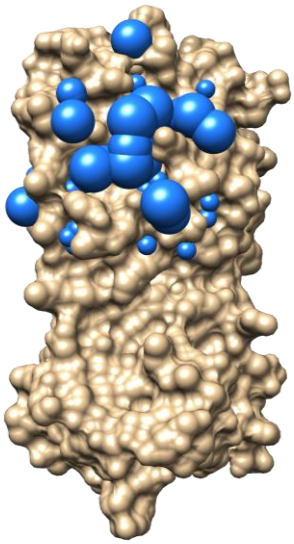
จากนั้น สังเกตขอบเขตที่สร้างขึ้น โดยเปิดไฟล์ 6LU7_receptor_woH.sph ควบคู่กับ 6lu7_rec_woH.mol2



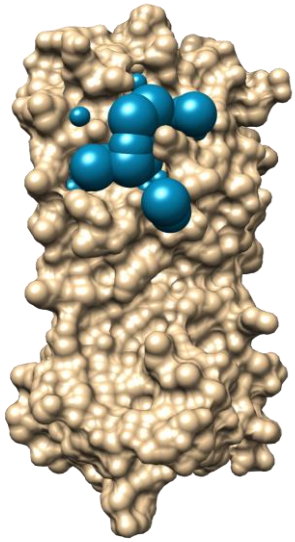
จากนั้น ทำการสร้างขอบเขตที่เป็นส่วนที่อยู่ของลิแกนด์ โดยใช้คำสั่ง sphere_selector

```
sphere_selector 6LU7_receptor_woH.sph 6lu7_ligand_wH.mol2 10.0
```

จะสังเกตได้ว่ามีไฟล์ selected_spheres.sph สร้างเพิ่มขึ้นมา ให้ผู้ใช้ทำการเปิดไฟล์ดังกล่าว ควบคู่กับ 6lu7_rec_woH.mol2 โดยคำสั่งดังกล่าวผู้ใช้งานสามารถเปลี่ยนขอบเขตที่ถูกเลือกได้จาก 10 Angstroms เป็น 8 หรือ 6 Angstroms ที่สามารถครอบคลุมขอบเขตที่เป็นที่อยู่ของลิแกนด์ ซึ่งในที่นี้ผู้ใช้งานเลือก 6 Angstroms



selected_spheres.sph 10 Angstroms



selected_spheres.sph 6 Angstroms

ขั้นตอน 5 – การสร้างกล่องและกริด

เนื่องจากการคำนวณค่าพลังงานเป็นการคำนวณที่มีราคาแพง เพื่อเป็นการลดราคาดังกล่าว DOCK 6 คำนวณค่าพลังงานโดยใช้กริด ซึ่งจะไม่พิจารณาอันตรกิริยากับลิแกนด์ที่อยู่ไกล

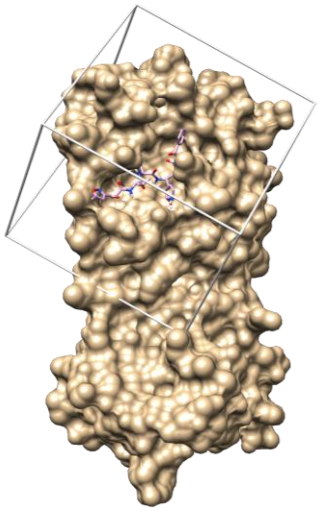
```
$ cd 003.gridbox
$ vi showbox.in
```

ให้ทำการใส่ข้อมูลนี้

```
Y                                          #generate box
8.0                                        #how many angstroms the box edges should be from the spheres
../002.surface_spheres/selected_spheres.sph #the location of the selected spheres
1                                          # the cluster number in the selected_spheres.sph file
6LU7.box.pdb                               #name of the output file
```

