Session 4: Introduction to *in silico* docking

Phil Biggin With input from Greg Ross, Ranjit Vijayan and Irfan Alibay
philip.biggin@bioch.ox.ac.uk

This document should also be available online at https://github.com/bigginlab/OxCompBio/tree/master/tutorials/Docking/docking-2019.html.

Table of Contents

1. Introduction
2. Exercise 1: Visualising the protein
3. Exercise 2: Preparing a ligand and protein for docking
4. Exercise 3: Docking indinavir into HIV-1 protease
5. Exercise 4: Docking nelfinavir into HIV-1 protease
6. Concluding remarks
7. References
8. Appendix I: Useful links/resources
9. Appendix II: Useful Unix/Linux commands

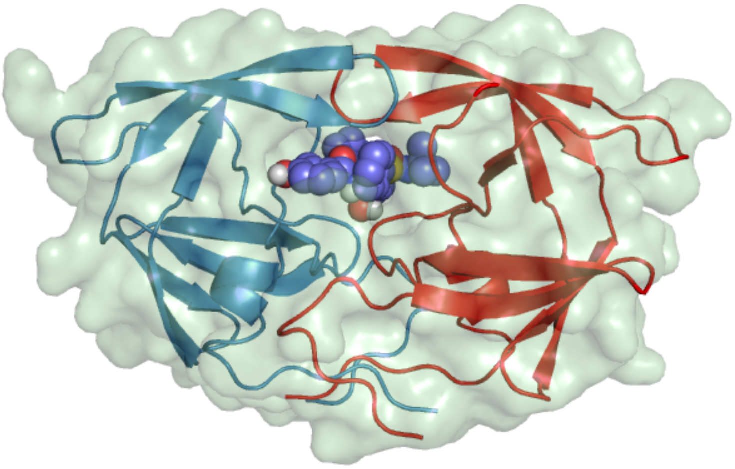
1. Introduction

Why is docking important?

The purpose of this practical is to give a flavour of how one could dock small flexible molecules to a protein structure. Such *in silico* methods are extremely useful for both finding potential binding sites and also to discover and/or engineer new molecules that could bind to a known site. This is a multi-billion dollar industry. Virtual screening and blind docking are often employed in an attempt to discover new medicines.

A case study: HIV-1 protease

The HIV-1 protease [1] is an enzyme that is vital for the replication of HIV. It cleaves newly formed polypeptide chains at appropriate locations so that they form functional proteins. Hence, drugs that target this protein could be vital for suppressing viral replication. A handful of drugs - called HIV-1 protease inhibitors (saquinavir, ritonavir, indinavir, nelfinavir, etc.) [2] - are currently commercially available that inhibit the function of this protein, by binding in catalytic site that binds the polypeptide.



HIV-1 Protease

What does this practical cover?

We will first look at the protein structure (PDB ID: 1HSG) [3] used in the molecular dynamics practical in a bit more detail. We will then try to dock a couple of drug molecules into the binding site to see how well docking can reproduce the binding pose. If time permits, we will generate the electrostatic surface of the protein to study how the drug interacts with the protein.

Make sure the tools work

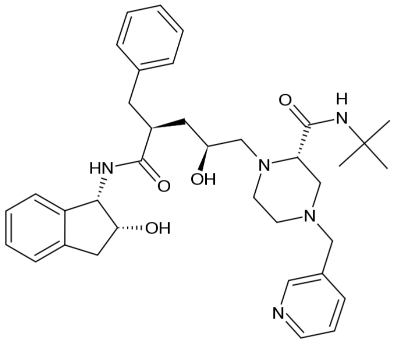
Anything that starts with the symbol `%` should be run in a Linux terminal/command prompt and `PyMOL>` should be run in a PyMOL command line.

