

BGGN-213: FOUNDATIONS OF BIOINFORMATICS (Lecture 12)

Structural Bioinformatics (Part 2)

https://bioboot.github.io/bgg213_S18/lectures/#12

Dr. Barry Grant

Section 1: *In silico* docking of drugs to HIV-1 protease

Today we will continue our analysis of the HIV-1 protease enzyme that we introduced in the last hands-on session (see Lecture 11). Recall that this enzyme cleaves newly formed polypeptide chains at appropriate locations so that they form functional proteins - a process essential for replication of HIV [1]. A number of drugs - called *HIV-1 protease inhibitors* (saquinavir, ritonavir, indinavir, nelfinavir, etc.) [2] - inhibit the function of this protein and act to suppress viral replication, by binding in the catalytic site that typically binds the polypeptide (**Figure 1**).

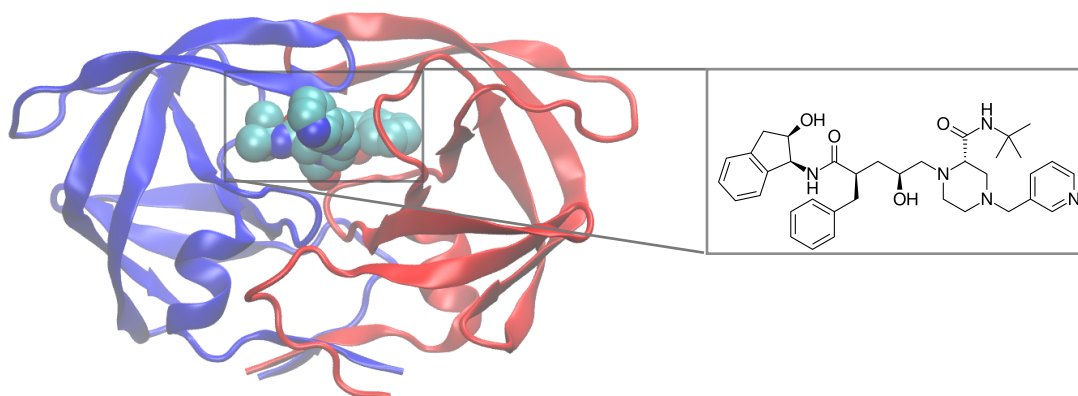


Figure 1. HIV-1 protease structure in complex with the small molecule indinavir. Note that this is similar to the figure you generated using VMD in the last hands-on session.

Last day we used the 2Å resolution X-ray crystal structure of HIV-1 protease with a bound drug molecule [indinavir](#) (PDB ID: 1HSG) [3]. Today we will continue with this structure and computationally dock drug molecules into the binding site of HIV-1 protease to see how well bioinformatics based docking can reproduce the crystallographically observed binding pose. If time permits, we will also explore the conformational dynamics and flexibility of the protein - important for its function and for considering during drug design.

NOTE: If you are using your own computer (i.e. not a classroom machine) you will require the **AutoDockTools** software package that comes as part of **MGLTools** software download available from:

<http://mgltools.scripps.edu/downloads>

For Windows based laptops please chose: **mgltools_win32_1.5.6_Setup.exe** and for Macs please chose the **(Snow) Leopard - Mac OS X 10.5 and 10.6 - Intel** binary installers. Launching these should begin the install process. There are also Linux versions available. Typically you will want the **x86_64** Linux version but please do ask if you are unsure.

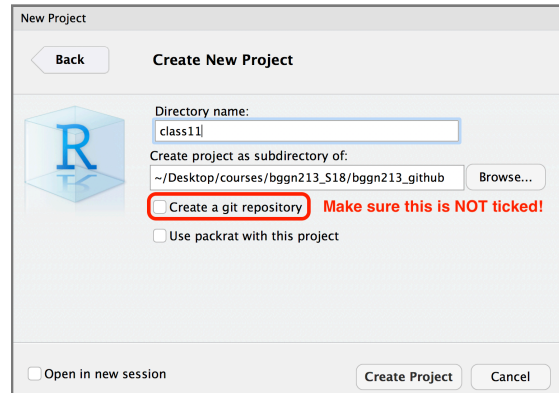
1.1 Obtaining and inspecting our input structure.