จากนั้นทำการรัน

```
$ showbox < showbox.in
```



จากนั้นจะทำการสร้างกริด โดยให้ทำการสร้างไฟล์ grid.in

```
$ vi grid.in
```

ให้ทำการใส่ข้อมูลนี้

compute_grids	yes
grid_spacing	0.4
output_molecule	no
contact_score	no
energy_score	yes
energy_cutoff_distance	9999
atom_model	a
attractive_exponent	6
repulsive_exponent	9
distance_dielectric	yes
dielectric_factor	4
bump_filter	yes
bump_overlap	0.75
receptor_file	../001.structure/6lu7_rec_dockprep.mol2
box_file	6LU7.box.pdb
vdw_definition_file	/dock6/parameters/vdw_AMBER_parm99.defn
score_grid_prefix	grid

Line by line:

1. compute scoring grids (yes)
2. what is the distance between grid points along each axis (in Angstroms).
3. write up coordinates of the receptor into a new file
4. compute contact grid? default is no
5. compute energy score? yes - we are using this method to compute force fields on probes
6. the max distance between atoms for the energy contribution to be computed
7. atom_model **u** means united atom model where atoms are attached to hydrogens, and **a** stands for all-atom model, where hydrogens on carbons are treated separately
8. attractive component stands for exponent of the attractive LJ term in VDW potential
9. repulsive component stands for exponent in the repulsive LJ term in VDW potential
10. distance dielectric stands for the dielectric constant to be linearly dependent on distance
11. distance dielectric factor is the coefficient of the dielectric
12. bump filter flag determines if we want to screen orientation for clashes before scoring and minimization
13. bump_overlap stands for the fraction of allowed overlap where 1 corresponds to no allowed overlap and 0 corresponds to full overlap being permitted.
14. our receptor file
15. the box file we generated in the Box section
16. VDW parameters file
17. Prefix for the grid file name. All the extensions will be generated automatically.

จากนั้นทำการรัน

```
$ grid -i grid.in -o gridinfo.out
```

เมื่อทำเสร็จสิ้น ผู้ใช้สามารถดูผลลัพธ์จากการประมวลผลได้จากไฟล์ gridinfo.out ซึ่งจะเห็นได้ว่า

Total charge on 6lu7.pdb	:	-4.000
Box center of mass	:	-10.265 15.071 67.562
Box dimensions	:	28.131 33.636 33.602
Number of grid points per side [x y z]	:	72 86 86
Total number of grid points	:	532512

โดยในการคำนวณพลังงานสำหรับขั้นกริดนี้ สามารถคำนวณได้จากสมการ ดังต่อไปนี้

The energy scoring function:

$$E = \sum_{i=1}^{lig} \left[\sqrt{A_{ii}} \sum_{j=1}^{rec} \frac{\sqrt{A_{jj}}}{r_{ij}^{a}} - \sqrt{B_{ii}} \sum_{j=1}^{rec} \frac{\sqrt{B_{jj}}}{r_{ij}^{b}} + 332.0 * q_i \sum_{j=1}^{rec} \frac{q_j}{Dr_{ij}} \right]$$

ขั้นตอน 6 – การทำ Energy minimization

ขั้นตอนนี้จะเป็นการ minimize ของลิแกนด์ เนื่องจากลิแกนด์ยังไม่ได้อยู่ในสภาวะที่พลังงานน้อยที่สุด (พลังงานน้อยสุด นั่นหมายถึง สภาวะที่เสถียรที่สุด)

ขั้นแรกจะทำการสร้างพารามิเตอร์สำหรับการทำด็อกกิ้ง โดย

```
$ cd 004.energy_min
$ vi min.in
```

