Teach-Discover-Treat 2014, Part 2; Molecular Docking with DOCK 3.7

Ryan G. Coleman[1], Joel Karpiak[1]
[1]Department of Pharmaceutical Chemistry, University of California San Francisco. San Francisco CA 94158.

Abstract

An anti-malaria protein target, lysyl tRNA synthetase, is used as a target for molecular docking. Molecular docking is described, using DOCK3.7 as a system for virtual screening. A pose of a known inhibitor, cladosporin, is predicted. Predicted affinity values for a held-out test set TCAMS (Tres Cantos Anti Malarial Set) are determined based on the docking energy, even though this approximation has many possible problems. Finally, prospective hit-picking against the target is conducted for fragments and small leadlike molecules from the purchasable eMolecules database.

-1. Philosophy.

Molecular Docking is the process of using a protein structure to evaluate the position and energy of small molecules in the binding site of the structure. Many approximations must be made in consideration of computer time, but a good program will use a reasonable energy function and consider many thousand poses of each small molecule relative to the protein binding site. Many programs for molecular docking exist, in this tutorial we use DOCK 3.7, a descendant of the original UCSF DOCK program. DOCK 3.7 has the advantage of pre-built small molecule databases, so docking is relatively easy and very fast once the small molecule databases have been built. In this tutorial, you will have many options to use pre-built ligand databases, or for those of you who want to install & test the database generation procedure you can do it yourself. Once DOCK 3.7 has done its calculations, the results can be examined visually with many programs, we will use UCSF Chimera for this.

In this project, we will be docking to an enzyme called lysyl tRNA synthetase. This enzyme is present in humans and plasmodium falciparum (the organism that carries malaria), however the genes (and therefore the proteins) are different. Structures of both have been solved with crystallography, and cladosporin has been identified as a selective hit. We will use molecular docking to hypothesize a pose of cladosporin to the plasmodium falciparum lysyl tRNA synthetase structure, we will proceed to use molecular docking to screen the ‘held out test set’ called TCAMS. Ideally, some of our predicted hits from virtual screening will work experimentally. Finally, we will dock purchasable molecules from eMolecules to the binding site and identify additional molecules that could be tested experimentally.

http://www.tdtproject.org/challenge-2---new-anti-malaria-compounds.html

0. Downloading & installing software.

DOCK3.7 runs on many flavors of GNU/Linux, as do all the accessory tools. Some popular versions include CentOS, Ubuntu and Debian. GNU/Linux is the most commonly used operating systems on large clusters of computers, so hopefully you have access to a small cluster already, but it is possible to install everything on a desktop GNU/Linux system. Visit http://dock.compbio.ucsf.edu/Online_Licensing/dock_license_application.html and apply for an academic/non-profit license for DOCK3.7, unfortunately this is the only license available at this time. You will also need academic/non-profit licenses for OpenEye tools OMEGA and the OpenEye Toolkit, ChemAxon's cxcalc and AMSOL from the Univ. of Minnesota if you want to build ligands on your own (alternatively you can use the ones built and available online as described later). Extra scripts for the TDT 2014 Challenge are located in https://github.com/ryancoleman/tdt2014-part2/, you can download, modify, etc. as these scripts are released as GPLv2. The DOCK3.7 documentation is at https://sites.google.com/site/dock37wiki/ which supplements this tutorial. UCSF Chimera, used as the visualization program, is available here: http://www.cgl.ucsf.edu/chimera/download.html

Downloading & installing this software is more difficult than the rest of this tutorial! For this reason, the results of each step are saved in the github repository where possible, so that you can follow along without being able to run any given piece of the software system.

1. Preparation of crystal structure for docking

The PDB (Protein Data Bank) contains the three dimensional coordinates for every solved protein structure. You can search by PDB code or by protein name (but many proteins have more than one name, a historical fact usually due to multiple scientists naming proteins before they knew which was which). In this case, you can search for PDB code 4H02 (that is a zero not a letter O) or just go to this URL: http://rcsb.org/pdb/explore/explore.do?structureId=4H02 to learn more about the structure. Since this protein has no ligand in the binding site, we will have to align it with a structure that does. Thankfully, the human ortholog of this protein has a ligand in the active site which we can use. In this case, you can search for PDB code 3BJU or just go to this URL: http://www.rcsb.org/pdb/explore.do?structureId=3bju

For docking, we are only going to use a single chain of the tetramer, as each ligand binds to a single monomer of the protein (though the protein can form a dimer or tetramer according to the literature). We make a file that looks like this:

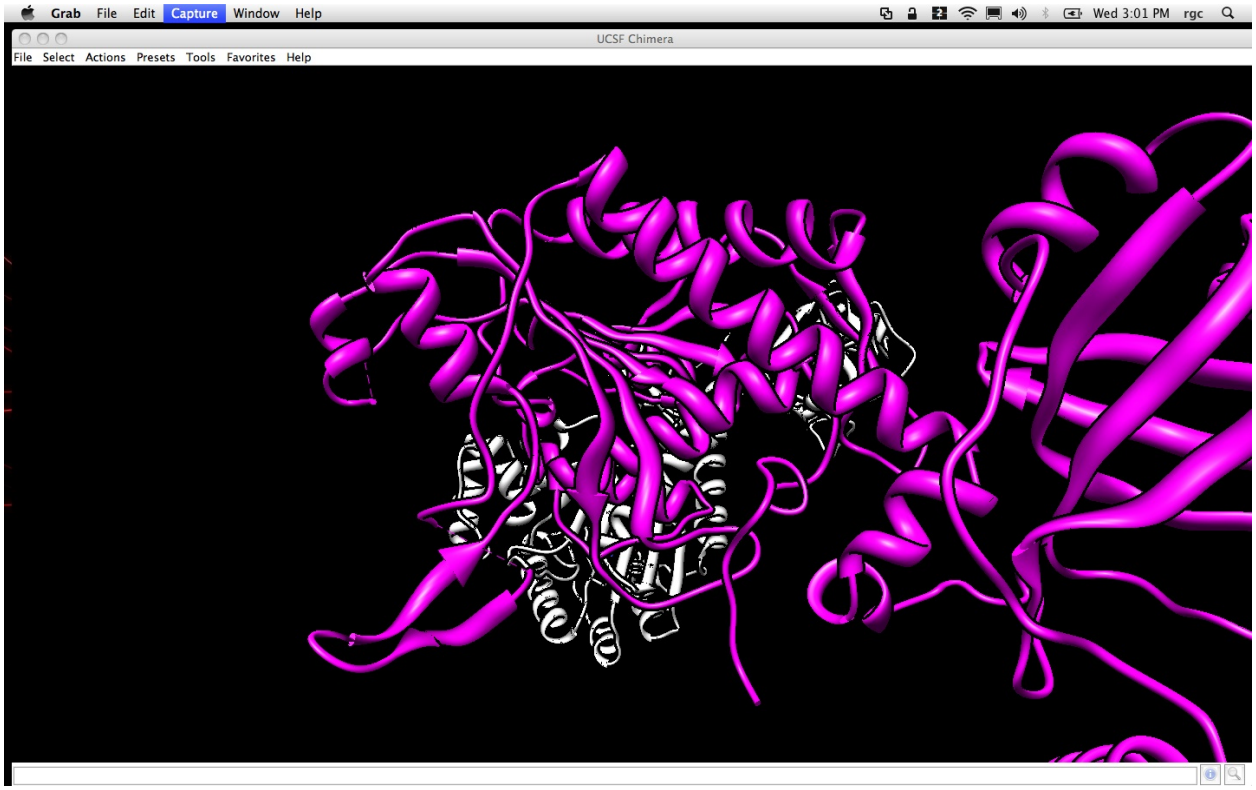
3BJUA
4H02A

Or you can download it from https://github.com/ryancoleman/tdt2014-part2/blob/master/step1files/pdb_code_list.txt Once you

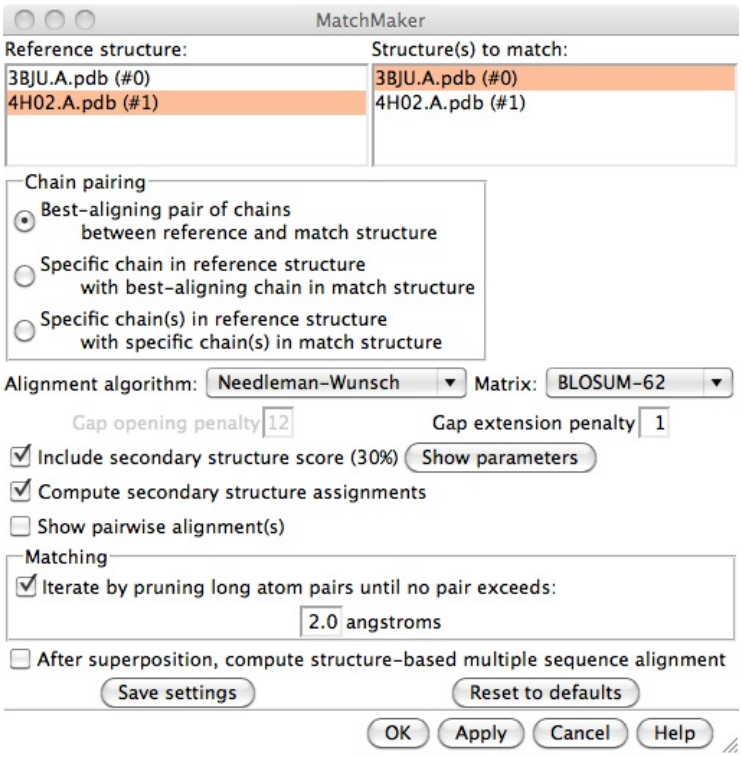
have this file and you've downloaded the scripts from https://github.com/ryancoleman/tdt2014-part2/tree/master/scripts, we want to run this command (the ` are backticks and are not part of the command):

```
`grab_pdb_chain.py pdb_code_list.txt`
```

You should have 4 PDB files, two complete files and 3BJU.A.pdb and 4H02.A.pdb which just contain chain A. You can compare these to the files here: https://github.com/ryancoleman/tdt2014-part2/blob/master/step1files/3BJU.A.pdb and https://github.com/ryancoleman/tdt2014-part2/blob/master/step1files/4H02.A.pdb Once you have these files, we need to align 3BJU (the human version) onto 4H02 (the plasmodium falciparum version). We will use UCSF Chimera's MatchMaker for this. http://www.cgl.ucsf.edu/chimera/docs/ContributedSoftware/matchmaker/matchmaker.html. First, load up both proteins using the File->Open dialogs, and UCSF Chimera should look like this:

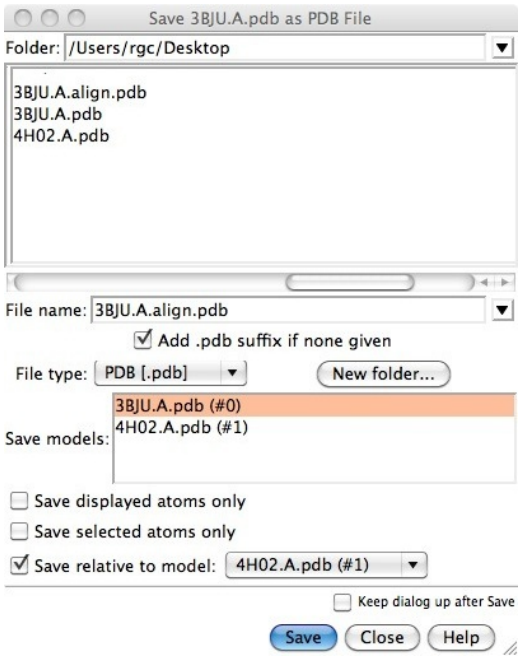


We want to align the human protein (3BJU) onto the p. fal. protein (4H02), so we open MatchMaker with Tools-Structure Comparison->MatchMaker. We want to select the appropriate chains, but we'll try using the default alignment settings first.



Once you've set it up, click "OK" and Chimera should align the two proteins and look like this:

Great! The two proteins are on top of each other. Remember, we are doing this so we can use the human ligand (Adenosine Tri-Phosphate or ATP) as a ligand for the p.fal. structure in docking preparation. We want to save the human protein now using File->Save PDB…



Here, we want to save 3BJU.A.pdb as 3BJU.A.align.pdb and we want to save it relative to 4H02.A.pdb (the bottom checkbox). Once you have your own 3BJU.A.align.pdb file, you can compare it to this one:
https://github.com/ryancoleman/tdt2014-part2/blob/master/step1files/3BJU.A.align.pdb

Now, we're ready to extract the ligand from the human protein. For this step, we could copy the HETATM lines from the 3BJU.A.align.pdb file, change them to ATOM lines and put them in a new file. Or, we can use this command, using two common tools in GNU/Linux, grep and sed.

```
`grep HETATM 3BJU.A.align.pdb | sed 's/HETATM/ATOM  /g' >
xtal-lig.pdb`
```

If you look at the binding site, you'll notice one sidechain with 2 positions. The following command removes that.
https://github.com/ryancoleman/tdt2014-part2/blob/master/scripts/pdbMostOccupied.py

```
`pdbMostOccupied.py 4H02.A.pdb rec.pdb`
```

Again, you can check your work at this stage by comparing to these files:
https://github.com/ryancoleman/tdt2014-part2/blob/master/step1files/xtal-lig.pdb and
https://github.com/ryancoleman/tdt2014-part2/blob/master/step1files/rec.pdb

Now we're going to run the docking preparation part of DOCK3.7. This program is called blastermaster.py and is part of the DOCK3.7 download.

```
`blastermaster.py -v >& blaster.log.txt &`
```

We run it like this so that the output from the program ends up in the log file called blaster.log.txt. The -v flag asks for extra verbose debugging. There is more information here: https://sites.google.com/site/dock37wiki/home/protein-target-preparation This process can take up to half an hour depending on your machine.