Start **RStudio** and create a **File > New Project...** new Project directory called ***class12*** within your GitHub tracked directory/folder where you are organizing your course work.

Again make sure the “*Create a git repository*” is **NOT** selected (see figure right)

Open a new GitHub flavored **R markdown** document to save your work as we proceed.

Now load the Bio3D package and enter the following commands:



```
library(bio3d)
file.name <- get.pdb("1hsg")
```

This will download the 1HSG pdb file from the RCSB PDB database to your current working directory (i.e. Your RStudio Project directory).

Next we will read this PDB structure into R so we can prepare it for further analysis.

```
hiv <- read.pdb(file.name)
```

You can get a quick summary of the contents of this pdb structure object by typing the name of the object we just created.

```
hiv
##
## Call: read.pdb(file = file.name)
##
## Total Models#: 1
## Total Atoms#: 1686, XYZs#: 5058 Chains#: 2 (values: A B)
##
## Protein Atoms#: 1514 (residues/Calpha atoms#: 198)
## Nucleic acid Atoms#: 0 (residues/phosphate atoms#: 0)
##
## Non-protein/nucleic Atoms#: 172 (residues: 128)
## Non-protein/nucleic resid values: [ HOH (127), MK1 (1) ]
##
## Protein sequence:
## PQITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWKPKMIGGIGGFIKVRQYD
## QILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNFPQITLWQRPLVTIKIGGQLKE
## ALLDTGADDTVLEEMSLPGRWKPKMIGGIGGFIKVRQYDQILIEICGHKAIGTVLVGPTP
## VNIIGRNLLTQIGCTLNF
```

```
##  
## + attr: atom, xyz, seqres, helix, sheet,  
##      calpha, remark, call
```

Q1: What is the name of the two non protein **resid** values in this structure? What does **resid** correspond to and how would you get a listing of all residue values in this structure?

If in doubt about any of the previous steps please refer to the last days handout for further details of working with single pdb structure objects in Bio3D - or ask Barry for assistance.

1.2 Prepare initial protein and ligand input files

The PDB file we just downloaded (1hsg.pdb) and imported into R (contained in the object `hiv` in your R session) contains protein, ligand and water atoms. Before proceeding with docking we first have to extract just the protein atoms.

If you examine the downloaded file you will see that the protein segment lines 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.) You could copy and paste the protein portion of the original PDB file into a new file. Here we will use R to do this potentially tedious and error prone step for us and save a new protein only PDB format file and a separate ligand only PDB format file. We will use these files to test our bioinformatics docking approach.

```
prot <- trim.pdb(hiv, "protein")  
lig <- trim.pdb(hiv, "ligand")  
  
write.pdb(prot, file="1hsg_protein.pdb")  
write.pdb(lig, "1hsg_ligand.pdb")
```

1.3 Using AutoDockTools to setup protein docking input

Docking algorithms require each atom to have a charge and an atom type that describes its properties. However, typical PDB structures don't contain this information. We therefore have to 'prep' the protein and ligand files to include these values along with their atomic coordinates. All this will be done in a tool called **AutoDock Tools** (ADT for short).

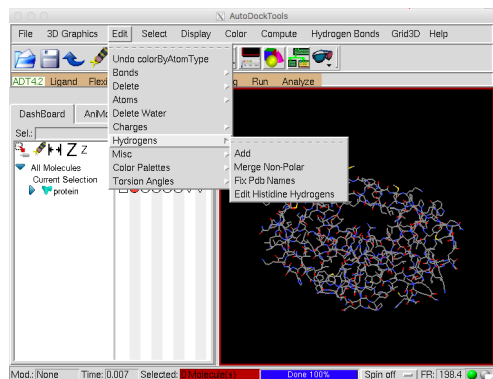
Launch AutoDock Tools (ADT) from the Desktop icon on Windows or using the Finder on Mac (from **Applications > MGLTools > AutoDocTools**).