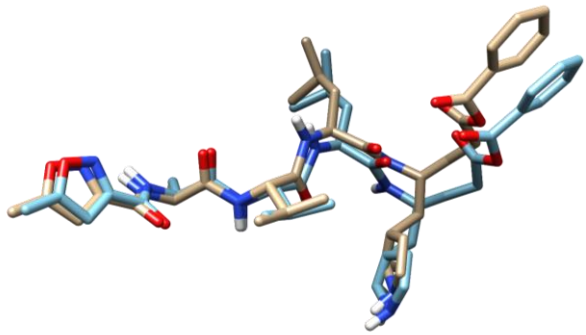
ให้ทำการใส่ข้อมูลนี้

conformer_search_type	rigid
use_internal_energy	yes
internal_energy_rep_exp	12
internal_energy_cutoff	100.0
ligand_atom_file	../001.structure/6lu7_ligand_wH.mol2
limit_max_ligands	no
skip_molecule	no
read_mol_solvation	no
calculate_rmsd	yes
use_rmsd_reference_mol	yes
rmsd_reference_filename	../001.structure/6lu7_ligand_wH.mol2
use_database_filter	no
orient_ligand	no
bump_filter	no
score_molecules	yes
contact_score_primary	no
contact_score_secondary	no
grid_score_primary	yes
grid_score_secondary	no
grid_score_rep_rad_scale	1
grid_score_vdw_scale	1
grid_score_es_scale	1
grid_score_grid_prefix	../003.gridbox/grid
multigrid_score_secondary	no
dock3.5_score_secondary	no
continuous_score_secondary	no
footprint_similarity_score_secondary	no
pharmacophore_score_secondary	no
descriptor_score_secondary	no

gbsa_zou_score_secondary	no
gbsa_hawkins_score_secondary	no
SASA_score_secondary	no
amber_score_secondary	no
minimize_ligand	yes
simplex_max_iterations	1000
simplex_tors_premin_iterations	0
simplex_max_cycles	1
simplex_score_converge	0.1
simplex_cycle_converge	1.0
simplex_trans_step	1.0
simplex_rot_step	0.1
simplex_tors_step	10.0
simplex_random_seed	0
simplex_restraint_min	yes
simplex_coefficient_restraint	10.0
atom_model	all
vdw_defn_file	/dock6/parameters/vdw_AMBER_parm99.defn
flex_defn_file	/dock6/parameters/flex.defn
flex_drive_file	/dock6/parameters/flex_drive.tbl
ligand_outfile_prefix	6LU7.lig.min
write_orientations	no
num_scored_conformers	1
rank_ligands	no

จากนั้นทำการรัน

```
$ dock6 -i min.in -o min.out
```



ลิแกนด์ที่ไม่ได้ minimization (สีฟ้า) กับ ลิแกนด์ที่ minimization (สีน้ำตาล)

ขั้นตอน 7 – การทำด็อกกิ้งตัวเอง (Self-docking) แบบ Flexible docking

ในขั้นตอนนี้จะเป็นการทำโมเลกุล่าร์ ด็อกกิ้ง แบบ Flexible ซึ่งมีราคาการคำนวณที่สูง โดยในการคำนวณลิแกนด์จะมีองศาการหมุนที่เป็นอิสระ

ขั้นแรกจะทำการสร้างพารามิเตอร์สำหรับการทำด็อกกิ้ง โดย

```
$ cd 005.dock
$ vi flex.in
```

ให้ทำการใส่ข้อมูลนี้

conformer_search_type	flex
write_fragment_libraries	no
user_specified_anchor	no
limit_max_anchors	no
min_anchor_size	5
pruning_use_clustering	yes
pruning_max_orients	1000
pruning_clustering_cutoff	100
pruning_conformer_score_cutoff	100.0
pruning_conformer_score_scaling_factor	1.0
use_clash_overlap	no
write_growth_tree	no
use_internal_energy	yes
internal_energy_rep_exp	12
internal_energy_cutoff	100.0
ligand_atom_file	../004.energy_min/6LU7.lig.min_scored.mol2
limit_max_ligands	no
skip_molecule	no
read_mol_solvation	no
calculate_rmsd	yes
use_rmsd_reference_mol	yes
rmsd_reference_filename	../004.energy_min/6LU7.lig.min_scored.mol2
use_database_filter	no
orient_ligand	yes
automated_matching	yes
receptor_site_file	../002.surface_spheres/selected_spheres.sph
max_orientations	1000
critical_points	no
chemical_matching	no