Since the data for docking (mostly grids of precomputed energy functions) are a bit large, they are available here: https://github.com/ryancoleman/tdt2014-part2/tree/master/step1files/blaster

What went on when you ran blastermaster? Quite a lot actually! If you read the blaster.log.txt you can see exactly what goes on. The grand philosophy of what is going on is that the binding site is being identified by the ligand, and the energy grids for each of the three energy functions are being constructed. Once these grids are constructed, docking proceeds very quickly since each atom can be scored against the grids for each energy function (van der Waals, ligand desolvation and electrostatics). While using grids is an approximation, it does make DOCK3.7 very fast.

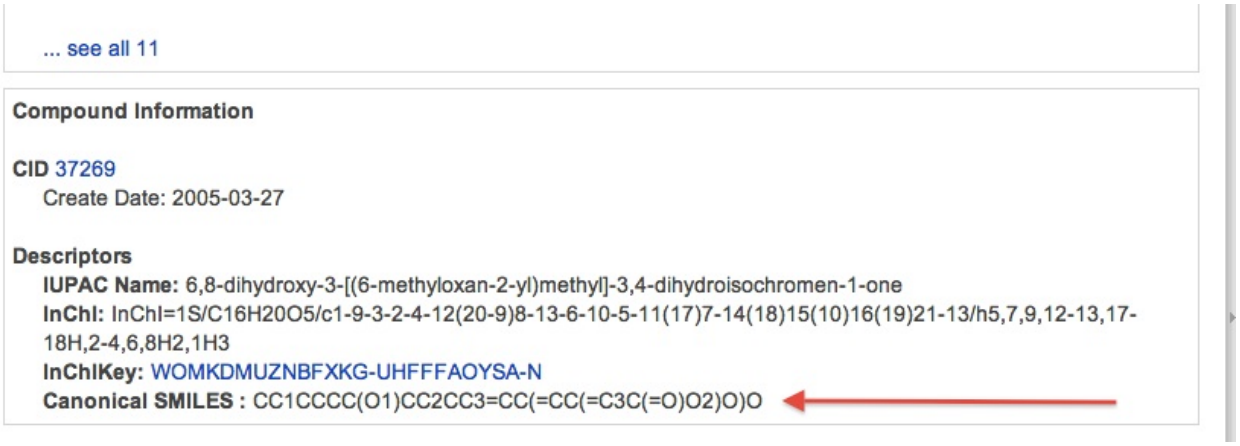
Once you've got these files, you're ready to build ligands and then move on to docking.

Preparing the Ligands

Cladosporin Database Building

Cladosporin has been identified as a selective inhibitor of the p.fal. ortholog of lysyl tRNA synthase. To build it for docking, we need to start with a SMILES string. We searched PubChem for cladosporin and found this page: http://pubchem.ncbi.nlm.nih.gov/summary/summary.cgi?cid=37269

Scroll down a bit to copy and paste the SMILES string



Copy this into a file. In the second column, you want to put some sort of identifier. Pick something short and descriptive, I chose CLADOSPORIN0. Save this file, feel free to compare it to this one https://github.com/ryancoleman/tdt2014-part2/blob/master/step1files/cladosporin.smi

Once you have the input file, you can build a docking database with the downloaded scripts (assuming you also set up the accessory programs Omega, AMSOL, cxcalc & sge for running the cluster). If you aren't using them, skip to the alternate heading "Downloading the Ligands". Use the program (backticks or ` surrounding the command are not typed):

```
`db2start.e.cxcalc.sh input.smi`
```

to start the ligand building process. Once all jobs have finished, a few molecules will not be correctly built, but can be rescued by simply re-running them. To do this, go into the marvin directory and then run the following command, like this:

```
`cd marvin`
`subprepG2.e.cxcalc.db2.redo.csh`
```

Which will start a few more jobs. Once they are all complete again, return to the starting directory and compile the many ligand databases into a few big files.

```
`cd ..`
`db2end-prefix.py name`
```

You should use a name here that is descriptive and meaningful instead of just name. For this project 'cladosporin' is a good one. Once that is done you should have a file called cladosporin-0000001.db2.gz Again, feel free to compare this file to https://github.com/ryancoleman/tdt2014-part2/blob/master/step1files/cladosporin-0000001.db2.gz

TCAMS Database Building

The TCAMS (Tres Cantos Anti Malarial Set) database is a set of compounds that were screened against malarial cells. The data has been withheld from participants in this challenge, so our goal is to predict which of the compounds did bind to the p.fal. lysyl tRNA synthetase protein.

More about the database is available here:
https://www.ebi.ac.uk/chemblntd/
https://www.ebi.ac.uk/chemblntd/download/#tcams_dataset

Eventually, the file we want to start with is here:

ftp://ftp.ebi.ac.uk/pub/databases/chembl/ChEMBLNTD/set1_gsk/chemblntd_gsk.txt.gz

The file is packaged with the common 'gzip' compression algorithm, which you can tell from the .gz extension. So, we use 'gunzip' to unzip it:

`gunzip chemblntd_gsk.txt.gz`

For various reasons, we only want the SMILES string along with the identifier. We want to use standard command-line tools to get the relevant columns, but there is sometimes an extra column on some rows. To remove this, run:

`sed -i 's/COMMERCIAL//g' chemblntd_gsk.txt`

Again the backticks ` aren't necessary, but the single quotes around the sed command are. This sed command simply looks for every occurrence of COMMERCIAL and removes them. We also want to remove the first line (the header line), which we do using any editor (GNU nano is the best, open the file, control-k cuts the first line, control-s saves the file!)

This file can be checked against this file here:
https://github.com/ryancoleman/tdt2014-part2/blob/master/step1files/chemblntd_gsk.txt

Now, we want to get the last column (the SMILES) and the 1st column (the TCAMS identifier), so we run (note that awk prints the last column as $NF means Number of Fields):

`cat chemblntd_gsk.txt | awk '{print $NF, "T"$1}' > tcams.smi`

(Note, for historical reasons, we have added T before every TCAMS identifier or some parts of the older DOCK3.7 code won't work. This is most unfortunate.)

You can check the tcams.smi file against this one:
https://github.com/ryancoleman/tdt2014-part2/blob/master/step1files/tcams.smi

Now that you have the input SMILES file, you can proceed with building this much larger docking database:

`db2start.e.cxcalc.sh tcams.smi`

to start the ligand building process. Once all jobs have finished, a few molecules will not be correctly built, but can be rescued by simply re-running them. To do this, go into the marvin directory and then run the following command, like this:

`cd marvin`
`subprepG2.e.cxcalc.db2.redo.csh`

Which will start a few more jobs, just like before. Once they are all complete again, return to the starting directory and compile the many ligand databases into a few big files.

```
`cd ..`
`db2end-prefix.py tcams`
```

Or some other name. This time, you'll get a lot of files (I got 581!). This is a lot of small molecules, but the hard part (building the ligand database) is over and docking is now relatively fast & easy.

Now, when you build this many files, sometimes one or two ligands aren't built correctly. I wish all these problems didn't exist, but they do! For now, run this script: https://github.com/ryancoleman/tdt2014-part2/blob/master/scripts/remove_incomplete_db2gz.py on all your docking database files and the wrong parts of each file will be removed.

Downloading the Ligands

If you can't get the software working, or want to skip ahead while the ligands build, or just want to move on with the tutorial, you can simply download this file:

https://github.com/ryancoleman/tdt2014-part2/blob/master/step1files/cladosporin-0000001.db2.gz

For the TCAMS data, you can download it here (it is too big for github):-

http://tools.bkslab.org/~rgc/tdt2014/tcams/

You'll want to download all the files in that directory.

(a) Binding pose for cladosporin

More about running DOCK3.7 is here: https://sites.google.com/site/dock37wiki/home/running-dock3-7

Now that we've prepared a protein and ligand database for cladosporin, docking is easy! First, set up the docking run with the following command (part of the DOCK3.7 download):

```
`setup_db2_lots.py 1 cladosporin /path/to/cladosporin_databases/`
```

Then run it with

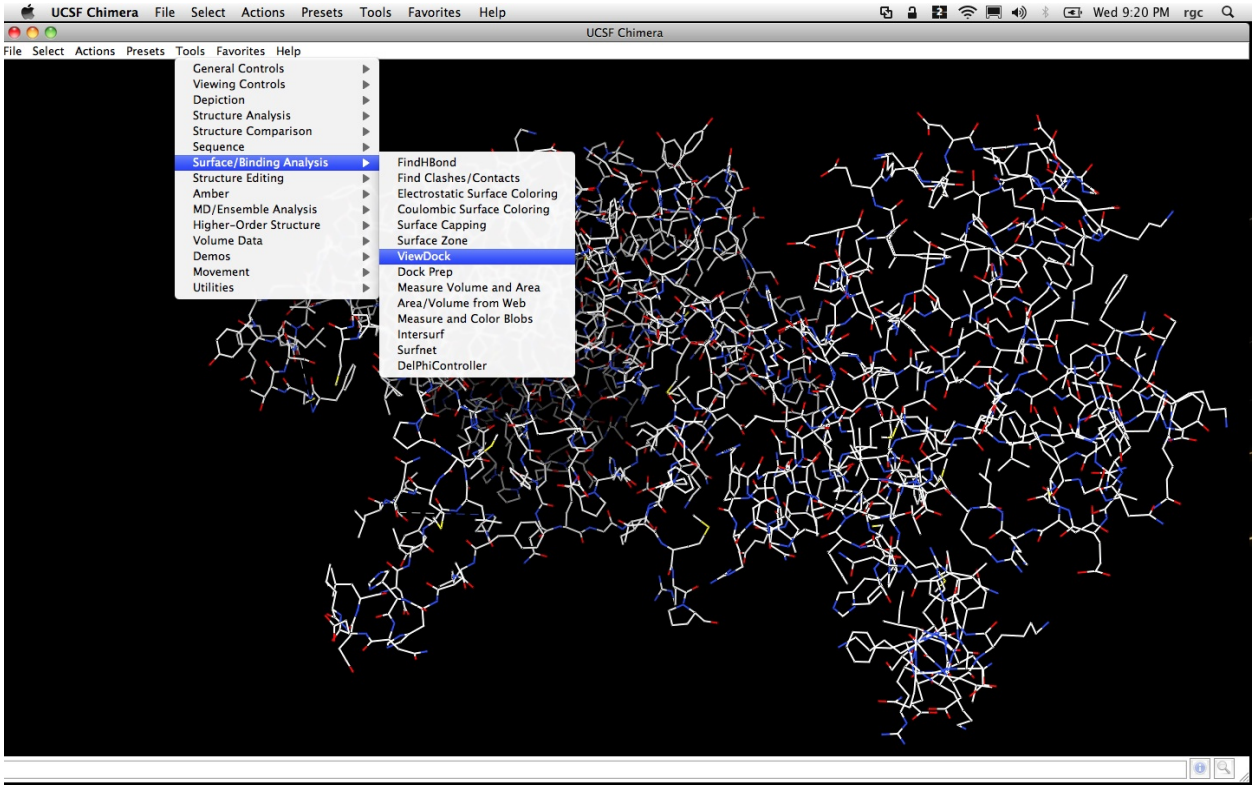
```
`submit.csh`
```

Alternatively, you can just run dock itself in the directory cladosporin, since there is only one docking run to do, you don't need a cluster setup, etc.

Once the run is done (it took less than 3 minutes on my computer), you can run the following 2 programs to get a file with the best pose for cladosporin. DOCK3.7 came up with hundreds of thousands of possible poses of cladosporin and scored them all.