NOTE: First time opening of ADT can be a rather slow process as it completes some initial configuration. Please be patient and resist the urge to click many times and thus open multiple instances.

In ADT, 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.

Bonds and atoms are shown in white. For better visualization, color the structure by atom type - **Color > By Atom Type**. Click **All Geometries** and then **OK**.



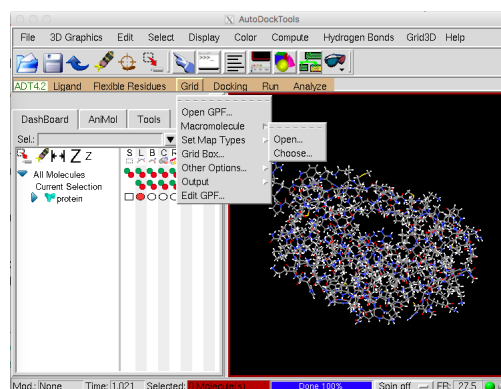
Q2: Can you locate the binding site visually? Note that crystal structures normally lack hydrogen atoms, why?

As we have already noted (remind me if we haven't) crystal structures normally lack hydrogen atoms. However, hydrogen atoms are required for appropriate treatment of electrostatics during docking so we need to add hydrogen atoms to the structure using **Edit > Hydrogen > Add**. Click **OK**. You should see a lot of white dashes where the hydrogens were added.

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 assign charges and prompt you to save the macromolecule.

Save this molecule to your `class12` folder with the default file extension of "pdbqt" (i.e. a new file called **1hsg_protein.pdbqt**). Open this in a text editor and look at the last two columns - these should be the charge and atom type for each atom. Note how they are different from the `1hsg_protein.pdb` file we input to ADT.



NOTE: Some users of ADT have reported that the file navigator for choosing the export file location does not fill in the correct path for their saved molecules. In such cases the files are output to the users home area and should be moved into the `class12` directory manually.

Q3: Look at the charges. Does it make sense (e.g. based on your knowledge of the physiochemical properties of amino acids)?

Now, we need to define the 3D search space where 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.

To define the box, use **Grid > Grid Box....** This will draw a box with opposite faces colored 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.

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** and **CONTINUE**.

1.4 Prepare the ligand

Like the protein, the ligand lacks hydrogen atoms. We need to add hydrogen atoms and also optionally define rotatable bonds that can be used for 'flexible docking'.

Load the ligand structure into ADT using **File > Read Molecule** and select 1hsg_ligand.pdb

Again, color by atom type. **Color > By Atom Type**. Click **All Geometries** and then **OK**.

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**.

Define this as the ligand in ADT so that ADT assigns partial charges and sets rotatable ligand bonds using **Ligand > Input > Choose....** Select 1hsg_ligand 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.

Click **Done** and save the ligand file in PDBQT format. Do this using **Ligand > Output > Save as PDBQT....** Click **Save** to save the file as **ligand.pdbqt**. Quit ADT using **File > Exit > OK**.

Again make sure this file lives in your **class12** folder.

1.5 Prepare a docking configuration file

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.

Use RStudio **File > New File... > Text File** (or use your favorite text editor such as TextEdit on Mac or NotePad on Windows (my current favorite is SublimeText)).