use_ligand_spheres	no
bump_filter	no
score_molecules	yes
contact_score_primary	no
contact_score_secondary	no
grid_score_primary	yes
grid_score_secondary	no
grid_score_rep_rad_scale	1
grid_score_vdw_scale	1
grid_score_es_scale	1
grid_score_grid_prefix	../003.gridbox/grid
multigrid_score_secondary	no
dock3.5_score_secondary	no
continuous_score_secondary	no
footprint_similarity_score_secondary	no
pharmacophore_score_secondary	no
descriptor_score_secondary	no
gbsa_zou_score_secondary	no
gbsa_hawkins_score_secondary	no
SASA_score_secondary	no
amber_score_secondary	no
minimize_ligand	yes
minimize_anchor	yes
minimize_flexible_growth	yes
use_advanced_simplex_parameters	no
simplex_max_cycles	1
simplex_score_converge	0.1
simplex_cycle_converge	1.0
simplex_trans_step	1.0
simplex_rot_step	0.1
simplex_tors_step	10.0
simplex_anchor_max_iterations	500
simplex_grow_max_iterations	500
simplex_grow_tors_premin_iterations	0
simplex_random_seed	0
simplex_restraint_min	no
atom_model	all
vdw_defn_file	/dock6/parameters/vdw_AMBER_parm99.defn
flex_defn_file	/dock6/parameters/flex.defn
flex_drive_file	/dock6/parameters/flex_drive.tbl

ligand_outfile_prefix	flex.out
write_orientations	no
num_scored_conformers	1
rank_ligands	no

จากนั้นทำการรัน

```
$ dock6 -i flex.in -o flex.out
```

เมื่อทำการรันเสร็จสิ้น สามารถดูผลลัพธ์ผ่านโปรแกรม Chimera โดยใช้ ViewDock

```
File > Open > 6lu7_rec_dockprep.mol2
File > Open > 6lu7_ligand_wH.mol2
Tools > Surface/binding Analysis > ViewDock > Select the Flex Dock output file. (flex.out_scored.mol2)
In the loaded dialog box select Dock 4, 5 or 6
```

จากนั้นทำการปรับเปลี่ยนการแสดงผลเพื่อให้โชว์คอลัมภ์ gridscore และ HA_RMSDs

```
Column > Show > Grid_Score
Column > Show > HA_RMSDs
```

(1) Standard heavy-atom RMSD (HA_RMSDs): This is the standard pair-wise RMSD calculation between the non-hydrogen atoms of a reference conformation *a* and a pose conformation *b* for a ligand with *N* total heavy atoms of index *i*

$$HA\_RMSDs = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\|a_i - b_i\|^2}$$

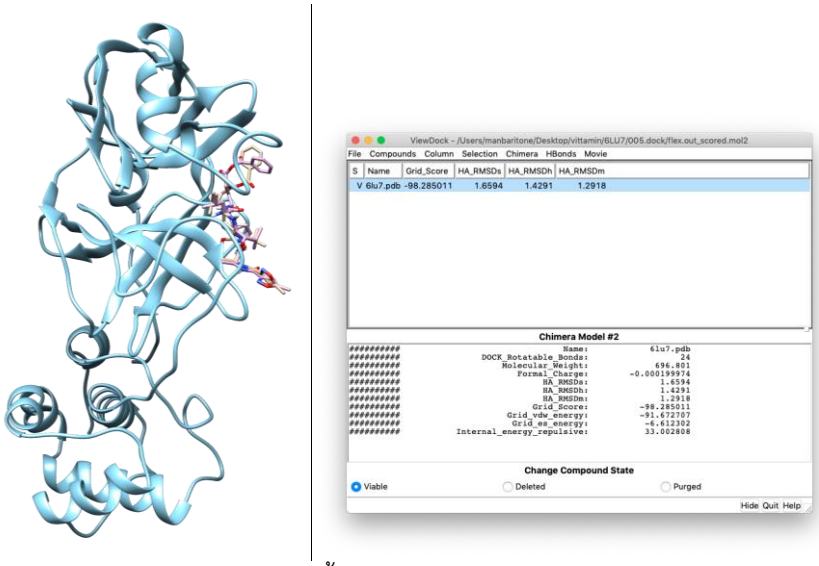
(2) Minimum-distance heavy-atom RMSD (HA_RMSDm): This measure is based on the RMSD implementation used in Autodock Vina (Trott and Olson, *J. Comput. Chem.* 2010), which does not explicitly enforce one-to-one mapping. Rather, atom pairings between reference conformation *a* and pose conformation *b* are determined by the minimum distance to any atom of the same element type, and it may be an under-prediction of the true RMSD.