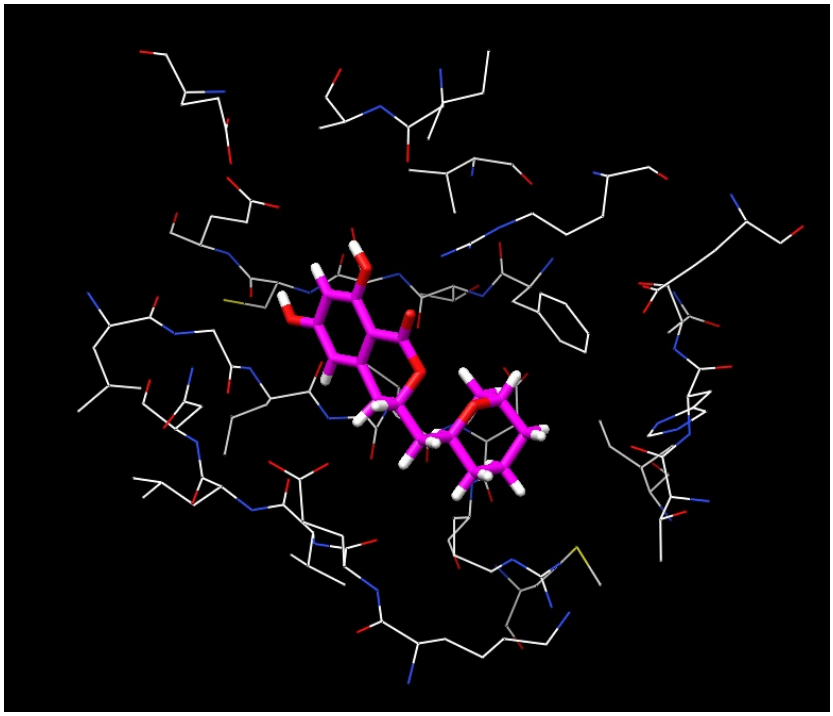
```
`extract_all.py`
`getposes.py -o cladosporin.mol2`
```

The best pose is now in the cladosporin.mol2 file. Again, you can check it https://github.com/ryancoleman/tdt2014-part2/blob/master/step1files/cladosporin.mol2 Let's look at it with UCSF Chimera. Open the protein (rec.pdb). I like to use Presets->Interactive 2 (all atoms). Once you've opened rec.pdb you want to use Tools->Surface/Binding Analysis->ViewDock to open the ligand file.

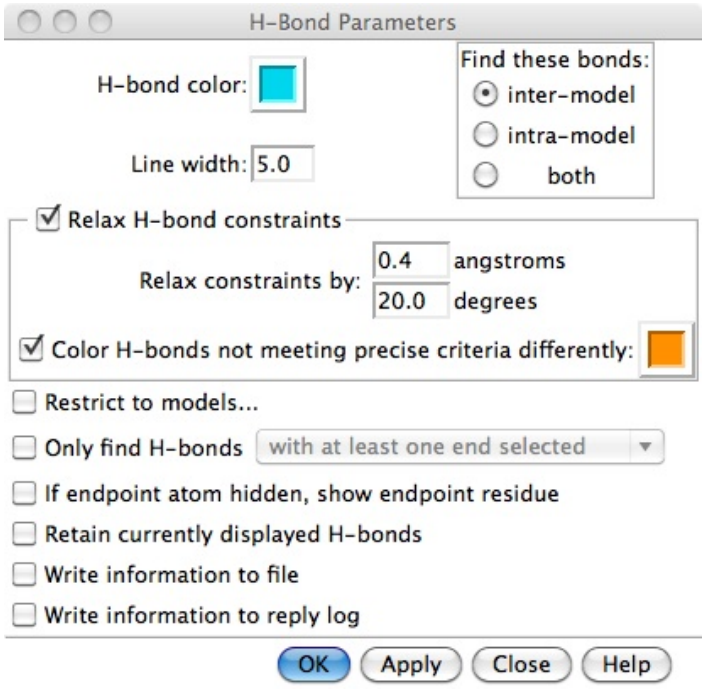


Choose cladosporin.mol2 from the menu.

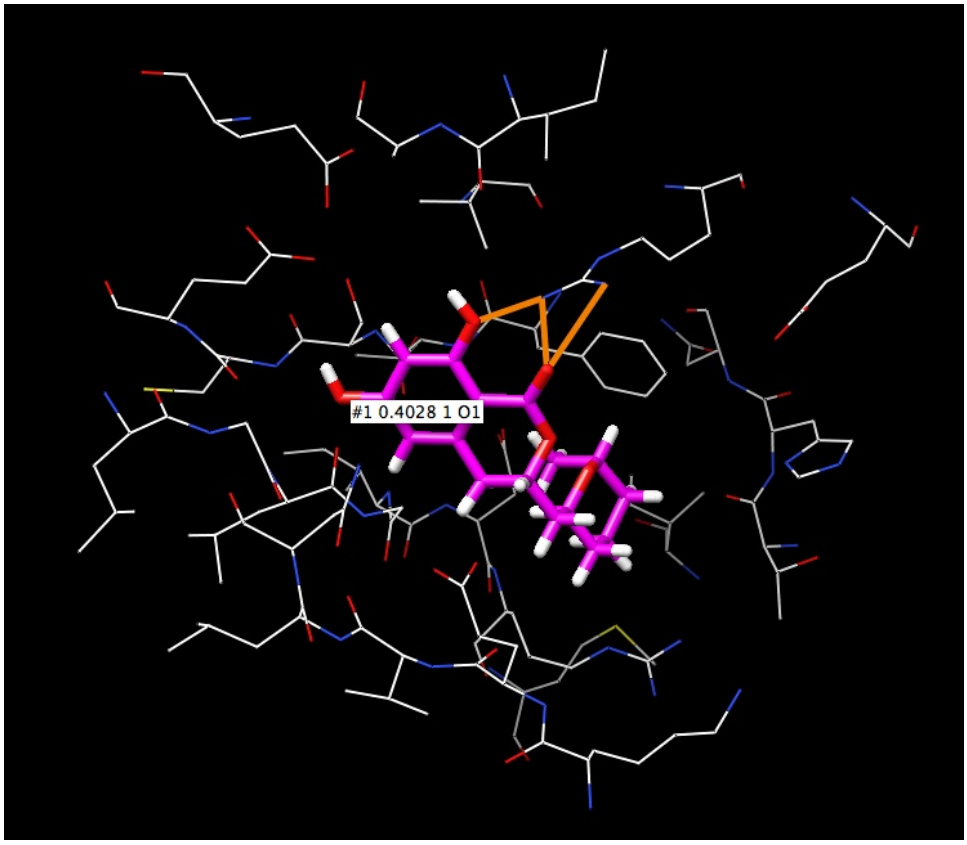
Now, you can see the whole ligand in context of the entire protein. We want to focus on the binding site. Select the ligand by control-clicking on an atom in the ligand then press the "up" arrow key to select the entire ligand. Once you've got it selected, choose Select->Zone, change the first number to 6.0, check the lowest box labeled "Select all atoms/bonds of any residue in the selection zone" and click OK. Once the ligand and the nearby residues are selected, click Action->Atoms/Bonds->Show only. Now you can right click and drag down to zoom in on the binding site and cladosporin. It should look like this (or it does if you download the cladosporin file above). Docking should be deterministic, but different computers, etc. can produce slightly different results.



Does this pose for cladosporin make sense? We always want to sanity check docking results, mostly to discover if we did something wrong in the protein or ligand build process, but also so we can learn what a good potential ligand looks like for virtual screening later. UCSF Chimera has some excellent built in analysis tools, so let's use those. Start with Tools->Surface/Binding Analysis->FindHBond. Set the parameters to inter-atom and show h-bonds as orange if they don't quite meet the criteria.



Once you click OK, you should see something like this, with 3 hydrogen bonds shown:



Does this pose make sense? Well, it isn't bad. The negative polarity on cladosporin is engaged with the ARG330 positively charged residue in the protein. The hydroxyls on cladosporin both point towards the negatively charged GLU346 and GLU308. Even though these are a bit too far to make hydrogen bonds, it is still a reasonable pose. There aren't major electrostatic clashes, and any stranded atoms are close enough to water (facing towards you in this image), that they can be rationalized as well.

To examine the van der Waals clashes, we select the ligand (control-clicking an atom and pressing "up" arrow), then Tools->Surface/Binding Analysis->Find Clashes/Contacts. Click designate at the top and then OK at the bottom. In this pose, no clashes are detected, meaning no atoms between the protein & ligand are too close.

In summary, this is a reasonable pose. Only an experiment (using x-ray crystallography of a protein-ligand complex) can tell us for sure.

If you want to download and open the UCSF Chimera session I made, it is here: https://github.com/ryancoleman/tdt2014-part2/blob/master/step1files/chimera_cladosporin.py Chimera sessions sometimes aren't backwards compatible, but this was made with an older version of Chimera, so hopefully it works with newer versions that you've just installed.

If we want to have a single PDB file with the receptor & cladosporin, run the following commands (convert.py is part of the DOCK 3.7 package that uses OpenEye tools to convert molecules):

```
`convert.py --i=cladosporin.mol2  --o=cladosporin.pdb`
`cat rec.pdb cladosporin.pdb > pfkrs1-cladosporin.pdb`
```

As always, compare your file to this one:
https://github.com/ryancoleman/tdt2014-part2/blob/master/step1files/pfkrs1-cladosporin.pdb

Now, we neglected to show the polar hydrogens on the protein in this file, which are sometimes useful. We could do this instead:

```
`cat working/rec.crg.pdb cladosporin.pdb > pfkrs1h-cladosporin.pdb`
```

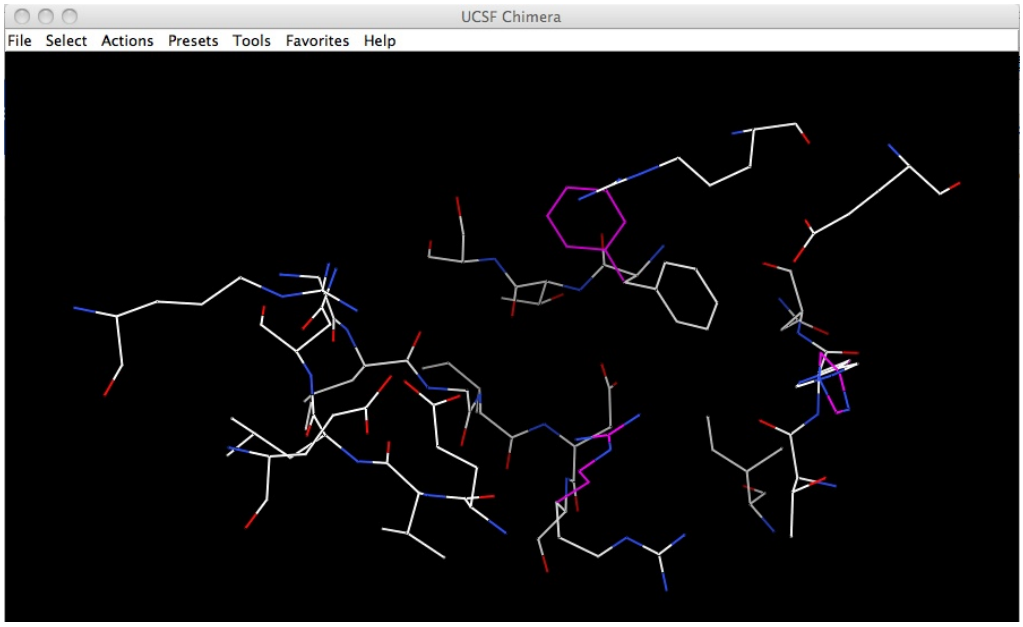
See this version for comparison:
https://github.com/ryancoleman/tdt2014-part2/blob/master/step1files/pfkrs1h-cladosporin.pdb

Adjusting the receptor conformation

The receptor conformation used is in an *apo* state, meaning without any ligand present. Since we had the structure of a human *holo* structure, with ATP present, we decided to examine what would happen if we moved some key sidechains to match the ATP-bound crystal structure.

Let's compare the two receptor conformations. Use the files here for the original:
https://github.com/ryancoleman/tdt2014-part2/blob/master/step1files/blaster_adjusted/rec.pdb
and for the adjusted:
https://github.com/ryancoleman/tdt2014-part2/blob/master/step1files/blaster/rec.pdb

As before, load them both in Chimera and zoom in on the binding site.



Here, the original conformation is in white, the adjusted sidechains are shown in purple. The phenylalanine is the most prominent shift, as the *apo* state clashes with the bound ATP (and presumably, other ligands like cladosporin and new ligands).