Create an input file for our docking calculations that looks something like the following:

```
receptor = 1hsg_protein.pdbqt
ligand = ligand.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 (`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 **Section 1.3** “*Prepare the protein*” above.

Save the file as `config.txt` in your **class12** folder.

Note: If necessary you can download a pre-prepared file from [here](#).

Section 2: *Docking ligands into HIV-1 protease*

For this section, we will use the program called **Autodock Vina** [4]. Autodock Vina is a fast docking program that requires minimal user intervention and is often employed for high-throughput virtual screening. We will run it from the command line.

Note: If you are working on your own laptop (i.e. NOT the classroom machines) you will need to first install the AutoDoc vina software for protein ligand docking from <http://vina.scripps.edu/download.html> chose the appropriate version for your OS.

Once downloaded the **Windows** versions can be installed by double clicking the resulting **autodock_vina_1_1_2_win32.msi** file and following the install prompts.

For **Mac** simply double click the downloaded **autodock_vina_1_1_2_mac.tar** file in your Downloads directory to yield a new directory called **autodock_vina_1_1_2_mac**. Within this new directory there is a bin directory containing the vina program.

On Macs you should now be able to run this command and see the associated help for the vina program:

```
> ~/Downloads/autodock_vina_1_1_2_mac/bin/vina --help
```

On Windows your command will look like the following:

```
> "\\Program Files (x86)\\The Scripps Research Institute\\Vina\\vina.exe" --help
```

2.2 Docking indinavir into HIV-1 protease

Now we are ready to run autodock vina to perform the docking. We will keep a log of all program output in a file **log.txt**

For Mac use the command:

```
> ~/Downloads/autodock_vina_1_1_2_mac/bin/vina --config config.txt --log log.txt
```

On Windows use the following:

```
> "\\Program Files (x86)\\The Scripps Research Institute\\Vina\\vina.exe" --config config.txt --log log.txt
```

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

Once the run is complete, you should have two new 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**.

2.3 Inspecting your docking results

In order to visualize the docks and compare to the crystal conformation of the ligand we will process the **all.pdbqt** to a PDB format file that can be loaded into VMD. To do this we will use R and the Bio3D package.

Back in **RStudio** running in your class12 directory and with the bio3d package loaded issue the following commands:

```
library(bio3d)
res <- read.pdb("all.pdbqt", multi=TRUE)

write.pdb(res, "results.pdb")
```

Now load both the original **1hsg.pdb** file and your new **results.pdb** file into VMD. You will need to use **VMD Main > File > New Molecule** separately for each file.

Begin by displaying the protein in cartoon representation, indinavir in licorice (also called stick representation) and the docked conformations as licorice as well.

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

Q4: Qualitatively, how good are the docks? Is the crystal binding mode reproduced? Is it the best conformation according to AutoDock Vina?

To assess the results quantitatively we will calculate the RMSD (root mean square distance) between each of the docking results and the known crystal structure using the bio3d package.

Back in **RStudio** read the original ligand with added hydrogens that you produced earlier and use the **rmsd()** function to compare to your docking results.

```
# res <- read.pdb("all.pdbqt", multi=TRUE)
ori <- read.pdb("ligand.pdbqt")
rmsd(ori, res)
```

Q5: Quantitatively how good are the docks? Is the crystal binding mode reproduced within 1Å RMSD for all atoms?

Q6: How would you determine the RMSD for heavy atoms only (i.e. non hydrogen atoms)?
HINT: The **atom.select()** function will be of help here along with the selection string "noh" for no hydrogens.

Side-Note: Remember to save your R markdown script document and Knit to generate a GitHub format .md report. Then stage, commit and push both these documents to GitHub by following the steps outlined in Section 4 - ask Barry if you are unsure of this process.

N.B. Muddy Point Assessment Form < <https://goo.gl/forms/Jz3C95WSPFk3WuE63> >

OPTIONAL: Section 3: Exploring the conformational dynamics of proteins

Visit the new web application Bio3D-web: <http://thegrantlab.org/bio3d/webapps> watch the introduction video and click start analysis to begin exploring the conformational dynamics and flexibility of protein structures.

If server link above is unresponsive or otherwise slow you can try the following mirror:

<http://129.177.232.111:3848/pca-app/>

Q7: What are the most flexible portions of HIV-1 protease? Would this flexibility likely effect docking calculations to this protein?

OPTIONAL: Class 11, Section 4 & 5 revisited

The web apps from the last section are simply fancy online interfaces to the Bio3D R package. In this section we will revisit our skipped section from last day to see how we can use Bio3D proper to do essentially the same analysis in a much more custom and controllable fashion.

First we need get the muscle alignment program working properly on windows - this was a stumbling point for some during the last days class.

5.1 Installing the stand-alone muscle alignment program

If you are not on one of the classroom computers you will need to download the appropriate version of the **muscle** multiple alignment program from: <https://www.drive5.com/muscle/downloads.htm>

Side-Note: For Mac you will most likely have done this successfully last day so you can skip this subsection.

On WINDOWS: After downloading MUSCLE, it should be moved to your RStudio project directory and renamed to **muscle.exe**. Use your windows file explorer to move the downloaded **muscle3.8.31_i86win32.exe** from your Downloads folder via copy and paste to your Project folder. Then right click to rename to **muscle.exe**

In your RStudio **Terminal** you should now be able to run the following without error:

```
./muscle.exe -version
```

If you get an error message then **something has not worked and you should ask Barry for help!**

On MAC: After downloading MUSCLE, it should be unzipped and renamed to just "**muscle**". If you are on Mac you can move this file to a directory such as **"/usr/local/bin"**

The easiest way to do this on Mac or Linux is to use the **Terminal/Shell** in your RStudio. At the shell prompt issue the following commands:

```
tar -xvf ~/Downloads/muscle3.8.31_i86darwin32.tar
sudo mv muscle3.8.31_i86darwin32 /usr/local/bin/muscle
```

If you now type **muscle** in your terminal you should see the help splash screen for the MUSCLE program. Instead, if you see the response “bash: muscle: command not found” **something has not worked and you should ask Barry for help!**

5.2 Search and retrieve Adenylate kinase structures

Below we perform a blast search of the PDB database to identify related structures to our query Adenylate kinase UniProt sequence (with identifier P69441). In this particular example we use the function `get.seq()` to fetch the query sequence directly from the UniProt sequence database.

```
aa <- get.seq("P69441")
## Fetching... Please wait. Done.
```

Next we use this sequence as input to the `blast.pdb()` function.

```
# Blast or hmmer search
b <- blast.pdb(aa)

## Searching ... please wait (updates every 5 seconds) RID =
ZM7GP50C014
## .
## Reporting 209 hits
```

Function `plot.blast()` facilitates the visualization and filtering of the Blast results. It will attempt to set a seed position to the point of largest drop-off in normalized scores (i.e. the biggest jump in E-values). In this particular case we specify a cutoff (after initial plotting) of 225 to include only the relevant *E.coli* structures:

```
# Plot a summary of search results
hits <- plot(b)