i. All the files you need for this practical can be found at https://github.com/bigginlab/OxCompBio/tree/master/tutorials/Docking. You can download the OxCompBio material to access it. This can be done by cloning the git directory `git clone https://github.com/bigginlab/OxCompBio.git` (*Note:* To do this, you migth have to remove any existing OxCompBio material using `rm -rf ./OxCompBio`).
ii. If going through this tutorial in person via the Biochemistry Linux VM, all necessary software should already have been installed for you. If not, see the **README.md** file for instructions what you will need to install. To test everything is working open a Terminal and make sure you have access to the following commands. Some of these may open up a graphical user interface (GUI). Close it and proceed to the next step. If you get an error or "`Command not found`" inform a demonstrator. (Note to demonstrators: might be worth just checking that the apbs plugin works within pymol - this is buggy across versions).

```
% cd ~/OxCompBio/tutorials/Docking
% vina --version
% apbs --version
% pdb2pqr --help
% pymol -c
% adt -h
```

2. Exercise 1: Visualising the protein

In this practical, we will use the structure of the HIV-1 protease used in the molecular dynamics practical - PDB ID: **1HSG**. This is a 2Å resolution X-ray crystal structure of HIV-1 protease with a bound drug molecule **indinavir**. We will use `pymol` to view the protein, the binding site and the drug molecule. `pymol` options/controls that you learned in the homology modelling practical will be handy here. An introduction to PyMOL can be found here.



Indinavir (Source: http://en.wikipedia.org/wiki/Indinavir)

i. In a terminal (open a new terminal if you don't have one open already), change to the docking practical directory.

```
% cd ~/OxCompBio/tutorials/Docking/dock-prac/1HSG
```

ii. Download the protease structure in PDB format (PDB ID: 1HSG) from the Protein Data Bank (http://www.pdb.org). Search using the PDB ID 1HSG. On the results page, click **Download Files** and select **PDB File (Text)**. Save it in `~/OxCompBio/tutorials/Docking/dock-prac/1HSG`. If the internet is down you can get a copy from `~/OxCompBio/tutorials/Docking/dock-prac/backup/data/1HSG.pdb`. (*Note:* Once again you might have to right click, select **Save Link As...** and browse to `/home/biocomp/OxCompBio/tutorials/Docking/dock-prac/1HSG`. Make sure you set the filename **Name** to 1HSG.pdb. Click **Save**)
iii. Load the structure into `pymol`. You should see the protein structure displayed as lines and water molecules as little red crosses.

```
% pymol 1HSG.pdb &
```

iv. Lets start by hiding everything. Click the **H** (for hide) next to **all** in PyMOL's **Object Control Panel** (the panel on the right with buttons **A**ctions, **H**ide, **S**how, **L**abel and **C**olour) and select **everything**. The screen should be clear now.
v. Now show (**S**) the protein (**1HSG**) using the **cartoon** representation and colour (**C**) by chain to show that it is a homodimer.
vi. Let's select the ligand indinavir and show it as sticks. In the PyMOL command line (with a "`PyMOL >_`") type

```
PyMOL> select indinavir, resn MK1
```

`resn MK1` selects the residue MK1, which is indinavir. You should now have an object in the object control panel called **(indinavir)**. Display it as sticks. Click anywhere in the display screen to unselect the selected atoms.
vii. Now, rotate (left mouse button), zoom (right mouse button) and move (middle mouse button) the molecule to get an idea where the binding site is. You will need to know where it is for the next exercise .
viii. Water molecules have the residue name HOH. Select and display all water molecules as red spheres. If you think the spheres are too big, type

```
PyMOL> set sphere_mode, 4
PyMOL> set sphere_scale, 0.4
```

ix. Q: Water molecules normally have 3 atoms. Why do we see just one atom per water molecule?
x. Q: There is a conserved water molecule in the binding site. Can you identify this water molecule?
xi. Q: Can you think of a way in which the ligand could enter the binding site?
xii. If you would like to save a nice image of the protein, ray trace it and save it as an image file.

```
PyMOL> bg_color white
PyMOL> set depth_cue,0
PyMOL> ray
PyMOL> png protein.png
```

xiii. Close PyMOL by typing `quit` in PyMOL command window or clicking the **x** in the PyMOL Tcl/Tk GUI window.
xiv. Using `eog`, you can visualise the image you just generated.

```
% eog protein.png
```

3. Exercise 2: Preparing the protein and ligand for docking

Docking algorithms require each atom to have a charge and an atom type that describes its properties. However, the PDB structure lacks these. So, we have to prep the protein and ligand files to include these values along with the atomic coordinates. Furthermore, for flexible ligand docking, we should also define ligand bonds that are rotatable. All this will be done in a tool called AutoDock Tools (`adt`).