As before, we prepare the receptor automatically using blastermaster. You can download the automatically prepared files from here:
https://github.com/ryancoleman/tdt2014-part2/tree/master/step1files/blaster_adjusted

Once that is complete, we dock cladosporin in the same way. To get the best pose, we chose to increase the sampling to 20000 orientations. In the INDOCK file, change

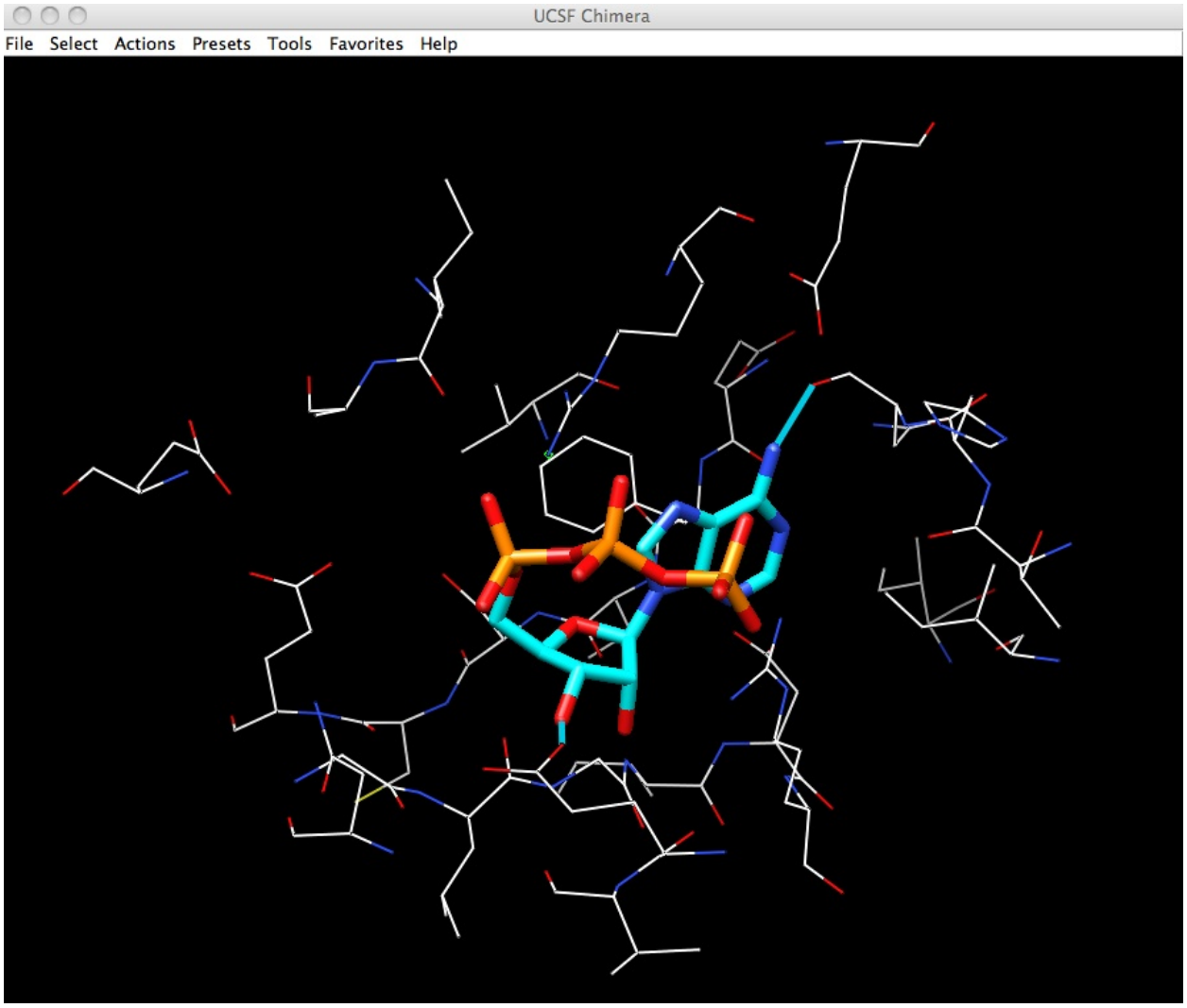
match_goal 5000

to:

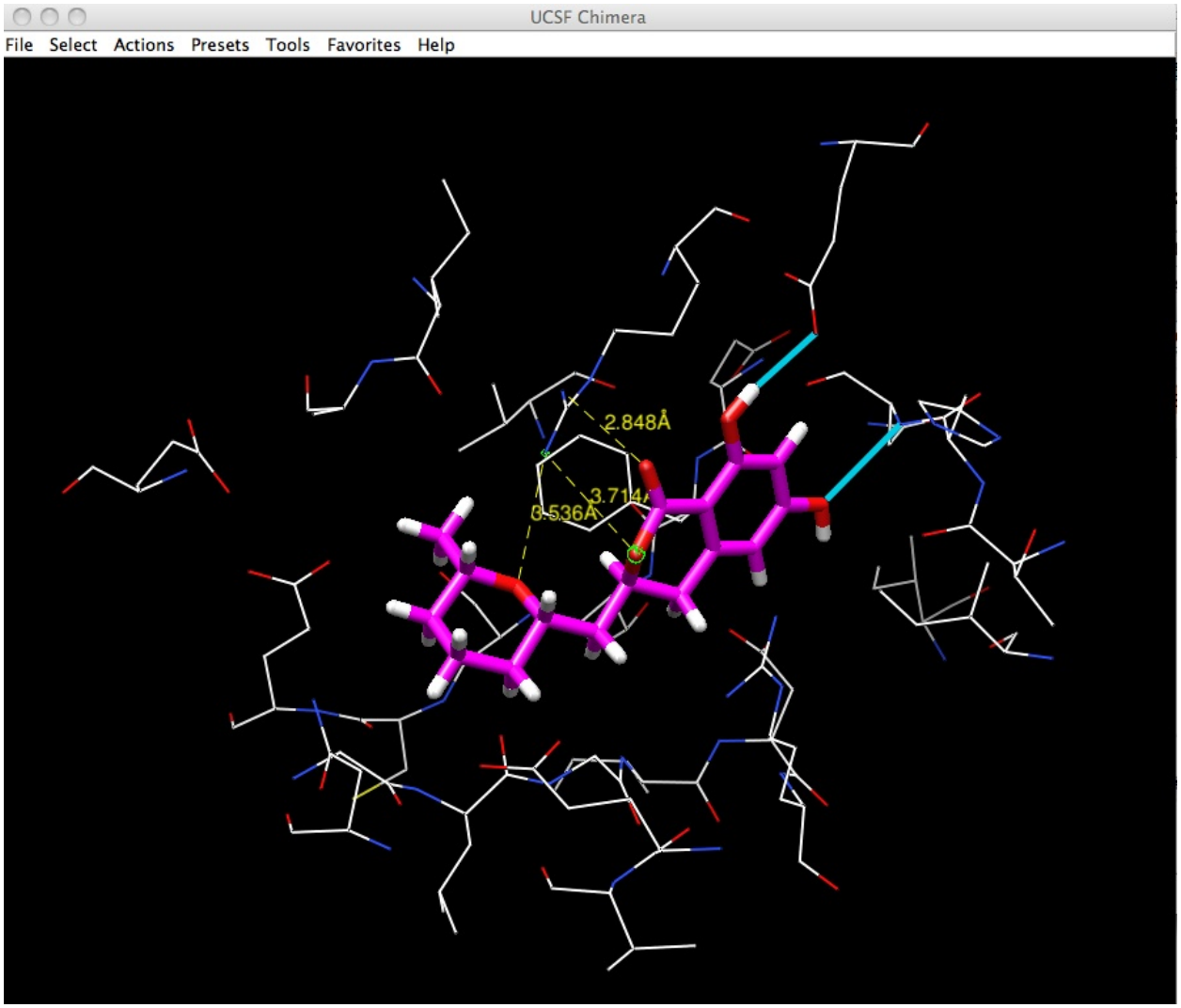
match_goal 20000

You can download the pose here:
https://github.com/ryancoleman/tdt2014-part2/blob/master/step1files/adjusted-cladosporin.mol2

First, see how ATP binds (this is the aligned ATP from the *holo* structure in the adjusted *apo* structure):

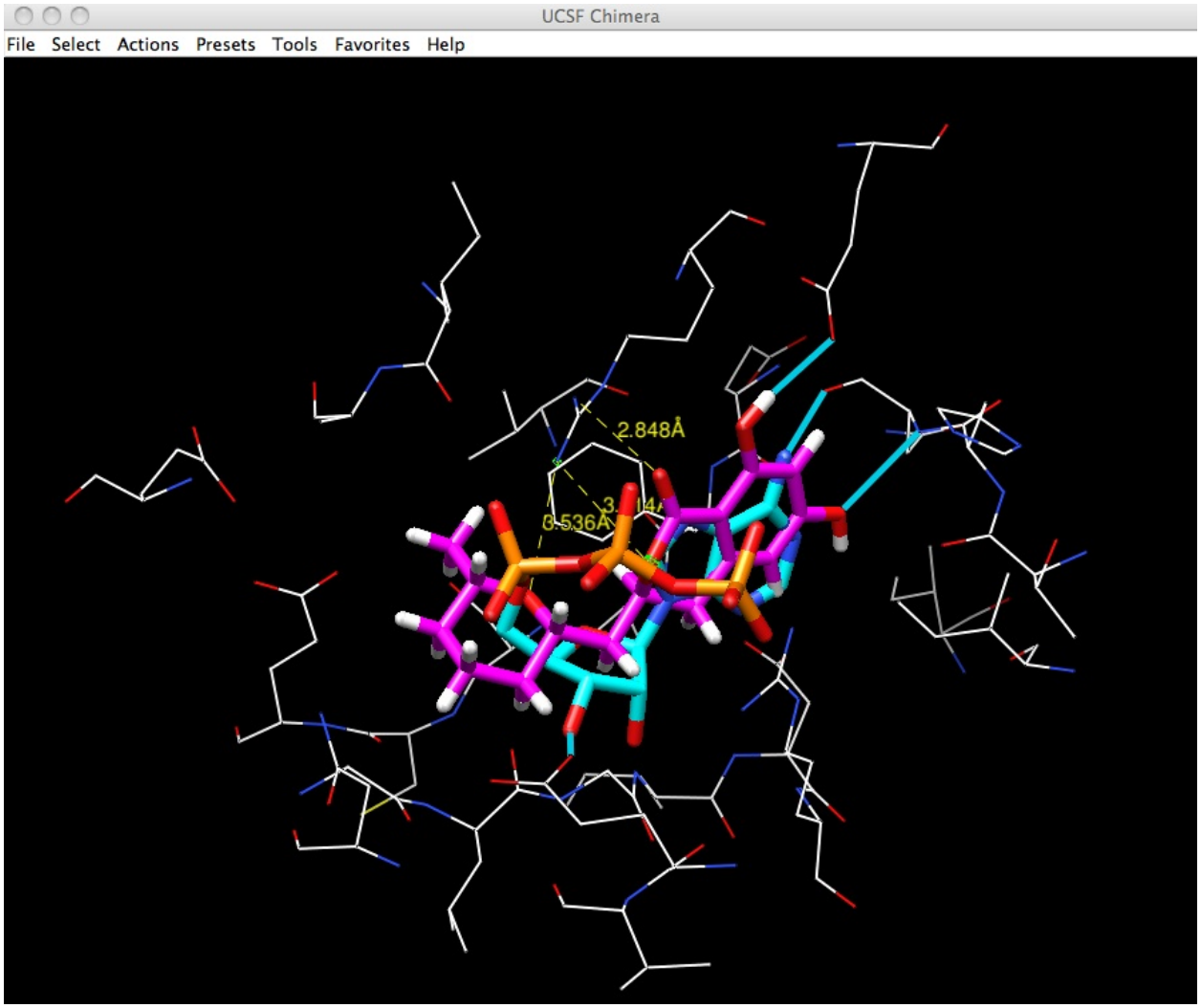


Now, look at the docked pose of cladosporin:



I've had Chimera draw some distances with ctrl-click then ctrl-shift-click and select "Show Distance", as some of the interactions don't show as hydrogen bonds but are likely good polar contacts between the charged arginine and the ligand.

When overlapped with ATP, this pose of cladosporin looks convincing:



As the ring structures align, as do many of the polar groups in the two ligands. We know from experimental data that ATP and cladosporin are competitive for each other, they cannot bind at the same time, so binding in the same site makes sense. The chimera session to examine on your own, as before, is here: https://github.com/ryancoleman/tdt2014-part2/blob/master/step1files/chimera-cladosporin-adjusted.py

For these reasons, we've chosen to use this adjusted structure for all later docking.