## * Possible cutoff values:      198 -3
##           Yielding Nhits:      39 209
##
## * Chosen cutoff value of:      198
##           Yielding Nhits:      39
```

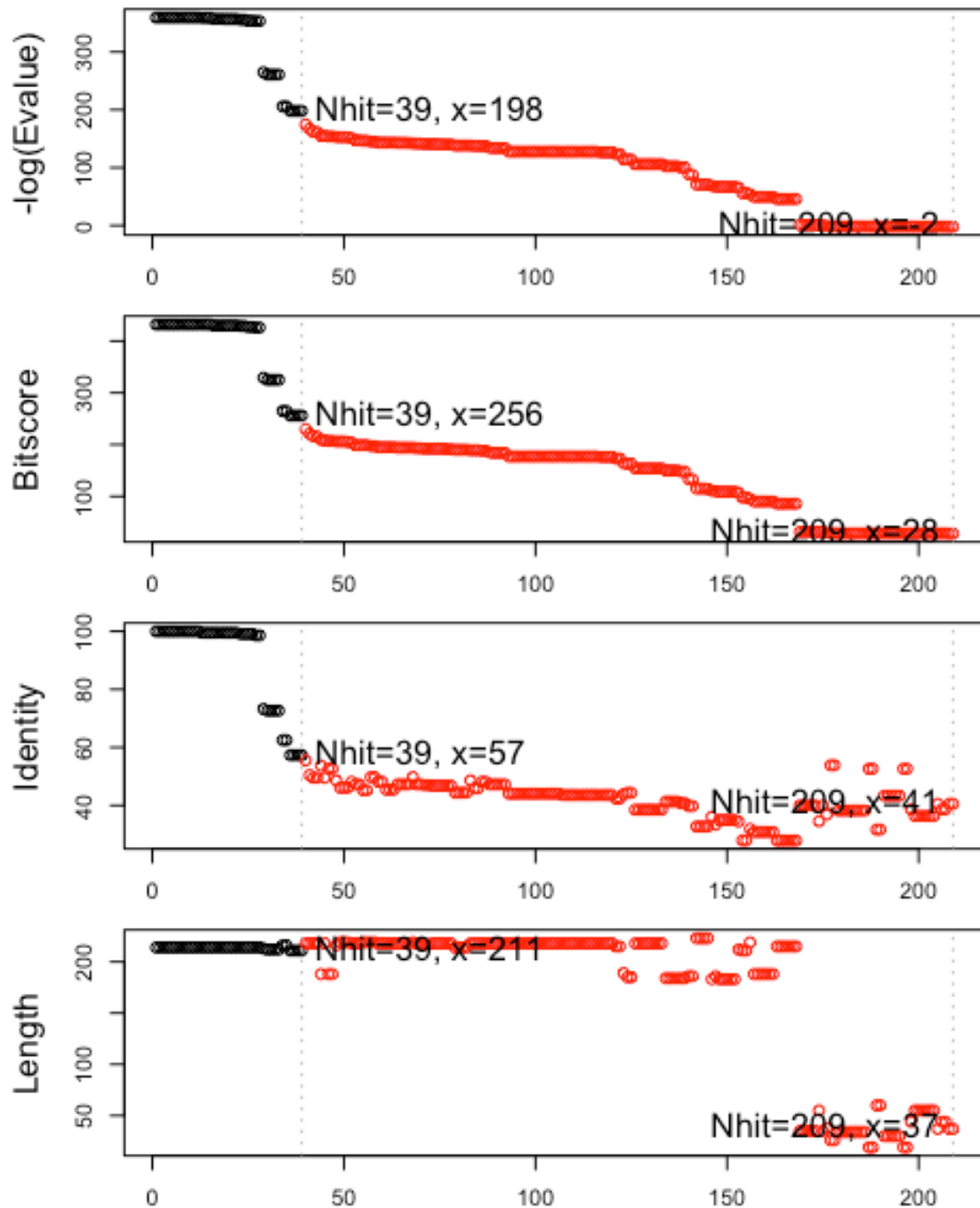


Figure 11: Blast results. Visualize and filter blast results through function `plot.blast()`. Here we proceed with only the top scoring hits (black).

```
head(hits$ pdb.id)
```

```
## [1] "1AKE_A" "1AKE_B" "1ANK_A" "1ANK_B" "4AKE_A" "4AKE_B"
```

The Blast search and subsequent filtering identified a total of 39 related PDB structures to our query sequence. The PDB identifiers of this collection are accessible through the `pdb.id` attribute to the `hits` object (`hits$pdb.id`). Note that adjusting the `cutoff` argument (to `plot.blast()`) will result in a decrease or increase of hits.

We can now use function `get.pdb()` and `pdbslit()` to fetch and parse the identified structures. Finally, we use `pdbaln()` to align the PDB structures.

```
# Fetch PDBs
files <- get.pdb(hits$pdb.id, path="pdb", split=TRUE, gzip=TRUE)
```

Next we use the `pdbaln()` function to align fit all the PDB structures.

```
# Align & superimpose structures
pdb <- pdbaln(files, fit=TRUE)
```

The Bio3D `pca()` function provides *principal component analysis* (PCA) of the structure data. PCA is a statistical approach used to transform a data set down to a few important components that describe the directions where there is most variance. In terms of protein structures PCA is used to capture major structural variations within an ensemble of structures.