Prepare the protein

i. The PDB file (`1HSG.pdb`) contains protein, ligand and water oxygen atoms. First we have to extract just the protein atoms, which are lines that start with the keyword **ATOM**. Each protein chain is terminated with a line that starts with **TER**. (If you would like to confirm this, open `1HSG.pdb` in a text editor and scroll through the text.)

```
% egrep "^(ATOM|TER)" 1HSG.pdb > 1HSG_protein.pdb
```

ii. Lauch AutoDock Tools (ADT) using the command `adt`

```
% adt &
```

iii. Load the protein using **File > Read Molecule**. Select `1HSG_protein.pdb`. Click **Open**.
Note: In ADT, you can translate the molecule by clicking and holding down the right mouse button while moving the mouse, rotate by clicking and holding down the middle button and zoom in/out by using the scroll wheel of the mouse.
iv. Bonds and atoms are shown in white. For better visualisation, colour the structure by atom type - **Color > By Atom Type**. Click **All Geometries** and then **OK**.
v. Q: Can you locate the binding site visually?
vi. Crystal structures normally lack hydrogen atoms. Q: Why? However, hydrogen atoms, or more specifically polar hydrogen atoms are required for appropriate treatment of electrostatics during docking. Add hydrogen atom to the structure using - **Edit > Hydrogen > Add**. Click **OK**. You should see a lot of white dashes where the hydrogens were added.
vii. Now we need to get ADT to assign charges and atom type to each atom in the protein. We do this with **Grid > Macromolecule > Choose...**. Choose **1HSG_protein** in the popup window and click **Select Molecule**. ADT will merge non-polar hydrogens, assign charges and prompt you to save the macromolecule. Click **Save**. This will create a file called `1HSG_protein.pdbqt` in the current folder. Open this in a text editor and look at the last two columns - these should be the charge and atom type for each atom.
Q: Look at the charges. Does it make sense?
viii. Now, we need to define the 3D search space were ligand docking will be attempted. Remember the binding site that you observed in one of the earlier steps. Ideally, if we do not know the binding site, we will either define a box that encloses the whole protein or perhaps a specific region of the protein. In this case, to speed up the docking process, we will define a search space that encloses the known binding site.
ix. To define the box, use **Grid > Grid Box...**. This will draw a box with opposite faces coloured in red, green and blue. Fiddle with the dials and see how you can enclose regions of the protein. In this instance we will use a **Spacing (angstrom)** of **1Å** (this is essentially a scaling factor). So set this dial to **1.000**. So that we all get consistent results, let us set the **(x, y, z) center** as **(16, 25, 4)** and the **number of points in (x, y, z)-dimension** as **(30, 30, 30)**. Make a note of these values. We will need it later. Close the **Grid Options** dialog by clicking **File > Close w/out saving**.
x. That is all we need to do with the protein file. Delete it from the display using **Edit > Delete > Delete Molecule** and select `1HSG_protein`. Click **Delete Molecule**.