As we are going to submit this as the predicted pose for cladosporin, we want to put it in a single PDB file like before. This time, we use the polar hydrogens placed on the receptor, in case those are useful, they can always be removed later.

```
`convert.py --i=adjusted-cladosporin.mol2
--o=adjusted-cladosporin.pdb`
```

```
`cat working/rec.crg.pdb adjusted-cladosporin.pdb >
pfkrs1-cladosporin-adj.pdb`
```

This file is here:
https://github.com/ryancoleman/tdt2014-part2/blob/master/step1files/pfkrs1-cladosporin-adj.pdb

(b) Activity Prediction for TCAMS

Docking this large database is just as easy as docking the small database for cladosporin, it only takes more computer time. Depending on your cluster, you may be able to run each database file built before on a separate machine, which means very little waiting at all. Again, we've decided to use the adjusted positions of the sidechains for this prediction. We will however, only use 5000 as the match goal instead of the 20000 used to generate the best pose for cladosporin, purely in the interest of computer time.

```
`setup_db2_lots.py 1000 tcams /path/to/tcams/database/db2/files/`
 `submit.csh`
```

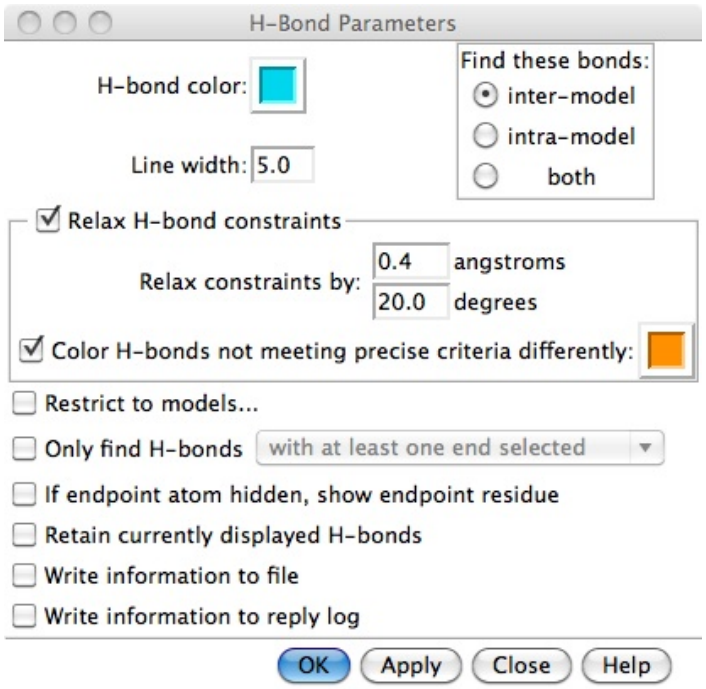
Now, just wait for those jobs to finish (this will take awhile). Then, like before, run

```
`extract_all.py` then
`getposes.py -o tcams.mol2`
```

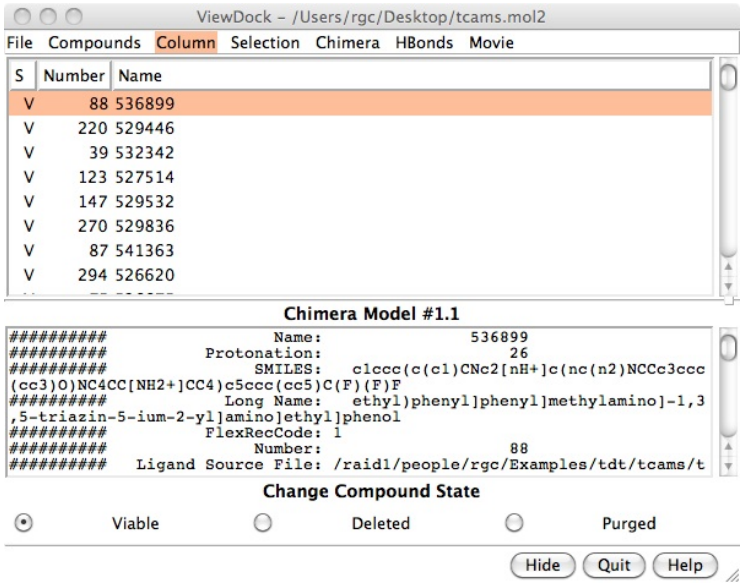
This will, by default, extract the top 500 molecules. As always, you can look & compare this file: https://github.com/ryancoleman/tdt2014-part2/blob/master/step1files/tcams.mol2 Nobody can really look through 500 molecules, and we'll be looking at them just to get a 'feel' for the overall docking accuracy, much in the same way we looked at the pose for cladosporin.

Let's look at it with UCSF Chimera. Open the protein (this time I'll use rec.crg.pdb, available here: https://github.com/ryancoleman/tdt2014-part2/blob/master/step1files/blaster_adjusted/working/rec.crg.pdb or in working/rec.crg.pdb in your files. I like to use Presets->Interactive 2 (all atoms). Once you've opened rec.pdb you want to use Tools->Surface/Binding Analysis->ViewDock to open the ligand file (tcams.mol2).

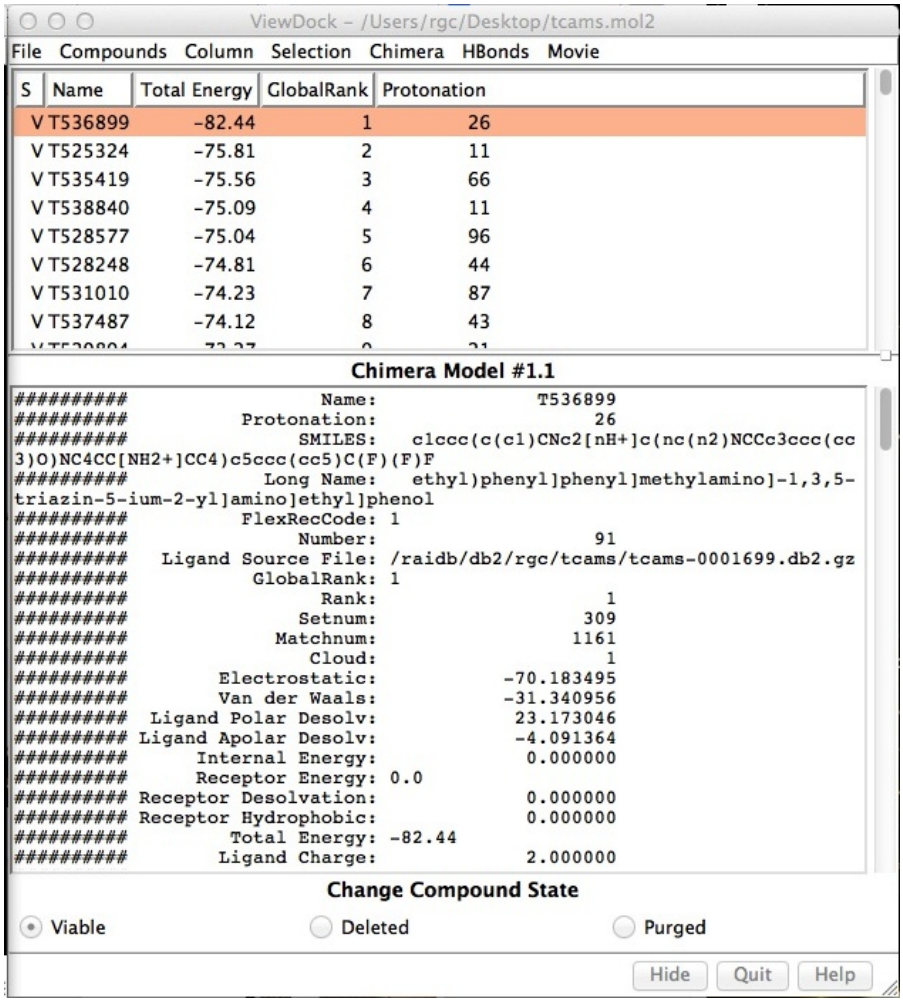
Once you've done that, arrange your viewing window to appropriately view the binding site. For ease of viewing, find & show hydrogen bonds again:



Once you've arranged it, switch to the Viewdock window. Note that it is very powerful and a good way of looking through many molecules (which is why we use UCSF Chimera).

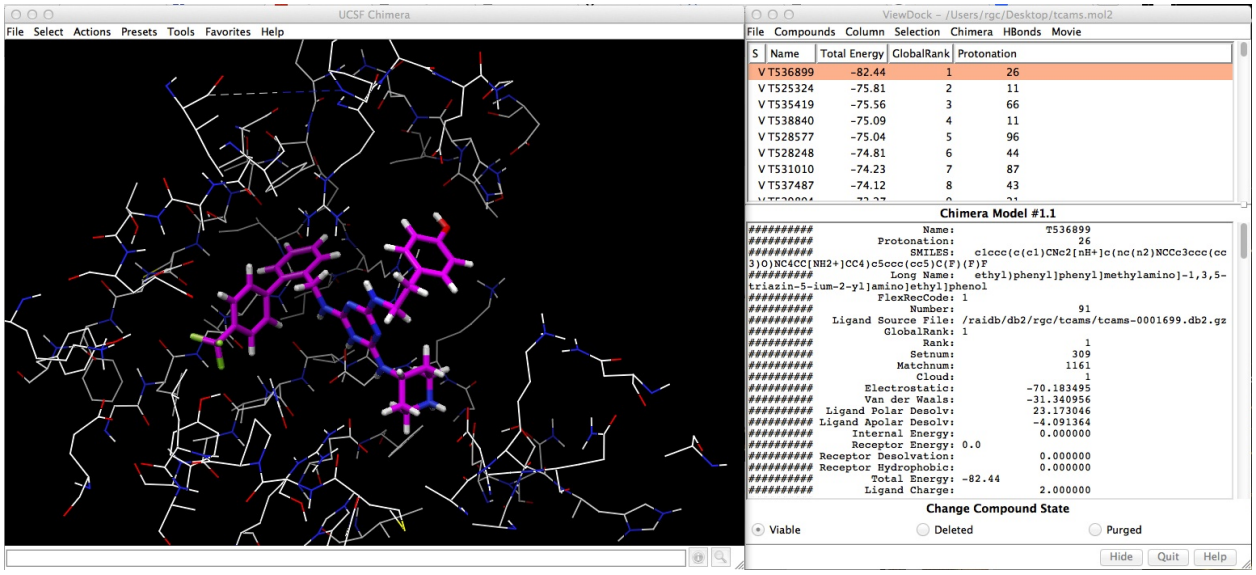


I like to hide/show different columns (Under the Column menu), like this (for instance):

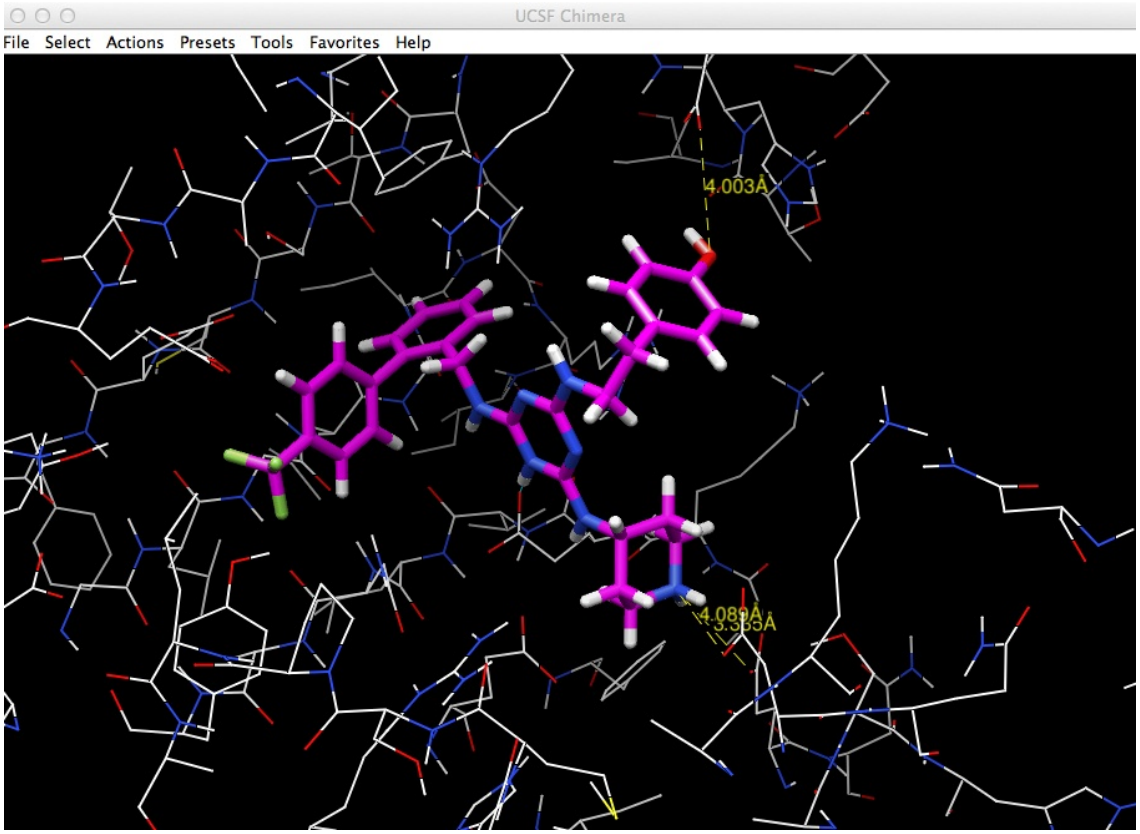


Looking at the many top molecules, note that they are very big. Currently, the DOCK scoring function probably overemphasizes larger molecules more than it realistically should, due to the many approximations and missing energy terms. This causes the largest molecules to get the best scores, even if they may not have a higher affinity in an experiment.