```
# Perform PCA & plot the results
pc.xray <- pca(pdb)
plot(pc.xray)
```

Function **rmsd()** will calculate all pairwise RMSD values of the structural ensemble. This facilitates clustering analysis based on the pairwise structural deviation:

```
# Calculate RMSD
rd <- rmsd(pdb)

# Structure-based clustering
hc.rd <- hclust(dist(rd))
grps.rd <- cutree(hc.rd, k=3)
```

Now we can plot our PCA results colored by conformational clusters. The resulting plot is a low-dimensional representation of the conformational variability within the ensemble of PDB structures. The plot is obtained by projecting the individual structures onto two selected PCs (e.g. PC-1 and PC-2). These projections display the inter-conformer relationship in terms of the conformational differences described by the selected PCs.

```
plot(pc.xray, 1:2, col="grey50", bg=grps.rd, pch=21, cex=1)
```

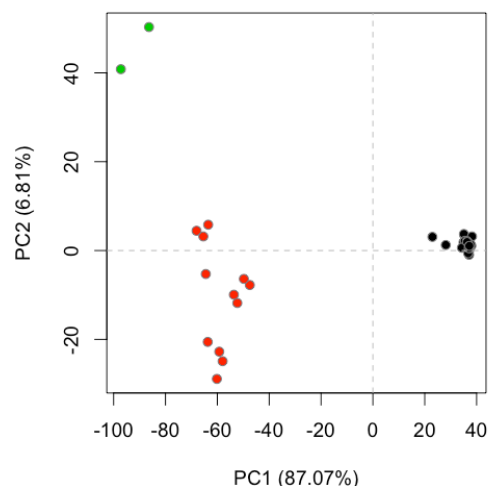
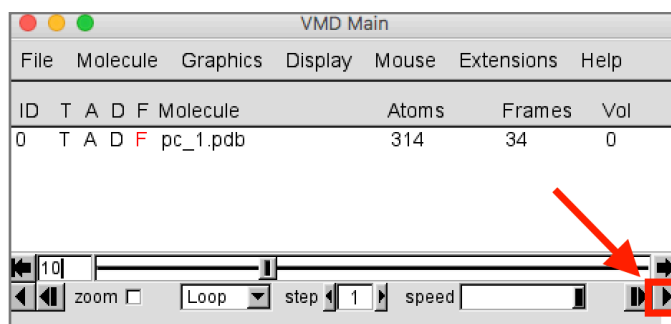


Figure 14: Projection of Adenylate kinase X-ray structures. Each dot represents one PDB structure.

To visualize the major structural variations in the ensemble the function **mktrj()** can be used to generate a trajectory PDB file by interpolating along a give PC (eigenvector):

```
# Visualize first principal component
mktrj(pc.xray, pc=1, file="pc_1.pdb")
```

You can open this file, `pc_1.pdb`, in VMD, chose the “Drawing Method” **Tube** and “Coloring Method” **Index**. Then click the play button shown below to animate the structure and visualize the major structural variations along PC1.



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 session, 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 solved protein structures continues to grow, *in silico* docking will play an increasingly important role in the drug discovery process.

Q8: What one part of this session or associated lecture material is still confusing?
If appropriate please also indicate the question number from this instruction pdf and answer the question in the following anonymous form: <https://goo.gl/forms/u5cPkBag5lBhgew42>

References:

1. Brik A, Wong CH. "HIV-1 protease: mechanism and drug discovery". *Org. Biomol. Chem.* (2003) 1:5–14.
<http://pubs.rsc.org/en/Content/ArticleLanding/2003/OB/b208248a>
2. Wensing AM, van Maarseveen NM, Nijhuis M. "Fifteen years of HIV Protease Inhibitors: raising the barrier to resistance." *Antiviral Res.* (2009).
<http://www.sciencedirect.com/science/article/pii/S0166354209004902>
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.
<http://www.jbc.org/content/269/42/26344.long>
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).
<http://onlinelibrary.wiley.com/doi/10.1002/jcc.21334/abstract>
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.
<http://www.pnas.org/content/98/18/10037>

Appendix I: Useful resources/links

This document and all associated input and output files are available online at <http://thegrantlab.org/teaching/> under "**BIMM-143** (*Lecture 12, Structural Bioinformatics 2*)". All required software is freely available from the sites listed below.

VMD	http://www.ks.uiuc.edu/Research/vmd/
AutoDock Tools	http://mgltools.scripps.edu/downloads
AutoDock Vina	http://vina.scripps.edu/download.html

NOTE: For installing AutoDock Tools on newer **mac OS** versions you will need to "control-click" (i.e. hold down the control key as you click your mouse on the downloaded .dmg file) to open the disc image and then select *Open* to begin the install.

Bio3D development version with interactive 3D viewers is available for **MAC** via