Prepare the ligand

i. Like the protein, the ligand lacks hydrogen atoms. We need to add hydrogen atoms and also define rotatable bonds that will be used for flexible docking.
ii. First, extract the ligand atoms from the PDB. As mentioned in Exercise 1, the ligand residue name for **indinavir** in the PDB file is **MK1** and the lines start with the keyword **HETATM** for heteroatoms. In a terminal type

```
% grep "^HETATM.*MK1" 1HSG.pdb > indinavir.pdb
```

iii. Load the ligand structure into ADT using **File > Read Molecule** and select `indinavir.pdb`
iv. Again, colour by atom type. **Color > By Atom Type**. Click **All Geometries** and then **OK**.
v. Now we have to add polar hydrogen atoms. Add all hydrogen atoms initially. Non polar hydrogens will be merged in the next step. **Edit > Hydrogens > Add**. Select **All Hydrogens** and click **OK**.
vi. Define this as the ligand in ADT so that ADT assigns partial charges and sets rotatable ligand bonds using **Ligand > Input > Choose...**. Select `indinavir` and click **Select Molecule for AutoDock4**. You should see a message that confirms that non-polar hydrogens have been merged, charges added and rotatable bond detected. Click **OK**. The ligand will now have only polar hydrogens.

vii. To check the rotatable bonds detected by ADT, go to **Ligand** > **Torsion Tree** > **Choose Torsions....** You should see 14 rotatable bonds.

viii. Q: What does the three colours - green, magenta and red - mean? Does it make sense?

ix. Click **Done** when you are done.

x. Save the ligand file in PDBQT format. Do this using **Ligand** > **Output** > **Save as PDBQT...**. Click **Save** to save the file as `indinavir.pdbqt`.

xi. Quit ADT using **File** > **Exit** and then **OK**.

Prepare a docking configuration file

i. Before we can perform the actual docking, we need to create an input file that defines the protein, ligand and the search parameters. We will create the input file in a text editor.

ii. If you have used Unix/Linux before, open your favourite text editor - `vi`, `emacs`, `nano` - or use a GUI based editor called `gedit`.

```
% gedit &
```

iii. The input file should look something like:

```
receptor = 1HSG_protein.pdbqt
ligand = indinavir.pdbqt

num_modes = 50

out = all.pdbqt

center_x = XX
center_y = XX
center_z = XX

size_x = XX
size_y = XX
size_z = XX

seed = 2009
```

This defines your protein (`receptor`), ligand, number of docking modes to generate (`num_modes`). All the docked modes will be collated in a file defined by `out` (`all.pdbqt`). You should replace the **XX** with the center of your 3D search space (`center_x/y/z`) and the size of the box (`size_x/y/z`) that you defined in the "Prepare the protein" section.

iv. Save the file as `config.txt` in the folder containing the protein and ligand `.pdbqt` files.

4. Exercise 3: Docking indinavir into HIV-1 protease

i. For this practical, we will use a program called **Autodock Vina** [4] for docking. Autodock Vina is a fast docking alogorithm that requires minimal user intervention. We will run it from a terminal.

ii. Make sure you are in the folder containing `1HSG_protein.pdbqt`, `indinavir.pdbqt` and `config.txt`

iii. Run `vina` to perform the docking. We will keep a log of all the program output in a file `log.txt`

```
% vina --config config.txt --log log.txt
```

This will take a few minutes depending on how fast your computer is. While you wait, if you are interested, read [1] for a review on HIV-1 protease structure, function and drug discovery.

iv. Once the run is complete, you should have a two files `all.pdbqt`, which contains all the docked modes, and `log.txt`, which contains a table of calculated affinities based on AutoDock Vina's scoring function [4]. The best docked mode, according to AutoDock Vina, is the first entry in `all.pdbqt`.

v. Lets visualise the docks and compare it to the crystal conformation of the ligand. Load the protein, the extracted ligand and the docks into `pymol`.

```
% pymol 1HSG_protein.pdb indinavir.pdb all.pdbqt &
```

vi. Begin by hiding everything. Now, display the **protein** in **cartoon** representation, **indinavir** in **stick** representation and the **docked conformations** as **sticks** as well.

vii. Cycle through the docks by clicking the playback control buttons on the lower right hand corner of the window.