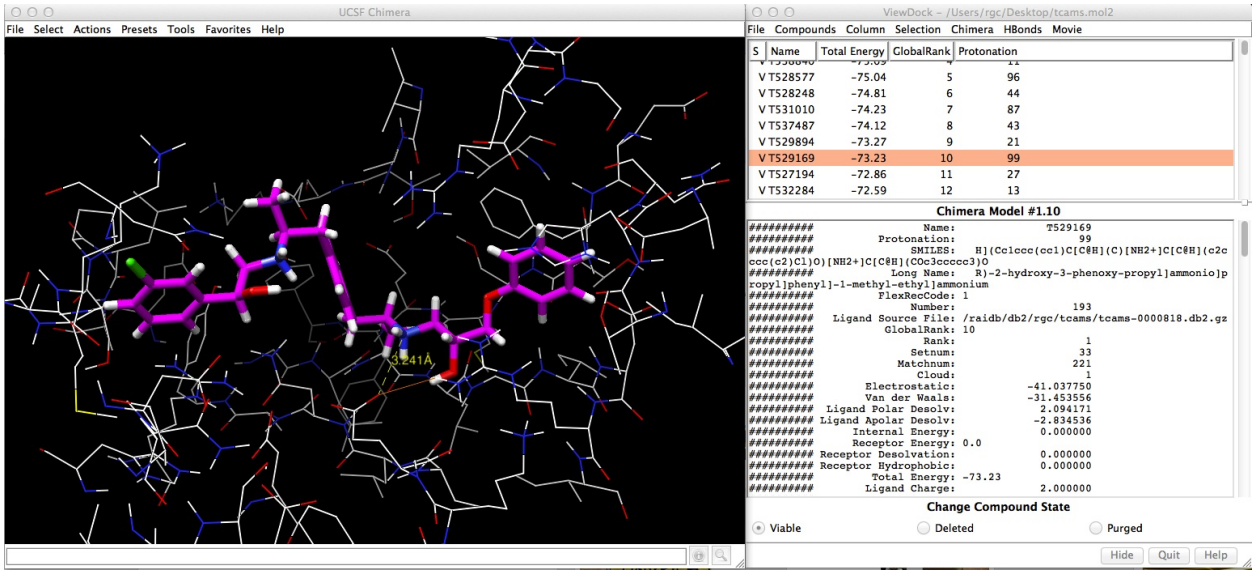
Examining the top molecule, we can see several good hydrogen bonds.



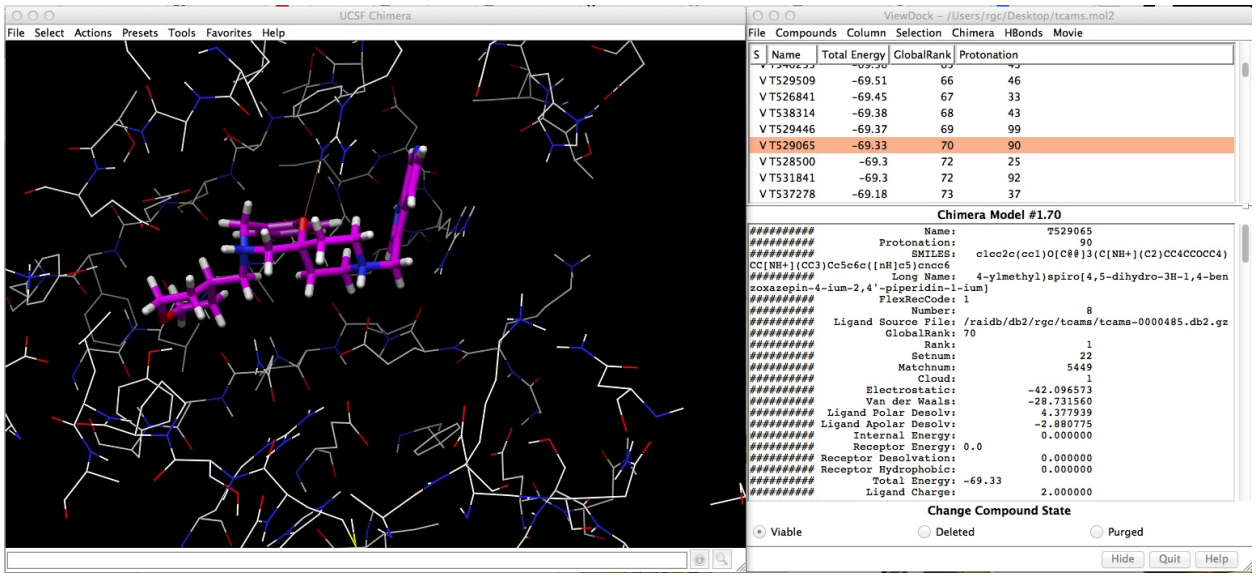
Additionally, I drew in some additional polar contacts not captured by the FindHBond module in UCSF Chimera. All together, this pose of the top-scoring molecule is very good. Most of the polar groups are engaged with complementary polar groups in the protein. Additionally, the putative ligand fills most of the space of the binding site. All these aspects are important for binding and not always captured by the docking energy function, which is why they should be examined visually.



The tenth molecule is exceptionally long and flexible. Again, there is no penalty for this in the scoring function. This molecule is likely scoring better than it should.



The 70th molecule scores very well (out of the over 13,000 molecules in the TCAMS set). In this case, it fills the pocket well and many polar contacts are satisfied.



Overall, though the current energy function has many known deficiencies, we will submit the scores as is, because we do not yet have a better way of adjusting the predicted scores to be more accurate. Again, recent work on SAMPL4 with the DOCK3.7 system showed a surprising correlation between predicted affinity and experimentally confirmed affinity (Pearson R = 0.64).

Of course, looking at the results is great but now we need to submit the predicted activity file. Though the ranges are wrong and the correlations between activity and DOCK energy score are usually weak at best, we will submit the DOCK energy scores as the predicted activity. The

output file format is (identifier, SMILES, activity). The data is in the original tcams.smi file and the activity is in the extract_all.sort.uniq.txt file, we use this script (https://github.com/ryancoleman/tdt2014-part2/blob/master/scripts/make_tcams_output.py) and following command to write the output file for the challenge. Note that this script removes the T prefix we added to the TCAMS identifiers for the output file.

```
`./make_tcams_output.py tcams.smi extract_all.sort.uniq.txt > tcams.predictions.txt`
```

This tcams.predictions.txt file will be part of the submission package. As usual, ours is here: https://github.com/ryancoleman/tdt2014-part2/blob/master/step1files/tcams.predictions.txt

2. Follow-up hit finding

a) Building eMolecules small molecule docking databases

eMolecules has provided a list of all available molecules for purchase for this challenge. While other options & vendors exist, we want to build what we can of the eMolecules database.

Quoting from the TDT2014 Site: “””Download most recent file of commercially available compounds from eMolecules, http://downloads.emolecules.com/ordersc/ and select most recent directory. We recommend you download this in January 2014 for good availability; use the “parent” file (without salts and solvates)”””

Note that there are over 5 million compounds in this file! First, we want to eliminate any duplicates by canonicalizing each smiles string.

```
`convert.py --i=emolecules.smi --o=canon.smi --smioCanonical`
```

After canonicalizing & removing duplicates with awk like this:

```
`cat canon.smi | awk '!($1 in a) {a[$1]; print}' > uniq.smi`
```

Now, we notice that a lot of these molecules are bigger than a typical lead for a drug. In fact, we may just want to dock fragments. To do this, we need the molecular weight. Use this script: https://github.com/ryancoleman/tdt2014-part2/blob/master/scripts/smilesmass.py

```
`smilesmass.py uniq.smi | sort -k 3 -n > uniq.mass.sort.smi`
```

Note that this script uses the OpenEye toolkit to calculate the molecular mass from the SMILES string.

This gives you a list sorted by molecular mass. We want to take out just the molecules with molecular mass < 250 (fragments) and with those between 250 and 350 (leads). There are several ways to do this, the laziest is to just use a text editor. Check out the final files here: https://github.com/ryancoleman/tdt2014-part2/blob/master/step2files/emolecules.canon.uniq.mass.sort.frags.txt and here https://github.com/ryancoleman/tdt2014-part2/blob/master/step2files/emolecules.canon.uniq.mass.sort.leads.txt

Now, you can remove the 3rd column of molecular mass with this command:

```
`cat emolecules.canon.uniq.mass.sort.frags.txt | awk '{print $1, "E"$2}' > frags.smi`
```

Again, we have to add a character to the beginning of the ID for historical reasons.

Now, use the same procedure from earlier to build the fragment database

Now that you have the input SMILES file, you can proceed with building this much larger docking database:

```
`db2start.e.cxcalc.sh frags.smi`
```

to start the ligand building process. Once all jobs have finished, a few molecules will not be correctly built, but can be rescued by simply re-running them. To do this, go into the marvin directory and then run the following command, like this:

```
`cd marvin`
`subprepG2.e.cxcalc.db2.redo.csh`
```

Which will start a few more jobs. Once they are all complete again, return to the starting directory and compile the many ligand databases into a few big files.

```
`cd ..`
`db2end-prefix.py frags`
```

Or some other name. This time, you'll get a lot of files (I got 507!). This is a lot of small molecules, but the hard part (building the ligand database) is over and docking is now relatively fast & easy.

Now, when you build this many files, sometimes one or two ligands aren't built correctly. I wish all these problems didn't exist, but they do! For now, run this script: https://github.com/ryancoleman/tdt2014-part2/blob/master/scripts/remove_incomplete_db2gz.py on all your docking database files and the wrong parts of each file will be removed.

Another step we decided to do on the fragments was to remove the fragments with more than one formal charge. The reason for this is that passively diffusable, orally bioavailable drugs often have this limitation, so docking extremely charged molecules is often unproductive. Run this script on the entire database and it will remove the molecules with charge less than -1 and greater than +1: https://github.com/ryancoleman/tdt2014-part2/blob/master/scripts/remove_non-q1_db2gz.py

You can do the same thing for leads, but there are a lot more leads and the build will take a very long time (weeks, depending on how many computers you have). In fact, this took so long for

me, that I just used leads less than 300 daltons. The input smile file is here:
https://github.com/ryancoleman/tdt2014-part2/blob/master/step2files/emolecules.leads.lt300.smi

Downloading the Ligands

For the eMolecules Fragment data, you can download it here (it is too big for github):

http://tools.bkslab.org/~rgc/tdt2014/frags/

You'll want to download all the files in that directory.

Leads (less than 300 daltons in mass) are in:

http://tools.bkslab.org/~rgc/tdt2014/leads/

b) Rank-order commercial compounds based on predicted activity against Pf Krs1

Two separate docking runs (fragments and leads less than 300 daltons) were done the adjusted receptor conformation.

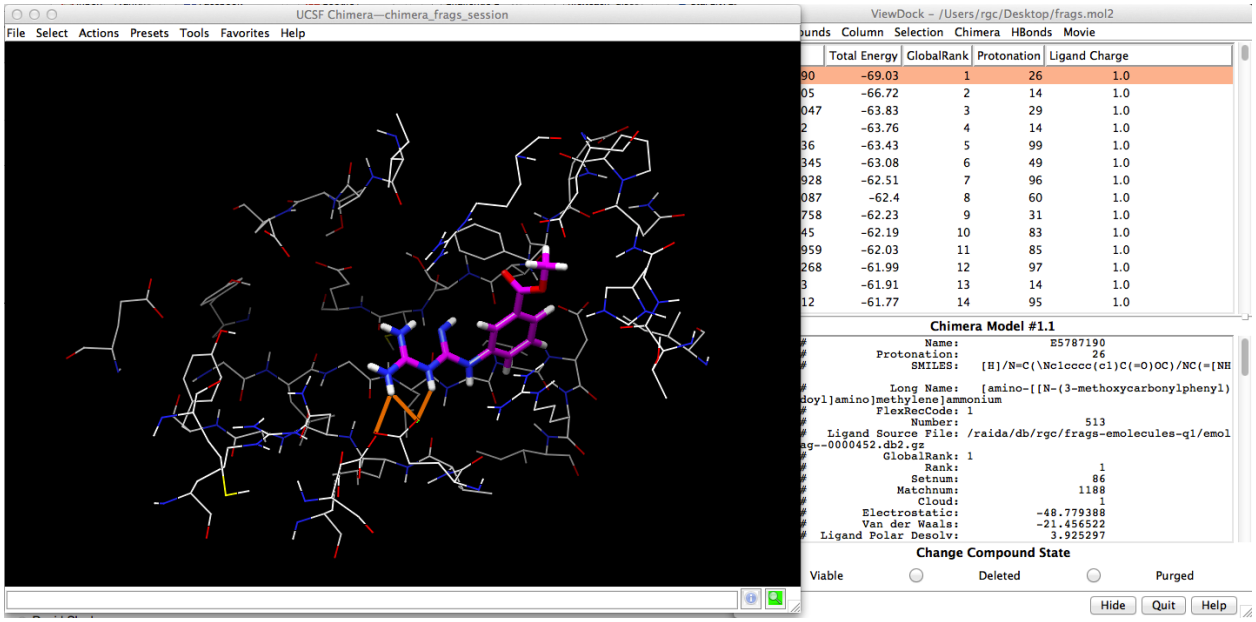
```
`setup_db2_lots.py 1000 tcams /path/to/fragments/database/db2/files/`
 `submit.csh`
```

Now, just wait for those jobs to finish (this will take awhile). Then, like before, run