```
download.file("https://tinyurl.com/bio3d-mac", "bio3d.tar.gz")
install.packages("bio3d.tar.gz", repos = NULL)
```

And for **windows** via:

```
install.packages("https://bioboot.github.io/bggn213_S18/class-
material/bio3d_2.3-4.9000.zip", repos = NULL)
```

With typical usage within your Rmd documents similar to the following code snippet:

```
rgl.pdbs <- view(pdbs)
rgl::rglwidget(elementId = "rgl.pdbs")
```

Appendix II: Basic Unix commands

ls	# list files in current directory.
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 file to the screen with more navigation
mkdir dirname	# make a new directory/folder

Windows equivalents

dir	# list files in current directory
CD dir	# change directory to the directory 'dir'
cd	# print the current working directory on the screen
del file	# delete (remove) 'file'
MD dirname	# make a new directory/folder
	# To run autodoc vina on windows
"%Program Files (x86)%\The Scripps Research Institute\Vina\vina.exe" --	
config config.txt --log log.txt	

Appendix III: Frequently Asked Questions (FAQ)

1. What library should I use for screening?

If you want to try and find novel compounds, you probably want to use a library designed for diversity, one which probes a large 'chemical space.' If there are small molecules which are

known to bind to your macromolecule, you may want to construct a tailored library of related compounds.

2. How much computational time should be invested in each compound? How many dockings, how many evaluations?

It depends on your receptor and on the computational resources available to you. One recent successful AutoDock Virtual Screening used 100 dockings with 5,000,000 evaluations per docking per compound.

3. How do I know which docking results are 'hits'?

When the results are sorted by lowest-energy, the compounds which bind as well as your positive control or better can be considered potential hits. (Remember to allow for the ~2.1 kcal/mol standard error of AutoDock). If you have no positive control, consider the compounds with the lowest energies as potential hits.

4. What's the best way to analyze the results?

Sort them by lowest energy first, then use VMD and Bio3D to inspect the quality of the binding.

5. Will I need to visualize the results with the best energies?

Generally it is wise to inspect the top 30 to 50 results. Some people advocate visually inspecting the top 100-400 hits.

6. What should I look for when I visualize a docked compound?

The first thing to check is that the ligand is docking into some kind of pocket on the receptor. The second is that there is a chemical match between the atoms in the ligand and those in the receptor. For example, check that carbon atoms in the ligand are near hydrophobic atoms in the receptor while nitrogens and oxygens in the ligand are near similar atoms in the binding pocket. Check for charge complementarity. Check whatever else you may know about your particular system: for instance, if you know that the enzymatic action of your protein involves a particular residue, examine how the ligand binds to that residue. In the case of HIV protease, good inhibitors bind in a mode which mimics the transition state.

7. How would we scale up to do virtual screening

In this class we prepared the ligand file using ADT, a graphical user interface. It is not reasonable to try to prepare thousands of ligand files using a graphical user interface. Tasks of this magnitude must be automated. We can use *prepare_ligand4.py*, a python script in the AutoDockTools module and run it via a Unix foreach loop. Alternatively you could write your own R function with bio3d to automate this task.

You can learn more about virtual screening with autodoc from the following review article:

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3083070/>