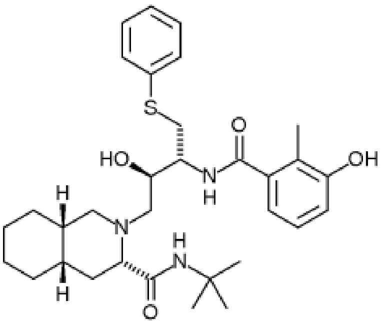
viii. Q: Qualitatively, how good are the docks?

ix. Q: Is the crystal binding mode reproduced? If not or if there is a difference, what could be the reason? Is it the best conformation according to AutoDock Vina?

x. Close PyMOL.

5. Exercise 4: Docking nelfinavir into HIV-1 protease

i. In the previous exercise, AutoDock Vina should have found a near native docked conformation for the drug **indinavir**. In this exercise, we will try to dock another drug **nelfinavir** into the protein structure 1HSG broadly along the lines of how one would dock a small molecule into a target structure. This approach could be refined and extended to the "virtual screening" approach.



Nelfinavir (Source: http://en.wikipedia.org/wiki/Nelfinavir)

ii. Start by creating a new directory for this docking run. Your current directory should now be: `~/OxCompBio/tutorials/Docking/dock-prac/1HSG`

```
% cd ~/OxCompBio/tutorials/Docking/dock-prac/1HSG_nelfinavir
```

iii. Since we will be using the same protein structure to dock `nelfinavir`, copy all the protein related files that you created in "Prepare the protein" section.

```
% cp ~/OxCompBio/tutorials/Docking/dock-prac/1HSG/1HSG* .
```

iv. We need a structure of `nelfinavir`. There should be a copy in `~/OxCompBio/tutorials/Docking/dock-prac/backup/nelfinavir.pdb`. Copy it to the current directory

```
% cp ~/OxCompBio/tutorials/Docking/dock-prac/backup/data/nelfinavir.pdb .
```

v. Prep the ligand file and generate the PDBQT file - `nelfinavir.pdbqt` - using the steps used in the "**Prepare the ligand**" section.

vi. Since the binding site is the same, copy the docking config file - `config.txt` - from the previous exercise to the current directory.

```
% cp ~/OxCompBio/tutorials/Docking/dock-prac/1HSG/config.txt .
```

vii. You have to change one entry in the configuration file. **Make the change.** (Hint: what files have changed?)

viii. Now you should have everything required by AutoDock Vina to generate a series of docking modes.

```
% vina --config config.txt --log log.txt
```

This will take a few minutes to run...

ix. Load the protein and the docked structures and show the protein as cartoon and the docks as sticks?

```
% pymol 1HSG_protein.pdb all.pdbqt &
```

x. Q: Qualitatively, how good are the docks?

xi. Q: Is it docked in the binding site?

xii. Q: Is the best docked conformation of `nelfinavir` similar to `indinavir`? (Note: you can load files by clicking on **File** > **Open..**

xiii. Q: Based on the literature, what is common among this class of HIV-1 protease inhibitors?

xiv. Q: How would you check whether the conformation makes sense?

xv. Q: How would you look at it from a "virtual screening" perspective?

xvi. Fortunately, in this instance, there are crystal structures of `nelfinavir` bound to HIV-1 protease inhibitor that you can use to validate your docking results.

xvii. Use the Protein Data Bank and download the PDB file for **1OHR**. Save it in the current directory - `~/OxCompBio/tutorials/Docking/dock-prac/1HSG_nelfinavir`. (If the internet is down, there should be a copy in `~/OxCompBio/tutorials/Docking/dock-prac/backup/data/1OHR.pdb`).

xviii. Load this structure into PyMOL using **File** > **Open...**. Select **1OHR.pdb** and click **OK**.

xix. Hide all representations of **1OHR** using **H** and then **everything**. Now, display **1OHR** as **cartoon** and **colour** by **chain**.

xx. You may have observed that the two HIV-1 protease structures are in different orientations in space. To make sense of the docking results, you have to align the two structures. You can use the PyMOL `align` command to do this. In the PyMOL command line, type:

```
PyMOL> align 1OHR, 1HSG_protein
```