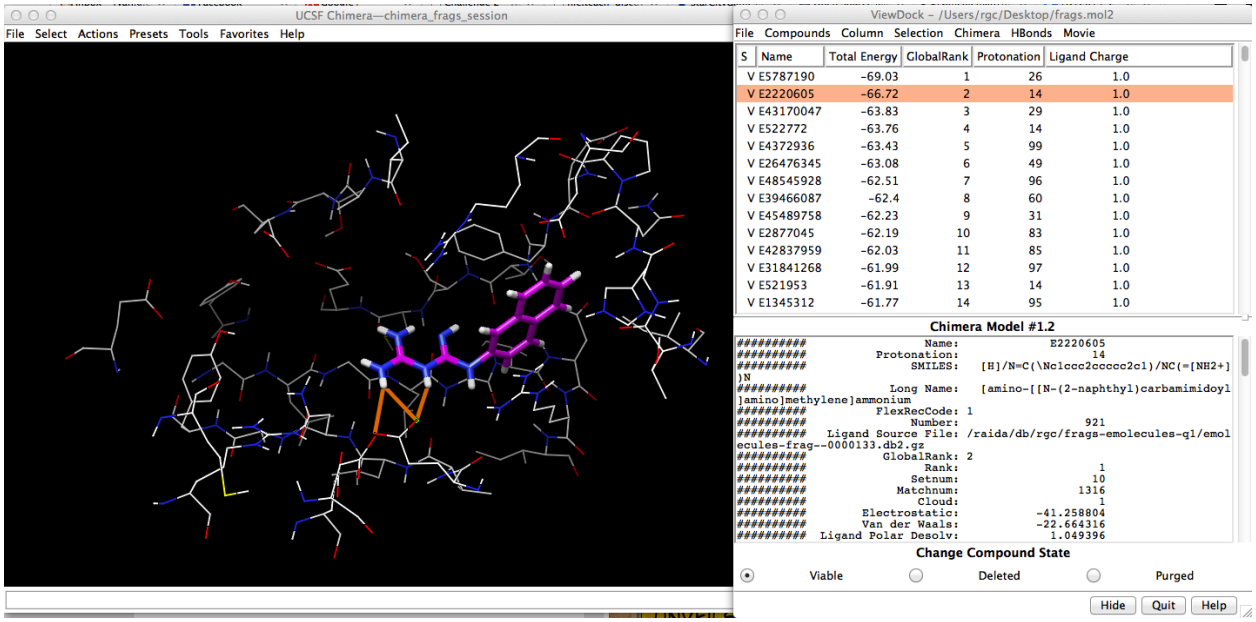
```
`extract_all.py` then
`getposes.py -o frags.mol2`
```

And a similar run for the leads (again, we're using a smaller version of the leadlike database, where only molecules up to 300 daltons in mass were considered).

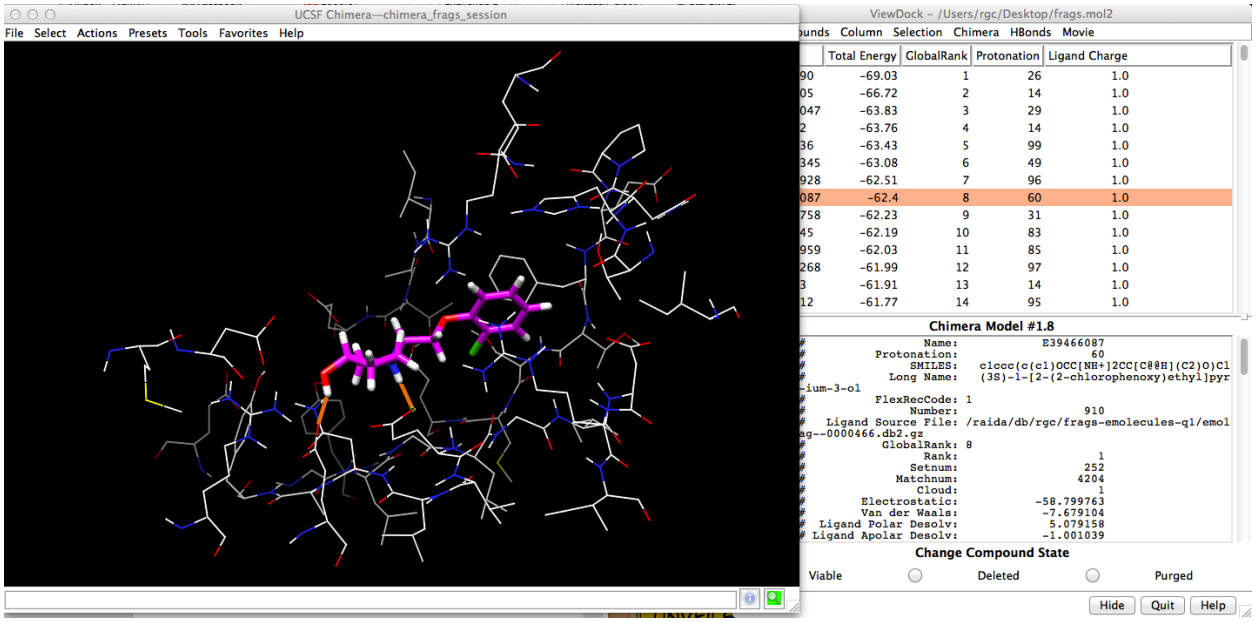
Both output files from our runs are here:
https://github.com/ryancoleman/tdt2014-part2/blob/master/step2files/frags.mol2 &
https://github.com/ryancoleman/tdt2014-part2/blob/master/step2files/leads.mol2
As before, we load them into Chimera using the ViewDock module. We're going to look through the fragments and make note of a few key features when examining putative ligands.



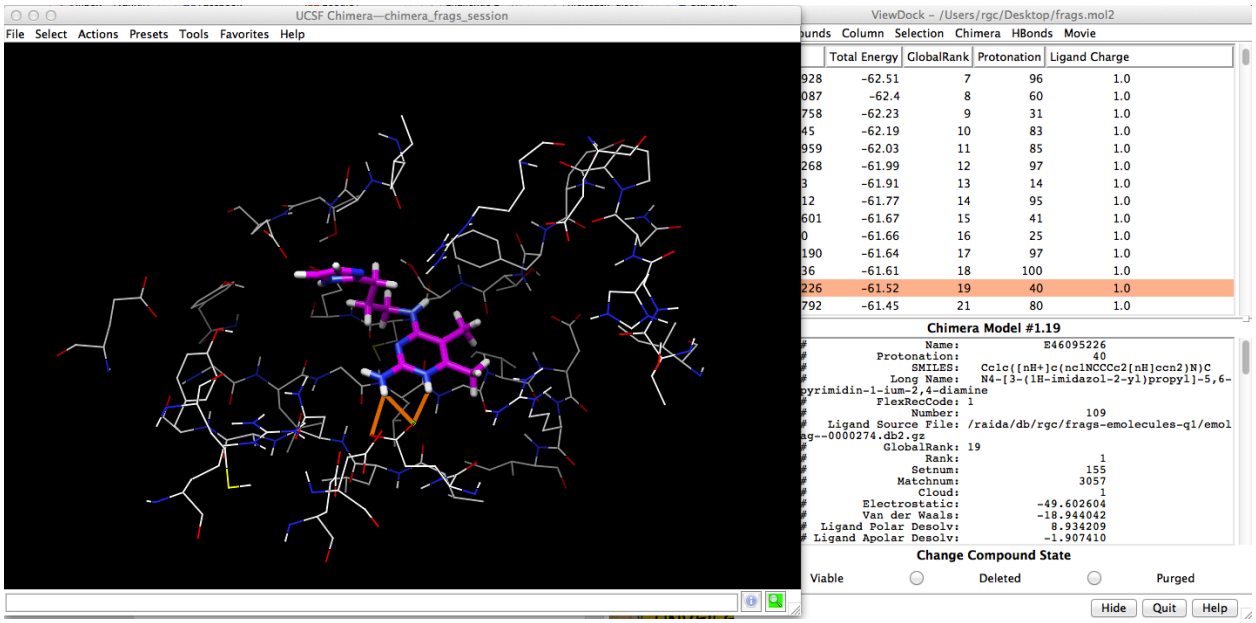
This is the top-ranked ligand. The charged group at left is usually worrisome because there are concerns about the exact protonation state. As you can see, ChemAxon's Marvin program assigns this exactly protonation state at 26% at pH 7.4. However, the good things about this pose are numerous, it fills the ATP/putative cladosporin site well. It doesn't strand many polar groups on either the protein on the ligand. Except for the concern about the charged group, this is a good putative ligand for follow-up testing.



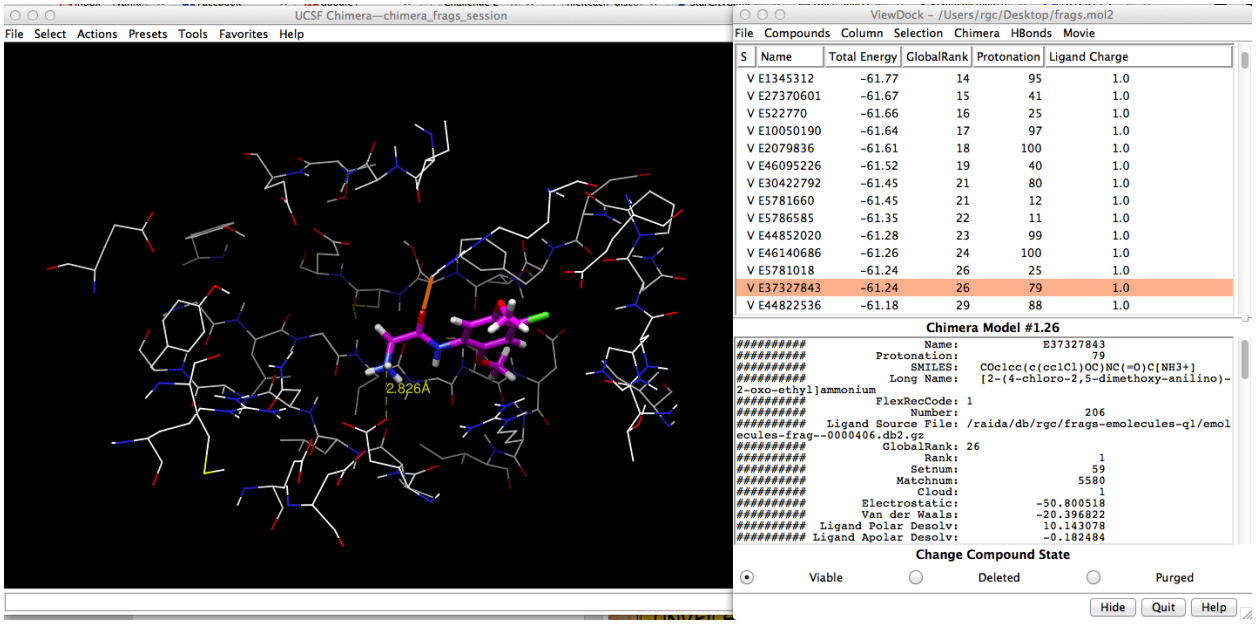
The second-ranked fragment looks very similar to the first one. The only difference is the hydrophobic group at right. This ring is even more hydrophobic, but the ring in the first-ranked structure wasn't making any critical polar contacts.



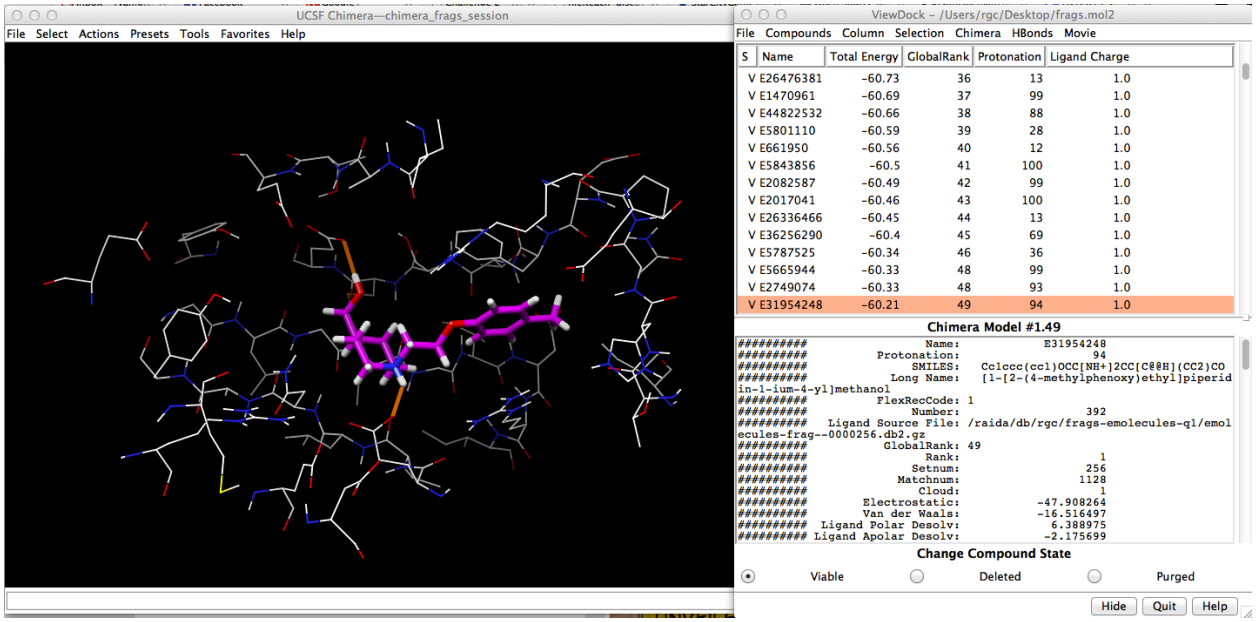
The protein has been rotated to show the excellent polar contacts made by the 8th ranked fragment. The ATP/putative cladosporin site is still filled well by the molecule.