$$RMSD_{\min}(A,B) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\min_j \|a_i - b_j\|\right)^2}$$
$$HA\_RMSDm = \max\left(RMSD_{\min}(A,B), RMSD_{\min}(B,A)\right)$$

(3) Hungarian (symmetry-corrected) heavy-atom RMSD (HA_RMSDh): The final RMSD implementation is based on an *O(N^4)* implementation of the *Hungarian algorithm* (Kuhn, *Nav. Res. Logist. Q.* 1955; Munkres, *J. Soc. Indust. Appl. Math.* 1957). The algorithm solves the optimal assignment between a set of reference ligand atoms *a* and a set of pose ligand atoms *b* of the same size. For all groups of atoms of the same Sybyl atom type, a cost matrix *M* is populated where each matrix element *mij* is equal to the distance-squared between reference atom *ai* and pose atom *bj*. The Hungarian algorithm is used to determine one-to-one assignments between reference and pose ligand atoms such that the total distance between atoms is minimized. The new assignments *c(i)* are fed into the standard

RMSD function in order to compute a symmetry-corrected RMSD. If the HA_RMSDh is "-1000.0", then there is an inconsistency in the number of atoms of at least one atom type between the reference and the docked conformer.

$$HA\_RMSDh = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left\|a_i - b_{c(i)}\right\|^2}$$



ลิแกนด์จากโครงสร้าง X-ray (สีน้ำตาล) โครงสร้างจากการทำ Self-docking (สีชมพู)

ขั้นตอน 8 – การทำ Molecular Footprint

สำหรับขั้นตอนนี้จะเป็นการวิเคราะห์อันตรกิริยาที่เกิดขึ้นระหว่างตัวรับและลิแกนด์ โดยจะพิจารณาถึง electrostatic interactions and Van der Waals interactions ก่อนและหลังการทำ minimization

ขั้นแรกจะทำการสร้างพารามิเตอร์สำหรับการทำ molecular footprint โดย

```
$ cd 006.footprint
$ vi footprint.in
```

ให้ทำการใส่ข้อมูลนี้

conformer_search_type	rigid
use_internal_energy	no
ligand_atom_file	../004.energy_min/6LU7.lig.min_scored.mol2
limit_max_ligands	no
skip_molecule	no
read_mol_solvation	no
calculate_rmsd	no
use_database_filter	no
orient_ligand	no
bump_filter	no

score_molecules	yes
contact_score_primary	no
contact_score_secondary	no
grid_score_primary	no
grid_score_secondary	no
multigrid_score_primary	no
multigrid_score_secondary	no
dock3.5_score_primary	no
dock3.5_score_secondary	no
continuous_score_primary	no
continuous_score_secondary	no
footprint_similarity_score_primary	yes
footprint_similarity_score_secondary	no
fps_score_use_footprint_reference_mol2	yes
fps_score_footprint_reference_mol2_filename	../001.structure/6lu7_ligand_wH.mol2
fps_score_foot_compare_type	Euclidean
fps_score_normalize_foot	no
fps_score_foot_comp_all_residue	yes
fps_score_receptor_filename	../001.structure/6lu7_rec_dockprep.mol2
fps_score_vdw_att_exp	6
fps_score_vdw_rep_exp	12
fps_score_vdw_rep_rad_scale	1
fps_score_use_distance_dependent_dielectric	yes
fps_score_dielectric	4.0
fps_score_vdw_fp_scale	1
fps_score_es_fp_scale	1
fps_score_hb_fp_scale	0
pharmacophore_score_secondary	no
descriptor_score_secondary	no
gbsa_zou_score_secondary	no
gbsa_hawkins_score_secondary	no
SASA_score_secondary	no
amber_score_secondary	no
minimize_ligand	no
atom_model	all
vdw_defn_file	/dock6/parameters/vdw_AMBER_parm99.defn
flex_defn_file	/dock6/parameters/flex.defn
flex_drive_file	/dock6/parameters/flex_drive.tbl
ligand_outfile_prefix	6lu7_footprint_min_cryst
write_footprints	yes

write_hbonds	yes
write_orientations	no
num_scored_conformers	1
rank_ligands	no

จากนั้นทำการรัน

```
$ dock6 -i footprint.in
```

เมื่อรันเสร็จจะปรากฏไฟล์ผลลัพธ์จำนวน 3 ไฟล์ ได้แก่

1. 6lu7_footprint_min_cryst_footprint_scored.txt
2. 6lu7_footprint_min_cryst_hbond_scored.txt
3. 6lu7_footprint_min_cryst_scored.mol2