1OHR should now be overlaid on **1HSG_protein**
Note: A copy of the aligned structure can be found in `~/OxCompBio/tutorials/Docking/dock-prac/backup/data/1OHR_aligned.pdb` if the `align` command does not work.

xxi. Q: What is the RMSD of the alignment? Is it qualitatively and quantitatively good?

xxii. You may have observed that moving the structure around the window is a bit difficult since the origin of the view has been altered when you loaded `1OHR.pdb`. To reset it, try:

```
PyMOL> reset
```

xxiii. **nelfinavir** has the residue name **1UN** in **1OHR**. Select and display it as sticks.

```
PyMOL> select nelfinavir, 1OHR and resn 1UN
```

Colour it **by element** and pick any colour combination.

xxiv. Q: Is the "best mode" generated by AutoDock Vina similar to the crystal conformation of `nelfinavir`? If not, why not?

xxv. Q: How about the "second best mode"? Why do you think this is the case? (Hint: Look at the affinities reported in the log file)

xxvi. Q: In this instance we assumed that chain A of 1HSG corresponds to chain A of 1OHR. Is this true always?

xxvii. Let us align chain A of 1OHR to chain B of 1HSG.

```
PyMOL> align 1OHR and chain A, 1HSG_protein and chain B
```

xxviii. Q: Compare the first two AutoDock Vina docks with the crystal conformation. What does this tell you? What could be a reason for this observation?

xxix. Close PyMOL.

6. Concluding remarks

In this practical, we have looked at how small molecules could be docked into a protein. This approach is widely used to detect binding sites and also to screen a library of small molecules to find potential drugs that could bind to a known binding site. Scoring functions play an important role in identifying a "good dock" and as such remains an area of active research. Several other considerations are worth noting including water molecules and ions in the binding site, flexibility of binding site residues, etc. Some docking programs (including AutoDock Vina) allow you to define a subset of flexible sidechains. Whilst this permits binding site rearrangement to accommodate distinct ligands, the computational search space increases many fold. Defining water molecules that are important in the binding site also remains an area of active research. In this practical, we have considered only protein-ligand docking. Protein-protein docking is also widely used, but not considered here due to the significantly higher search space that has to be considered. As the number of published and unpublished protein structures continue to grow, *in silico* docking will continue to play an important role in the drug discovery process.

7. References

1. Brik A, Wong CH. "HIV-1 protease: mechanism and drug discovery". Org. Biomol. Chem. (2003) 1:5–14.
2. Wensing AM, van Maarseveen NM, Nijhuis M. "Fifteen years of HIV Protease Inhibitors: raising the barrier to resistance." Antiviral Res. (2009) Online in advance of print.
3. Chen Z, Li Y, Chen E, Hall DL, Darke PL, Culberson C, Shafer JA, Kuo LC. "Crystal structure at 1.9-A resolution of human immunodeficiency virus (HIV) II protease complexed with L-735,524, an orally bioavailable inhibitor of the HIV proteases." J. Biol. Chem. (1994) 269:26344-26348.
4. Trott O, Olson AJ. "AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization and multithreading", J. Comput. Chem. (2009) Online in advance of print.
5. Baker NA, Sept D, Joseph S, Holst MJ, McCammon JA. "Electrostatics of nanosystems: application to microtubules and the ribosome." Proc. Natl. Acad. Sci. (2001) 98:10037-10041.

8. Appendix I: Useful links/resources

PyMOL	http://www.pymolwiki.org/index.php/Main_Page (might need to install with conda install -c samoturk pymol)
AutoDock Tools	http://autodock.scripps.edu
AutoDock Vina	http://vina.scripps.edu

9. Appendix II: Basic Linux commands

`ls -lrt`	provides a "long" list of all files in the current directory in reverse order of time.
`cd dir`	change directory to the directory 'dir'
`pwd`	print the current working directory on the screen
`rm file`	delete (remove) 'file'
`mv file newfile`	rename file to newfile
`cat file`	print the contents of file to the screen
`more file`	print the contents of file to the screen but with more navigation possible.