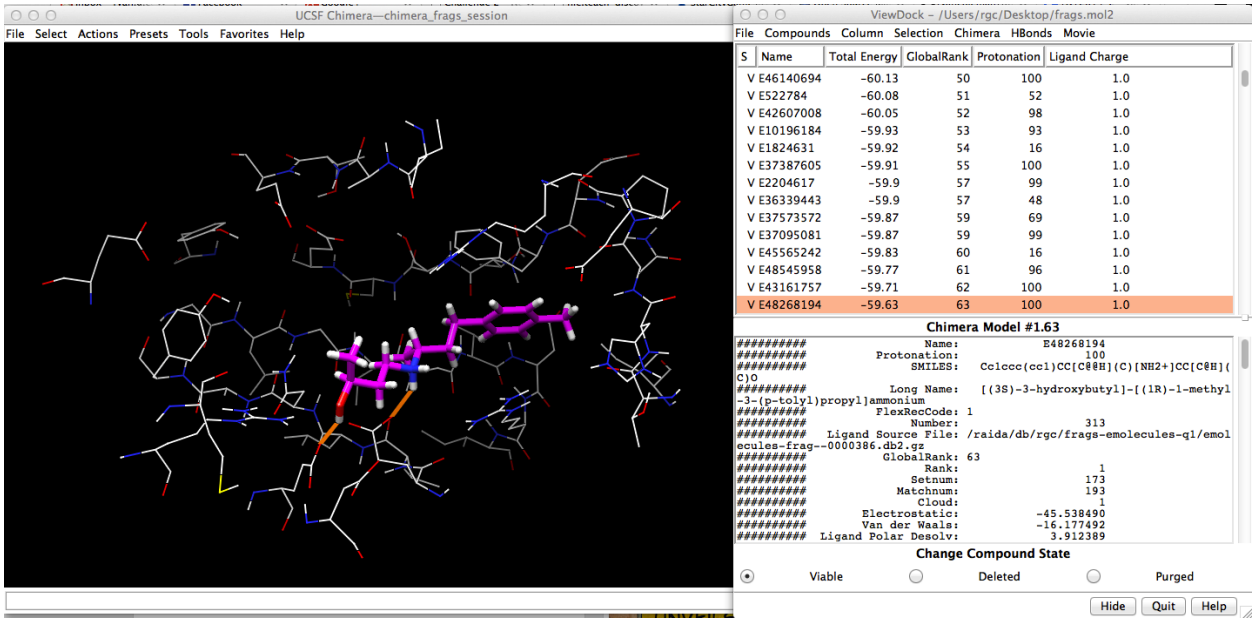
Here is an example of a particularly bad molecule in rank number 19. It doesn't fill the presumed binding site. It does make some polar contacts, but strands several others. Though it receives a good score, we presume it will not actually bind, however experimental testing is the only way to be sure.



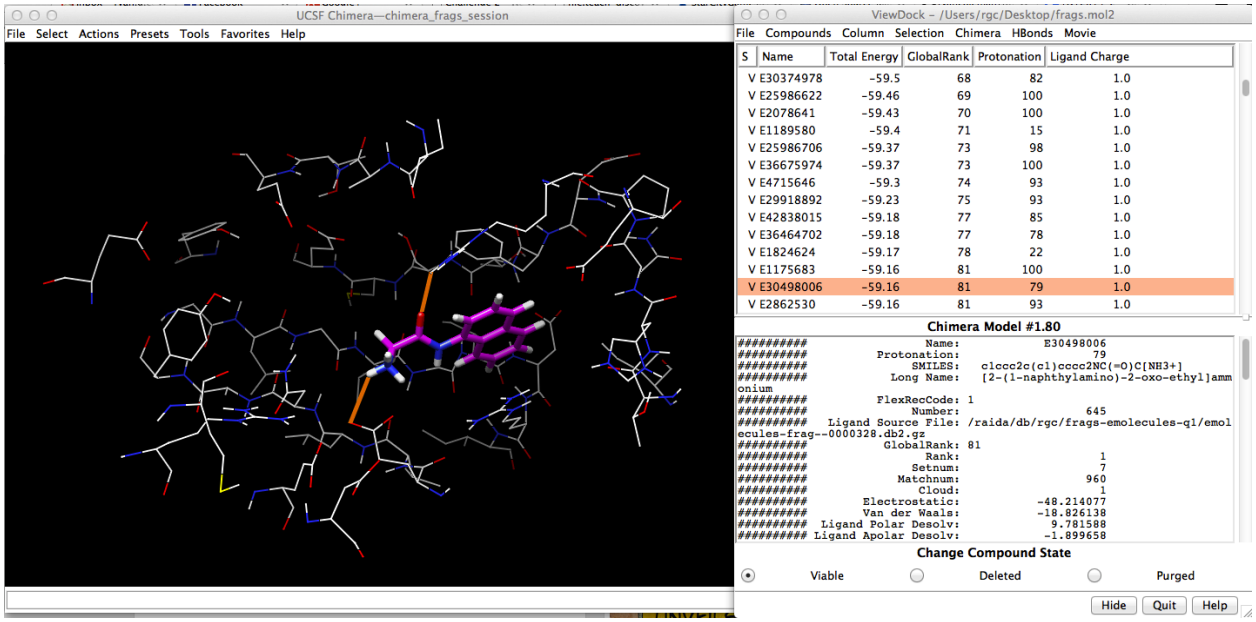
This is another excellent molecule. It makes a charge-charge interaction with the tertiary amine on the ligand and the negatively charged carboxylate on the protein. It also makes a polar contact between the carbonyl group of the ligand and the charged arginine on the protein. It still fills the ATP/putative cladosporin site well for a fragment. This is an excellent molecule for follow-up testing.



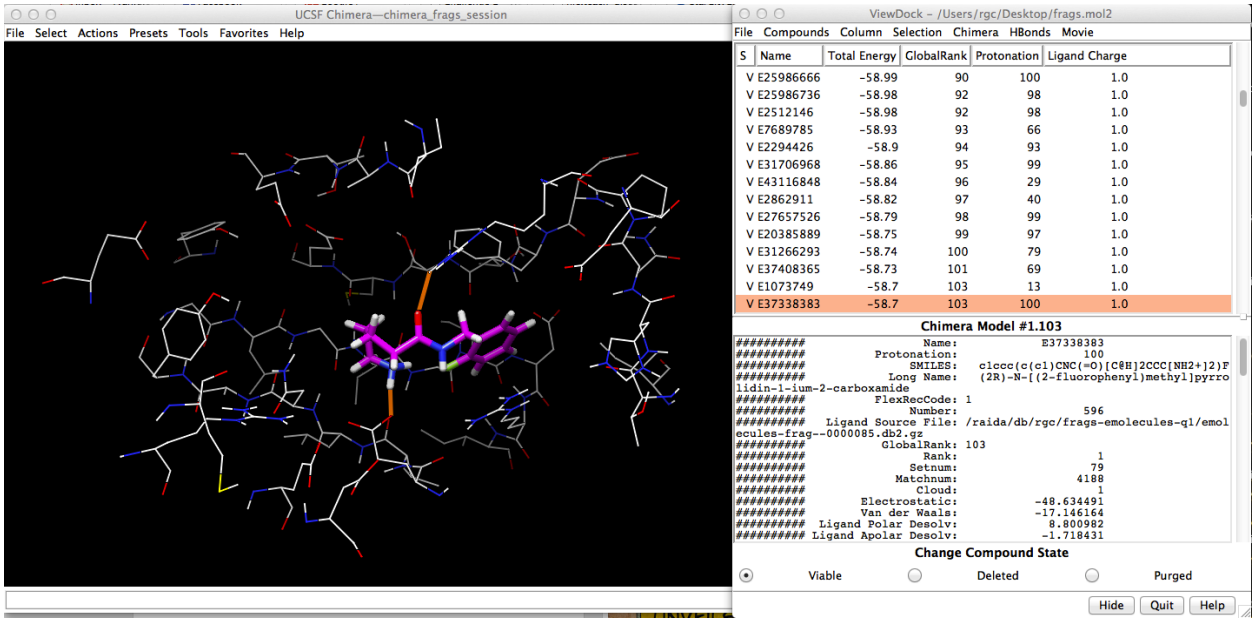
This molecule fills the binding site well and makes polar contacts in multiple places. It does this without stranding other polar groups. Again, this molecule (rank #49) is a good candidate for follow-up testing.



Again, this molecule is an excellent candidate for followup screening. The hydrophobic pocket at right is well filled by the fragment. There are multiple polar contacts and all the polar groups on the ligands are involved in a polar contact.



This molecule (rank #81) is similar to the fragment ranked #26 on the left. It has a more hydrophobic group on the right, but like molecule #2, this is probably a good thing overall as that small pocket is very hydrophobic.



This molecule again makes multiple polar contacts and fills the binding site well. This, along with many other fragments found using molecular docking, are good candidates for follow-up experimental testing.

UCSF Chimera sessions for both prospective screens are here:
https://github.com/ryancoleman/tdt2014-part2/blob/master/step2files/chimera_frags_session.py
&
https://github.com/ryancoleman/tdt2014-part2/blob/master/step2files/chimera_leads_session.py

As before, we need to run a script to make the submission file. The ranked order will be fragments then leadlike molecules. So, go into each directory for fragments and leadlike molecules and run this script:
https://github.com/ryancoleman/tdt2014-part2/blob/master/scripts/make_prediction_output.py
Using this as the emolecules.smi file:
https://github.com/ryancoleman/tdt2014-part2/blob/master/step2files/emolecules.smi. First, in the frags directory.

```
`./make_prediction_output.py emolecules.smi extract_all.sort.uniq.txt 500 > prospective.emolecules.hits.txt`
```

Then, in the leads directory:

```
`./make_prediction_output.py emolecules.smi extract_all.sort.uniq.txt
500 >> prospective.emolecules.hits.txt`
```

This makes a prospective file as required for the competition: https://github.com/ryancoleman/tdt2014-part2/blob/master/step2files/prospective.emolecules.hits.txt

Adding this tutorial https://github.com/ryancoleman/tdt2014-part2/blob/master/submission/tutorial.pdf, once completed, to the submission files, results in the submitted tarball: https://github.com/ryancoleman/tdt2014-part2/blob/master/submission/submission.tar.gz

Specific files to include for this challenge

1. PDB file with the Pf Krs1 protein and predicted binding mode for cladosporin: https://github.com/ryancoleman/tdt2014-part2/blob/master/submission/pfkrs1-cladosporin-adj.pdb
2. Predictions for held-out, external test set in Pf Krs1 (identifier, smiles, measure of predicted activity): https://github.com/ryancoleman/tdt2014-part2/blob/master/submission/tcams.predictions.txt
3. Rank-ordered list of top-1000 commercial compounds predicted to be active (identifier, smiles): https://github.com/ryancoleman/tdt2014-part2/blob/master/submission/prospective.emolecules.hits.txt

All these files, and this tutorial, are available in the following .tar.gz file: https://github.com/ryancoleman/tdt2014-part2/blob/master/submission/submission.tar.gz

Citations & Further Reading

Ryan G. Coleman*, Michael Carchia, Teague Sterling, John J. Irwin, Brian K. Shoichet. Ligand Pose and Orientational Sampling in Molecular Docking. PLOS ONE. October 1, 2013. PDF. PubMed Central Free Full Text. Code. Documentation. Auto-DUD-E Test Set.

Gaulton, et al. ChEMBL: a large-scale bioactivity database for drug discovery"" Nucleic Acids Research. Volume 40. Issue D1. Pages D1100-D1107. 2011. http://nar.oxfordjournals.org/content/40/D1/D1100

Coleman, Sterling & Weiss. SAMPL4 & DOCK3.7: Lessons for automated docking procedures. Journal of Computer-Aided Molecular Design. 2014. http://link.springer.com/article/10.1007/s10822-014-9722-6

H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, P.E. Bourne (2000) The Protein Data Bank. Nucleic Acids Research, 28: 235-242.

Hoepfner et al. Selective and specific inhibition of the plasmodium falciparum lysyl-tRNA synthetase by the fungal secondary metabolite cladosporin. Cell Host Microbe. 2012 Jun 14;11(6):654-63. http://www.ncbi.nlm.nih.gov/pubmed/22704625

Guo et al. Crystal structure of tetrameric form of human lysyl-tRNA synthetase: Implications for multisynthetase complex formation. http://www.pnas.org/content/105/7/2331.long

Khan et al. Structural analysis of malaria-parasite lysyl-tRNA synthetase provides a platform for drug development. Acta Cryst. (2013). D69, 785-795. http://scripts.iucr.org/cgi-bin/paper?S0907444913001923