ซึ่งสามารถนำผลลัพธ์ดังกล่าวไปใช้ในการวิเคราะห์ Key residue ที่สำคัญต่อการจับระหว่างโปรตีนตัวรับกับลิแกนด์

ขั้นตอน 9 – การคัดกรองเสมือนจริง (Virtual Screening)

สำหรับการคัดกรองเสมือนจริง เราจะใช้ลิแกนด์ที่เตรียมได้จากการค้นหาผ่าน similarity จาก Google Colab ผ่านฐานข้อมูล ChEMBL (ไฟล์ cov_compounds.mol2) ซึ่งมีลิแกนด์จำนวน 64 ลิแกนด์

โดยขั้นแรกให้ผู้ใช้นำไฟล์ cov_compounds.mol2 ไปไว้ที่โฟลเดอร์ 006.virtual_screen จากนั้นทำการสร้างพารามิเตอร์สำหรับการทำด็อกกิ้ง

```
$ cd 007.virtual_screen
$ vi virtual.in
```

ให้ทำการใส่ข้อมูลนี้

conformer_search_type	flex
write_fragment_libraries	no
user_specified_anchor	no
limit_max_anchors	no
min_anchor_size	5
pruning_use_clustering	yes
pruning_max_orients	1000
pruning_clustering_cutoff	100
pruning_conformer_score_cutoff	100.0
pruning_conformer_score_scaling_factor	1.0
use_clash_overlap	no
write_growth_tree	no
use_internal_energy	yes

internal_energy_rep_exp	9
internal_energy_cutoff	100.0
ligand_atom_file	cov_compounds.mol2
limit_max_ligands	no
skip_molecule	no
read_mol_solvation	no
calculate_rmsd	no
use_database_filter	no
orient_ligand	yes
automated_matching	yes
receptor_site_file	../002.surface_spheres/selected_spheres.sph
max_orientations	1000
critical_points	no
chemical_matching	no
use_ligand_spheres	no
bump_filter	no
score_molecules	yes
contact_score_primary	no
contact_score_secondary	no
grid_score_primary	yes
grid_score_secondary	no
grid_score_rep_rad_scale	1
grid_score_vdw_scale	1
grid_score_es_scale	1
grid_score_grid_prefix	../003.gridbox/grid
multigrid_score_secondary	no
dock3.5_score_secondary	no
continuous_score_secondary	no
footprint_similarity_score_secondary	no
pharmacophore_score_secondary	no
descriptor_score_secondary	no
gbsa_zou_score_secondary	no
gbsa_hawkins_score_secondary	no
SASA_score_secondary	no
amber_score_secondary	no
minimize_ligand	yes
minimize_anchor	yes
minimize_flexible_growth	yes
use_advanced_simplex_parameters	no
simplex_max_cycles	1

simplex_score_converge	0.1
simplex_cycle_converge	1.0
simplex_trans_step	1.0
simplex_rot_step	0.1
simplex_tors_step	10.0
simplex_anchor_max_iterations	500
simplex_grow_max_iterations	500
simplex_grow_tors_premin_iterations	0
simplex_random_seed	0
simplex_restraint_min	no
atom_model	all
vdw_defn_file	/dock6/parameters/vdw_AMBER_parm99.defn
flex_defn_file	/dock6/parameters/flex.defn
flex_drive_file	/dock6/parameters/flex_drive.tbl
ligand_outfile_prefix	virtual.out
write_orientations	no
num_scored_conformers	1
rank_ligands	no

จากนั้นทำการรัน

```
$ dock6 -i virtual.in -o virtual.out
```